Tallinna Ülikool

Digihumanitaaria tehnoloogiad

Jaagup Kippar

Tallinn 2018

Sisukord

Sissejuhatus **6**

Naiste ja meeste arv veebilehelt **7**

Andmed veebilehelt 7

Eesnimede eraldamine 9

Ülesandeid 16

Arvude võrdlemine ja esitamine 16

Veel suhteid 17

Ülesandeid 18

Regulaaravaldised **19**

Otsing 19

Asendamine 20

Ülesandeid 21

Andmete avaldamine veebis **22**

Putty klient 26

Kokkupuuted Linuxi käskudega 27

Sisevõrgu masina veebi vaatamine väljastpoolt 29

Shelli käsurida **31**

ls 31

more 31

echo 31

Tulemuse saatmine faili 32

Tekstiredaktor pico 32

Loendamine - wc 33

Read faili algusest ja lõpust 35

Harjutus 37

grep 37

Skriptid, sortimine 38

Harjutus 42

Failide kopeerimine 43

Tulba arvude summa 46

Harjutus 47

Aritmeetilise keskmise leidmine 47

Harjutus 49

curl 49

wget 50

Python **51**

split, tükeldamine 52
Muutujad 53
Tingimus 53
Massiiv 54
Harjutus 55
Regulaaravaldised 56
Harjutus 57
Hulgad 57
Harjutus 57
Tehted hulkadega 58
Harjutus 59
Loendamine 61

Pandas 62
Sortimine 63
Rühmitamine 66
Sõnaliikide uuring 67
Tabelite ühendamine 70
Sõnaliikide osakaal 73
Programmikood failis 76

Estnltk 77
Sõnade andmed 77
Veebist andmete lugemine. 78
Kõikidest liikidest sõnade loendamine 79
Harjutus 83

Tähepaarid 87
Abivahendiks Pandas 88
Andmed veebilehel 90
Veebilehe loomine programmikoodi abil 92
Uuritava teksti andmed veebilehel 93
DataFrame veebilehel tabelina 94
Valitud andmed veebilehele 96

Joonised 99
Selgitustekstidega joonis 100
Tulpdiagramm 102
Sektordiagramm 103
Joonis veebist loetud andmete põhjal 104
Käsklus DataFrame tulba küljes 105
Veebilehe loomine andmete põhjal 107
Mitme tunnusega tulpdiagramm 109
Tulbad üksteise peal 110

Järjestatud horisontaalsed tulbad 110
Standardviga tulpdiagrammil 111
Karpdiagramm 112
Karpdiagramm skaleeritud andmetega 113
XY-diagramm 115
Punktid ja joon ekraanil 117

SQL 120
Andmete loomine 120
Andmebaasi loomine 121
Andmebaasi sisenemine 121
Andmetabeli loomine 121
Andmete sisestamine 122
Andmete päring 123
Harjutus 124
Agregaatfunktsioonid 126
Harjutus 127
Andmete muutmine 128

Näited keelekorpuse andmetega 130
Tekstide võrdlus 133
Andmebaasiskeem 140
Sõnaliigipaaride sageduste võrdlus 143
Dokumentide metaandmed 145
Sõnaliikide paarid 147
Sõnaliigipaaride sageduste võrdlus keeleti 148
Harjutus 149

Python ja MySQL 151
Joonis SQL-tabelist tulnud andmete põhjal 153
Sõnaliikide paarid 155
Harjutus 158
Joonis genereeritud veebilehel 160
Andmed mitme emakeele kohta 163
Vastused omaette kataloogis 165

PHP 169
Päring andmebaasist 169
Sisestus kasutajalt 171
Sõnaliigipaarid vastavalt emakeelele 173
Harjutus 174
Sisestus rippmenüüst 178
Sisestatu säilitamine lehel 185
Sessioonimuutuja 187

Andmete lisamine 190
Tekstide võrdlemine 196
Harjutus 201

Kordamisküsimused **206**

Sissejuhatus

Kõigeoskajate aeg hakkab tagasi tulema. Või vähemasti tasub oma valdkonna töö juures kasutada mujalt avanenud võimalusi. Humanitaarteadustes kasutatakse digitehnoloogiaid järjest julgemalt ning need pakuvad vähemasti triangulatsioonina võimalusi seniste järelduste kinnitamiseks või kahtluse alla seadmiseks, lähemal uurimisel toovad aga välja uusi ja vahel ootamatuidki seoseid. Siinne tutvustav materjal annab kätte tehnoloogilised vahendid muuhulgas humanitaarvaldkonnas ettetulevate andmetega ümber käimiseks.

Vahel öeldakse, et matemaatika, millega tegelevad matemaatikud, võib olla suhteliselt lihtne - või vähemasti kitsalt piiritletud. Füüsikutel ja mitmesugustel muudel loodusteadlastel tuleb rinda pista juba märgatavalt keerukama matemaatikaga. Nende uuritavatel probleemidel pole sageli küljes selgeid võrrandeid, küll aga on vaja leida lahendusi, kus ka võrrandid vahel kasulikuks osutuvad ning siis tuleb neid sobivalt kombineerida. Näitena tuuakse vastu kaljut paiskuv merelaine, selle käigus mõjuvad jõud ning lendavate piiskade trajektoorid ja ühinemised. Matemaatika aga, millega tegelevad ajaloolased, ühiskonnateadlased, muusikud ja filoloogid võib olla veel märgatavalt komplitseeritum. Samas usinal katsetajal ja süvenejal on selle abil mõndagi võita.

Sama paistab olema digitehnoloogiatega. Informaatikud õpivad esimesel kursusel küllalt selgepiirilisi põhimõtteid ja tehnikaid. Loodusteadlastel on juba esimestel semestritel vaja leida lahendusi küsimustele, millele ei ole head otsest ja mugavat digimaailma lahendust. Humanitaarerialadel tuleb mõnigikord otsida ja kombineerida tehnikaid, et neile vajalikud olukorrad digimaailmale lahendatatavaks teha ning tulemuseks on mõnigikord lahendused, mille puhul “tavalised” informaatikud peavad tüki aega nuputama, et nähtud lahendus tehniliselt läbi hammustada rääkimata sellest, et suurema osa lahenduse sisust ja tähtsusest moodustab selle erialane pool olgu filoloogias, muusikas, ajaloos või mujal.

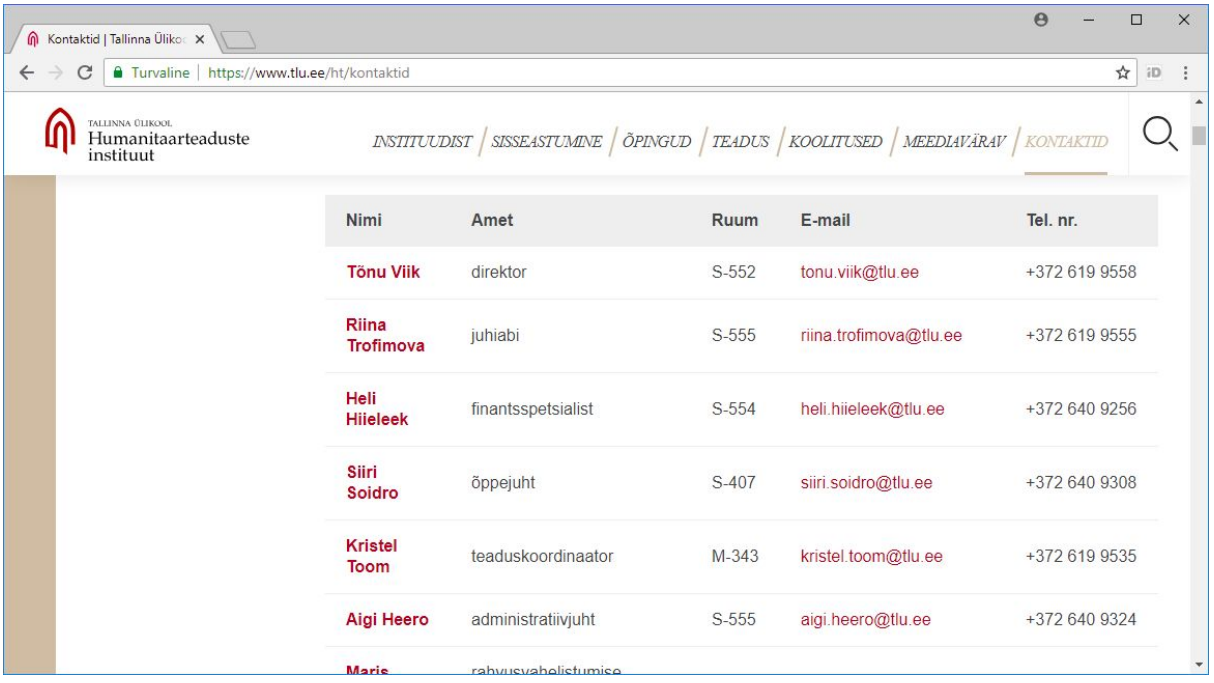
Siinses õpikus keskendutakse olemasolevate tehnoloogiate kasutamisele humanitaarvaldkonna näidete juures. Mõnigikord jäetakse lihtsustuse huvides märgatav osa vastava vahendi taustast või võimalustest märkimata. See osalt küll takistab detailset süvenemist, kuid kogemused esimeste õppurikursustega näitavad, et vajalikes ja rohkem kasutatud valdkondades jõutakse näidete kaudu ringiga ka tehniliste põhimõteteni - vähemasti sel määral, et õpitud vahendeid suudetakse edukalt oma töös rakendada.

Naiste ja meeste arv veebilehelt

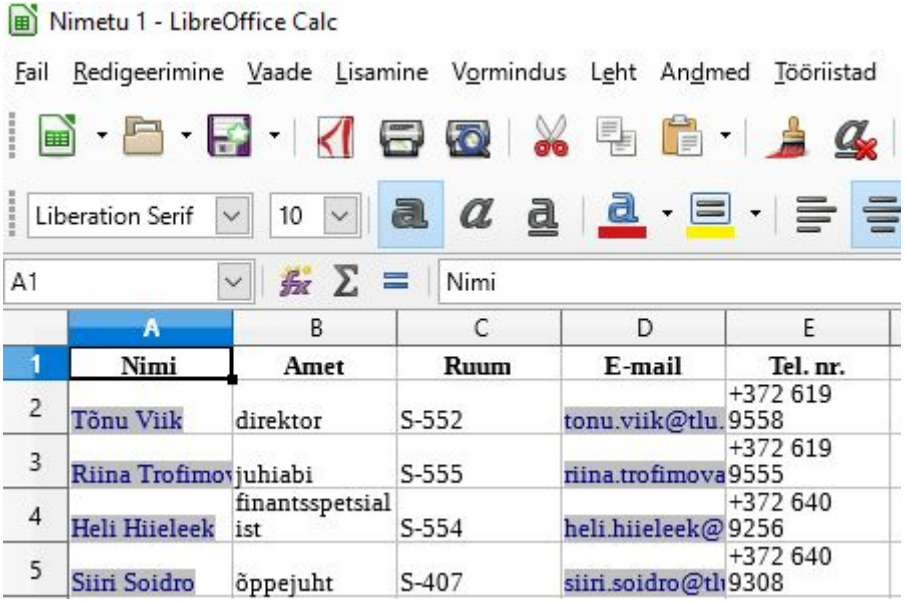
Paari tuttava nimega loetelu puhul võib korraks peale vaadata ning juba ongi teada, mitu mees- ja naishäält lauluseltskonnas on ja sealtkaudu häälte jagunemise ja repertuaari kohta otsuseid teha. Suuremate andmehulkade puhul aga jõuab mingil ajal arvuti kiiruse poolest nobedamalt toimetada. Kui suurte puhul - see iseküsimus. Paarsada või ka paar tuhat nime käsitsi ühekordselt üle vaadata õnnestub tõenäoliselt rutem kui selle jaoks eraldi programmi kirjutama hakata. Tuleb aga sellist kontrolli pidevalt korduvalt teha või tõesti on juba inimeste arv kümnetes tuhandetes, siis võib arvutist märgatavalt kasu olla. Mugavaimal juhul täisautomaatsena, kus algandmete muutuse korral varsti uued tulemused välja arvutatakse ja sellest ka teada antakse. Mõnigikord on aga lihtsam leppida, et arvuti aitab üksikute aeganõudvamate etappide juures ning osa samme tuleb siiski käsitsi läbi teha.

Andmed veebilehelt

Tänapäeval saab märgatava koguse andmeid kätte veebi kaudu. Samas ei pruugi kättesaadav olla edasiseks töötluseks kuigi mugaval kujul - siis tuleb leida kohandamiseks mooduseid. Üsna mitmekülgne moodus on lehelt teksti kopeerimine lihtsasse tekstiredaktorisse (näiteks Notepad, pico), seal sobivate asenduste tegemine ning siis juba vajaliku osa sihtkohta sättimine. Mõni tabel õnnestub ka otse tabelarvutusprogrammi üle tuua - näiteks Tallinna Ülikooli humanitaarteaduste Instituudi töötajate loetelu.



Kui tabelarvutusprogramm tunneb vahekohad ära, siis jõuavad lahtrid veebilehelt lahtriteks arvutustabelisse. Selliste ülekandmiste puhul ei saa kunagi kindel olla, et moodus, mis ühe lehe ja tabelarvutusvahendiga kehtis, kehtiks ka pärast veebilehe kujunduse muutust või arvutustabeli tarkvara versiooniuuendust. Samas digitehnoloogi töö märgatavalt osalt ongi käepäraste võimaluste otsimine ja rakendamine ning kui parasjagu nõndamoodi sobivalt lahenduseni jõuab, siis tasub seda moodust parasjagu pruukida.



	A	B	C	D	E
1	**Nimi**	**Amet**	**Ruum**	**E-mail**	**Tel. nr.**
2	Tõnu Viik	direktor	S-552	tonu.viik@tlu.	+372 619 9558
3	Riina Trofimov	juhiabi	S-555	riina.trofimova	+372 619 9555
4	Heli Hiieleek	finantsspetsialist	S-554	heli.hiieleek@	+372 640 9256
5	Siiri Soidro	õppejuht	S-407	siiri.soidro@tl	+372 640 9308

Kord andmed käes, tasub neid sobivaks puhastama hakata. Kuna kavas on eesnimede järgi määrata, kui palju millisest soost inimesi nimekirjas leidub, siis loobun muudest tulpadest ning kopeerin esimese tulba andmed uuele lehele. Nagu näha, leiduvad seal nii ees- kui perekonnanimi ning vahele tulevad ka read allüksuste nimedega. Et veebileht vahepeal muutunud, siis ka eri joonistel olevad andmed veidi erinevad.

Nimi
Tõnu Viik
Riina Trofimov
Siiri Soidro
Kristel Toom
Aigi Heero
Maris Peters
Helina Puksma
Signe Steinpilr
Jana Lehtsalu
Kristiina Rann
Tiiu Rumen
Jaanika Pern
Heli Hiieleek
Denis Kuzmin
Doris Feldman
Doris-Marii Kc
Elina Tahvel
Akadeemiline suund: Aasia uuringud
Nimi
Otto Jastrow
Alexander Hor

Vormingu eemaldamiseks on mugav andmed tõsta korraks Notepadi või muude lihtsasse vorminguta redaktorisse ning pärast jälle tagasi. Mõnel tabelarvutusvahendil ka omal olemas vormingute alt käsklus vormingu eemaldamiseks. Andmete ritta sättimiseks tabelarvutusprogrammil olemas sortimiskäsk. Nii tulevad ka näiteks välja kohad, kus sama inimene on mitmel korral tabelis (näiteks eri üksustes)

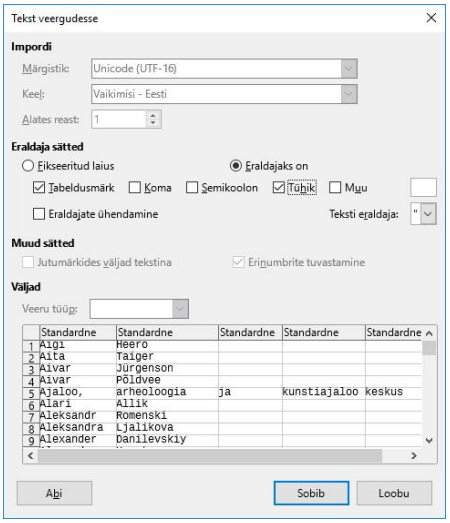
	A	B	C
1	Aigi Heero		
2	Aita Taiger		
3	Aivar Jürgenson		
4	Aivar Põldvee		
5	Ajaloo, arheoloogia ja kunstiajaloo keskus		
6	Alari Allik		
7	Aleksandr Romenski		
8	Aleksandra Ljalikova		
9	Alexander Danilevskiy		
10	Alexander Horstmann		
11	Ana Sofia Chermont de Magalhaes		
12	Andine Friederike Frick		
13	Andres Luure		
14	Anna Verschik		
15	Anne Lange		
16	Annekatrin Kaivapalu		
17	Anneli Kõvamees		
18	Anneli Kõvamees		

Eesnimede eraldamine

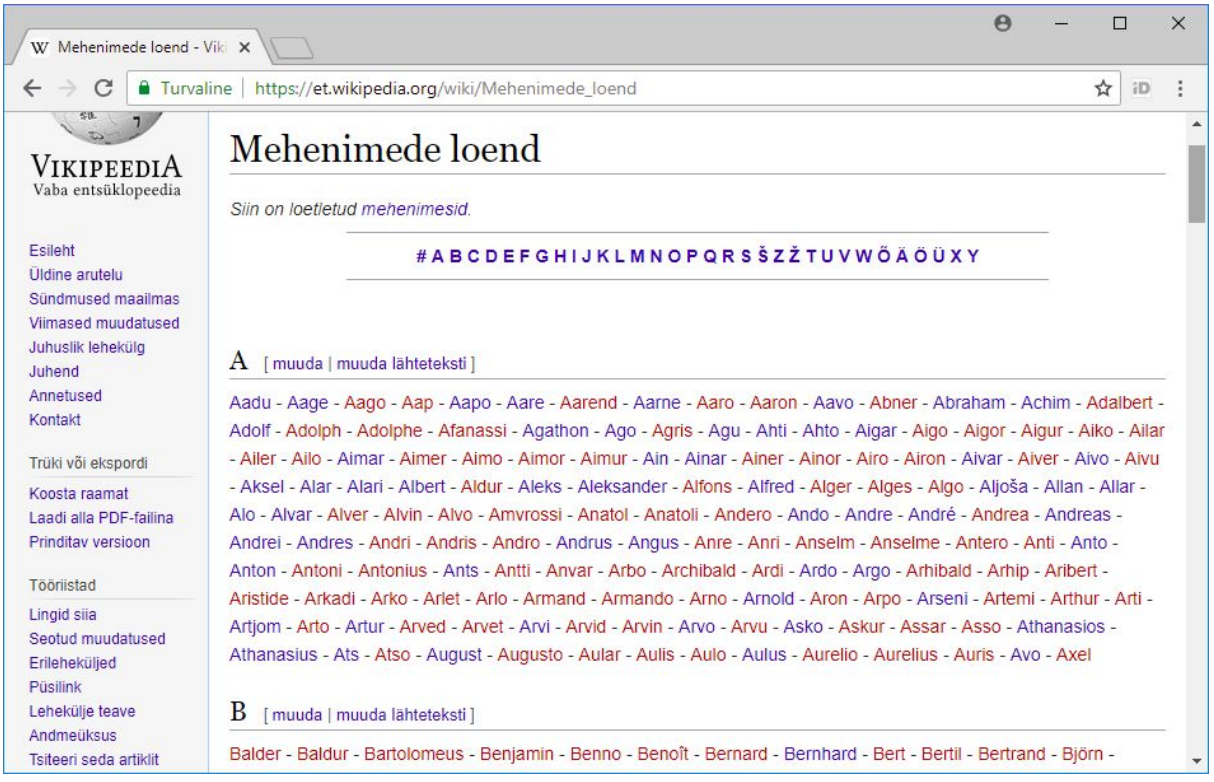
Soo kindlaks tegemiseks on vaja eraldi eesnime. Kus lähteandmete juures eesnimed eraldi, seal neid nõnda mugav pruukida. Küllalt sageli aga andmete veebilehele paneku puhul pole mõeldud nende edasise automaatkasutuse peale (või isegi püütakse sellest hoiduda) ning siis tuleb sobiva kätte saamiseks eraldi tegutseda.

Inimestel võib olla rohkem kui üks eesnimi. Õnneks vähemasti esimese nime saab siin enamasti eesnimeks lugeda. Perekonnanime puhul tasub jälle viimast otsida. Kus kahe sõnaga piirdutakse, seal on tõenäoline, et ees- ja perekonnanimi on suhteliselt selgelt määratud. Samas mitmesugused üksuste nimed on ka loetelus sees ning nemad enamasti pikemad kui kaks sõna. Nii tasub pikematesse ridadesse ettevaatlikumalt suhtuda ning vajadusel käsitsi korrektiive teha. Mõni kahesõnaline üksus saab ka nimede vahele peitu pugeda, aga nende otsimise juhul tasub juba hinnata, et kui suur osa neid on ning kas meil sellist täpsust vaja, et nende otsimine end ära tasub.

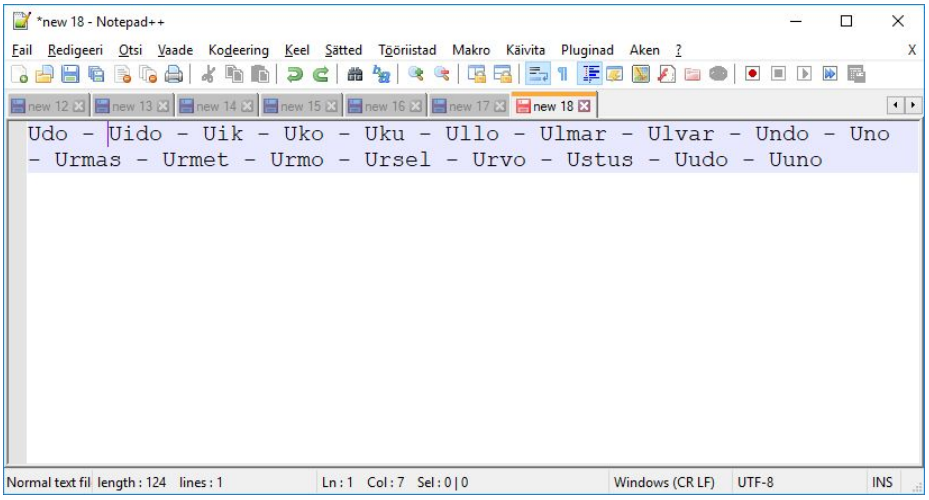
Tabelarvutusprogrammi juures üldjuhul leiab teksti veergudesse paigutamise vahendi, praegusel juhul määran, et tulpade eraldajateks oleksid tühikud sõnade vahel

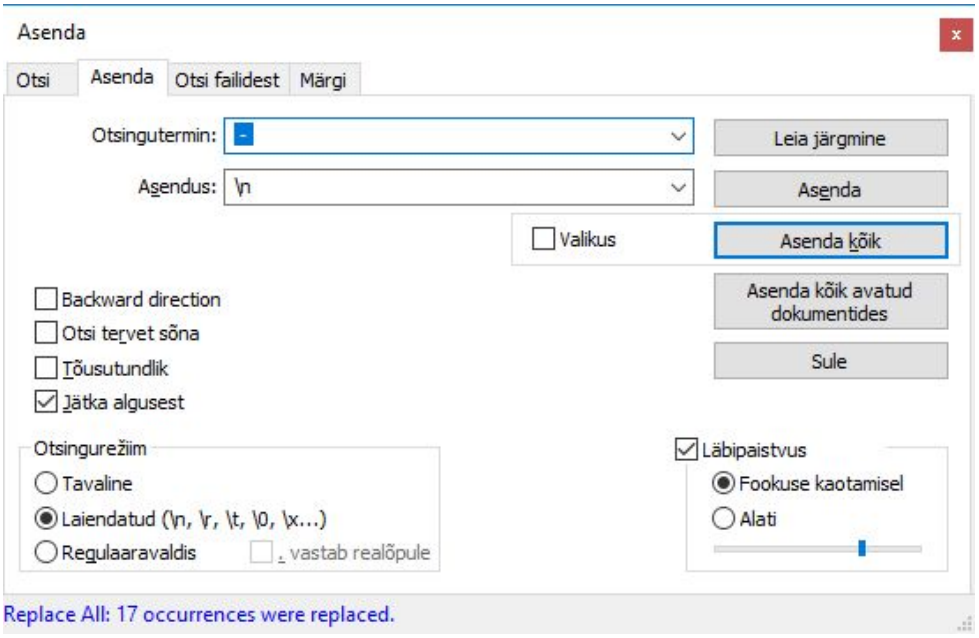


Nüüd hulk nimesid olemas. Arvuti aga endiselt ei tea, et keda meheks ja keda naiseks pidada - ning vastavat standardfunktsiooni ei kipu ka olema. Õnneks leidub Wikipedias lehekülg mehenimede loendiga, sarnase leiab ka naisenimede kohta. Päris kõiki nimesid küll seal sees ei ole ning mõni nimi võib ka mõlemale sobida, aga mingi enamvähem hinnangu saab niimoodi kätte sellegipoolest. Jällegi peab muidugi mõtlema, et saavutatav täpsus meid rahuldab, aga see küsimus on tarvilik enamvähem iga uuringu juures.

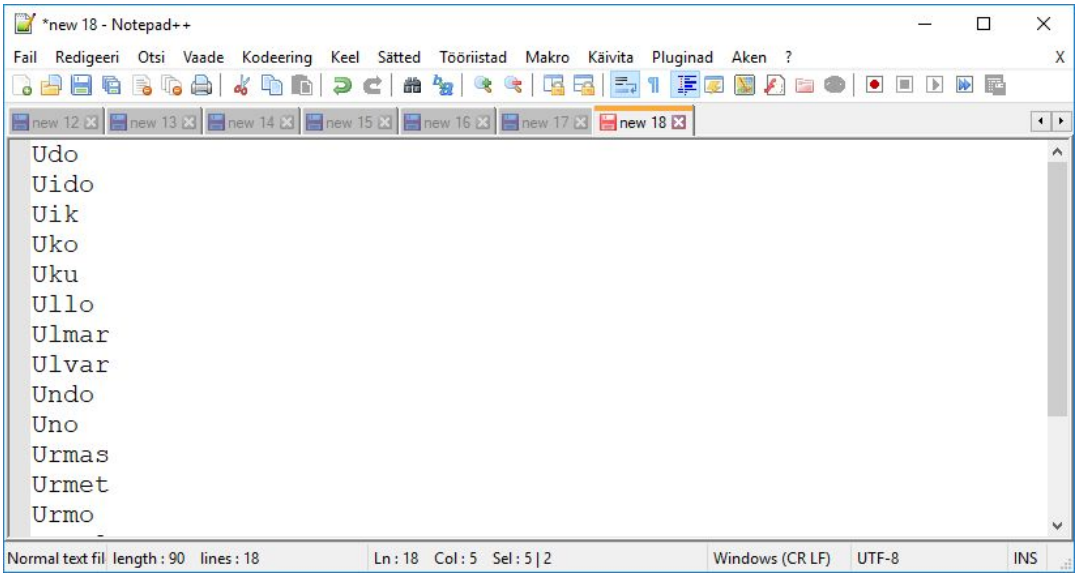


Nimede ühte tulpa saamiseks on üks võimalus kopeerida nad tekstiredaktorisse (nt. notepad++), kus asendada sõnu eraldav miinusmärk reavahetussümboliga. Tavaline notepad ei sobi, sest too ei suuda reavahetusmärki asendusse määrata.

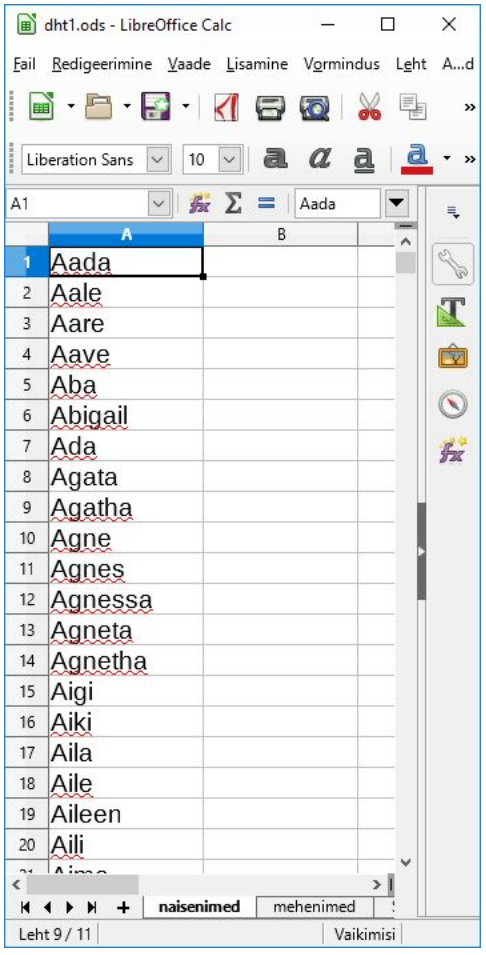


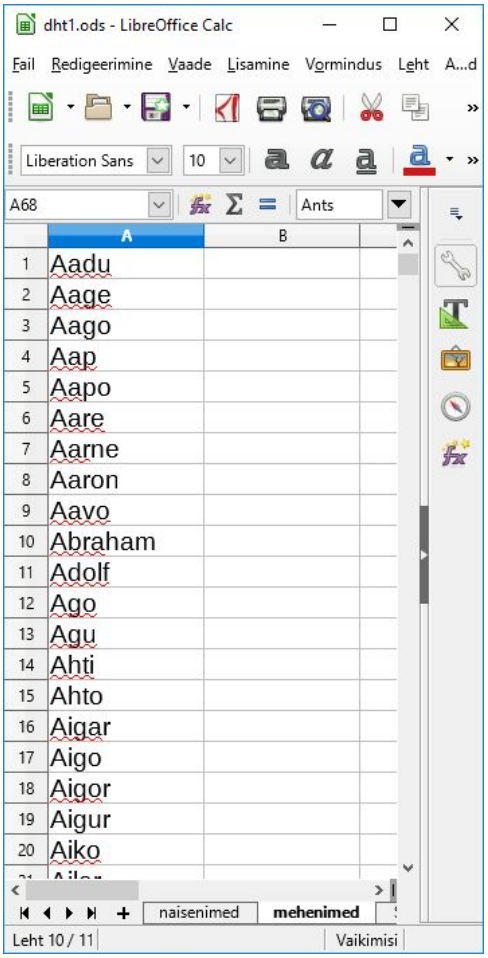


Tulemusena ongi nimed üksteise all ning nood saab tagasi tabelarvutuslehele kopeerida.

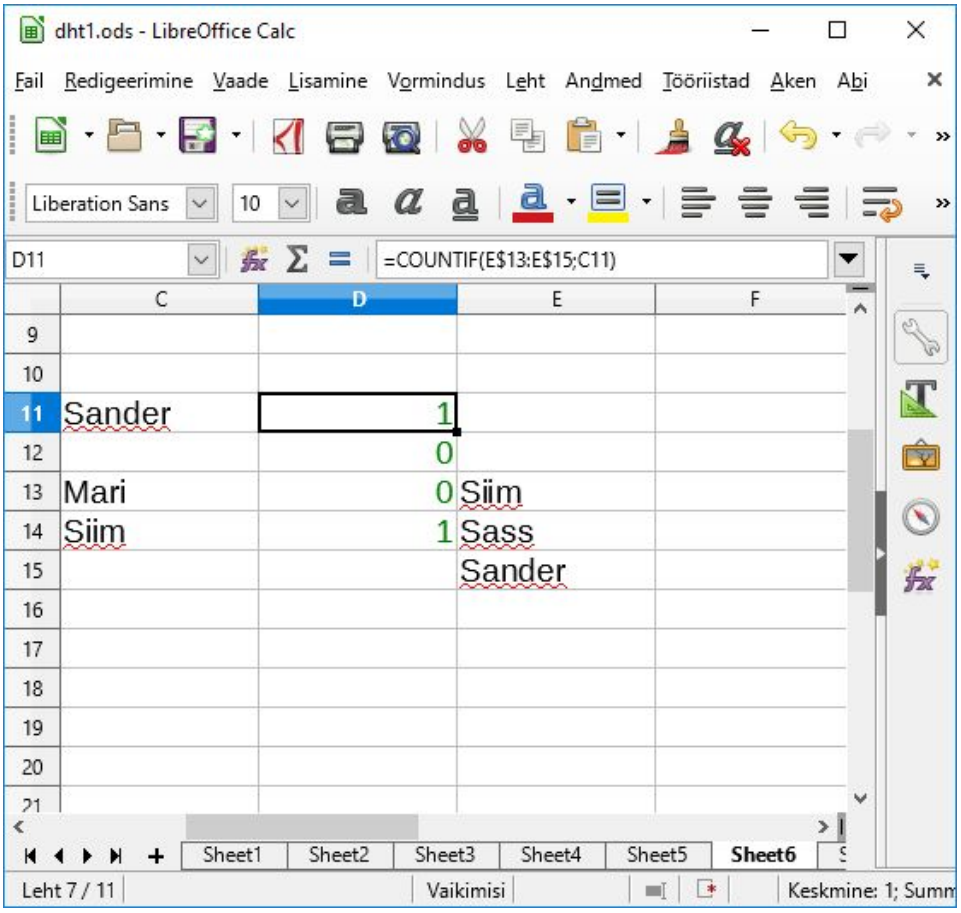


Nii õnnestus saada eraldi tabelilehed naise- ja mehenimede tarbeks.





Vaja veel kindlaks teha, et millisest soost otsitav nimi on. Enne pika loetelu juurde minekut katsetame väiksemaga. Lahtritesse E13 kuni E:15 panin kolm mehenime - Siim, Sass, Sander. Uuritava nime panen praegu lahtrisse E11 (Sander). Koostan funktsiooni =COUNTIF(E13:E15; C11), mis siis loeb kokku, et mitu korda tagumine element leidub esimeses loetelus. Kuna nimed ühekordselt, siis leidmise puhul väljastab 1, muidu 0. Et valemit saaks tulpa mööda alla paljundada, siis tulid meestenimede loetelu aadressil reanumbritele dollarimärgid ette - ehk siis E$13:E$15. Nii ongi nime kohta märge olemas, et kas võiks tegemist olla mehega.



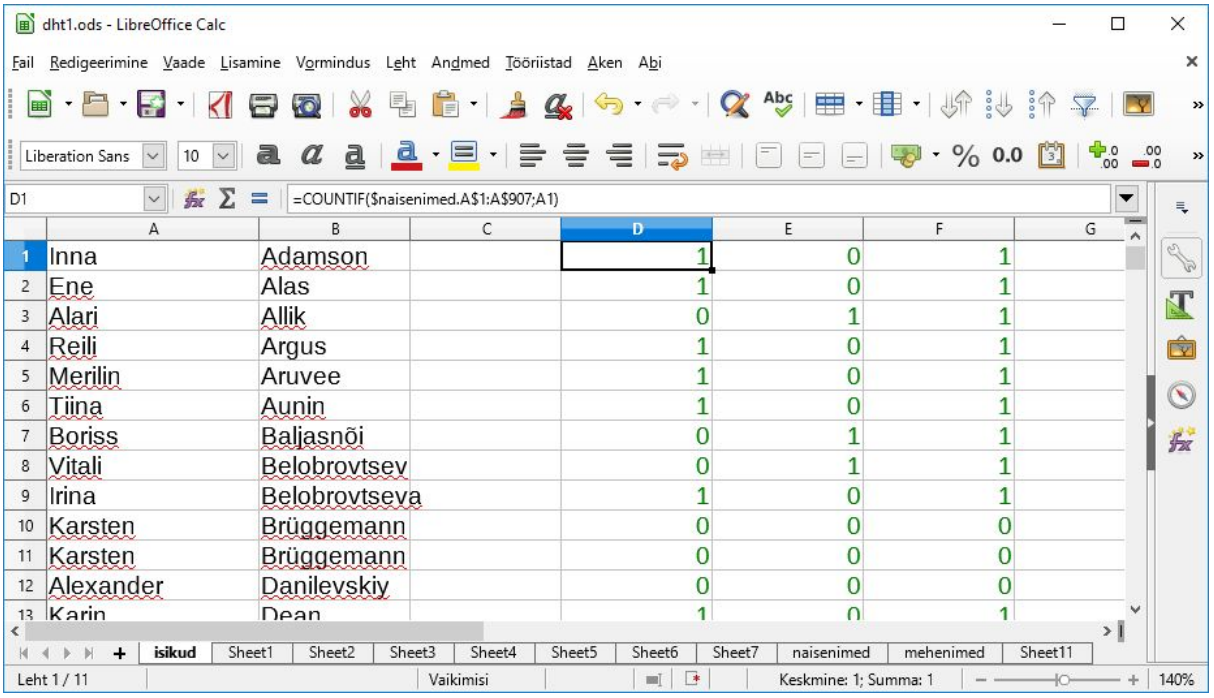
Edasi saab nimesid juba suuremast tabelist otsida. Naise- ja mehenimed kumbki eraldi lehele. Nimede järgi tuleb kolm eraldi arvutulpa. Esimene näitab, et kas vastav nimi leidub naistenimede tabelis, teine, et kas meestenimede tabelis ning kolmandas tulbas liidame kahe eelmise väärtused kokku, et kas nimi on nende andmete põhjal selgelt määratud.

=COUNTIF($naisenimed.A$1:A$907;A1)
=COUNTIF($mehenimed.A$1:A$1092 ;A1)

kui arvutusi tehakse Excelis, siis teise lehe poole pöördumine näeb välja
=COUNTIF(naisenimed!A$1:A$907;A1)

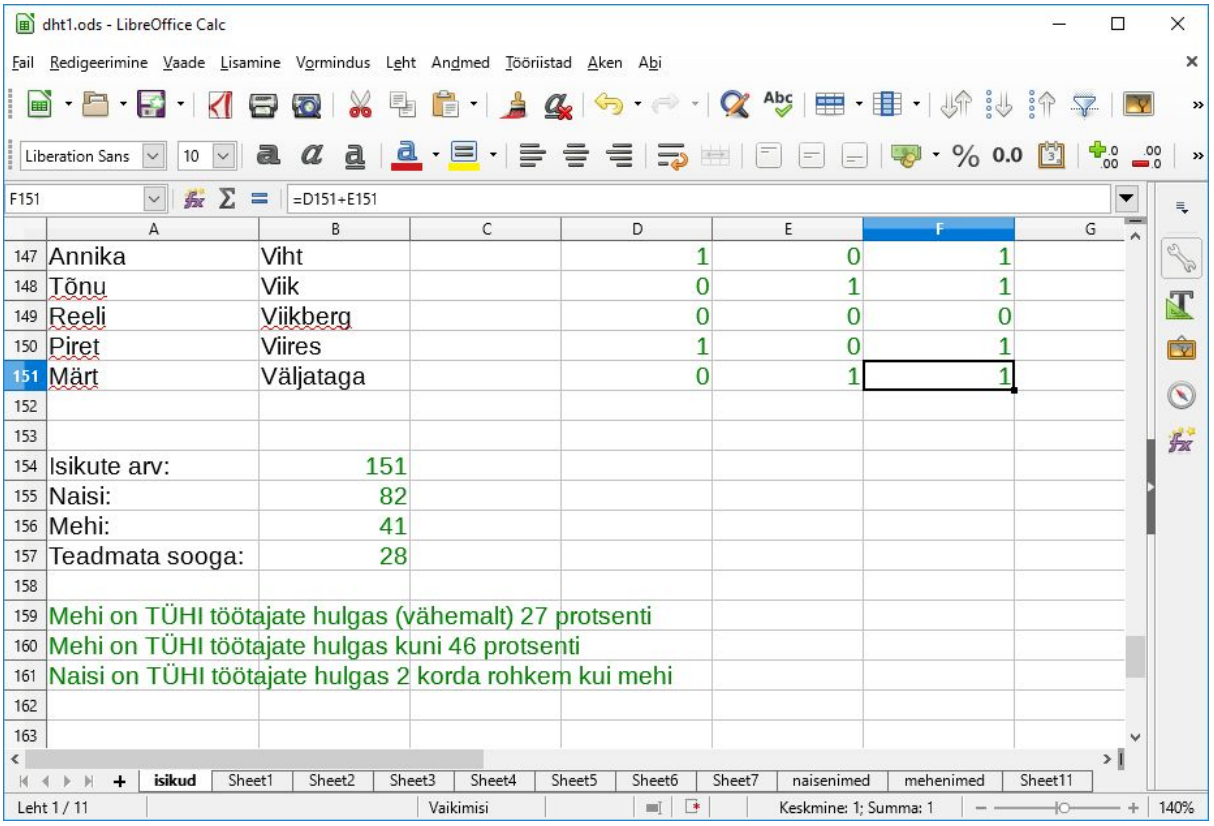
Kui kahe esimese arvu summaks on 1 (ehk siis on kas naine või mees), siis jah, muidu on midagi kahtlast.

=D1+E1



	A	B	C	D	E	F	G
1	Inna	Adamson		1	0	1	
2	Ene	Alas		1	0	1	
3	Alari	Allik		0	1	1	
4	Reili	Argus		1	0	1	
5	Merilin	Aruvee		1	0	1	
6	Tiina	Aunin		1	0	1	
7	Boriss	Baljasnõi		0	1	1	
8	Vitali	Belobrovtsev		0	1	1	
9	Irina	Belobrovtseva		1	0	1	
10	Karsten	Brüggemann		0	0	0	
11	Karsten	Brüggemann		0	0	0	
12	Alexander	Danilevskiy		0	0	0	
13	Karin	Dean		1	0	1	

Tabeli lõppu kannatab olemasoleva teabe põhjal kokkuvõtted koostada:



Isikute (isikutega ridade) arv
=ROWS(A1:A151)

Naiste tulba arvude summa:
=SUM(D1:D151)

Meeste tulba arvude summa
=SUM(E1:E151)

Nullide arv kolmandas tulbas
=COUNTIF(F1:F151; "=0")

Pikema jutu puhul läheb tekst jutumärkide vahele, &-märkide abil ühendatakse osad kokku, märkide ümber on vaja tühikud jätta. Meeste protsendi leidmiseks jagan meeste arvu (B156) isikute üldarvuga (B154).
="Mehi on TÜHI töötajate hulgas (vähemalt) " & ROUND(B156/B154*100) & " protsenti"

Kuna 28 inimese puhul polnud sugu teada, siis ei saa me välistada, et nad on mehed:
="Mehi on TÜHI töötajate hulgas kuni " & ROUND((B154-B155)/B154*100) & " protsenti"

Naiste puhul sarnane arvutus
="Naisi on TÜHI töötajate hulgas " & B155/ B156 & " korda rohkem kui mehi "

Ülesandeid

- Otsi veebileht isikute nimedega. Puhasta välja eesnimed. Märgi punktidena üles, mida ja kuidas tegid.
- Otsi naisenimede loetelu, puhasta nimed välja. Koosta vahend kontrollimaks, kas uuritav nimi leidub loetelus. Näita välja, millised esimeselt lehelt võetud nimed on naisenimed.
- Kuva mitu protsenti lehel olevatest nimedest on naisenimed, mitu protsenti mehenimed, mitu protsenti teadmata.

Arvude võrdlemine ja esitamine

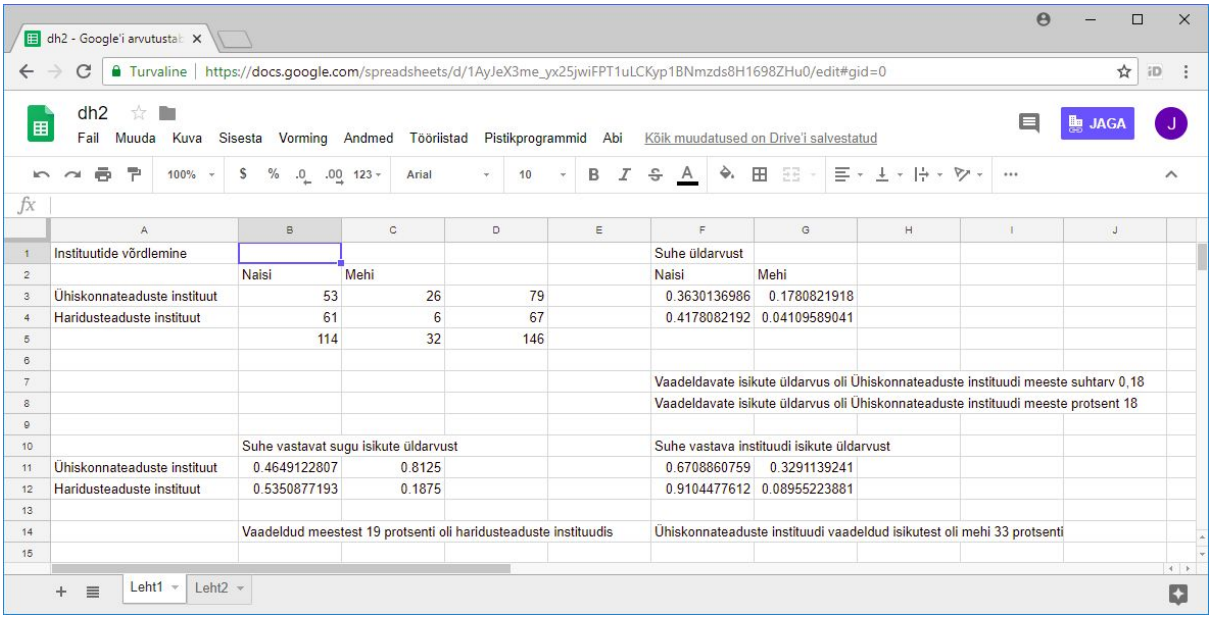
Samadest arvulistest andmetest saab mitmesuguseid näitajaid välja arvutada. Milliseid just, seda tuleb otsustada näitajate kasutusvajaduse järgi. Samas võib sõnastusi valides ja sobivaid kohti rõhutades jätta lugejale samadest andmetest märgatavalt erineva mulje.

Võrdlemisel on levinud andmestikuks 2 x 2 tabel. Praeguses näites siis TLÜ Ühiskonnateatuste instituudis ning Haridusteaduste instituudis töötavate naiste ja meeste arvud.

	Naisi	Mehi	
Ühiskonnateaduste instituut	53	26	79
Haridusteaduste instituut	61	6	67
	114	32	146

Ridu pidi kokku on kummaski instituudis töötavate inimeste arv, veerge pidi kokku naiste ja meeste arv ning all paremas nurgas kõigi isikute arv kahe instituudi peale kokku. Näites arvestame, et kõikide uuritavate inimeste sugu on teada.

Suhet saab arvutada nii üldarvu suhtes, vastavat sugu isikute üldarvu suhtes kui vastava instituudi isikute üldarvu suhtes. Samuti tuleb valida, kas tulemus esitada suhtarvuna või protsendina - viimasel juhul vajalik arv sajaga korrutada.



	A	B	C	D	E	F	G	H	I	J
1	Instituutide võrdlemine					Suhe üldarvust				
2		Naisi	Mehi			Naisi	Mehi			
3	Ühiskonnateaduste instituut	53	26	79		0.3630136986	0.1780821918			
4	Haridusteaduste instituut	61	6	67		0.4178082192	0.04109589041			
5		114	32	146						
6										
7						Vaadeldavate isikute üldarvus oli Ühiskonnateaduste instituudi meeste suhtarv 0,18					
8						Vaadeldavate isikute üldarvus oli Ühiskonnateaduste instituudi meeste protsent 18					
9										
10		Suhe vastavat sugu isikute üldarvust				Suhe vastava instituudi isikute üldarvust				
11	Ühiskonnateaduste instituut	0.4649122807	0.8125			0.6708860759	0.3291139241			
12	Haridusteaduste instituut	0.5350877193	0.1875			0.9104477612	0.08955223881			
13										
14		Vaadeldud meestest 19 protsenti oli haridusteaduste instituudis				Ühiskonnateaduste instituudi vaadeldud isikutest oli mehi 33 protsenti				
15										

Suhte üldarvu saan, kui jagan vastava arvu isikute kogusummaga. Näiteks Ühiskonnateaduste instituudi 26 meest moodustavad kahe instituudi uuritud isikute koguarvust 26/146, suhtarvu 0,178 ehk 0,18 ehk 18 protsenti
=C3/D5

Ühiskonnateaduste instituudi 26 meest moodustavad kokku 32st mehest 26/32 ehk 0,812 ehk 81 protsenti.
=C3/C$5

Ühiskonnateaduste instituudi 26 meest moodustavad selle inistituudi isikutest 26/79 ehk 33 protsenti
=C3/$D3

Veel suhteid

Suhe algväärtuste vahel
Kui tahan arvude erinevust rõhutada, siis saan näidata mitte väärtuse osakaalu summas, vaid väärtuste omavahelist suhet. Kui märgin, et Ühiskonnateaduste instituudis on naisi 67 protsenti ja mehi 33 protsenti, siis võetakse neid kergemini kui lihtsalt arve. Kui aga kirjutan, et Ühiskonnateaduste instituudis on naisi kaks korda rohkem kui mehi, siis jääb see mõnelegi ehk selgemalt silma. Ja kui sinna juurde märkida, et Haridusteaduste instituudis on naisi kümme korda rohkem kui mehi, siis see võib ehk ka lugeja silmi märgatavalt suurendada.

Suhe osakaalude vahel

Saab välja tuua osakaalude suhte - Ühiskonnateaduste instituudis on mehi 33 protsenti, Haridusteaduste instituudis 9 protsenti. Järelikult seal on Ühiskonnateaduste instituudis on meeste osakaal 3,7 korda suurem.

Suhe algväärtuste suhete vahel

Kõige reljeefsemalt rõhutab erinevusi algväärtuste vaheliste suhete omavaheline suhe - Ühiskonnateaduste instituudis on kaks naist ühe mehe kohta, Haridusteaduste instituudis kümme naist ühe mehe kohta - järelikult Haridusteaduste instituudis on naisi mehe kohta viis korda rohkem. Ehk kui eelnevalt arvasin, et kokkusaamisele tulev inimene on Ühiskonnateaduste instituudist ning tegelikult tuleb ta Haridusteaduste instituudist, siis tõenäosus, et ta on naine, on eelnevast viis korda suurem - ja juba enne oli kaks korda tõenäolisem, et tuleb naine kui et mees - muidugi juhul, kui kohtumisele tulek ei sõltu soost.

	Naisi	Mehi			Naisi mehe kohta	Sugude suhte suhe instituutide vahel
Ühiskonnateaduste instituut	53	26	79		2.038461538	4.987421384
Haridusteaduste instituut	61	6	67		10.16666667	
	114	32	146			
					Kui saan teada, et isik töötab Ühiskonnateaduste instituudi asemel Haridusteaduste instituudis,	
Ühiskonnateadlast haridusteadlase kohta	0.868852459	4.333333333			siis võin endisest 5 korda väiksema tõenäosusega eeldada, et ta on mees	
Tööinstituutide suhte suhe sugude vahel	4.987421384				Kui saan teada, et isik pole mitte naine, vaid on mees,	
					siis võin endisest 5 korda suurema tõenäosusega eeldada, et ta töötab Ühiskonnateaduste instituudis	

Ülesandeid

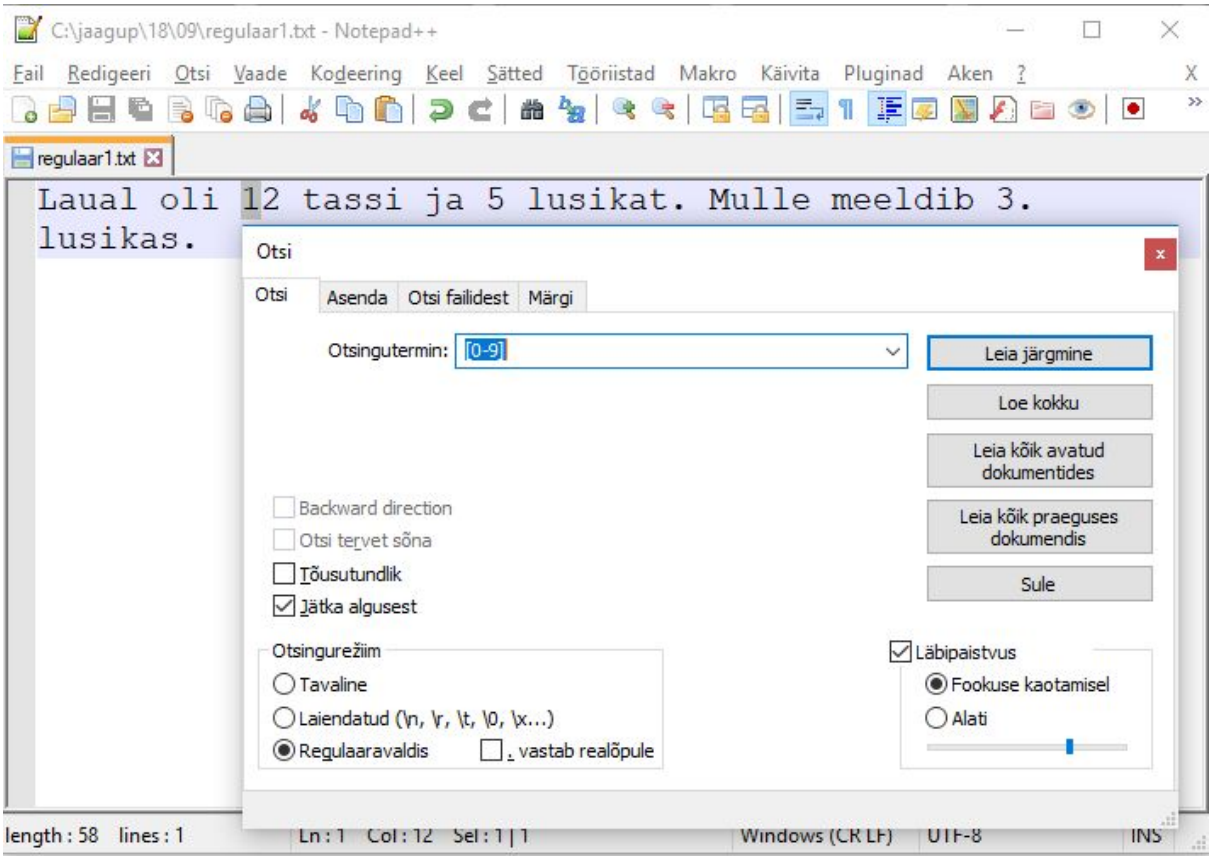
- Koosta 2x2 tabel veebist leitud andmete põhjal - näiteks uudisvoo kohta: kas autor on naine/mees? kas kommentaare on vähemalt 10?
- Arvuta rea- ja veerusummad ning üldarv, sõnasta mõne arvu kohta neist lause
- Leia osakaalud ridade kaupa, veergude kaupa ning üldarvu suhtes, sõnasta
- Leia kummagis suunas suhted algandmed, sõnasta
- Leia osakaalude suhted, sõnasta
- Leia algandmete suhete suhted, sõnasta, too lugejale selgitav näide.

Regulaaravaldised

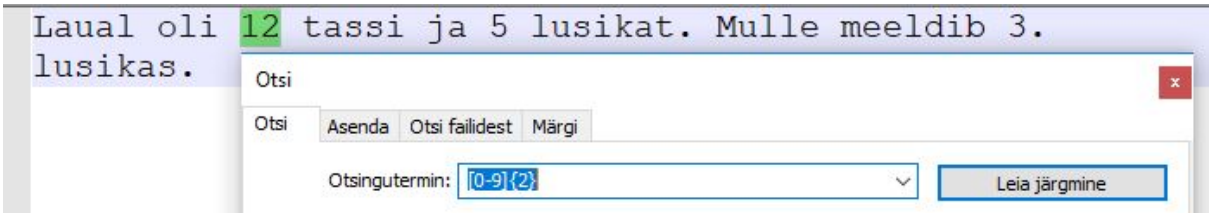
Tekstidest sobivate andmete kätte saamiseks aitavad lisaks “tavalisele” otsingukäsule regulaaravaldised. Ehk siis saab määrata, kas otsitakse numbrit sisaldavat sõna, viietähelist sõna, sõna, mille alguses on “a” ja lõpus “o”, või hoopis midagi keerulisemat. Vastavalt vahenditele võib regulaaravaldiste süntaks olla mõnevõrra erinev, kuid võimalused siiski suhteliselt sarnased. Praegused näited tehakse tekstiredaktoriga Notepad++, aga hiljem kasutatakse samu võimalusi näiteks Pythoni programmeerimiskeeles.

Otsing

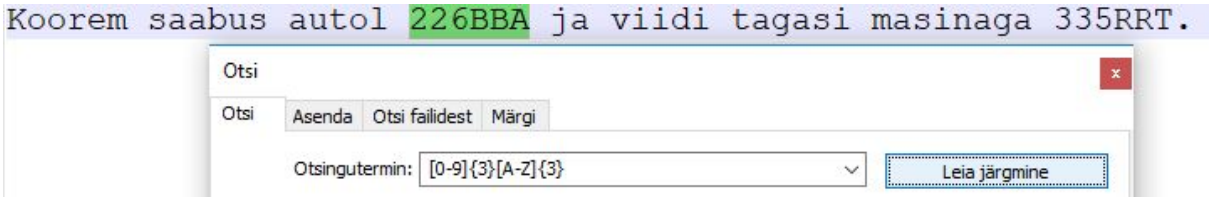
Katsetuseks loon lause mõnede arvudega. Otsinguaknast valin otsingutüübiks “Regulaaravaldis” ning otsinguterminiks [0-9], ehk siis kantsulgude vahel sümbolid vahemikus nullist üheksani. Vajutades järgmise leidmise nuppu, märgitakse lehel ära esimene number, praegusel juhul siis tekstis “12” sümbol “1”.



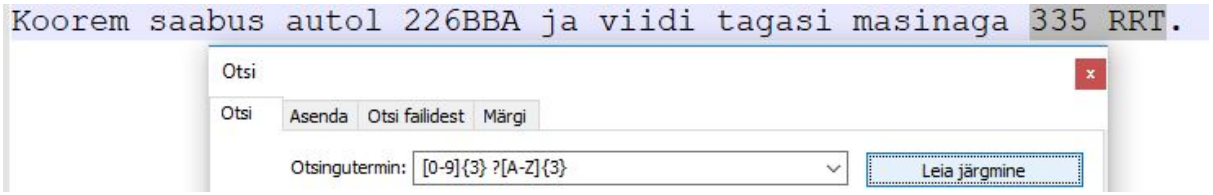
Kui märgin otsinguterminiks [0-9]{2}, siis see tähendab, et otsitakse kahte järjestikust numbrit. Praegusel juhul on tekstis selleks vaid arv 12.



Eesti autonumbri leidmiseks saab märkida, et järjest peavad olema kolm numbrit ja kolm tähte [0-9]{3}[A-Z]{3}



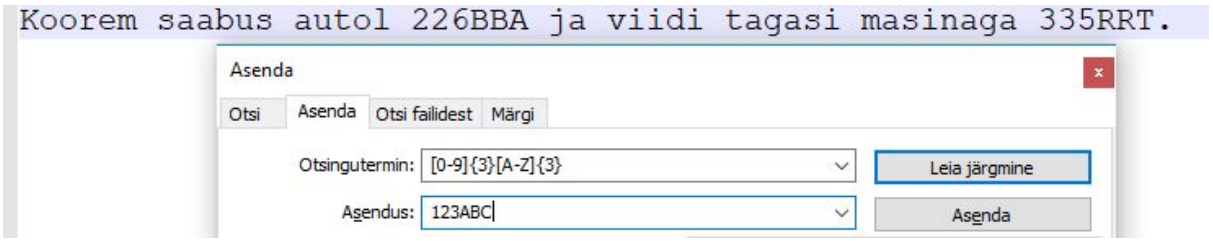
Kui luban, et numbrite ja tähtede vahel võib olla tühik, kuid ei pruugi, siis saab avaldise kirjutada kujul [0-9]{3} ?[A-Z]{3}, ehk siis keskel on tühik koos sellele järgneva küsimärgiga. Nii leitakse tekstis olevad mõlemad autonumbrid üles.



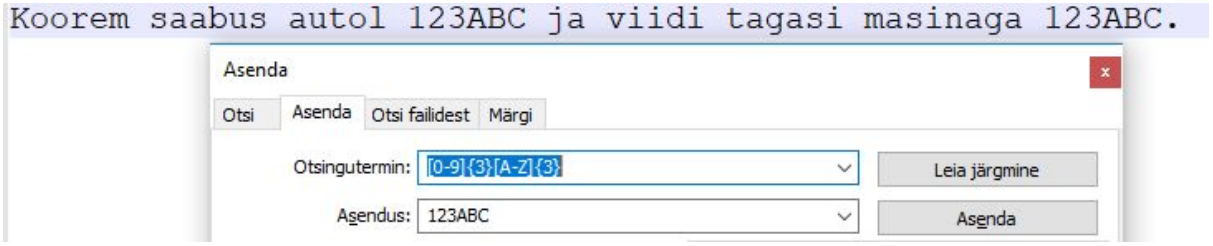
Kui küsimärgi asemel oleks pluss, siis oleks lubatud üks või rohkem eelnevat sümbolit (praegusel juhul tühikut). Erimärkidest veel: punkt tähistab ühte suvalist sümbolit. Kui aga otsitakse punkti ennast, siis peab tema ette panema langjoone \. Näiteks punktiga lõppevad sõnad saab kätte kujul [a-z]+\. Kui täpitähti ja numbreid ka arvestada, siis [a-zõäöü0-9]+\. ning suured tähed saab veel omakorda lisada. Või siis kahekohalise päeva ja kuu ning neljakohalise aastaga kuupäeva leiab kujul [0-9]{2}\.[0-9]{2}\.[0-9]{4}

Asendamine

Pikemate tekstide puhul tahetakse lisaks üles leidmisele nende kohtadega ka midagi peale hakata. Tekstiredaktoris on tavalisimaks operatsiooniks asendamine. Näiteks kui soovin kõik autonumbrid asendada anonüümsuse huvides kujule 123ABC, siis see täiesti õnnestub



Näha tulemus pärast asendust:



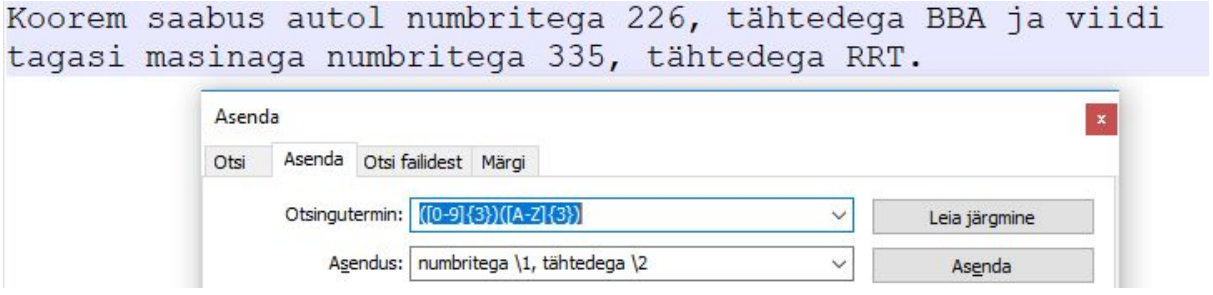
Asendus võib olla aga ka paindlikum. Kui tahan avaldise osi asenduse juures edaspidi kasutada, siis tuleb need paigutada sulgudesse. Nii on edaspidi võimalik järjekorranumbri järgi viidata sulgude sisule. Praegusel juhul siis saan autonumbri numbrite osale viidata kujul \1, tähtede osale kujul \2, sest nood olid vastavalt esimeste ja teiste sulgude sees. Lasen algse teksti läbi otsinguteminiga

([0-9]{3})([A-Z]{3})

ning asendusega

numbritega \1, tähtedega \2

Tulemuseks on



Ülesandeid

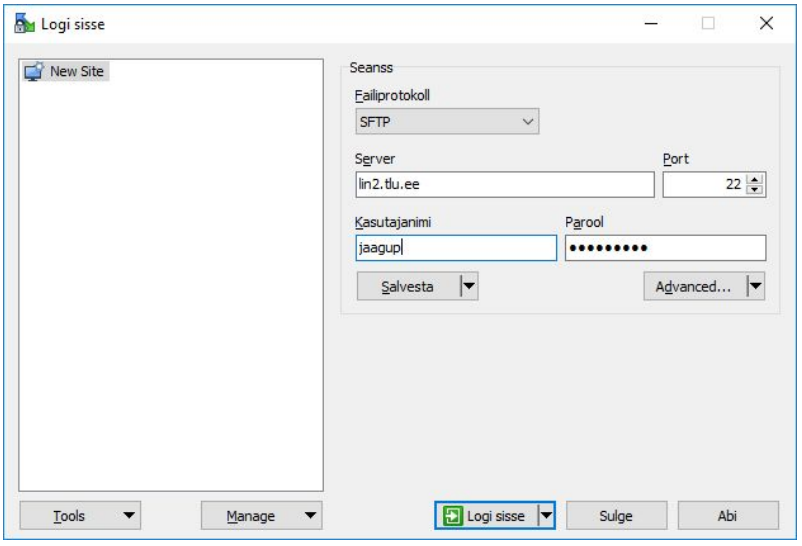
- Leia tekstis a-tähte sisaldavad sõnad
- Leia tekstis .ee-lõpulised veebiaadressid
- Leia tekstis kuupäevad kujul pp.kk.aaaa
- Asenda need kuupäevad kujule aaaa-kk-pp

Andmete avaldamine veebis

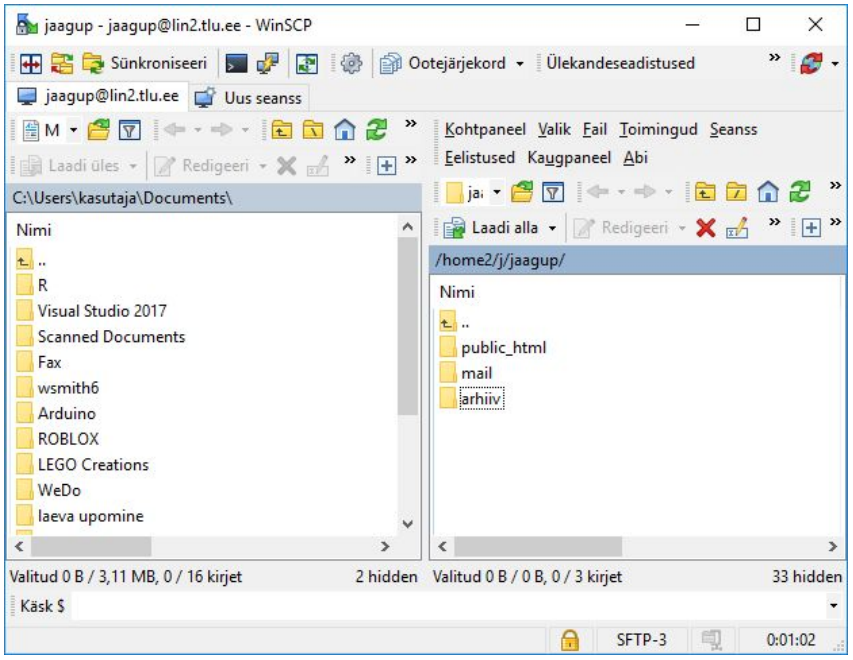
Teistele kättesaadavaks tegemiseks tuleb andmefail enamasti vastavasse kausta kopeerida. Windowsi masinate juures on aastaid mugavaks vahendiks olnud WinSCP. Kui vaja faile üles panna ja vaadata, siis enamasti tasub teenusepakkuja juurest uurida järgmised andmed:

kasutajanimi
parool
paigutuskataloog serveris
aadress veebis

Kusjuures kopeerimise sihtkoha serveri ning vaatamiseks veebis väljas oleva serveri aadress ei pruugi kattuda. Järgnevalt näide lihtsa teksti ülespaneku kohta Apache veebiserveris.

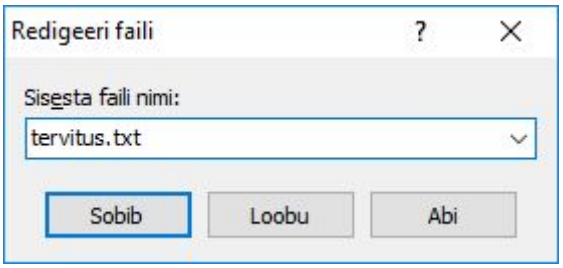
- 

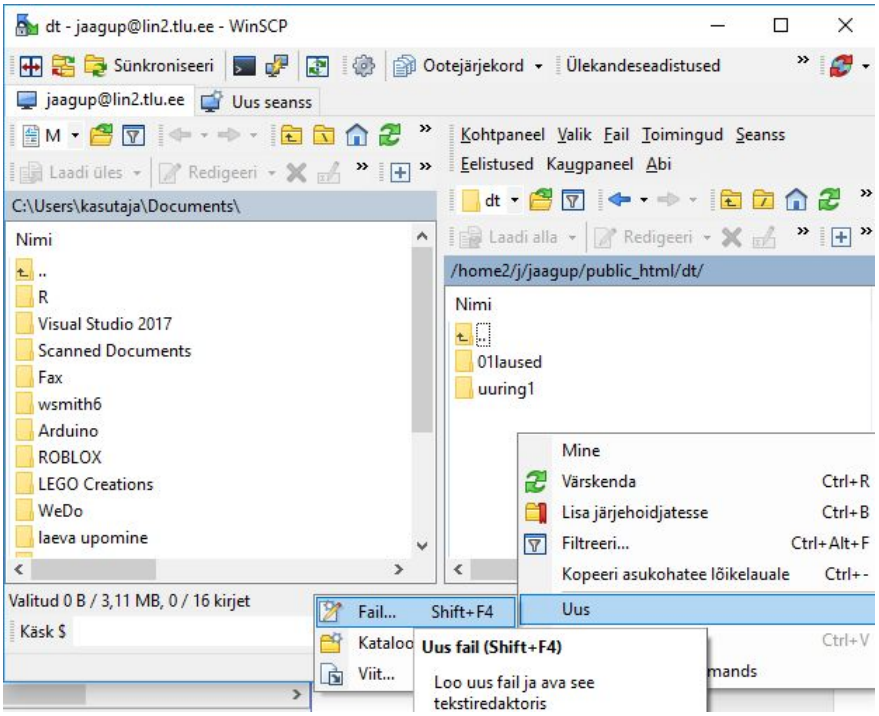
Sisselogimise järgselt ilmub aken, kus ühel pool näha kohalikud failid ja kataloogid ning teisel pool võrguserveri omad. Veebis nähtav kaust on sageli nimega public_html - aga see võib ka serveri seadetest sõltuda. Levinud on ka näiteks nimed htdocs ning www. Sinna kausta pandud failid võivad olla juba veebis nähtavad. Vastava kausta sisse on mõnigikord kasulik luua oma alamkaustade süsteem, et andmehulga sisse ära ei upuks.



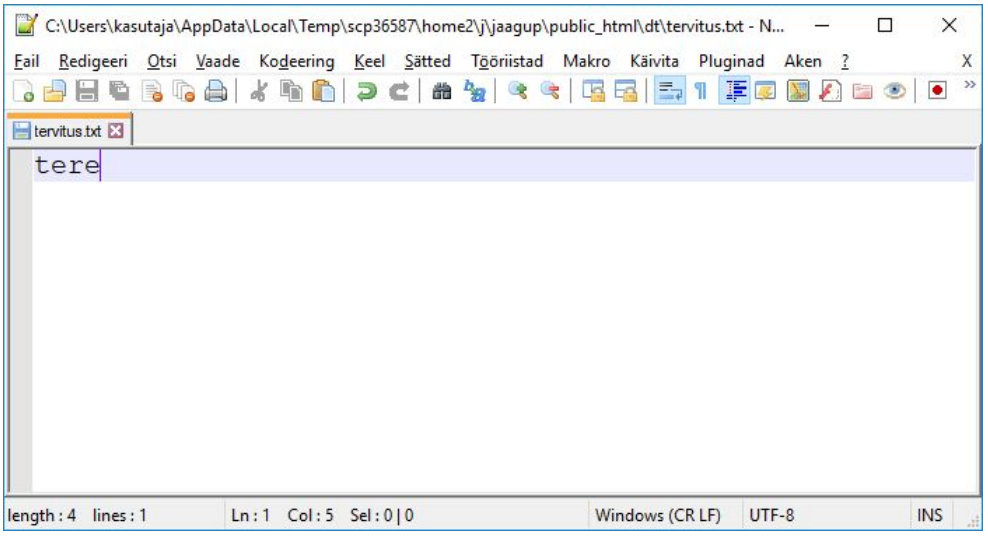
Siin näites teen public_html-i sisse digihumanitaaria tehnoloogiate tarbeks alamkausta dt.

Sinna sisse hiire parema klahviga vajutades uus fail, näiteks nimega tervitus.txt





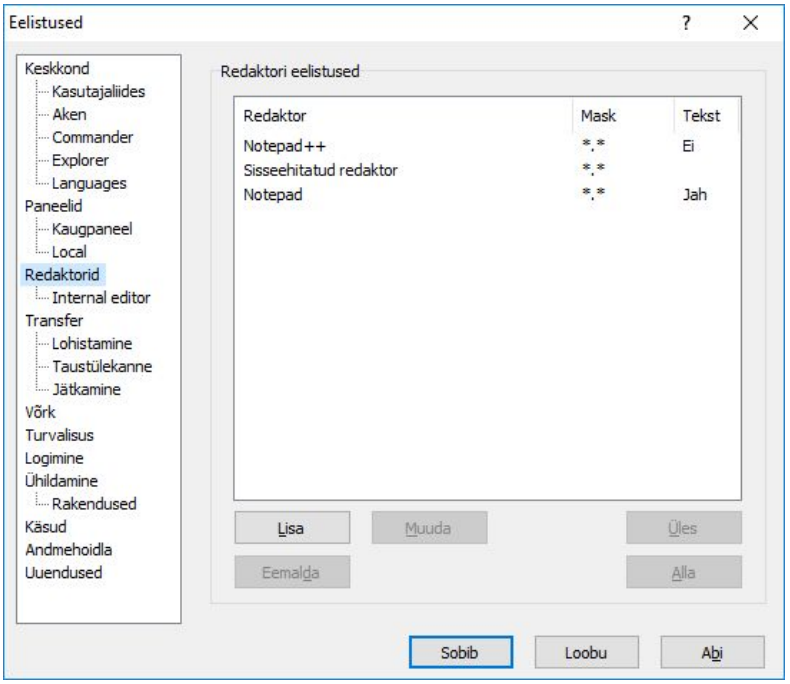
Faili sisse teade, mida soovida veebis vaadata



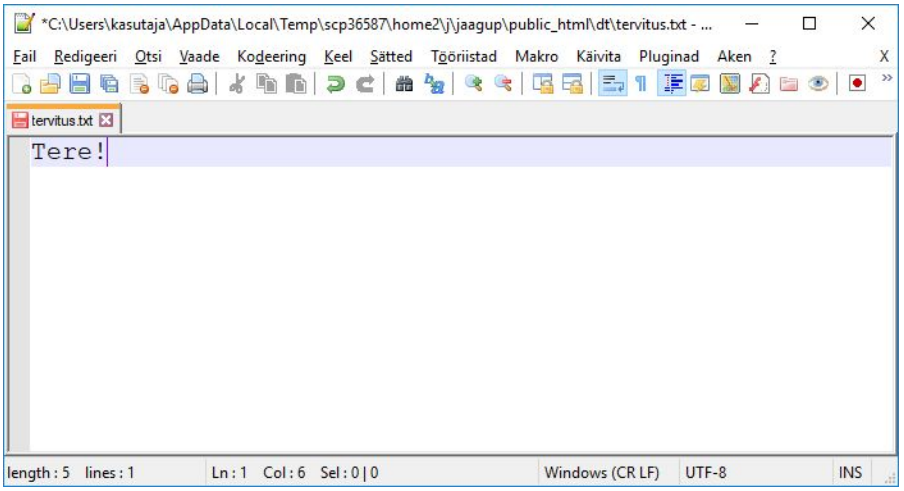
Tulemuse vaatamiseks on vaja teada, kust selle veebis leida võib. Tallinna Ülikoolis näiteks õppuritele avatud masina lin2.tlu.ee public_html-kaustas olevad kontod leiab vaatamiseks

hoopis aadressilt http://www.tlu.ee/~kasutajanimi/ alt - ehk siis praegune leht järgneval aadressil:

Pärast WinSCP paigaldamist arvaneb rakendusele sisse ehitatud redaktor. Mõne lause kirjutamiseks on see täiesti sobilik, pikemate tekstide kirjutamiseks ja treppimiseks aga sobib mõni andekam tekstiredaktor paremini. Eelistuste alt saab valida, et millise programmi abil teksti kirjutatakse. Siin näites valiti redaktorite alt uus redaktor, otsiti see Program Files kaustast üles Notepad++ nime alt ning nihutati esimeseks.



Kui tekst muuta ja salvestada, siis seda kohe veel uuena veebis ei näe



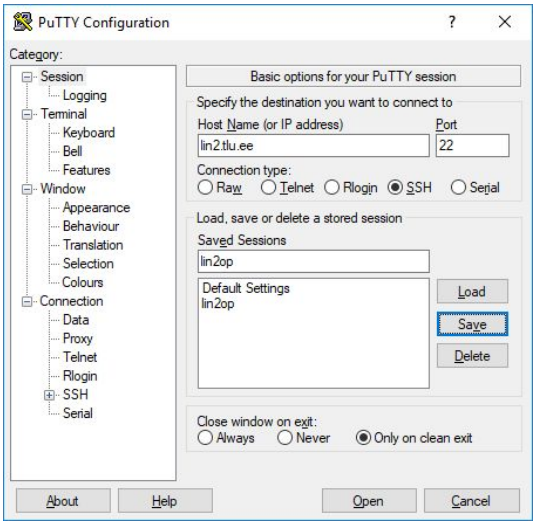
‘

Uuenenud teksti leiab alles pärast lehe uuesti laadimise nupu vajutamist

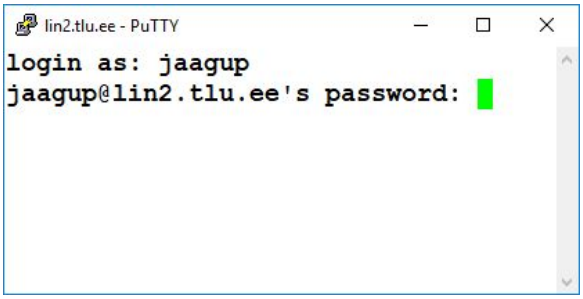
.

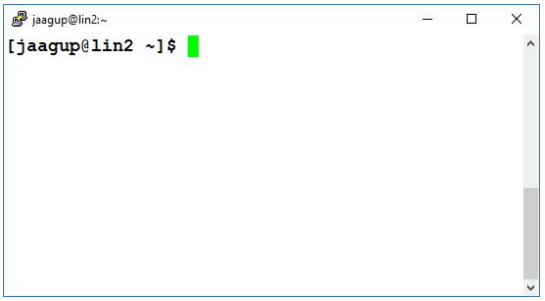
Putty klient

Üheks lihtsaks kliendiks Windowsi all on putty. Serverisse logides tuleb kõigepealt määrata selle nimi

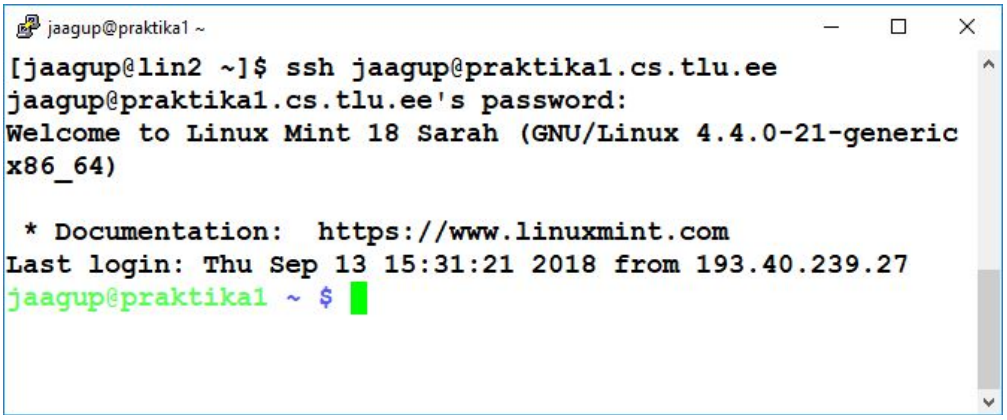


Edasi küsitakse kasutajanimi ja parool. Ei tasu ehmatada, et paroolitähed kaovad nagu musta auku - kui andmed õiged, siis pärast sisestusklahvi vajutamist lastakse edasi





Siinses näites on serveritega keerulisem. Välisvõrgust otse pääses ainult ülikooli üldserverisse (lin2.tlu.ee). Mitmedki vajalikud programmid on kättesaadavad aga ainult sisevõrgus olevas masinas praktika1.cs.tlu.ee - nii tuleb omakorda edasi liikuda ssh-käskluse abil ning ollaksegi siseserveri käsurea otsas.



Kokkupuuted Linuxi käskudega

pwd (print working directory) ehk asukoht kasutaja masinas - praegusel juhul /home/jaagup

```
jaagup@praktika1 ~ $ pwd
/home/jaagup
```

Kausta loomine siseveebis nähtavate failide jaoks

```
jaagup@praktika1 ~ $ mkdir public_html
```

Kausta sisse liikumiseks käsklus cd. Veebikausta alla eraldi kaustad aasta, kursuse ja tunni tarbeks, nii ei upu kohe andmetesse ära.

```
jaagup@praktika1 ~/public_html/2018 $ mkdir dt
jaagup@praktika1 ~/public_html/2018 $ cd dt
jaagup@praktika1 ~/public_html/2018/dt $ mkdir 04kasurida
jaagup@praktika1 ~/public_html/2018/dt $ cd 04kasurida/
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ pwd
/home/jaagup/public_html/2018/dt/04kasurida
```

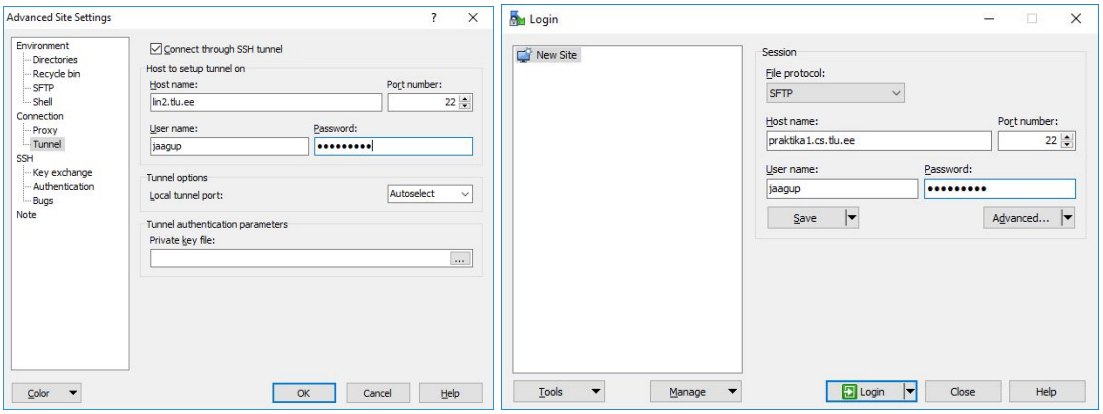
Kataloogipuus ülespoole liikumiseks sobib cd ..

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ cd ..
jaagup@praktika1 ~/public_html/2018/dt $ pwd
/home/jaagup/public_html/2018/dt
```

Pärast siis jälle tagasi

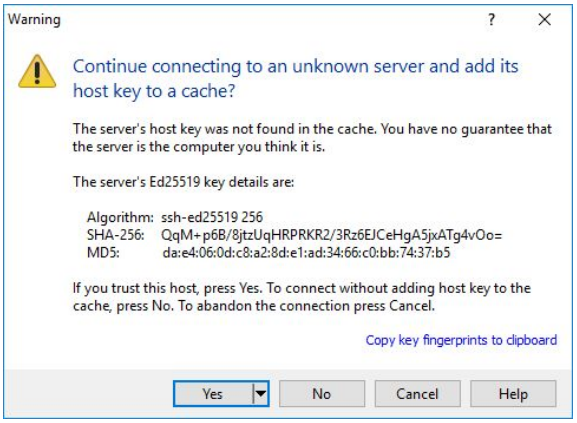
```
jaagup@praktika1 ~/public_html/2018/dt $ cd 04kasurida/
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $
```

Faile kannatab luua ka käsureal töötava redaktoriga. Mõnelegi aga on mugavam oma tuttavat kohalikku tekstiredaktorit kasutada ning lasta WinSCP-l või mõnel muul rakendusel faile vajadust mööda kopeerida. Et praktikaserver sisevõrgus, siis tuleb ka siin tunnel luua. Advanced -> Connection -> Tunnel. Linnuke sisse, et soovitakse seda pruukida ning praegusel juhul vahemasinaks lin2.tlu.ee oma kasutajanime ja parooliga



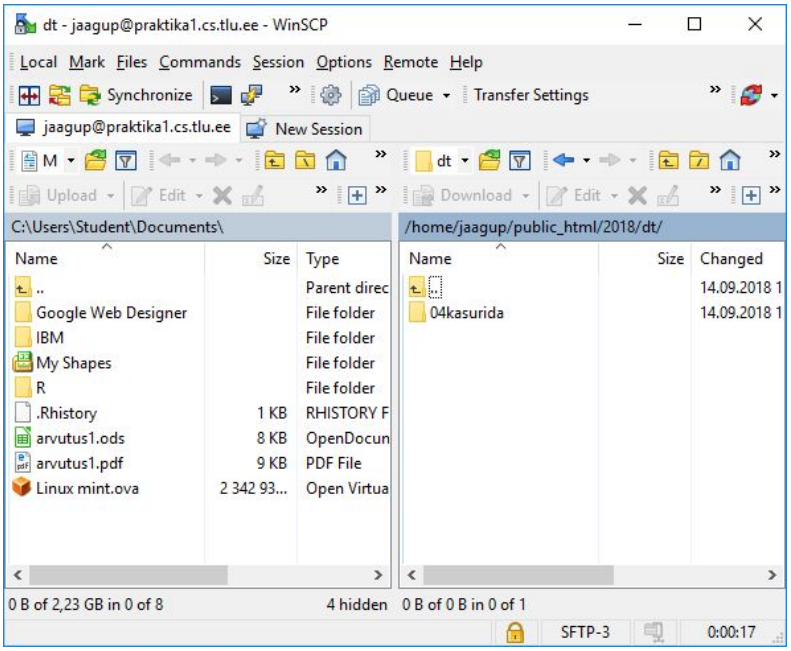
Kui tunnelimasin määratud, siis edasi vaja anda sihtmasina andmed

Esmakordsel ühendumisel võidakse küsida, et kas ikka usaldan pakutavat ühendust - või kahtlustan, et keegi parasjagu tahab pealt kuulata.



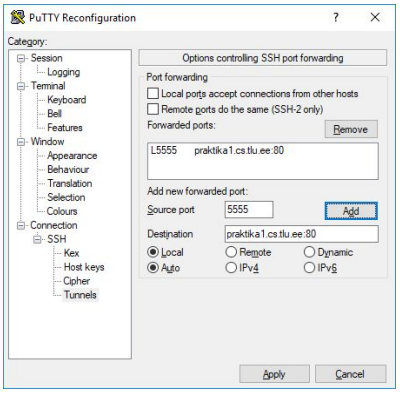
Tõsise kahtluse korral on võimalik serveriadministraatori juurde kontrollima minna, et kas näidatavad koodid ikka õiged on.

Pääsen ennist tehtud kataloogidesse ning võin asuda faile looma.

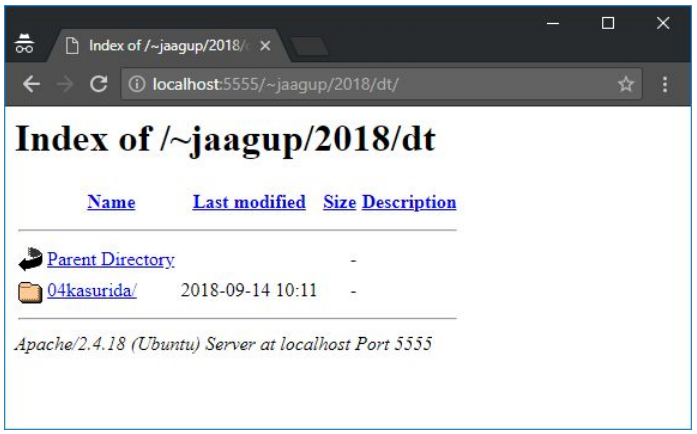


Sisevõrgu masina veebi vaatamine väljastpoolt

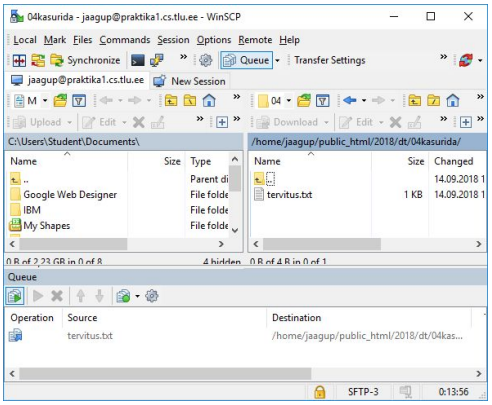
Ka sisevõrgus olevas veebiserveris paiknevate failide välisvõrgust vaatamiseks tuleb omaette tunnel luua. Selleks leiab putty vasakust ülanurgast seadete akna ning Connection -> SSH -> Tunnels kaudu saab määrata, et mida mille kaudu kuhu läbi lastakse.

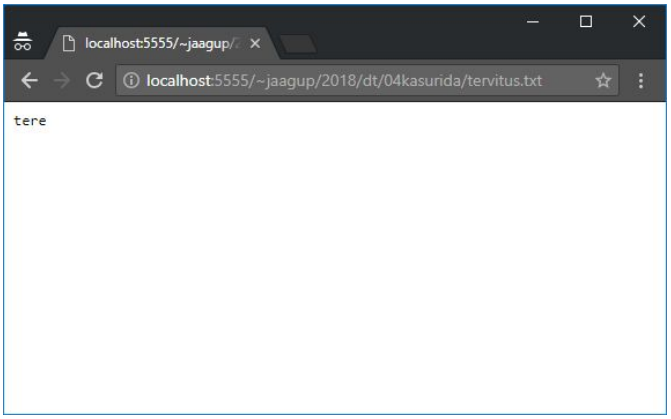


Siin siis suunatakse kohaliku masina (st. mille taga inimene istub) andmed pordist 5555 (laest võetud number, mille poole hiljem pöörduda) tulemüüri taga oleva praktika1.cs.tlu.ee-nimelise masina porti 80 - mille all jookseb sealne veebiserver. Et ise logitakse lin2.tlu.ee-masinasse, siis see on vaheseadmeks.
Brauseris avades tuleb aadressiks määrata localhost:5555 ning sellele järgnev veebiaadress sisevõrguserveris. Kui kõik õnneks läheb, võib kataloogi sisu vaadata.



Pärast sinna faili loomist ja salvestamist näeb brauseris selle sisu.





Shelli käsurida

Kataloogipuus sobivas kohas saab hakata käsureaga tegutsema. Olemasolevate failide loetelu nägemiseks sobib käsklus ls

ls

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ ls
tervitus.txt
```

Detailsema info kuvamiseks lisatakse võti -l (nagu long).

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ ls -l
total 4
-rw-rw-r-- 1 jaagup jaagup 4 Sep 14  2018 tervitus.txt
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $
```

Nii paistavad välja faili õigused, faili omanik ja grupi nimi kuhu fail kuulub. Faili suurus baitides (tere puhul iga tähe kohta bait) ning viimane muutmisaeg, nimi ka. Õiguste puhul r tähendab lugemisõigust (read) ning w kirjutusõigust (write). Esimene kolmik on omaniku enese kohta, viimane tähekolmik võõraste kasutajate (nt veebiserver) kohta. Ehk siis praegu kasutaja ise saab faili lugeda ja sinna muutusi salvestada, võõrad võivad ainult lugeda.

more

Faili sisu vaatamiseks sobib käsklus more

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more tervitus.txt
tere
```

Kui fail juhtub pikem olema, siis hoolitseb more, et korraga tuleks ette ainult ühe lehekülje jagu. Edasi tuleb juba järgmise vajutuse pärast - et ikka jõuaks vajaliku läbi lugeda.

echo

Millegi ekraanile trükkimiseks sobib echo.

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ echo "ahoi"
ahoi
```

Mitte, et see niisama kuigi kasulik oleks. Kui aga hiljem skriptijuppe kokku panna, siis küll. Ja praegu on ta abivahendiks näitamisel, kuidas käsust saabunud tekst on võimalik faili lõppu lisaks saata. Kaks suurem-kui märki enne failinime ning väljahõigatud tekst saadetaksegi tervitus.txt lõppu

Tulemuse saatmine faili

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ echo "ahoi" >> tervitus.txt
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more tervitus.txt
tereahoi
```

more abil nägi, et mis seal sees oli.
Teist korda veel lõppu lisades näeb uut ahoi-d järgmisel real, sest lisades lisandus ka reavahetus

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ echo "ahoi" >> tervitus.txt
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more tervitus.txt
tereahoi
ahoi
```

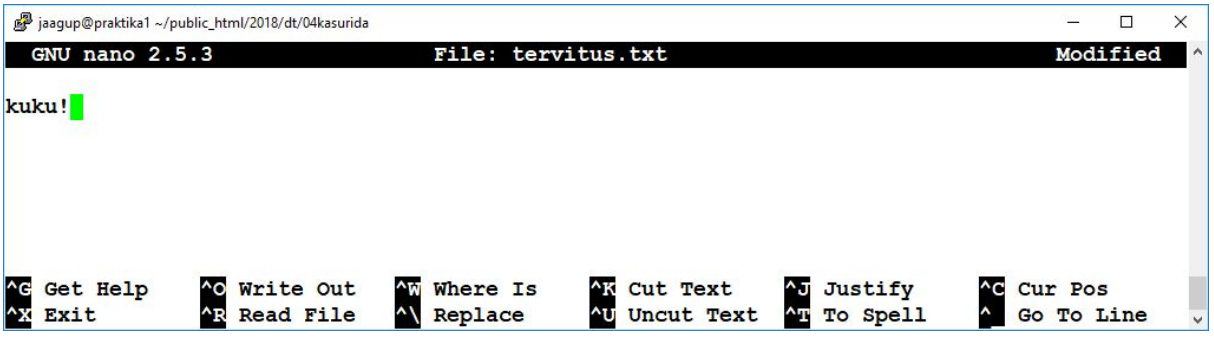
Kui fail luua või olemasolev üle kirjutada, siis läheb vaja üht >-märki.

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ echo "kuku" > tervitus.txt
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more tervitus.txt
kuku
```

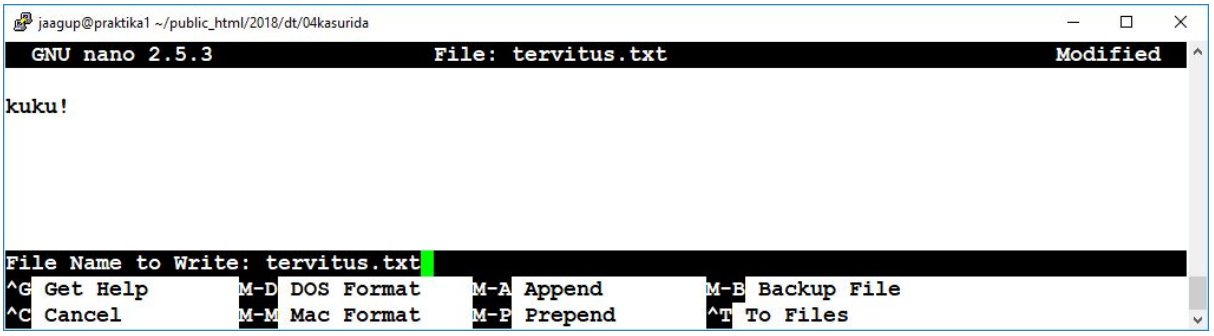
Terminaliaknas tekstide kirjutamiseks on hulk rakendusi olemas - peab ainult vaatama, et millise on haldur sinna lisanud. Üks lihtsamaid on pico

Tekstiredaktor pico

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ pico tervitus.txt
```



Nagu allolev õpetus näitab, siis salvestamiseks sobib CTRL+O, hilisemaks väljumiseks CTRL+X



Käsu more abil võib veenduda, et muutus jõudis kohale.

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more tervitus.txt
kuku!
```

Faili sisu hindamiseks sobib käsk wc (word count). Praegusel juhul siin üks rida, üks sõna ja kokku kuus baiti (hüüumärk ja reavahetus võtavad ka baidi enese alla).

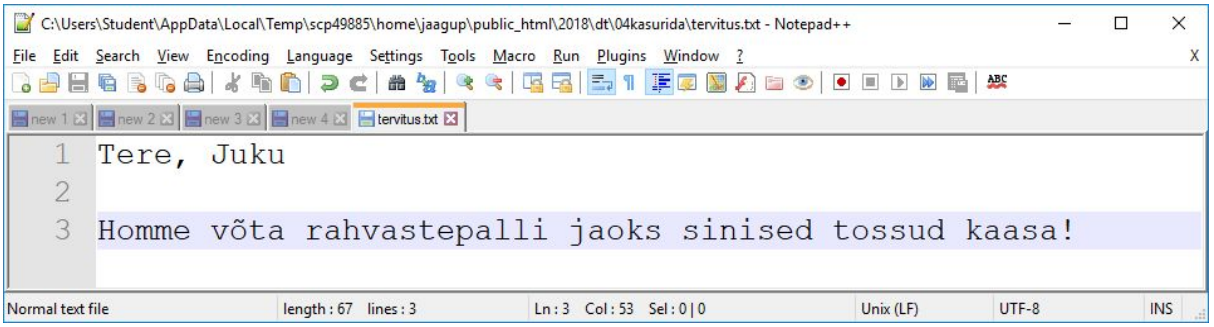
Loendamine - wc

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc tervitus.txt
1 1 6 tervitus.txt
```

Linux pakuba käskluste juurde abi - enamasti võtmega --help

wc puhul võib lugeda, et saab eraldi välja küsida baite ja tähti, samuti sõnade arve.

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc --help
Usage: wc [OPTION]... [FILE]...
  or:  wc [OPTION]... --files0-from=F
Print newline, word, and byte counts for each FILE, and a total line if
more than one FILE is specified.  A word is a non-zero-length sequence of
characters delimited by white space.

With no FILE, or when FILE is -, read standard input.

The options below may be used to select which counts are printed, always in
the following order: newline, word, character, byte, maximum line length.
  -c, --bytes            print the byte counts
  -m, --chars            print the character counts
  -l, --lines            print the newline counts
      --files0-from=F    read input from the files specified by
                           NUL-terminated names in file F;
                           If F is - then read names from standard input
  -L, --max-line-length  print the maximum display width
  -w, --words            print the word counts
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/wc>
or available locally via: info '(coreutils) wc invocation'
```



Uurimiseks veidi pikem fail

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc tervitus.txt
 2  9 67 tervitus.txt
```

Kui eraldi määran, et soovin näha nii tähti kui baite, siis selgub, et esimesi on 66, teisi aga 67 - õ võtab utf-8 kodeeringus kaks baiti oma alla.

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc -l -w --chars --bytes tervitus.txt
 2  9 66 67 tervitus.txt
```

Käsk wc suudab hakkama saada ka mitme failiga

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more tervitus2.txt
Tere, Mati

Võta pall ka!
```

Nii kuvatakse iga faili andmed eraldi ning kõik kokku ka.

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc *.txt
 2  5 28 tervitus2.txt
 2  9 67 tervitus.txt
 4 14 95 total
```

Faili loomiseks üks moodus on echo-käsuga kuvada tekst ning suunata see faili.

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ echo "algus kell 10" > teade.txt
```

Nii kataloogis näha, et üks fail juures ja mida sealt seest leida võib. 1 reavahetus, kolm sõna ja kokku 14 sümbolit

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc *.txt
  1   3  14 teade.txt
  2   5  28 tervitus2.txt
  2   9  67 tervitus.txt
  5  17 109 total
```

Metamärgi * abil näen kõiki faile, mis algavad “tervi”-ga ja lõppevad “.txt”-ga

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc tervi*.txt
```

```
 2  5 28 tervitus2.txt
 2  9 67 tervitus.txt
 4 14 95 total
```

Read faili algusest ja lõpust

Tervitusfaili sisu

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more tervitus.txt
Tere, Juku

Homme võta rahvastepalli jaoks sinised tossud kaasa!
```

Viimane rida failist

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ tail -n 1 tervitus.txt
Homme võta rahvastepalli jaoks sinised tossud kaasa!
```

Viimane rida wc-käsu väljundist.

```
/jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ wc *.txt | tail -n 1
  5  17 109 total
```

Mitu käsku üheskoos

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt
Juku
Kati
Anu
Madis
Mati
```

järjestatuna

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt | sort
Anu
Juku
Kati
Madis
Mati
```

Kahanevalt järjestatuna

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt | sort -r
Mati
Madis
Kati
Juku
Anu
```

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt
Juku
Kati
Anu
Madis
Mati
```

Kaks viimast nime

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ tail -n 2 eesnimed.txt
Madis
Mati
```

Alates teisest nimest

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ tail -n +2 eesnimed.txt
Kati
Anu
Madis
Mati
```

Faili algusots (ehk praegu kogu fail)

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ head eesnimed.txt
Juku
Kati
Anu
Madis
Mati
```

Kaks esimest

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ head -n 2 eesnimed.txt
Juku
Kati
```

Ilma kahe viimaseta

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ head -n -2 eesnimed.txt
Juku
Kati
Anu
```

Harjutus

Sortige eesnimed ning salvestage kaks esimest neist eraldi faili

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt | sort | head -n 2 >
vastus.txt
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more vastus.txt
Anu
Juku
```

grep

Tähte a sisaldavad nimed

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt | grep a
Kati
Madis
Mati
```

at- sisaldavad read

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt | grep at
Kati
Mati
```

tähesuurust arvestamata (ignore case)

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ more eesnimed.txt | grep -i a
Kati
Anu
Madis
Mati
```

ühest failist

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ grep a eesnimed.txt
Kati
Madis
Mati
```

mitmest failist

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ grep at *.txt
eesnimed.txt:Kati
eesnimed.txt:Mati
tervitus2.txt:Tere, Mati
```

regulaaravaldis otsingus

```
jaagup@praktika1 ~/public_html/2018/dt/04kasurida $ grep --basic-regexp "[A-Z][a-z]" *.txt
eesnimed.txt:Juku
eesnimed.txt:Kati
eesnimed.txt:Anu
eesnimed.txt:Madis
eesnimed.txt:Mati
tervitus2.txt:Tere, Mati
tervitus2.txt:Võta pall ka!
tervitus.txt:Tere, Juku
tervitus.txt:Homme võta rahvastepalli jaoks sinised tossud kaasa!
vastus.txt:Anu
vastus.txt:Juku
```

Skriptid, sortimine

Kui käsud muutuvad pikemaks, siis pole neid mugav sageli uuesti sisestada ning vigu kipub ka sealjuures tekkima. Lahenduseks on vajalik käsk või käskude rida faili kirja panna ning vajalikul ajal välja kutsuda. Siin tervituskäsklus failis kirjas ning tööle lükkamisel näeb selle tulemust.

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more tervitus.sh
echo "tere"
jaagup@praktika1 ~/public_html/2018/dt/05skript $ sh tervitus.sh
tere
```

Järgnevalt majandame laste pikkuste ja masside andmetega. Käsud cat ning more kuvavad mõlemad andmed ekraanile, more ootab iga ekraanitäie järel klahvivajutust, cat-i abil saab nad aga korraga kätte.

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ cat viiesklass.txt
eesnimi,pikkus,mass,sugu
Juku,170,45,m
Kati,160,35,n
Mati,160,72,m
Madis,165,53,m
Mati,163,60,m
Katrin,165,43,n
Siim,151,38,m
Martin,159,46,m
Kadri,164,57,n
Katariina,148,35,n
Maria,143,38,n
Marta,169,550,n
Madis,156,65,m
Mihkel,165,69,m
Tiina,170,38,n
Miia,145,68,n
Siim,151,55,m
Juhan,175,110,m
```

```
Priit,156,63,m
Kristjan,164,59,m
Kristi,155,53,n
Kristiina,158,62,n
Killu,164,49,n
Mart,170,69,m
Kert,143,36,m
Gert,152,67,m
Lauri,156,53,m
Moonika,164,58,n
Jaanika,165,59,n
Jaanus,164,63,m
Jaan,162,65,m
```

Sort-käsuga pannakse read tähestiku järjekorda

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ sort viiesklass.txt
eesnimi,pikkus,mass,sugu
Gert,152,67,m
Jaan,162,65,m
Jaanika,165,59,n
Jaanus,164,63,m
Juhan,175,110,m
Juku,170,45,m
```

....

Sama tulemuse saab, kui väljakuvatud andmed torukäsu abil järjekorda sättida

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more viiesklass.txt | sort
eesnimi,pikkus,mass,sugu
Gert,152,67,m
Jaan,162,65,m
Jaanika,165,59,n
Jaanus,164,63,m
Juhan,175,110,m
Juku,170,45,m
Kadri,164,57,n
```

....

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ sort < viiesklass.txt
eesnimi,pikkus,mass,sugu
Gert,152,67,m
Jaan,162,65,m
Jaanika,165,59,n
Jaanus,164,63,m
Juhan,175,110,m
Juku,170,45,m
Kadri,164,57,n
```

Kahanevasse järjestusse paigutamiseks aitab võti -r (reverse)

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more viiesklass.txt | sort -r
Tiina,170,38,n
Siim,151,55,m
Siim,151,38,m
Priit,156,63,m
Moonika,164,58,n
...
Jaanika,165,59,n
Jaan,162,65,m
Gert,152,67,m
eesnimi,pikkus,mass,sugu
```

Siin aga näha, et ka tulpade pealkirjad läksid nõnda viimasele kohale - väikesed tähed pandi omaette. Pealkirjarea ette jätmiseks sobib see eraldi välja küsida, tagumine ots ära järjestada ning siis jälle mõlemad osad kokku panna.

Esimene rida failist

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ head -n 1 viiesklass.txt
eesnimi,pikkus,mass,sugu
```

Read alates teisest

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ tail -n +2 viiesklass.txt
Juku,170,45,m
Kati,160,35,n
Mati,160,72,m
Madis,165,53,m
Mati,163,60,m
Killu,164,49,n
...

Mart,170,69,m
Kert,143,36,m
Gert,152,67,m
Lauri,156,53,m
Moonika,164,58,n
Jaanika,165,59,n
Jaanus,164,63,m
Jaan,162,65,m
```

Skriptina kokku pandud - vastusefaili kõigepealt esimene rida ning siis ülejäänud järjestatuna sinna uue faili lõppu juurde.

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more nimesort1.sh
more viiesklass.txt | head -n 1 > vastus.txt
more viiesklass.txt | tail -n +2 | sort >> vastus.txt
```

Tulemusena näeb faili jõudnud andmeid

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ cat vastus.txt
```

```
eesnimi,pikkus,mass,sugu
Gert,152,67,m
Jaan,162,65,m
Jaanika,165,59,n
Jaanus,164,63,m
Juhan,175,110,m
Juku,170,45,m
Kadri,164,57,n
Katariina,148,35,n
Kati,160,35,n
Katrin,165,43,n
Kert,143,36,m
Killu,164,49,n
Kristi,155,53,n
Kristiina,158,62,n
Kristjan,164,59,m
Lauri,156,53,m
Madis,156,65,m
Madis,165,53,m
Maria,143,38,n
Mart,170,69,m
Marta,169,550,n
Martin,159,46,m
Mati,160,72,m
Mati,163,60,m
Mihkel,165,69,m
Miia,145,68,n
Moonika,164,58,n
Priit,156,63,m
Siim,151,38,m
Siim,151,55,m
Tiina,170,38,n
```

Selgitused saab ka skripti abil vastusefaili lisada, siis pärast selgem lugeda, et millega tegu

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more nimesort2.sh
echo "Nimede järjestamise tulemused" > vastus.txt
echo "" >> vastus.txt
more viiesklass.txt | head -n 1 >> vastus.txt
more viiesklass.txt | tail -n +2 | sort >> vastus.txt
```

jaagup@praktika1 ~/public_html/2018/dt/05skript $ more vastus.txt

Nimede järjestamise tulemused

```
eesnimi,pikkus,mass,sugu
Gert,152,67,m
Jaan,162,65,m
Jaanika,165,59,n
Jaanus,164,63,m
```

Käsklusele sort võib öelda ka tulba, mitmenda järgi soovitakse sortida. Tulpadesse jagamisel kasulik määrata tulpade eraldaja, praegusel juhul koma. Ning kolmanda tulba järgi

sortmiseks sobib võti -k3,3 (kolmandast alates ja kuni kolmandani). Lõpus olev -n (numeric) tähendab, et järjestatakse arvulise väärtuse järgi, ehk siis 95 < 125, tähestiku järgi vaadataks eelkõige esimest sümbolit

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more viiesklass.txt | sort
--field-separator="," -k3,3 -n

eesnimi,pikkus,mass,sugu
Katariina,148,35,n
Kati,160,35,n
Kert,143,36,m
Maria,143,38,n
Siim,151,38,m
Tiina,170,38,n
Katrin,165,43,n
Juku,170,45,m
Martin,159,46,m
...
```

Harjutus

Koostage skript, mille abil kuvatakse klassi viis kergemat ja viis raskemat õpilast, lisage tekstina selgitused juurde

Lahendus

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more nimesort3.sh
echo "Klassi viis kergemat" > vastus.txt
echo "" >> vastus.txt
more viiesklass.txt | sort --field-separator="," -k 3,3 -n | head -n 6 >> vastus.txt

echo "" >> vastus.txt
echo "Klassi viis raskemat" >> vastus.txt
echo "" >> vastus.txt
more viiesklass.txt | head -n 1 >> vastus.txt
more viiesklass.txt | tail -n +2 | sort --field-separator="," -k3,3 -n | tail -n 5 >>
vastus.txt
```

```
jaagup@praktika1 ~/public_html/2018/dt/05skript $ more vastus.txt
Klassi viis kergemat

eesnimi,pikkus,mass,sugu
Katariina,148,35,n
Kati,160,35,n
Kert,143,36,m
Maria,143,38,n
Siim,151,38,m
```

Klassi viis raskemat

```
eesnimi,pikkus,mass,sugu
Mart,170,69,m
Mihkel,165,69,m
Mati,160,72,m
```

```
Juhan,175,110,m
Marta,169,550,n
```

Failide kopeerimine

Graafiliselt failide ümber tõstmine tänapäeval ehk levinum. Kui aga neid vaja eraldiseisvas serveris ühe koha pealt hulgem teise paigutada, siis võib käsureavahenditega kiirem või mugavam olla. Samuti kuluvad kopeerimiskäsud ära sobivaid skripte kokku pannes. Järgnev näide tuleb uue töökataloogi ettevalmistamise kokku.

Käsk pwd (print working directory) näitab kausta, kus parajasti ollakse. Siinsete käsureasätete juures osa kataloogist kodukaustast(~) alates näha, mõnel pool aga pwd-käsklus ainsaks mooduseks teada saada, kus jooksev kataloog parajasti on.

```
jaagup@praktika1 ~/public_html/2018/dt $ pwd
/home/jaagup/public_html/2018/dt
Kataloogis näha kaks alamkataloogi


jaagup@praktika1 ~/public_html/2018/dt $ ls
04kasurida  05skript
```

Alamkataloogi sisu:

```
jaagup@praktika1 ~/public_html/2018/dt $ ls 05skript/
nimesort1.sh  nimesort3.sh  vastus.txt   viiesklass.txt
nimesort2.sh  tervitus.sh   vastus.txt?
```

Esialgu huvitab meid edasiseks töötluseks sealt fail viiesklass.txt

Ise dt-kataloogis olles loon uue alamkataloogi nimega 06skript

```
jaagup@praktika1 ~/public_html/2018/dt $ mkdir 06skript
```

ja veendun selle olemasolus

```
jaagup@praktika1 ~/public_html/2018/dt $ ls
04kasurida  05skript 06skript
```

Kopeerin andmefaili uude kataloogi

```
jaagup@praktika1 ~/public_html/2018/dt $ cp 05skript/viiesklass.txt 06skript/
```

ja veendun, et see sinna ka kohale jõudis

```
jaagup@praktika1 ~/public_html/2018/dt $ ls 06skript/
viiesklass.txt
```

Lähen ise ka vastavasse kausta

```
jaagup@praktika1 ~/public_html/2018/dt $ cd 06skript/
jaagup@praktika1 ~/public_html/2018/dt/06skript $
```

cut - tulpade eraldamine

Käsu võimaluste kohta leiab lähemad seletused näiteks Wikipediast

https://en.wikipedia.org/wiki/Cut_(Unix)

Siin mõned katsed käsuga. Kõigepealt meenutame viienda klassi laste andmefaili sisu

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ head viiesklass.txt
eesnimi,pikkus,mass,sugu
Juku,170,45,m
Kati,160,35,n
Mati,160,72,m
Madis,165,53,m
Mati,163,60,m
Katrin,165,43,n
Siim,151,38,m
Martin,159,46,m
Kadri,164,57,n
```

Esimese tulba andmete nägemiseks määran eraldajaks koma (-d “,”) ning teatan, et soovin näha esimest tulpa (-f 1)

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ cut -d "," -f 1 < viiesklass.txt
eesnimi
Juku
Kati
Mati
Madis
Mati
Katrin
Siim
```

Esimese ja kolmanda tulba nägemiseks lihtsalt need tulpade numbrid loetellu

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ cut -d "," -f 1,3 < viiesklass.txt
eesnimi,mass
Juku,45
Kati,35
Mati,72
Madis,53
Mati,60
Katrin,43
Siim,38
```

Võib küsida andmeid ka tähe kaupa - praegul siis esitähed igast reast

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ cut -c 1 < viiesklass.txt
e
J
```

```
K
M
M
M
K
S
M
```

paste - tekstide ühendamine

Eraldi faili massid

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ cut -d "," -f 3 < viiesklass.txt >
massid.txt
```

ning esitähed

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ cut -c 1 < viiesklass.txt > esitahed.txt
```

Kontroll, et mis faili jõudis

```
more massid.txt
mass
45
35
72
53
60
43
38
46
57

jaagup@praktika1 ~/public_html/2018/dt/06skript $ more esitahed.txt
e
J
K
M
M
M
K
S
M
```

paste-abil pannakse failide andmed rida-realt kõrvuti

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ paste esitahed.txt massid.txt
e       mass
J       45
K       35
M       72
M       53
M       60
K       43
```

```
S       38
M       46
K       57
```

Võtmega (-d “-”) saab tulpade eraldajaks miinusmärk

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ paste -d "-" esitahed.txt massid.txt
e-mass
J-45
K-35
M-72
M-53
M-60
K-43
S-38
M-46
```

Soovides ühe faili ridade andmed ühte ritta kokku, aitab võti (-s).

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ paste -s -d "," esitahed.txt
e,J,K,M,M,M,K,S,M,K,K,M,M,M,M,T,M,S,J,P,K,K,K,K,M,K,G,L,M,J,J,J
```

Tulba arvude summa

Kõigepealt kolmanda tulba andmed

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ cut -d "," -f 3 < viiesklass.txt
mass
45
35
72
53
60
43
38
```

Samuti need olemas juba eraldi masside failis - tail-käsuga hoolitsen, et pealkirjarida ei jääks arvude sisse

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ tail -n +2 < massid.txt
45
35
72
53
60
43
38
46
57
35
```

Arvutuskäsuks sobib bc - ehk siis eelmise käsu väljundist tulnud tehe arvutatakse välja

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ echo "3+4" | bc
7
```

Sama triki abil saab siin tulpade summat arvutada: saabuvad arvud pannakse ühte ritta ning arvude vahele eraldajaks plussmärk

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ tail -n +2 < massid.txt | paste -s -d "+"
45+35+72+53+60+43+38+46+57+35+38+550+65+69+38+68+55+110+63+59+53+62+49+69+36+67+53+58+59+63
+65
```

Sellele otsa veel bc-arvutuskäsklus ning summa ongi käes.

```
tail -n +2 < massid.txt | paste -s -d "+" | bc
2233
```

Harjutus

- Kirjutage lause nõnda, et masside andmed võetakse otse failist viiesklass.txt ning leitakse masside summa

lahendus

```
tail -n +2 <viiesklass.txt | cut -d "," -f 3  | paste -s -d "+" | bc
2233
```

Aritmeetilise keskmise leidmine

Keskmise saamiseks tuleb summa jagada elementide arvuga. Summa on eelmisest käsust käes. Ridade arvu saab kätte wc-käsust

Reavahetuste, sõnade ja tähtede arv

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ wc viiesklass.txt
 32  32 533 viiesklass.txt
```

Sobiva arvu eraldamiseks muudan kõigepealt kõik tühikud ja muud “valged sümbolid” ühekordseks tr-käsuga.

Topelttühikud ühekordseks

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ wc viiesklass.txt | tr -s " "
 32 32 533 viiesklass.txt
```

Eemadan algusesse jäänud tühiku

Alles andmed alates teisest tähest

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ wc viiesklass.txt | tr -s " " | cut -c 2-
32 32 533 viiesklass.txt
```

cut-käsu abil saan kätte esimese tulba sisu, eraldajaks tühik

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ wc viiesklass.txt | tr -s " " | cut -c 2-
| cut -d " " -f 1
32
```

Kokku saab nõnda pika käsu. Langevate ülakomade vahele pandud väärtus läheb echo-käsus käima. Jagamismärk lause keskel jääb tavaliseks sümboliks ning käivitamise tulemusena saab tehte kokku. Praegu veel tekitab pealkirjarida arvutusanomaalia, samuti laste üllatavalt suur keskmine mass tuleneb testimiseks sisse pandud paarist väga raskest lapsest.

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ more keskmine1.sh
echo `tail -n +2 <viiesklass.txt | cut -d "," -f 3  | paste -s -d "+" | bc`/`wc
viiesklass.txt | tr -s " " | cut -c 2- | cut -d " " -f 1` | bc
jaagup@praktika1 ~/public_html/2018/dt/06skript $ sh keskmine1.sh
69
```

Kui käsk liialt pikale läheb, saab selle mitmesse ritta jagada, jättes eelmise rea lõppu langjoone. See toimib aga ainult juhul, kui redaktor salvestab reavahetused Linuxi-režiimis (\n)

Ainult Unixi reavahetuste korral saab käsu jagada mitmesse ritta

```
echo `tail -n +2 <viiesklass.txt | cut -d "," -f 3  | paste -s -d "+" | bc` / \
  `wc viiesklass.txt | tr -s " " | cut -c 2- | cut -d " " -f 1` | bc
```

Käsu võib paigutada faili, siis ta kergemini kättesaadav. Ning siis kannatab juba failinime ette anda käsurea parameetrina - esimene sõna programminime järel on kättesaadav muutujast $1

```
more keskmine2.sh
echo `tail -n +2 <$1 | cut -d "," -f 3  | paste -s -d "+" | bc` / \
  `wc $1 | tr -s " " | cut -c 2- | cut -d " " -f 1` | bc
```

Sinna see failinimi “viiesklass.txt” lähebki

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ sh keskmine2.sh viiesklass.txt
69
```

Juurde veel teine parameeter, et millise tulba keskmist leitakse, tema nimeks siis $2

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ more keskmine2.sh
echo `tail -n +2 <$1 | cut -d "," -f $2  | paste -s -d "+" | bc` / \
  `wc $1 | tr -s " " | cut -c 2- | cut -d " " -f 1` | bc
```

ja võibki näha keskmist pikkust

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ sh keskmine2.sh viiesklass.txt 2
154
```

Harjutus

- Koosta programm, millele antakse ette uuritava faili nimi. Programmi töö tulemusena kirjutatakse vastusefaili aruanne, kus sisaldub uuritava faili nimi ning nii teise kui kolmanda tulba arvude summa ning keskmine

Lahendus

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ sh uuring.sh viiesklass.txt
jaagup@praktika1 ~/public_html/2018/dt/06skript $ more vastus.txt
Faili viiesklass.txt uuring
Teise tulba arvude summa:

4952
jaagup@praktika1 ~/public_html/2018/dt/06skript $ more uuring.sh
echo "Faili $1 uuring" > vastus.txt
echo "" >> vastus.txt
echo "Teise tulba arvude summa: " >> vastus.txt
echo `tail -n +2 <$1 | cut -d "," -f 2  | paste -s -d "+" | bc` >> vastus.txt
```

curl

Üksiku faili küsimine veebist

```
jaagup@praktika1 ~/public_html/2018/dt/06skript $ curl
http://www.tlu.ee/~jaagup/andmed/muu/5klass.txt
eesnimi,pikkus,mass,sugu
Juku,170,45,m
Kati,160,35,n
Mati,160,72,m
Madis,165,53,m
Mati,163,60,m
Katrin,165,43,n
```

Mida võib endiselt edasi töödelda teiste käskudega

```
curl http://www.tlu.ee/~jaagup/andmed/muu/5klass.txt | cut -d "," -f 3

mass
45
35
72
53
60
43
```

wget

Rekursiivne (linkide järgi) failikogumiku alla laadimine. Võti (-r), rekursiivne näitab, et minnakse viidete sisse, (-l 2) ehk level 2, et piirdutakse teise taseme viidetega.

```
wget -r -l 2 http://www.tlu.ee/~jaagup/andmed/muu/
```

Tekkis kataloog nimega www.tlu.ee, mille sisse paigutati sobivad failid

```
jaagup@praktika1 ~/public_html/2018/dt/06skript/hoidla $ ls -l
total 4
drwxrwxr-x 4 jaagup jaagup 4096 Sep 25 10:32 www.tlu.ee
```

Python

Shelli skriptid on failidega töötamiseks levinud ja mugavad. Kui tahta andmestiku detailide kallal pikemalt nokitseda, siis selleks on “päris” programmeerimiskeeled levinumad ja mugavamad. Ehkki - mugav on enamasti see, millega rohkem harjunud oled ning ka käsureaskriptide abil saab faili üksikute ridade kaupa ette võtta ning tingimuste järgi määrata, mida kusagil ette võtta.

Märgatav osa programmeerimiskeeli nõuab, et kõigepealt tuleb käskudest kood valmis kirjutada, siis masinale arusaadavaks kompileerida ning alles seejärel võib käivitama ja tulemusi vaatama hakata. Pythoni teeb muuhulgas alustajale mugavaks võimalus käske kohe ka ühekaupa käivitada ja tulemusi vaadata.

Pythoni keele kohta seletavaid materjale algajatele leiab näiteks TLÜ õppejõud Inga Petuhhovi veetava Programmeerimise algkursuse lehelt
http://www.cs.tlu.ee/~inga/progbaas/

Interpretaatori käivitamiseks tuleb käsureal sisestada selle nimi - siinses näitmasinas on selleks käsklus python3.5.

```
jaagup@praktika1 ~/public_html/2018/dt $ python3.5
Python 3.5.1+ (default, Mar 30 2016, 22:46:26)
[GCC 5.3.1 20160330] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Edasi võib ükshaaval korraldusi jagada. Alustuseks lihtne liitmistehe, millele arvuti kuvab ka kohe vastuse

```
>>> 3+2
5
```

Sama tulemuse saab, kui paluda töö tulemus print-käsu abil välja kuvada.

```
>>> print(3+2)
5
```

Käsureal käivitades kuvatakse tulemus välja, kui sellele muud ülesannet pole antud. Tekstid tuleb paigutada jutumärkide või ülakomade vahele. Ja ka ühe väärtuse sisestamine on käsk, mille tulemus trükitakse välja

```
>>> "Tere"
'Tere'
```

Tekst on Pythoni jaoks ühtlasi tähtede kogum. Loendamist alustatakse nullist. Esimese tähe väljatrükk

```
>>> "Tere"[0]
'T'
```

Pikema tekstilõigu puhul tuleb märkida, et kust alates ja kuhuni trükitakse. Kusjuures vahed on nummerdatud vastavate tähtede eest

```
|T|e|r|e|
0 1 2 3 4
```

```
>>> "Tere"[0:2]
'Te'
```

andis siis tulemuseks kaks esimest tähte.

Python võimaldab ka lõpuotsast küsida

```
>>> "Tere"[-1]
'e'
```

annab tulemuseks teksti viimase tähe
Tähtede arv sõnas käsuga len

```
>>> len("Tere")
4
```

Ning veidi nagu naljanumbrina saab tekste ka arvuga korrutada

```
>>> 5*"Tere"
'TereTereTereTereTere'
```

split, tükeldamine

Teksti puhul sai kantsulgudes oleva järjekorranumbri abil pöörduda üksikute tähtede poole. Pöördutavaks üksuseks võib olla aga ka midagi muud, näiteks sõnad. Käsklus split muudab teksti massiiviks, sulgudes oleva jutumärkides/ülakomades sümboli(te) abil näidatakse, et mille kohalt tekst tükeldatakse.

```
>>> "Juku tuli kooli".split(' ')
['Juku', 'tuli', 'kooli']
```

Jällegi algab loendamine nullist.

```
>>> "Juku tuli kooli".split(' ')[0]
'Juku'
```

Miinusmärgi abil võib lõpust lugeda

```
>>> "Juku tuli kooli".split(' ')[-1]
'kooli'
```

ning len näitab, mitmest osast kogum koosneb.

```
>>> len("Juku tuli kooli".split(' '))
3
```

Et lugemist alati nullist, siis teine sõna on järjekorranumbriga 1

```
>>> "Juku tuli kooli".split(' ')[1]
'tuli'
```

Teisest alates kuni lõpuni kirjutatakse arv üks koos sellele järgneva kooloniga

```
>>> "Juku tuli kooli".split(' ')[1:]
['tuli', 'kooli']
```

Muutujad

Enamikes programmeerimiskeeltes saab (vahe)tulemusi meelde jätta märksõnade ehk muutujate (variable) abil, nii võimalik neid hiljem sobivas kohas jälle pruukida

```
>>> vanus=5
>>> vanus
5
>>> print(vanus)
5
```

Arvutades käituvad nad nagu tavalised väärtused

```
>>> vanus+1
6

>>> print(vanus)
5
```

Tehte tulemusena saab aga arvutada uue väärtuse ning selle siis muutujasse salvestada

```
>>> vanus=vanus+1
>>> vanus
6
```

Tingimus

Vastavalt tingimusele kannatab programmeerimiskeeles valida, mida tehakse või väljastatakse. Siinses näites väljastatakse kuuest eluaastast suurema vanuse korral, et tegemist on koolilapsega, muul juhul on käsu väljundiks lihtsalt laps

```
>>> "koolilaps" if vanus>6 else "laps"
'laps'
```

Suurema vanuse korral ka vastus teistsugune

```
>>> vanus=8
>>> "koolilaps" if vanus>6 else "laps"
'koolilaps'
```

Massiiv

Teksti sai massiiviks muuta split-käskluse abil. Väärtused saab aga massiivi ka otse sisse anda. Loetelu ümber kandilised sulud ning elementide vahele komad. Praegusel juhul elementideks täisarvud

```
>>> vanused=[5, 6, 8, 11]
>>> vanused
[5, 6, 8, 11]
```

Taas saab neid järjekorranumbri järgi küsida. Olgu algusest

```
>>> vanused[0]
5
```

või lõpust

```
>>> vanused[-1]
11
```

Elementide arv ka samamoodi

```
>>> len(vanused)
4
```

Pythoni omapäraks on võimalus massiivi sees elemente lühidalt ümber arvutada. Järgnev käsklus loob vana põhjal uue massiivi, kus iga lapse vanus on ühe aasta jagu suurem. Tsüklikäsklus for käib ükshaaval läbi vanuste massiivi elemendid. Iga ringi juures saab vanus konkreetse lapse vanuse väärtuse ning uude väljastatavasse massiivi jäetakse endisest ühe võrra suurem väärtus

```
>>> [vanus+1 for vanus in vanused]
[6, 7, 9, 12]
```

vanuste massiiv ise aga jäi endiseks

```
>>> vanused
[5, 6, 8, 11]
```

Korduse ja tingimuse saab ka kokku panna - et millise vanuse juures on inimene lihtsalt laps ja millal koolilaps

```
>>> ["koolilaps" if vanus>6 else "laps" for vanus in vanused]
['laps', 'laps', 'koolilaps', 'koolilaps']
```

Harjutus

- Koosta lause, pane muutujasse
- split-käsu abil paiguta lause sõnad massiivi
- Kuva lause viimane sõna
- Kuva lause viimase sõna pikkus
- Koosta uus massiiv, kus igaks elemendiks on lause vastava sõna tähtede arv

lahendus

```
>>> lause="Juku tuli kooli"
>>> m=lause.split()
>>> m[-1]
'kooli'
>>> len(m[-1])
5
>>> pikkused=[len(sona) for sona in m]
>>> pikkused
[4, 4, 5]
```

Mõned täiendavad näited harjutuse juurde.

Uue massiivi loomise tsüklit saab rakendada ka värskelt split-käsuga loodud massiivile

```
>>> [len(sona) for sona in "Juku tuli kooli".split()]
[4, 4, 5]
```

Loomistsükkel ilma andmeid muutmata

```
>>> [sona for sona in "Juku tuli hommikul kooli".split()]
['Juku', 'tuli', 'hommikul', 'kooli']
```

juurde tingimus, et loetellu jäävad alles vaid neljast tähest pikemad sõnad

```
>>> [sona for sona in "Juku tuli hommikul kooli".split() if len(sona)>4]
['hommikul', 'kooli']
```

len-käsu abil nende pikkused

```
> [len(sona) for sona in "Juku tuli hommikul kooli".split() if len(sona)>4]
[8, 5]
```

mitu käsku kokku pannes on tulemuseks lauses olevate neljast pikemate sõnade tähtede arvu summa

```
> sum([len(sona) for sona in "Juku tuli hommikul kooli".split() if len(sona)>4]
13
```

Regulaaravaldised

Kasutada saab neid mitme keele ja vahendi sees, põhiomadused on enamikes paikades sarnased. Pythonis on nende pruukimiseks mugavaks mitmekülgseks käskluseks findall paketist re (regular expressions). Näitena numbrid tekstist

```
>>> import re
>>> re.findall("[0-9]", "2 kassi ja 3 koera")
['2', '3']
```

Kui otsinguks on sümbol loetelust null kuni üheksa, siis näidatakse arvu 12 numbrid eraldi

```
>>> re.findall("[0-9]", "12 kassi ja 3 koera")
['1', '2', '3']
```

Plussmärk teatab, et otsitakse järjestikust üht või rohkemat sümbolit ning sellisel juhul tuleb arv 12 kokku.

```
>>> re.findall("[0-9]+", "12 kassi ja 3 koera")
['12', '3']
```

Ülakomad leitud vastete ümber tähendavad, et tulemused on tekstitüüpi. Neid arvutamiseks kasutades on nad vaja enne arvuks muuta. Teksti muudab täisarvuks käsklus int (sõnast integer). Käsklus map võimaldab käsu rakendada massiivi kõigile liikmetele. map-i tulemus välja trükkides väljastatakse lihtsalt objekti aadress - ehkki soovitud andmed on seal sees. Nende nägemiseks ilmutatud kujul sobib käsklus list.

```
>>> map(int, re.findall("[0-9]+", "12 kassi ja 3 koera"))
<map object at 0x7fcc98908be0>
>>> list(map(int, re.findall("[0-9]+", "12 kassi ja 3 koera")))
[12, 3]
```

Lauses olevate arvude summa leidmiseks sobib leitud arvud sum-käsuga kokku liita

```
>>> sum(list(map(int, re.findall("[0-9]+", "12 kassi ja 3 koera"))))
15
```

Vahepealse - list-käsu võib ka vahelt välja jätta, sest andmed mõistetakse välja võtta ka otse map-käsu väljundist.

```
>>> sum(map(int, re.findall("[0-9]+", "12 kassi ja 3 koera")))
15
```

Harjutus

- kuvage findall-käsu abil kõik tekstis leiduvad a-tähed
- näidake arv, mitu neid on

lahendus

```
>>> re.findall("a", "12 kassi ja 3 koera")
['a', 'a', 'a']
>>> len(re.findall("a", "12 kassi ja 3 koera"))
3
```

Hulgad

Programmeerimiskeeled eristavad loetelusid (list) ja hulki (set). Esimeses neist võivad elemendid korduda, nende omavaheline järjekord on tähtis. Teises vastupidi - mitut sama väätusega elementi ei säilitata, nende järjekorda ei arvestata - vähemasti Pythoni standardi juurde kuuluva hulgaklassi juures. Kirjapildis on loetelu elemendid kandiliste sulgude vahel, hulga omad (ja hiljem ka assotsiatiivmassiivi/dictionary omad) loogeliste sulgude vahel

```
>>> set(['a', 'a', 'e', 'u', 'u'])
{'a', 'e', 'u'}
```

Elementide arvu saab kätte samamoodi

```
>>> len(set(['a', 'a', 'e', 'u', 'u']))
3
```

Harjutus

- kuvage, millised erinevad täishäälikud on lauses
- kuvage, mitu erinevat täishäälikut on lauses

```
>>> set(re.findall("[aeiouõäüö]", "12 kassi, 3 koera ja 1 küülik"))
{'a', 'e', 'ü', 'i', 'o'}
>>> len(set(re.findall("[aeiouõäüö]", "12 kassi, 3 koera ja 1 küülik")))
```

```
5
```

Kui (findalli väljastatav) massiiv on tühi, siis if-tingimus loeb selle eitavaks ehk vääraks väärtuseks. Nii saab kirjutada soovitud otsingutingimuse.

```
>>>  "7 olemas" if '7' in re.findall("[0-9]", "12 varblast ja 3 koera") else "seitset pole"

 'seitset pole'
```

Kui otsitu leitakse, siis antakse sellest ka teada

```
>>> "7 olemas" if '7' in re.findall("[0-9]", "17 varblast ja 3 koera") else "seitset pole"

'7 olemas'
```

Loetellu kuulumist kontrollitakse in-operaatori abil. Kuna ülakomade vahel olevat teksti loetakse täheloeteluks, siis on Pythonile arusaadav ka järgnev kontroll. Ehk kui küsitakse, kas sümol seitse sisaldub sümbolite loetelus, kus asuvad sümbolid üks ja seitse, siis vastus on jah.

```
>>> '7' in '17'
True
```

Samuti tuleb jaatav vastus küsimusele, kas seitse sisaldub loetelus, kus on üks, seitse ja kolm

```
>>> '7' in ['1', '7', '3']
True
```

Tehted hulkadega

Hulkade liitmise, ühisosa või vahe tehetega saab kätte vastuseid, mille leidmine muul programmeerimise moel võib pikemaks osutuda. Mõned näited

```
>>> jukukeeled={"eesti", "vene"}
>>> katikeeled={"eesti", "soome", "inglise"}
>>> jukukeeled
{'vene', 'eesti'}
>>> katikeeled
{'eesti', 'soome', 'inglise'}
```

Hulkade ühend, ehk kahe peale kõik erinevad väärtused kokku

```
>>> jukukeeled.union(katikeeled)
{'eesti', 'vene', 'soome', 'inglise'}
```

Sama tulemuse saab ka püstkriipsu-tehte abil

```
>>> jukukeeled | katikeeled
{'eesti', 'vene', 'soome', 'inglise'}
```

Hulkade ühisosa, koos vastava tehtemärgiga

```
>>> jukukeeled.intersection(katikeeled)
{'eesti'}
>>> jukukeeled & katikeeled
{'eesti'}
```

Hulkade vahe

```
>>> jukukeeled.difference(katikeeled)
{'vene'}
>>> jukukeeled - katikeeled
{'vene'}
```

Harjutus

- Kuvage keeled, mida oskab Kati, aga mitte Juku

```
>>> katikeeled-jukukeeled
{'soome', 'inglise'}
```

XOR ehk keeled, mida oskab vaid üks, aga mitte mõlemad

```
>>> jukukeeled ^ katikeeled
{'vene', 'soome', 'inglise'}
```

Hulkade näide failiandmetega

Kahe tekstifaili sisu - ühes jooksjate, teises ujujate nimed

```
jaagup@praktika1 ~/public_html/2018/dt/08pyfailid $ more jooksjad.txt
Juku
Kati
Madis
Mati

jaagup@praktika1 ~/public_html/2018/dt/08pyfailid $ more ujujad.txt
Anu
Kati
Madis
```

Samas kataloogis sisenen Pythoni käsureale

```
jaagup@praktika1 ~/public_html/2018/dt/08pyfailid $ python3.5
Python 3.5.1+ (default, Mar 30 2016, 22:46:26)
[GCC 5.3.1 20160330] on linux
```

```
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Faili sisu lugemiseks saab faili avada ning read-käsuga selle sisus muutujatesse paigutada

```
>>> j=open("jooksjad.txt").read()
```

Väljatrükil näha ka failis olevad reavahetused \n

```
>>> j
'Juku\nKati\nMadis\nMati\n'
```

Regulaaravaldise abil leiab välja näiteks tekstifailis paiknevad a-tähed

```
>>> import re
>>> re.findall("a", j)
['a', 'a', 'a']
```

Võimalik andmed ka kohe massiivi lugeda - selleks käsklus readlines

```
>>> jooksjad=open("jooksjad.txt").readlines()
>>> jooksjad
['Juku\n', 'Kati\n', 'Madis\n', 'Mati\n']
```

Nagu näha, siis niimoodi jäävad reavahetused massiivi elementidesse sisse
Kui faili sisu reavahetuste kohalt tükeldada, siis nimede taha reavahetusi ei jää, küll aga säilub faili lõpus olnud reavahetus

```
>>> jooksjad=open("jooksjad.txt").read().split("\n")
>>> jooksjad
['Juku', 'Kati', 'Madis', 'Mati', '']
```

Üheks mooduseks on vana massiiv läbi käies luua uus massiiv nendest ridadest, mille pikkus on rohkem kui 0 sümbolit, ehk kus on midagi.

```
>>> [jooksja for jooksja in jooksjad if len(jooksja)>0]
['Juku', 'Kati', 'Madis', 'Mati']
```

Või siis lugeda read massiiviks ning edasi strip-käsuga eemaldada otstesse jäänud läbipaistvad sümbolid (tühikud, tabulaatorid ja reavahetused)

```
>>> jooksjad=[jooksja.strip() for jooksja in open("jooksjad.txt").readlines()]
>>> jooksjad
['Juku', 'Kati', 'Madis', 'Mati']
```

Edasi juba võimalikud tavalised hulgatehted. Ehk siis leida jooksjad, kes pole ujujad (tuleb ka enne sisse lugeda)

```
>>> set(jooksjad)-set(ujujad)
{'Mati', 'Juku'}
```

Hulgatehete jaoks tasub andmed kohe hulka ja mitte loetellu lugeda - ehk siis andmete ümber panna loogelised sulud.

```
>>> jooksjad={jooksja.strip() for jooksja in open("jooksjad.txt").readlines()}
>>> jooksjad
{'Mati', 'Madis', 'Juku', 'Kati'}

>>> ujujad={ujuja.strip() for ujuja in open("ujujad.txt").readlines()}
>>> ujujad
{'Madis', 'Anu', 'Kati'}
```

Hulkade tehted ja nende tulemused

```
>>> jooksjad | ujujad
{'Madis', 'Juku', 'Kati', 'Mati', 'Anu'}
>>> jooksjad ^ ujujad
{'Mati', 'Juku', 'Anu'}
>>> #ainult jooksis või ujus

>>> jooksjad-ujujad
{'Mati', 'Juku'}
```

Hulkade sisaldumist saab kontrollida operaatori <= abil. Ehk siis uuritakse, kas Mati ja Juku on mõlemad jooksjad, vastuseks saadi “jah”

```
>>> jooksjad
{'Mati', 'Madis', 'Juku', 'Kati'}
>>> {'Mati', 'Juku'} <= jooksjad
True
```

Kuna Mari pole jooksjate loetelus, siis kontroll, kas Mati +´Juku + Mari on jooksjad annab vastuseks “ei”

```
>>> {'Mati', 'Juku', 'Mari'} <= jooksjad
False
```

Loendamine

Alustuseks korduvate väärtustega massiiv

```
>>> loomad="koer kass koer koer kass".split()
>>> loomad
['koer', 'kass', 'koer', 'koer', 'kass']
```

Paketi collections klass Counter aitab väärtuste kordade arvu lugeda

```
>>> from collections import Counter
>>> loendaja=Counter(loomad)
>>> loendaja
Counter({'koer': 3, 'kass': 2})
```

Hiljem võimalik üksikutele võtmetele vastavad väärtused eraldi välja küsida.

```
>>> loendaja["koer"]
3
```

Pandas

Pythoni keelde on tavavahenditele lisaks loodud andmetöötluspakett Pandas, mida mõnedki peavad põhjuseks Pythonit oma uuringute juures kasutada. Installimine võib vahel veidi keerukas olla, kuid üheks mooduseks Pandas tööle saada on see lisada Anaconda-nimelise komplekti koosseisus.

Et oleks andmeid mida töödelda, selleks kopeerime eelnevalt kasutatud viiesklass.txt-nimelise faili uude kataloogi pandase-harjutuste tarbeks.

```
jaagup@praktika1 ~/public_html/2018/dt $ cp 06skript/viiesklass.txt 09pandas/
jaagup@praktika1 ~/public_html/2018/dt $ cd 09pandas/
jaagup@praktika1 ~/public_html/2018/dt/09pandas $ python3.5
Python 3.5.1+ (default, Mar 30 2016, 22:46:26)
[GCC 5.3.1 20160330] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

Uus pakett tuleb eraldi sisse lugeda. Tüüpiliseks kujuks selle puhul talle anda alias pd

```
>>>import pandas as pd
```

Käsuga pd.read_csv saab andmetabeli samast kataloogist kätte

```
>>> lapsed=pd.read_csv("viiesklass.txt")
```

head-käsk kuvab tabeli alguse

```
>>> lapsed.head()
  eesnimi  pikkus  mass sugu
0    Juku     170    45    m
1    Kati     160    35    n
2    Mati     160    72    m
3   Madis     165    53    m
4    Mati     163    60    m
```

Üksikute tulpade kohta saab arvutusi küsida - suurim, vähim ja keskmine, samuti mediaan, ehk väärtus, millest pooled on suuremad ja pooled väiksemad. Kui aritmeetiline keskmine on mediaanist väiksem, siis see vihjab, et hulgas on mõned ebaproportsionaalselt väikesed väärtused (ehk lühikesed lapsed), kes mõjutavad aritmeetilist keskmist tunduvalt rohkem kui üksikväärtustest vähem sõltuvat mediaani.

```
>>> lapsed.pikkus.max()
175
>>> lapsed.pikkus.min()
143
>>> lapsed.pikkus.mean()
159.74193548387098
>>> lapsed.pikkus.median()
162.0
```

Tabelist üksikute väärtuste kätte saamine on veidi tülikam ettevõtmine. Niisama values-käsklus tulbale annab numpy-paketi massiivi

```
>>> lapsed.pikkus.values
array([170, 160, 160, 165, 163, 165, 151, 159, 164, 148, 143, 169, 156,
       165, 170, 145, 151, 175, 156, 164, 155, 158, 164, 170, 143, 152,
       156, 164, 165, 164, 162])

>>> type(lapsed.pikkus.values)
<class 'numpy.ndarray'>
```

selle otsa aga tolist() kirjutades tuleb välja tavaline Pythoni list ehk massiiv.

```
>>> lapsed.pikkus.values.tolist()
[170, 160, 160, 165, 163, 165, 151, 159, 164, 148, 143, 169, 156, 165, 170, 145, 151, 175,
156, 164, 155, 158, 164, 170, 143, 152, 156, 164, 165, 164, 162]
```

Sortimine

Andmestiku järjestamiseks tunnuse järgi sobib sort_values - näha mõned lühemad lapsed

```
>>> lapsed.sort_values(by="pikkus").head()
      eesnimi  pikkus  mass sugu
24       Kert     143    36    m
10      Maria     143    38    n
15       Miia     145    68    n
9   Katariina     148    35    n
6        Siim     151    38    m
```

Nimekirja tagumise otsa saab käsuga tail

```
>>> lapsed.sort_values(by="pikkus").tail()
   eesnimi  pikkus  mass sugu
11   Marta     169   550    n
14   Tiina     170    38    n
23    Mart     170    69    m
0     Juku     170    45    m
17   Juhan     175   110    m
```

muuhulgas saab määrata, mitut kirjet tahetakse

```
>>> lapsed.sort_values(by="pikkus").tail(3)
   eesnimi  pikkus  mass sugu
23    Mart     170    69    m
0     Juku     170    45    m
17   Juhan     175   110    m
```

Lisaparameeter ascending=False keerab järjestuse tagurpidi, suuremad ette

```
>>> lapsed.sort_values(by="pikkus", ascending=False).head(3)
```

```
   eesnimi  pikkus  mass sugu
17   Juhan     175   110    m
0     Juku     170    45    m
23    Mart     170    69    m
```

Eesnimed pikkuste järjekorras

```
>>> lapsed.sort_values(by="pikkus", ascending=False).eesnimi.values.tolist()
['Juhan', 'Juku', 'Mart', 'Tiina', 'Marta', 'Madis', 'Katrin', 'Jaanika', 'Mihkel',
'Jaanus', 'Moonika', 'Kadri', 'Killu', 'Kristjan', 'Mati', 'Jaan', 'Kati', 'Mati',
'Martin', 'Kristiina', 'Priit', 'Madis', 'Lauri', 'Kristi', 'Gert', 'Siim', 'Siim',
'Katariina', 'Miia', 'Maria', 'Kert']
```

Omakorda pääseb nende andmete juures rakendama tavalisi Pythoni käske - tulemuseks on esitähtede loetelu laste pikkuste järjekorras

```
>>> [eesnimi[0] for eesnimi in lapsed.sort_values(by="pikkus",
ascending=False).eesnimi.values.tolist()]
['J', 'J', 'M', 'T', 'M', 'M', 'K', 'J', 'M', 'J', 'M', 'K', 'K', 'K', 'M', 'J', 'K', 'M',
'M', 'K', 'P', 'M', 'L', 'K', 'G', 'S', 'S', 'K', 'M', 'M', 'K']
```

Soovides nad kokku ühendada aitab Pythoni stringi/sõne käsklus join. Nätieks “-”.join([‘a’, ‘b’,’c’]) annab tulemuseks ‘a-b-c’ ning kogu lasteseltskonna eesnimede esitähtede loetelu tuleb järgnevalt

```
>>> "-".join([eesnimi[0] for eesnimi in lapsed.sort_values(by="pikkus",
ascending=False).eesnimi.values.tolist()])
'J-J-M-T-M-M-K-J-M-J-M-K-K-K-M-J-K-M-M-K-P-M-L-K-G-S-S-K-M-M-K'
```

Uue tulba saab tekitada andmed sellele omistades. Valemi saab luua terve tulba arvutamiseks korraga

```
>>> lapsed["pikkusmeetrites"]=lapsed["pikkus"]/100.0
>>> lapsed.head()
  eesnimi  pikkus  mass sugu  pikkusmeetrites
0    Juku     170    45    m             1.70
1    Kati     160    35    n             1.60
2    Mati     160    72    m             1.60
3   Madis     165    53    m             1.65
4    Mati     163    60    m             1.63
```

Nagu Pythonis mujalgi, saab ka siin del-käsuga loetelu ühe elemendi - ehk siis praegusel juhul meetrites arvutatud pikkuse tulba ära kustutada

```
>>> del lapsed["pikkusmeetrites"]
>>> lapsed.head()
  eesnimi  pikkus  mass sugu
0    Juku     170    45    m
1    Kati     160    35    n
2    Mati     160    72    m
3   Madis     165    53    m
4    Mati     163    60    m
```

Arvutile on arusaadav tehe kuvada, et millise lapse pikkus on suurem kui 165 sentimeetrit - vastuseks siis jah-id ja ei-d laste järjekorras

```
>>> lapsed.pikkus>165
0      True
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11     True
12    False
13    False
14     True
15    False
16    False
17     True
18    False
19    False
20    False
21    False
22    False
23     True
24    False
25    False
26    False
27    False
28    False
29    False
30    False
Name: pikkus, dtype: bool
```

Tegemist sarnase pandase tabeliga kui muudki - saab rakendada näiteks head-käsku

```
>>> (lapsed.pikkus>165) . head()
0     True
1    False
2    False
3    False
4    False
Name: pikkus, dtype: bool
```

Saadud loetelu võib aga laste kuvamiseks kantsulgudesse anda - nii näidatakse lapsi, kelle pikkus on suurem kui 165

```
>>> lapsed[lapsed.pikkus>165]
   eesnimi  pikkus  mass sugu
0     Juku     170    45    m
11   Marta     169   550    n
14   Tiina     170    38    n
17   Juhan     175   110    m
```

```
23    Mart     170    69    m
```

Päringu tulemuseks on taas tavaline pandase dataframe - nii et võin küsida, milline on pikkade laste juures vähim mass

```
>>> lapsed[lapsed.pikkus>165].mass.min()
38
```

Ridade arv tabelis

```
>>> len(lapsed)
31
```

ning veergude arv tabelis

```
>>> len(lapsed.columns)
4
```

160st sentimeetrist pikemate laste arv

```
>>> len(lapsed[lapsed.pikkus>160])
16
```

Ja vastavad lapsed ise. Nagu näha, siis id-d ei lähe järjest - aga ei peagi minema, sest ainult osa ju loetellu alles jäänud ning identifikaator samal lapsel/real/kirjel ikka sama

```
>>> lapsed[lapsed.pikkus>160]
     eesnimi  pikkus  mass sugu
0       Juku     170    45    m
3      Madis     165    53    m
4       Mati     163    60    m
5     Katrin     165    43    n
8      Kadri     164    57    n
11     Marta     169   550    n
13    Mihkel     165    69    m
14     Tiina     170    38    n
17     Juhan     175   110    m
19  Kristjan     164    59    m
22     Killu     164    49    n
23      Mart     170    69    m
27   Moonika     164    58    n
28   Jaanika     165    59    n
29    Jaanus     164    63    m
30      Jaan     162    65    m
```

Rühmitamine

Niisama rühmitamise peale saame teada lihtsalt, et lapsi on kahest soost

```
>>> lapsed.groupby("sugu")
<pandas.core.groupby.DataFrameGroupBy object at 0x7fb2cae2d160>
```

```
>>> len(lapsed.groupby("sugu"))
2
```

Kui tahame teada saada kummastki soost laste arvu, võime kokku lugeda, mitu eesnimi kumbagi rühma sai

```
>>> lapsed.groupby("sugu").eesnimi.count()
sugu
m    18
n    13
Name: eesnimi, dtype: int64
```

Samuti leiab kummagi rühma suurima pikkuse

```
>>> lapsed.groupby("sugu").pikkus.max()
sugu
m    175
n    170
Name: pikkus, dtype: int64
```

- Kuvage nii poiste kui tüdrukute puhul suurim pikkus laste hulgas, kelle mass on alla 60 kg

```
>>> lapsed[lapsed.mass<60].groupby("sugu").pikkus.max()
sugu
m    170
n    170
Name: pikkus, dtype: int64
```

Sõnaliikide uuring

Keeleteadus on digihumanitaaria üks märgatav haru. Tekstide uuringu juures sõna küllalt lihtsaks tunnuseks on sõnaliik. Nende automaatseks määramiseks kasutatakse parsereid. Pythoni vahendite hulgas saab neid kasutada estnltk kaudu (Natural Language Toolkit). Praeguses näites kasutame juba tekstianalüsaatori olemasolevat väljundit.

http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_sonad_lemmad_sonaliigid.txt

```
word_texts,lemmas,postags
Kui,kui,D|J
Kungla,Kungla,H
rahvas,rahvas,S
kuldsel,kuldne,A
aal,aal,S
kord,kord,D
istus,istuma,V
maha,maha,D
```

```
sööma,sööma,V
",",",",Z
siis,siis,J
Vanemuine,Vanemuine,H
murumaal,murumaa,S
läks,minema,V
kandle,kannel,S
lugu,lugu,S
lööma,lööma,V
.,.,Z
```

Tuntud laul “Kungla rahvas”. Igal real sõna, sõna kohta kolm väärtust - laulus esinev sõna ise, selle algvorm (lemma) ning sõnaliik. Hiljem näeb ka sõnaliikide pikemaid kirjeldusi, esimesest näitest paistab välja, et Z-iga tähistatakse kirjavahemärki, V ehk verb on tegusõna ning S ehk substantiiv on nimisõna. Loeme faili andmed pandas-teegi abil mällu

```
>>> import pandas as pd
>>>sonad=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_sonad_lemmad_sonaliigid.txt")
>>>sonad
   word_texts        lemmas postags
0         Kui           kui     D|J
1      Kungla        Kungla       H
2      rahvas        rahvas       S
3     kuldsel        kuldne       A
4         aal           aal       S
5        kord          kord       D
```

Edasi juba võib andmestikku uurima hakata. Kõik loetelus olevad nimisõnad (S) saab nõnda:

```
>>> sonad[sonad.postags=='S'].word_texts.values.tolist()
['rahvas', 'aal', 'murumaal', 'kandle', 'lugu', 'metsa', 'laande', 'lauluga', 'saivad',
'lind', 'lehepuu', 'loomad', 'laululugu', 'mets', 'mere', 'suu', 'rahva', 'sugu',
'lauluviis', 'pärjad', 'pähe', 'murueide', 'tütreid', 'rahvas', 'mättal', 'mäe', 'õhtu',
'õues', 'kandle', 'hääl', 'põues']
```

ehk siis tabelist jäetakse vastavad read alles, küsitakse tulba “word_texts” väärtus ning tavaliseks Pythoni listiks saamiseks veel juurde käsud values.tolist()

Harjutus

- Pange võrdluses S-i (Substantiiv, nimisõna) asemele V (Verb, tegusõna), vaadake tulemust

ja lahendus

```
>>> sonad[sonad.postags=='V'].word_texts.values.tolist()
     ['istus', 'sööma', 'läks', 'lööma', 'Läks', 'mängima', 'läks', 'laulis', 'kõlas',
'pandi', 'sai', 'näha', 'laulan', 'põksub']
```

Rühmitada saab siin andmeid nagu ennegi - seekord loeme kokku sõnade esinemise sagedused (ehk arvud) sõnaliikide kaupa.

```
>>> sonad.groupby("postags").postags.count()
postags
A       2
D       7
D|J     1
H       5
J      11
K       1
P       3
S      31
V      14
Z      11
```

Veidi pikem katsetus andmetega: kui ei soovi sõnaliikide kaupa ainult arve, vaid sõnu endid, siis saab rühma kuuluva komplekti kinni püüda. Funktsiooni apply parameetriks antakse iga rühma (erineva sõnaliigi) puhul massiiv uuritud sõnadest (kuna pärast groupby-d valitud omadus word_texts). Praegusel juhul sõnad ühendatakse join-käsu abil sidekriipsuga

```
>>> sonad.groupby("postags")["word_texts"].apply(lambda m: "-".join(m))
postags
A                                        kuldsel-kaunis
D                    kord-maha-Sealt-siis-Siis-siis-hilja
D|J                                                 Kui
H                 Kungla-Vanemuine-Eesti-Eesti-Vanemuise
J                   siis-aga-aga-ja-ja-ja-ja-ja-Ja-ja-ja
K                                                  peal
P                                           Ma-see-minu
S      rahvas-aal-murumaal-kandle-lugu-metsa-laande-l...
V      istus-sööma-läks-lööma-Läks-mängima-läks-lauli...
Z                                  ,-.-,-.-;-.-.-.-,-,-.
```

Harjutus

- grupeerige sõnad lemma (algsõna) järgi, kuvage iga lemma kohta millised sõnad on

Lahendus

```
>>> sonad.groupby("lemmas")["word_texts"].apply(lambda m: "-".join(m))
lemmas
,                               ,-,-,-,
.                            .-.-.-.-.-.
;                                      ;
Eesti                        Eesti-Eesti
Kungla                            Kungla
Vanemuine           Vanemuine-Vanemuise
aal                                  aal
aga                              aga-aga
```

Nagu näha, siis ka kirjavahemärgid loetakse eraldi üksusteks

Korduvatest sõnadest lahti saamiseks aitab käsklus set

```
>>> sonad.groupby("postags")["word_texts"].apply(lambda m: "-".join(set(m)))
```

```
postags
A                                        kaunis-kuldsel
D                          maha-hilja-Sealt-Siis-siis-kord
D|J                                                 Kui
H                         Vanemuise-Kungla-Eesti-Vanemuine
J                                        siis-Ja-ja-aga
```

Kui ka suur- ja väiketähti ei taheta eristada, siis saab loetelus olevad sõnad läbi käia ning igaühele anda käskluse lower()

```
>>> sonad.groupby("postags")["word_texts"].apply(lambda m: "-".join(set([sona.lower() for
sona in m])))
postags
A                                        kaunis-kuldsel
D                                sealt-siis-maha-hilja-kord
D|J                                                 kui
H                         kungla-eesti-vanemuise-vanemuine
J                                           siis-ja-aga
```

Tabelite ühendamine

Praegusel juhul on sõnaliigist näha ainult ühetäheline lühend

```
>>> sonad.head()
  word_texts  lemmas postags
0        Kui     kui     D|J
1     Kungla  Kungla       H
2     rahvas  rahvas       S
3    kuldsel  kuldne       A
4        aal     aal       S
```

Samas sõnaliikide pikemad kirjeldused on olemas eraldi teises failis

```
sonaliigid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/sonaliikide_lyhendid.txt")
>>> sonaliigid.head()
  liigilyhend         liigikirjeldus
0           A    omadussõna algvõrre
1           C   omadussõna keskvõrre
2           D               määrsõna
3           G  käändumatu omadussõna
4           H              pärisnimi
```

Tabelid saab omavahel kokku panna, pandase juures selleks käsuks merge.

Allolev käsklus teatab, et tabel (dataframe) nimega sonad ühendatakse tabeliga sonaliigid nõnda, et esimese, vasaku tabeli sonad tulbale vastab parempoolse tabeli sonaliigid tulp liigilyhend, tulemus salvestatakse tabelisse nimega “koos”.

```
>>> koos=sonad.merge(sonaliigid, left_on="postags", right_on="liigilyhend")
```

Väljatrükil näha, et tabeli tulbad pandi kõrvuti nii et tulbad postags ja liigilühend võrduvad ning liigikirjeldus siis vastavalt kõrval

```
>>> koos
   word_texts        lemmas postags liigilyhend liigikirjeldus
0       Kungla        Kungla       H           H      pärisnimi
1    Vanemuine     Vanemuine       H           H      pärisnimi
2        Eesti         Eesti       H           H      pärisnimi
3        Eesti         Eesti       H           H      pärisnimi
4    Vanemuise     Vanemuine       H           H      pärisnimi
5       rahvas        rahvas       S           S       nimisõna
6          aal           aal       S           S       nimisõna
7     murumaal       murumaa       S           S       nimisõna
8       kandle        kannel       S           S       nimisõna
9         lugu          lugu       S           S       nimisõna
10       metsa          mets       S           S       nimisõna
```

Kui tahan vaid sõna ennast ning eestikeelset liigikirjeldust näha, siis tuleb soovitud tulpade loetelu koos-dataframe kandilistesse sulgudesse ette anda. Et selle peale kahekordsed kandilised sulud tekivad, pole midagi ohtlikku.

```
>>> koos[["word_texts", "liigikirjeldus"]]
   word_texts liigikirjeldus
0       Kungla      pärisnimi
1    Vanemuine      pärisnimi
2        Eesti      pärisnimi
3        Eesti      pärisnimi
4    Vanemuise      pärisnimi
5       rahvas       nimisõna
6          aal       nimisõna
7     murumaal       nimisõna
```

Sõnade väljatrüki juures näha, et need tulid esialgse lauluga võrreldes teises järjekorras. Kui tahta sõnu samasse järjekorda jätta, tuleb sõna järjekorranumber sõnaga kaasa panna.

Praeguse seisuga on failist sisselugemisel tekkinud järjekorranumber omaette indeksis, mis on küll unikaalne identifikaator, aga pandase dataframe jaosk nagu päris tavaline tulp ei ole ning merge-käsuga tehtud ühendamise juures kaasa ei lähe.

```
>>> sonad.head()
  word_texts  lemmas postags
0        Kui     kui     D|J
1     Kungla  Kungla       H
2     rahvas  rahvas       S
3    kuldsel  kuldne       A
4        aal     aal       S
```

Tavaliseks tulbaks saamiseks kopeerin indeksi eraldi tulbaks nimega “sonanr”

```
>>> sonad["sonanr"]=sonad.index
```

Nüüd need järjekorranumbrid ilusasti tulbana olemas

```
>>> sonad.head()
  word_texts  lemmas postags  sonanr
0        Kui     kui     D|J       0
1     Kungla  Kungla       H       1
2     rahvas  rahvas       S       2
3    kuldsel  kuldne       A       3
4        aal     aal       S       4
```

ning tulevad merge-ga kaasa ka

```
>>> koos=sonad.merge(sonaliigid, left_on="postags", right_on="liigilyhend")
>>> koos.head()
  word_texts     lemmas postags  sonanr liigilyhend liigikirjeldus
0     Kungla     Kungla       H       1           H      pärisnimi
1  Vanemuine  Vanemuine       H      11           H      pärisnimi
2      Eesti      Eesti       H      44           H      pärisnimi
3      Eesti      Eesti       H      62           H      pärisnimi
4  Vanemuise  Vanemuine       H      77           H      pärisnimi
```

Väljatrükil järjestades saab sõnad soovitult ritta. Samas paistab, et esimene “Kui” on loetelust kaduma läinud. Lähemalt uurides selgub, et sealset sõnaliiki pole parser suutnud kindlalt määrata. Kuna (D|J) kujul vastet liigilühendite tabelis pole, siis tavaline merge jättiski selle rea välja.

```
>>> koos.sort_values(by="sonanr").head()
   word_texts  lemmas postags  sonanr liigilyhend       liigikirjeldus
0      Kungla  Kungla       H       1           H            pärisnimi
5      rahvas  rahvas       S       2           S             nimisõna
36    kuldsel  kuldne       A       3           A  omadussõna algvõrre
6         aal     aal       S       4           S             nimisõna
38       kord    kord       D       5           D             määrsõna
```

Kui tahta, et esimesest (vasakust) tabelist kõik väärtused sees oleksid - sõltumata sellest, kas teiselt poolt talle vastet leiab - siis aitab, kui merge-le lisada parameeter how=”left”. Nii näeb ka laulu esimest kui-d loetelus. Liigikirjelduse pool jääb tal lihtsalt tühjaks

```
>>> koos=sonad.merge(sonaliigid, how="left", left_on="postags", right_on="liigilyhend")
>>> koos.sort_values(by="sonanr").head()
  word_texts  lemmas postags  sonanr liigilyhend       liigikirjeldus
0        Kui     kui     D|J       0         NaN                  NaN
1     Kungla  Kungla       H       1           H            pärisnimi
2     rahvas  rahvas       S       2           S             nimisõna
3    kuldsel  kuldne       A       3           A  omadussõna algvõrre
4        aal     aal       S       4           S             nimisõna
```

Kokku lugemise puhul Na-d (Not accessible) ei taha hästi toimida. Kui aga puuduvad väärtused asendada näiteks sõnaga “teadmata”, siis saab nad kokku lugeda küll

```
>>> koos.fillna('teadmata').groupby("liigilyhend").postags.count()
liigilyhend
A            2
D            7
```

```
H            5
J           11
K            1
P            3
S           31
V           14
Z           11
teadmata     1
```

Sõnaliikide osakaal

Eri pikkusega tekstide võrdlemisel on sõnaliikide esinemise arvu võrdlemisel kasulikum võrrelda nende osakaale. Alustuseks aga ikkagi loendamise tulemusel saadud arvud

```
>>> sonad.groupby("postags").word_texts.count()
postags
A       2
D       7
D|J     1
H       5
J      11
K       1
P       3
S      31
V      14
Z      11
Name: word_texts, dtype: int64
```

Arvutuse tulemustüübiks on series, ehk siis võti ja ühetulbaline väärtus

```
>>> type(sonad.groupby("postags").word_texts.count())
<class 'pandas.core.series.Series'>
```

Indeksiks on sõnaliigid, ehk siis tähed, mille järgi grupeeriti

```
>>> sonad.groupby("postags").word_texts.count().index.values.tolist()
['A', 'D', 'D|J', 'H', 'J', 'K', 'P', 'S', 'V', 'Z']
```

Ja väärtusteks kogused

```
>>> sonad.groupby("postags").word_texts.count().values.tolist()
[2, 7, 1, 5, 11, 1, 3, 31, 14, 11]
```

Kui sõnade arv teada, saab sagedused koguarvuga läbi jagada ning tulevad suhted tervikusse

```
>>> sonad.groupby("postags").word_texts.count()/86
postags
A      0.023256
```

```
D      0.081395
D|J    0.011628
H      0.058140
J      0.127907
K      0.011628
P      0.034884
S      0.360465
V      0.162791
Z      0.127907
Name: word_texts, dtype: float64
```

koguarvu annab käsklus len

```
>>> sonad.groupby("postags").word_texts.count()/len(sonad)
postags
A      0.023256
D      0.081395
D|J    0.011628
H      0.058140
J      0.127907
K      0.011628
P      0.034884
S      0.360465
V      0.162791
Z      0.127907
Name: word_texts, dtype: float64
```

Hilisema arvutuse tarbeks vahetame tüübi dataframeks

```
>>> kogused=sonad.groupby("postags").word_texts.count().to_frame()
>>> kogused
         word_texts
postags
A                 2
D                 7
D|J               1
H                 5
J                11
K                 1
P                 3
S                31
V                14
Z                11
```

Indeksiks endiselt sõnaliikide tähed

```
>>> kogused.index
Index(['A', 'D', 'D|J', 'H', 'J', 'K', 'P', 'S', 'V', 'Z'], dtype='object', name='postags')
```

Nii saab sinna osakaalu tulba juurde arvutada

```
>>> kogused["osakaal"]=kogused["word_texts"]/kogused.word_texts.sum()
>>> kogused
         word_texts   osakaal
postags
A                 2  0.023256
D                 7  0.081395
```

```
D|J               1  0.011628
H                 5  0.058140
J                11  0.127907
K                 1  0.011628
P                 3  0.034884
S                31  0.360465
V                14  0.162791
Z                11  0.127907

>>> kogused
         word_texts   osakaal sonaliik
postags
A                 2  0.023256        A
D                 7  0.081395        D
D|J               1  0.011628      D|J
H                 5  0.058140        H
J                11  0.127907        J
K                 1  0.011628        K
P                 3  0.034884        P
S                31  0.360465        S
V                14  0.162791        V
Z                11  0.127907        Z
```

Hilisemaks kasutamiseks tulemus faili

```
>>> kogused.to_csv("osakaalud.txt")


postags,word_texts,osakaal,sonaliik
A,2,0.023255813953488372,A
D,7,0.08139534883720931,D
D|J,1,0.011627906976744186,D|J
H,5,0.05813953488372093,H
J,11,0.12790697674418605,J
K,1,0.011627906976744186,K
P,3,0.03488372093023256,P
S,31,0.36046511627906974,S
V,14,0.16279069767441862,V
Z,11,0.12790697674418605,Z
```

Faili lugemine tekstina

```
>>> open("osakaalud.txt").read()
'postags,word_texts,osakaal,sonaliik\nA,2,0.023255813953488372,A\nD,7,0.08139534883720931,D
\nD|J,1,0.011627906976744186,D|J\nH,5,0.05813953488372093,H\nJ,11,0.12790697674418605,J\nK,
1,0.011627906976744186,K\nP,3,0.03488372093023256,P\nS,31,0.36046511627906974,S\nV,14,0.162
79069767441862,V\nZ,11,0.12790697674418605,Z\n'
```

või siis tabelisse tagasi

```
>>> osakaalud=pd.read_csv("osakaalud.txt")
>>> osakaalud
  postags  word_texts   osakaal sonaliik
0       A           2  0.023256        A
1       D           7  0.081395        D
2     D|J           1  0.011628      D|J
```

```
3       H           5  0.058140        H
4       J          11  0.127907        J
5       K           1  0.011628        K
6       P           3  0.034884        P
7       S          31  0.360465        S
8       V          14  0.162791        V
9       Z          11  0.127907        Z
```

Programmikood failis

Mitu käsku korraga ühe Pythoni koodiga

```
jaagup@praktika1 ~/public_html/2018/dt/10pandas $ more osakaaluarvutaja.py
import pandas as pd
sonad=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/lambipirn_sonad_lemmad_
sonaliigid.txt")
kogused=sonad.groupby("postags").word_texts.count().to_frame()
kogused["osakaal"]=kogused["word_texts"]/kogused.word_texts.sum()
print(kogused["osakaal"])


jaagup@praktika1 ~/public_html/2018/dt/10pandas $ python3.5 osakaaluarvutaja.py

postags
A      0.043909
A|V    0.007082
C      0.002833
D      0.121813
D|J    0.004249
H      0.001416
I      0.004249
J      0.070822
K      0.029745
N      0.009915
O|P    0.001416
P      0.072238
S      0.260623
V      0.202550
Y      0.005666
Z      0.161473
Name: osakaal, dtype: float64
```

Estnltk

Eestikeelsete tekstide andmeid kätte leida aitab pythoni pakett estnltk.

Sõnade andmed

```
>>> from estnltk import Text
>>> t=Text("Juku tuli kooli")
>>> t
{'text': 'Juku tuli kooli'}
```

Teksti sõnade algvormid ja sõnaliigid pandase dataframena

```
>>> t.get.word_texts.lemmas.postags.as_dataframe
  word_texts  lemmas postags
0       Juku    Juku       H
1       tuli  tulema       V
2      kooli    kool       S
```

Vajadusel juurde ka sõnaliikide seletused

```
>>> t.get.word_texts.lemmas.postags.postag_descriptions.as_dataframe
  word_texts  lemmas postags postag_descriptions
0       Juku    Juku       H           pärisnimi
1       tuli  tulema       V            tegusõna
2      kooli    kool       S            nimisõna
```

Sõnade andmed saab ka tüüpide kaupa loetelus välja küsida

```
>>> t.get.word_texts.lemmas.postags.postag_descriptions.as_dict
{'postag_descriptions': ['pärisnimi', 'tegusõna', 'nimisõna'], 'word_texts': ['Juku',
'tuli', 'kooli'], 'lemmas': ['Juku', 'tulema', 'kool'], 'postags': ['H', 'V', 'S']}
```

või siis lihtsalt sobiva massiivina

```
>>> t.postags
['H', 'V', 'S']
```

Samas kataloogist faili avamine

```
>>> open("salm1.txt").read()
'Paar päeva pärast paasa pühi palusid pisikesed punase peaga poisid papalt'
```

Failis olevate sõnade liigid

```
>>> Text(open("salm1.txt").read()).postags
['N', 'S', 'K', 'S', 'S', 'V', 'A', 'A', 'S', 'S', 'S']
```

Nende sageduse kokku lugemine Counteri abil

```
>>> from collections import Counter
>>> Counter(Text(open("salm1.txt").read()).postags)
Counter({'S': 6, 'A': 2, 'N': 1, 'K': 1, 'V': 1})
```

Veebist andmete lugemine.

Veebist andmete kätte saamiseks sobib pakett urllib.request
Andmetüübi paika saamiseks vajalik decode-käsklus pärast andmete saabumist

```
>>>import urllib.request
>>>
urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt").read().dec
ode("utf8")
'Kui Kungla rahvas kuldsel aal\r\nkord istus maha sööma,\r\nsiis Vanemuine murumaal\r\nläks
kandle lugu lööma.\r\n\r\nLäks aga metsa mängima,\r\nläks aga laande lauluga.\r\n\r\nSealt
saivad lind ja lehepuu\r\nja loomad laululugu;\r\nsiis laulis mets ja mere suu\r\nja Eesti
rahva sugu.\r\n\r\nSiis kõlas kaunis lauluviis\r\nja pärjad pandi pähe.\r\nJa murueide
tütreid siis\r\nsai Eesti rahvas näha.\r\n\r\nMa laulan mättal, mäe peal\r\nja õhtu hilja
õues\r\nja Vanemuise kandle hääl,\r\nsee põksub minu põues.\r\n'
```

Tekstis leiduvate sõnade liigid

```
>>>
Text(urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt").read(
).decode("utf8")).postags
['D|J', 'H', 'S', 'A', 'S', 'D', 'V', 'D', 'V', 'Z', 'J', 'H', 'S', 'V', 'S', 'S', 'V',
'Z', 'V', 'J', 'S', 'V', 'Z', 'V', 'J', 'S', 'S', 'Z', 'D', 'S', 'S', 'J', 'S', 'J', 'S',
'S', 'Z', 'D', 'V', 'S', 'J', 'S', 'S', 'J', 'H', 'S', 'S', 'Z', 'D', 'V', 'A', 'S', 'J',
'S', 'V', 'S', 'Z', 'J', 'S', 'S', 'D', 'V', 'H', 'S', 'V', 'Z', 'P', 'V', 'S', 'Z', 'S',
'K', 'J', 'S', 'D', 'S', 'J', 'H', 'S', 'S', 'Z', 'P', 'V', 'P', 'S', 'Z']

>>>
kungladf=Text(urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.tx
t").read().decode("utf8")).get.postags.as_dataframe.groupby("postags").postags.count().to_f
rame()
>>> kungladf
         postags
postags
A              2
D              7
D|J            1
H              5
J             11
K              1
P              3
S             31
V             14
Z             11
```

Kõikidest liikidest sõnade loendamine

Ühe teksti puhul saab grupeerimise ja loendamise abil kätte, kui palju ja millisest liigist sõnu selles tekstis oli. Mitme teksti omavahelise võrdlemise puhul aga vajame, et võrreldavate liikide loetelu oleks ühesugune. Nii tasubki ette võtta eraldi failis peitub terviklik loetelu, et teaksime tekstis puuduvate liikide sagedused nulliks määrata

```
>>> import pandas as pd
>>> lyhendid=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/sonaliikide_lyhendid.txt")
>>> lyhendid
   liigilyhend            liigikirjeldus
0            A       omadussõna algvõrre
1            C      omadussõna keskvõrre
2            D                  määrsõna
3            G     käändumatu omadussõna
```

…

Kungla rahva laulust leitud sõnaliikide meeldetuletus

```
>>> kungladf.head()
         postags
postags
A              2
D              7
D|J            1
H              5
J             11
```

Sõnaliik ka indeksi kõrval tavaliseks tulbaks, et see tabelite ühendamisel ilusasti kaasa tuleks

```
>>> kungladf["sonaliik"]=kungladf.index
>>> kungladf.head()
         postags sonaliik
postags
A              2        A
D              7        D
D|J            1      D|J
H              5        H
J             11        J
```

Sisu nimele vastama panekuks nimetame tulba postags ümber koguseks

```
>>> kungladf=kungladf.rename(columns={"postags":"kogus"})
>>> kungladf.head()
         kogus sonaliik
postags
A            2        A
D            7        D
D|J          1      D|J
H            5        H
```

```
J           11        J
```

Lühendite tabeli ning Kungla rahva laulu sõnaliikide sageduse tabeli ühendamine. Et laulust puuduvad sõnaliigid alles jääks, selleks how=”left”.

```
>>> vastus1=lyhendid.merge(kungladf, how="left", left_on="liigilyhend",
right_on="sonaliik")
>>> vastus1.head()
  liigilyhend         liigikirjeldus  kogus sonaliik
0           A    omadussõna algvõrre    2.0        A
1           C   omadussõna keskvõrre    NaN      NaN
2           D               määrsõna    7.0        D
3           G  käändumatu omadussõna    NaN      NaN
4           H              pärisnimi    5.0        H
```

Asendame tühjad väärtused (NaN) nullidega ning jätame alles vaid tulbad “liigilyhend” ja “kogus”

```
>>> vastus1=vastus1.fillna(0)[["liigilyhend", "kogus"]]
>>> vastus1.head()
  liigilyhend  kogus
0           A    2.0
1           C    0.0
2           D    7.0
3           G    0.0
4           H    5.0
```

Samad operatsioonid teise tekstiga

```
>>>
laulikdf=Text(urllib.request.urlopen("http://www.tlu.ee/~jaagup/andmed/keel/eestilaulud.txt
").read().decode("utf8")).get.postags.as_dataframe.groupby("postags").postags.count().to_fr
ame()
>>> laulikdf.head()
         postags
postags
A           2069
A|V          298
C            119
D           4291
D|I           18


>>> laulikdf["sonaliik"]=laulikdf.index
>>> laulikdf=laulikdf.rename(columns={"postags": "laulikkogus"})
>>> laulikdf.head()
         laulikkogus sonaliik
postags
A               2069        A
A|V              298      A|V
C                119        C
D               4291        D
D|I               18      D|I
```

Näeme, kuivõrd on mingit tüüpi sõnu ühes laulus, kui palju terves laulikus

```
>>> vastus2=vastus1.merge(laulikdf, how="left", left_on="liigilyhend",
right_on="sonaliik").fillna(0)[["liigilyhend", "kogus", "laulikkogus"]]
>>> vastus2.head()
  liigilyhend  kogus  laulikkogus
0           A    2.0         2069
1           C    0.0          119
2           D    7.0         4291
3           G    0.0           29
4           H    5.0         1396
```

Andmed tekstifaili edasise töötlemise tarbeks

```
>>> vastus2.to_csv("vastus2.txt", index=False)
```

ja seal nad on

```
liigilyhend,kogus,laulikkogus
A,2.0,2069
C,0.0,119
D,7.0,4291
G,0.0,29
H,5.0,1396
I,0.0,393
J,11.0,1653
K,1.0,593
N,0.0,114
O,0.0,14
P,3.0,4447
S,31.0,9979
U,0.0,13
V,14.0,6621
X,0.0,10
Y,0.0,293
Z,11.0,8930
```

Vajadusel saab failist tagasi kätte ka

```
>>> vastus2=pd.read_csv("vastus2.txt")
>>> vastus2.head()
  liigilyhend  kogus  laulikkogus
0           A    2.0         2069
1           C    0.0          119
2           D    7.0         4291
3           G    0.0           29
4           H    5.0         1396
```

Kuna tekstid on märgatavalt erineva pikkusega, siis sagedusi saab võrrelda nende osakaalude kaudu - ehk siis jagada vastava sümboli sagedus tekstis olevate sümbolite üldarvuga

```
>>> vastus2.kogus.sum()
85.0
>>> vastus2.laulikkogus.sum()
```

```
40964
>>> vastus2["kunglaosakaal"]=vastus2.kogus/vastus2.kogus.sum()
>>> vastus2["laulikosakaal"]=vastus2.laulikkogus/vastus2.laulikkogus.sum()
>>> vastus2.head()
  liigilyhend  kogus  laulikkogus  kunglaosakaal  laulikosakaal
0           A    2.0         2069       0.023529       0.050508
1           C    0.0          119       0.000000       0.002905
2           D    7.0         4291       0.082353       0.104751
3           G    0.0           29       0.000000       0.000708
4           H    5.0         1396       0.058824       0.034079
```

Et mõnda sõnaliiki Kungla rahva laulus üldse ei esine, siis ohutum on esialgu osakaalud teineteisest lahutada ning leida vahe

```
>>> vastus2["osakaaluvahe"]=vastus2.kunglaosakaal-vastus2.laulikosakaal
```

Miinusmärkidega väärtuste puhul on osakaal väiksem Kungla rahva laulus, plussmärkide puhul jällegi on selles laulus vastavate sõnaliikide suhteline sagedus kogu lauliku tekstiga võrreldes suurem

```
>>> vastus2.head()
  liigilyhend  kogus  laulikkogus  kunglaosakaal  laulikosakaal  osakaaluvahe
0           A    2.0         2069       0.023529       0.050508     -0.026978
1           C    0.0          119       0.000000       0.002905     -0.002905
2           D    7.0         4291       0.082353       0.104751     -0.022398
3           G    0.0           29       0.000000       0.000708     -0.000708
4           H    5.0         1396       0.058824       0.034079      0.024745
```

Järjestades tuleb vahe paremini välja

```
>>> vastus2.sort_values("osakaaluvahe")
   liigilyhend  kogus  laulikkogus  kunglaosakaal  laulikosakaal  osakaaluvahe
16           Z   11.0         8930       0.129412       0.217996     -0.088585
10           P    3.0         4447       0.035294       0.108559     -0.073265
0            A    2.0         2069       0.023529       0.050508     -0.026978
2            D    7.0         4291       0.082353       0.104751     -0.022398
5            I    0.0          393       0.000000       0.009594     -0.009594
15           Y    0.0          293       0.000000       0.007153     -0.007153
1            C    0.0          119       0.000000       0.002905     -0.002905
8            N    0.0          114       0.000000       0.002783     -0.002783
7            K    1.0          593       0.011765       0.014476     -0.002711
3            G    0.0           29       0.000000       0.000708     -0.000708
9            O    0.0           14       0.000000       0.000342     -0.000342
12           U    0.0           13       0.000000       0.000317     -0.000317
14           X    0.0           10       0.000000       0.000244     -0.000244
13           V   14.0         6621       0.164706       0.161630      0.003076
4            H    5.0         1396       0.058824       0.034079      0.024745
6            J   11.0         1653       0.129412       0.040353      0.089059
11           S   31.0         9979       0.364706       0.243604      0.121102
```

Vaatan vahede järjestuses esimest kolme

```
>>> vastus2.sort_values("osakaaluvahe").head(3)
   liigilyhend  kogus  laulikkogus  kunglaosakaal  laulikosakaal  osakaaluvahe
```

```
16           Z   11.0         8930       0.129412       0.217996     -0.088585
10           P    3.0         4447       0.035294       0.108559     -0.073265
0            A    2.0         2069       0.023529       0.050508     -0.026978
```

Kirjavahemärke on Kungla rahva laulus 13%, suures laulikus 22%, vahe 9 protsendipunkti või 1,7 korda. Asesõnade (P) vahe 8 protsendipunkti Kungla rahva kahjuks, omadussõnade algvõrdel (A) 2,5 protsendipunkti

```
>>> vastus2.sort_values("osakaaluvahe").tail(3)
   liigilyhend  kogus  laulikkogus  kunglaosakaal  laulikosakaal  osakaaluvahe
4            H    5.0         1396       0.058824       0.034079      0.024745
6            J   11.0         1653       0.129412       0.040353      0.089059
11           S   31.0         9979       0.364706       0.243604      0.121102
```

Pärisnimesid (H), sidesõnu (J) ning nimisõnu (S) on jällegi Kungla rahva laulus tuntavalt rohkem kui suures laulikus üldiselt.

Erinevust saab vaadata ka sageduste osakaalu suhte abil

```
>>> vastus2["osakaalusuhe"]=vastus2.kunglaosakaal/vastus2.laulikosakaal
```

Esimese laulu lühiduse tõttu ei saa puuduvate sõnaliikide järgi kuigivõrd otsustada. Samas suuremate sageduste puhul (kirjavahemärk ehk Z 11 korda) pea kahekordne vahe on juba täiesti märgatav. Samuti jääb silma sidesõnade 3,2 korda suurem osakaal Kungla rahva laulus

```
 vastus2.sort_values("osakaalusuhe")[["liigilyhend", "kogus", "osakaalusuhe"]]
   liigilyhend  kogus  osakaalusuhe
8            N    0.0      0.000000
1            C    0.0      0.000000
14           X    0.0      0.000000
3            G    0.0      0.000000
12           U    0.0      0.000000
5            I    0.0      0.000000
9            O    0.0      0.000000
15           Y    0.0      0.000000
10           P    3.0      0.325115
0            A    2.0      0.465857
16           Z   11.0      0.593642
2            D    7.0      0.786182
7            K    1.0      0.812697
13           V   14.0      1.019032
11           S   31.0      1.497125
4            H    5.0      1.726108
6            J   11.0      3.207032
```

Harjutus

- Võtke samad kaks teksti (Kungla rahvas + laulik). Tooge välja tähtede sageduste osakaalud ja nende erinevused

Abiks paar katsetust DataFrame tegemisel. Käsuga list saab kätte üksikud tähed tekstis

```
>>> list("Tere")
['T', 'e', 'r', 'e']
```

Kui list ette anda DataFrame-le, siis paigutatakse andmed tulpa numbriga 0

```
>>> pd.DataFrame(['a', 'b'])
   0
0  a
1  b
```

Kaks käsku koos

```
>>> tahed1=pd.DataFrame(list("Tere"))
>>> tahed1
   0
0  T
1  e
2  r
3  e
```

Mugavamaks pöördumiseks saab tulba nime ümber muuta

```
>>> tahed1.columns=[["tahed"]]
>>> tahed1
  tahed
0     T
1     e
2     r
3     e
```

ja siis kokku lugeda, millist tähte kui mitu korda esines

```
>>> tahed1.groupby("tahed").tahed.count()
tahed
T    1
e    2
r    1
Name: tahed, dtype: int64
```

Tulbale aga saab ka kohe nime anda

```
>>> pd.DataFrame({"tahed": list("Tere")})
  tahed
0     T
1     e
```

```
2     r
3     e
```

Kui ei soovita eristada suur-ja väiketähti, siis

```
>>> "Tere".lower()
'tere'
```

Lahenduse loomine failis.

```
jaagup@praktika1 ~/public_html/2018/dt/12nltk $ more tahevordleja1.py
import urllib.request
import pandas as pd
aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()
df=pd.DataFrame({"tahed":list(tekst)})
print(df.head())
df2=df.groupby("tahed").tahed.count().to_frame()
print(df2.sort_values("tahed", ascending=False).head(10))
```

Kõigepealt faili tähed eraldi ridadel ning edasi järjestatuna kokku loetuna, et mitu korda milline täht esines

```
jaagup@praktika1 ~/public_html/2018/dt/12nltk $ python3.5 tahevordleja1.py
  tahed
0     k
1     u
2     i
3
4     k
       tahed
tahed
          57
a         56
s         35
u         34
l         33
e         29
i         27
\n        22
\r        22
m         18
```

Juurde osakaalu arvutamine

```
jaagup@praktika1 ~/public_html/2018/dt/12nltk $ more tahevordleja2.py
import urllib.request
import pandas as pd

aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()
```

```
df=pd.DataFrame({"tahed":list(tekst)})
df2=df.groupby("tahed").tahed.count().to_frame()
df2["symbol"]=df2.index
df2["osakaal"]=df2.tahed/df2.tahed.sum()
print(df2.sort_values("osakaal", ascending=False).head(10))
```

ja nähtav tulemus

```
jaagup@praktika1 ~/public_html/2018/dt/12nltk $ python3.5 tahevordleja2.py
       tahed symbol   osakaal
tahed
           57         0.121535
a          56      a  0.119403
s          35      s  0.074627
u          34      u  0.072495
l          33      l  0.070362
e          29      e  0.061834
i          27      i  0.057569
\n         22     \n  0.046908
\r         22     \r  0.046908
m          18      m  0.038380
```

Edasi juba kahe faili sisu võrdlemine

```
import urllib.request
import pandas as pd

aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()
df=pd.DataFrame({"tahed":list(tekst)})
df2=df.groupby("tahed").tahed.count().to_frame()
df2["symbol"]=df2.index
df2["osakaal"]=df2.tahed/df2.tahed.sum()
#print(df2.sort_values("osakaal", ascending=False).head(10))

aadress="http://www.tlu.ee/~jaagup/andmed/keel/eestilaulud.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()
df=pd.DataFrame({"laulutahed":list(tekst)})
df3=df.groupby("laulutahed").laulutahed.count().to_frame()
df3["symbol"]=df3.index
df3["lauluosakaal"]=df3.laulutahed/df3.laulutahed.sum()
#print(df3.sort_values("lauluosakaal", ascending=False).head(10))

koos=df2.merge(df3, left_on="symbol", right_on="symbol")
koos["suhe"]=koos.osakaal/koos.lauluosakaal
print(koos.sort_values("suhe"))
```

Tavalise (inner) ühendamise korral jäävad alles vaid sümbolid, mis olemas mõlemis tabelis. Eesotsas sümbolid, mida Kungla rahva laulus märgatavalt vähem

```
    tahed symbol   osakaal  laulutahed  lauluosakaal       suhe
7       1      b  0.002132        1801      0.008248   0.258523
28      1      ü  0.002132        1756      0.008042   0.265148
18      3      o  0.006397        4928      0.022568   0.283441
3       4      ,  0.008529        5077      0.023250   0.366830
22     11      t  0.023454        8765      0.040139   0.584323
```

```
14      12      k  0.025586        8392      0.038431   0.665776
17      14      n  0.029851        8454      0.038715   0.771042
20       9      r  0.019190        5144      0.023557   0.814617
12      27      i  0.057569       15173      0.069484   0.828523
8       12      d  0.025586        6462      0.029593   0.864622
26       5      õ  0.010661        2690      0.012319   0.865426
9       29      e  0.061834       15561      0.071261   0.867706
24       7      v  0.014925        3743      0.017141   0.870744
4        6      .  0.012793        2987      0.013679   0.935251
2       57         0.121535       28333      0.129750   0.936687
16      18      m  0.038380        7100      0.032514   1.180392
19       7      p  0.014925        2743      0.012561   1.188186
21      35      s  0.074627       13455      0.061617   1.211146
1       22     \r  0.046908        8388      0.038413   1.221171
0       22     \n  0.046908        8388      0.038413   1.221171
6       56      a  0.119403       20836      0.095418   1.251370
15      33      l  0.070362       11503      0.052678   1.335719
23      34      u  0.072495       10219      0.046798   1.549112
13      10      j  0.021322        2973      0.013615   1.566092
10       8      g  0.017058        2376      0.010881   1.567674
11      10      h  0.021322        2865      0.013120   1.625128
25      11      ä  0.023454        2741      0.012552   1.868512
27       4      ö  0.008529         651      0.002981   2.860824
5        1      ;  0.002132          45      0.000206  10.346648
```

Ning lõpus sellised, mida jälle märgatavalt rohkem

Tähepaarid

Lisaks üksikutele tähtedele, sõnadele, sõnaliikidele, käänetele ja muudele väärtustele saab tekste eristada järgnevusi võrreldes, praeguse lihtsaima näite juures siis sümbolite paare arvestades. Sisendiks juba tuttav Kungla rahva laulu tekst üle võrgu sisse loetuna ning väiketähtedeks tehtuna

```
>>> import urllib.request
>>>
>>> aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
>>> tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()

>>> tekst[0:10]
'kui kungla'
```

Paar katsetust arvudega. Käsklus range annab välja soovitud arvuvahemiku. Selle ilmutatud kujul kuvamiseks tuleb aga lisada käsklus list.

```
>>> range(10)
range(0, 10)
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Arvuvahemikku saab kasutada uue massiivi loomisel. Ruutuvõtmise tehte kaudu tuleb algsest järjekorranumbrite loetelust arvude ruutude loetelu.

```
>>> [arv*arv for arv in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Samal moel õnnestub ka tekstist tähepaare välja võtta. Näitena tähepaarid laulusõnade teksti esimesest kümnest tähest. Paare mahub sinna ühe võrra vähem - nii ka miinusmärk tehte juures

```
>>> [tekst[arv:arv+2] for arv in range(10-1)]
['ku', 'ui', 'i ', ' k', 'ku', 'un', 'ng', 'gl', 'la']
```

Veidi üldisemal kujul kirja pannes saab jada pikkuse ette määrata ning selle järgi arvutada.

```
>>> pikkus=3
>>> [tekst[arv:arv+pikkus] for arv in range(10-(pikkus-1))]
['kui', 'ui ', 'i k', ' ku', 'kun', 'ung', 'ngl', 'gla']
```

Paarid kogu tekstist

```
>>> pikkus=2
>>> paarid=[tekst[arv:arv+pikkus] for arv in range(len(tekst)-(pikkus-1))]
```

Neist esimesed ekraanile

```
>>> paarid[0:5]
['ku', 'ui', 'i ', ' k', 'ku']
>>> len(paarid)
468
```

Abivahendiks Pandas

Pythoni “tavavahenditega” saab enamiku arvutusi ette võtta. Pandas-paketi DataFrame klassi sisse pandud käsud aga aitavad andmestikuga lisaks mitmekülgselt toimetada. Massiivi paarid DataFrame sees kasutatavaks tegemiseks tuleb tulemus muuta tabeliks, kus praegusel juhul on vaid üks tulp, nimeks sai sellele “paar”.

```
>>> import pandas as pd
>>> df=pd.DataFrame({"paar":paarid})
```

```
>>> df2=df.groupby("paar").paar.count().to_frame()
>>> df2.head()
      paar
paar
\n\r     4
\nj      6
\nk      1
\nl      3
\nm      1
```

```
>>> df2.sort_values("paar", ascending=False).head(10)
      paar
paar
\r\n    22
a       16
s       13
 l      11
ja      10
 m      10
la       9
e        9
is       9
ma       7

>>> len(df2)
167
>>>
```

Edasine analüüs sarnane kui varem

```
>>> tekst[0:10]
'kui kungla'
>>>
>>> from estnltk import Text
>>> t=Text(tekst)
>>> t.postags[0:10]
['d|j', 's', 's', 'a', 's', 'd', 'v', 'd', 'v', 'z']
>>> 'D|J'.split('|')
['d', 'j']
>>> 'S'.split('|')
['s']
>>> 'D|J'.split('|')[0]
'd'
>>> [ x.split('|')[0] for x in t.postags[0:10]]
['d', 's', 's', 'a', 's', 'd', 'v', 'd', 'v', 'z']

>>> sonaliigid=[ x.split('|')[0] for x in t.postags]
>>> sonaliigid[:5]
['d', 's', 's', 'a', 's']
>>> sonaliigid[0:5]
['d', 's', 's', 'a', 's']


>>> paarid=[sonaliigid[koht:koht+2] for koht in range(len(sonaliigid)-1)]
>>> paarid[0:10]
[['d', 's'], ['s', 's'], ['s', 'a'], ['a', 's'], ['s', 'd'], ['d', 'v'], ['v', 'd'], ['d',
'v'], ['v', 'z'], ['z', 'j']]

>>> paarid=["".join(sonaliigid[koht:koht+2]) for koht in range(len(sonaliigid)-1)]
>>> paarid[0:10]
['ds', 'ss', 'sa', 'as', 'sd', 'dv', 'vd', 'dv', 'vz', 'zj']
>>>
```

- Kolm levinumat paari

```
>>> pd.DataFrame({"paar": paarid}).groupby("paar").paar.count().to_frame().
               sort_values("paar", ascending=False).head(3)
```

paar

paar

SS 11
JS 10
SZ 7

Andmed veebilehel

Alustuseks lihtne veebileht oma struktuuriga. HTML-keeles määratakse märgendite abil, et millised tekstid kus ja kuidas näidatakse. Üldjuhul iga alustav märgend kusagil ka lõpeb. Nt. kogu dokumenti alustav <html> lõpeb </html> juures. Sama lugu ka dokumendi struktuuri kahe suurema elemendi - head ja body puhul. Esimeses neist andmed lehe sisu kohta, teises sisu ise. Mõni element aga piirdubki ainult ühe märgendiga - näiteks kooditabelit määrav `<meta charset="utf-8" />` lõpeb samas kus algab, selleks pannakse viisakusest (ja XML-standardiga sobitumiseks) kaldkriips märgendi enese lõppu.

```
<!doctype html>
<html>
   <head>
      <title>Tekstide võrdlemise leht</title>
      <meta charset="utf-8" />
   </head>
   <body>
      <h1>Tekstide võrdlemine</h1>

   </body>
</html>
```

Paigutades faili veebis kättesaadavasse kataloogi näeb kujundatud tulemust. Samuti võimalik faili brauseriga kohaliku masina kataloogis vaadata.

Lehe sisse loetelu näide. Tahtes laulude nimed saada nummerdamata loetellu (unordered list), aitavad meid elemendid ul ja li (list item). Iga loetelu sees siis sobivad elemendid

```
<h2>Tekstid</h2>
```

```
<ul>
  <li>Kungla rahvas</li>
  <li>Lambipirni anekdoot</li>
</ul>
```

Tekstid

- Kungla rahvas
- Lambipirni anekdoot

Kui teksti osa tahetakse viitama panna, tuleb abiks element nimega a (anchor) koos parameetriga href (hyperlink reference). Elemendi piirkonnas olevale tekstile vajutades avaneb viidatud lehekülg.

```
  <h2>Tekstid</h2>
  <ul>
    <li><a href="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt">
            Kungla rahvas</a></li>
   <li><a href="http://www.tlu.ee/~jaagup/andmed/keel/lambipirn.txt">
            Lambipirni anekdoot</a></li>
  </ul>
```

Tekstid

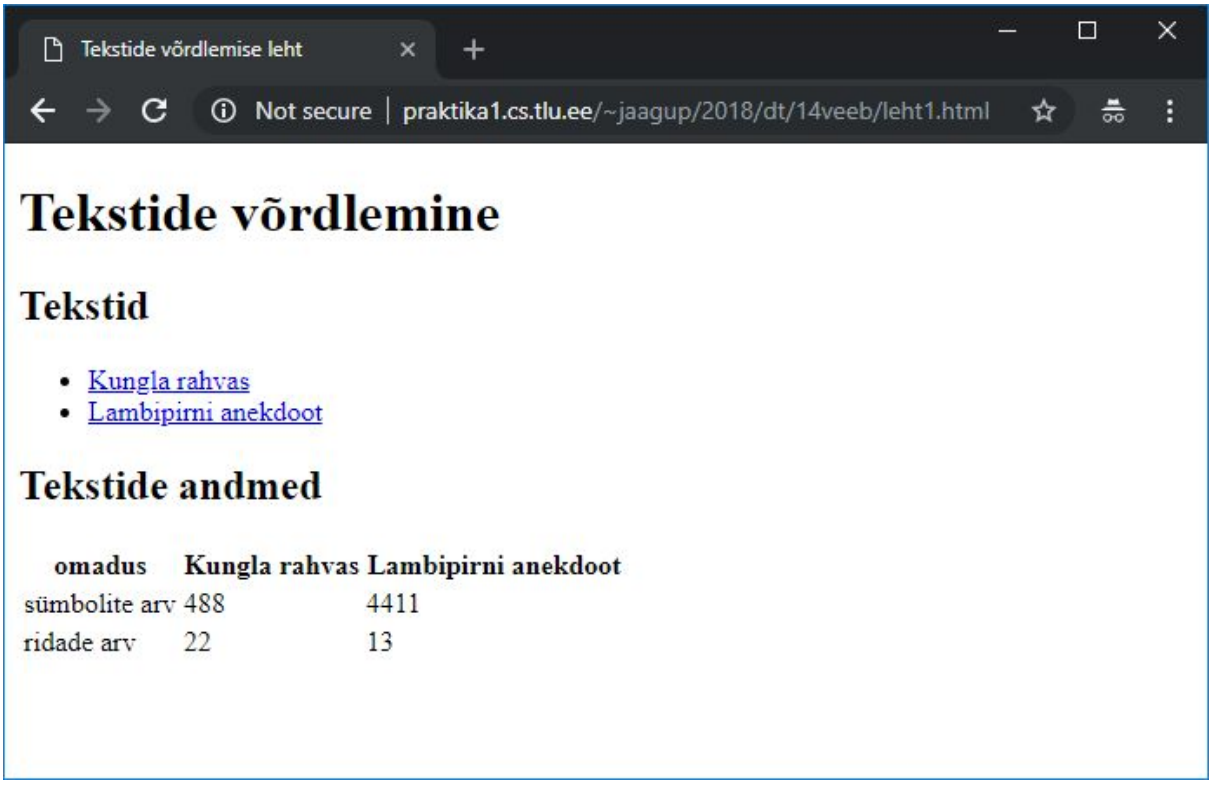
- Kungla rahvas
- Lambipirni anekdoot

Tabelikujul andmete esitamiseks on küllalt mugav HTMLi vastav element. Algusesse ja lõppu märgendid <table> ning </table>. Iga rea algusesse ja lõppu <tr> ning </tr> (table row). Pealkirjalahtrite elementideks <th></th> (table head) ning sisulahtriteks <td></td> (table data).

```
<!doctype html>
<html>
   <head>
      <title>Tekstide võrdlemise leht</title>
         <meta charset="utf-8" />
   </head>
   <body>
      <h1>Tekstide võrdlemine</h1>

      <h2>Tekstid</h2>
         <ul>
            <li><a href="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt">Kungla
rahvas</a></li>
               <li><a
href="http://www.tlu.ee/~jaagup/andmed/keel/lambipirn.txt">Lambipirni anekdoot</a></li>
         </ul>

         <h2>Tekstide andmed</h2>
         <table>
           <tr>
                <th>omadus</th>
                <th>Kungla rahvas</th>
                <th>Lambipirni anekdoot</th>
```

```
            </tr>
            <tr>
              <td>sümbolite arv</td>
              <td>488</td>
              <td>4411</td>
            </tr>
            <tr>
              <td>ridade arv</td>
              <td>22</td>
              <td>13</td>
            </tr>
          </table>
   </body>
</html>
```

Andmed tabelina nähtavad veebilehel



Veebilehe loomine programmikoodi abil

Ühekordselt saab lehe käsitsi valmis kirjutada. Kui aga on vaja luua ülevaadet korduvalt, siis on mugavam, kui programmilõik selle töö meie eest ära teeb. Esialgu tundub ehk imelik kirjutada ühes programmeerimiskeeles teise programmeerimiskeele teksti, aga seda võimalust pruugitakse mitmel pool.

Alustuseks koodinäide, mis väljastab lihtsalt pealkirjaga HTML-lehe

Avatakse fail, kirjutatakse read sisse ning lõpuks suletakse. Sulgemiskäsu puudumisel võib juhtuda, et teelepandud tekst ei jõua kettale, nüüd aga see käsklus programmi lõpus ilusasti olemas

loomine1.py

```
f=open("kirjeldus1.html", "w")
f.write("<!doctype html>\n")
f.write("<html>\n")
f.write("  <head>\n")
f.write("    <title>Tekstide kirjeldus</title>\n")
f.write("  </head>\n")
f.write("  <body>\n")
f.write("    <h1>Kungla rahvas</h1>\n")
f.write("  </body>\n")
f.write("</html>\n")
f.close()
```

Programm käima

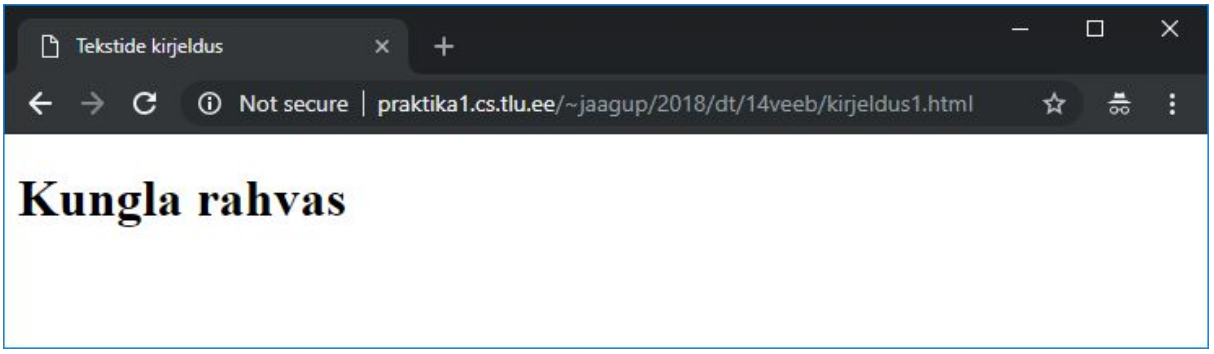
jaagup@praktika1 ~/public_html/2018/dt/14veeb $ python3.5 loomine1.py

ning saab loodud faili sisu vaadata. Tekstina

jaagup@praktika1 ~/public_html/2018/dt/14veeb $ more kirjeldus1.html

```
<!doctype html>
<html>
  <head>
    <title>Tekstide kirjeldus</title>
  </head>
  <body>
    <h1>Kungla rahvas</h1>
  </body>
</html>
```

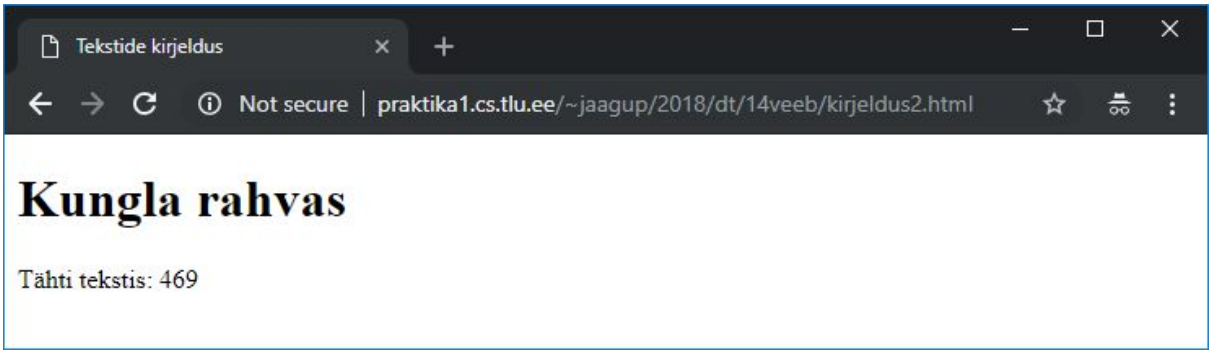
ja graafiliselt



Uuritava teksti andmed veebilehel

Näitena siis varasemate oskuste ühendus. Kõigepealt loetakse veebist laulusalmi andmed sisse. Loodava veebilehe keskele kirjutatakse teksti tähtede arv.

```
import urllib.request

aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()


f=open("kirjeldus2.html", "w")
f.write("<!doctype html>\n")
f.write("<html>\n")
f.write("  <head>\n")
f.write("    <title>Tekstide kirjeldus</title>\n")
f.write("    <meta charset='utf8' />\n")
f.write("  </head>\n")
f.write("  <body>\n")
f.write("    <h1>Kungla rahvas</h1>\n")
f.write("     Tähti tekstis: "+str(len(tekst))+"\n")
f.write("  </body>\n")
f.write("</html>\n")
f.close()
```



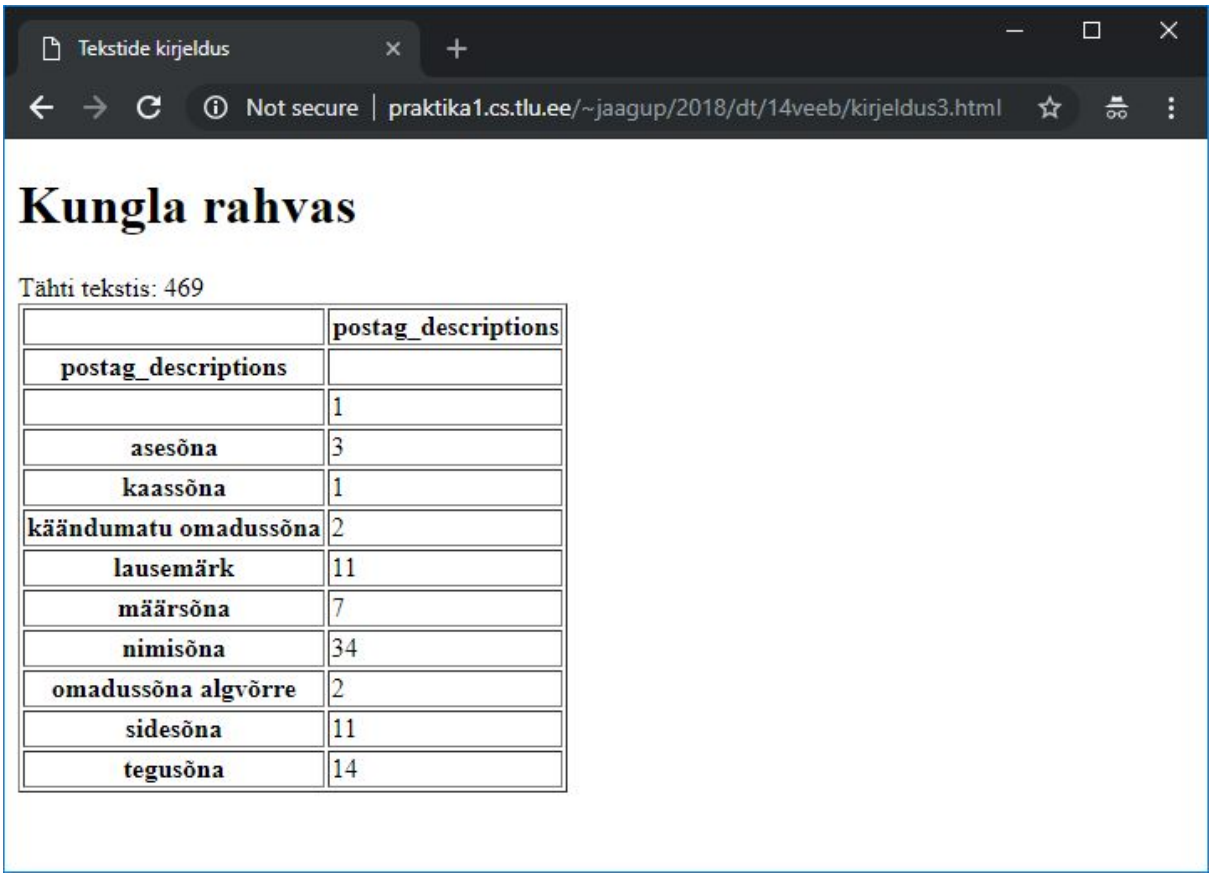
DataFrame veebilehel tabelina

Pandase paketi DataFramel on küljes hulgem käsklusi. Üks neist võimaldab ta sisu mugavalt muuta HTMLi tabeliks. Piisab objektile lihtsalt anda käsklus to_html() ja see siis sobivasse kohta trükkida. Enne loetakse kokku, et mitu korda milline sõnaliik tekstis esines.

```
import urllib.request

aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()

from estnltk import Text
t=Text(tekst)
df=t.get.postag_descriptions.as_dataframe.groupby("postag_descriptions").postag_description
s.count().to_frame()

f=open("kirjeldus3.html", "w")
f.write("<!doctype html>\n")
```

```
f.write("<html>\n")
f.write("  <head>\n")
f.write("    <title>Tekstide kirjeldus</title>\n")
f.write("    <meta charset='utf8' />\n")
f.write("  </head>\n")
f.write("  <body>\n")
f.write("    <h1>Kungla rahvas</h1>\n")
f.write("     Tähti tekstis: "+str(len(tekst))+"\n")
f.write(df.to_html())
f.write("  </body>\n")
f.write("</html>\n")
f.close()
```

Leht veebilehitsejas nähtav



Kungla rahvas

Tähti tekstis: 469

	postag_descriptions
postag_descriptions	
	1
asesõna	3
kaassõna	1
käändumatu omadussõna	2
lausemärk	11
määrsõna	7
nimisõna	34
omadussõna algvõrre	2
sidesõna	11
tegusõna	14

ning valminud HTML ka täiesti loetav

```
<!doctype html>
<html>
  <head>
    <title>Tekstide kirjeldus</title>
    <meta charset='utf8' />
  </head>
  <body>
    <h1>Kungla rahvas</h1>
     Tähti tekstis: 469
<table border="1" class="dataframe">
  <thead>
    <tr style="text-align: right;">
      <th></th>
```

```
      <th>postag_descriptions</th>
    </tr>
    <tr>
      <th>postag_descriptions</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th></th>
      <td>1</td>
    </tr>
    <tr>
      <th>asesõna</th>
      <td>3</td>
    </tr>
    <tr>
      <th>kaassõna</th>
      <td>1</td>
    </tr>
    <tr>
      <th>käändumatu omadussõna</th>
      <td>2</td>
    </tr>
    <tr>
      <th>lausemärk</th>
      <td>11</td>
    </tr>
    <tr>
      <th>määrsõna</th>
      <td>7</td>
    </tr>
    <tr>
      <th>nimisõna</th>
      <td>34</td>
    </tr>
    <tr>
      <th>omadussõna algvõrre</th>
      <td>2</td>
    </tr>
    <tr>
      <th>sidesõna</th>
      <td>11</td>
    </tr>
    <tr>
      <th>tegusõna</th>
      <td>14</td>
    </tr>
  </tbody>
</table>  </body>
</html>
```

Valitud andmed veebilehele

Ennist kuvati tabel tervikuna välja. Selline valmiskäsklus on mugav senikaua, kuni meil just tema väljundit on vaja. Kui aga tuleb andmestikust valida omale vajalikke väärtusi ning neid soovitud kujul esitada, siis tuleb oma koodiga rohkem andmetega tegeleda.

DataFramest võtme järgi üksikute väärtuste välja võtmiseks aitab käsklus “at”

```
f.write(str(df.at["lausemärk", "postag_descriptions"])+", \n")
```

kirjutab näiteks tulba “postag_descriptions” väärtuse reast indeksi väärtusega “lausemärk” - ehk siis mitu lausemärki uuritud tekstis oli.

Järjekorranumbri järgi indekstulba väärtuse lugemiseks saab indeksi seest kandiliste sulgudega välja küsida vajaliku järjekorranumbriga elemendi, lugemine algab nullist. Tulpadest järjekorranumbri järgi välja võtmiseks sobib massiiv “iat”. Praegusel juhul siis väärtus veerust number 0 (indeksiveerg on eraldi arvestusega) ning realt number 0 ehk kõige ülemisest reast.

```
f.write("levinumaid: "+sortdf.index[0]+" - " + str(sortdf.iat[0, 0])+"<br />\n")
```

Ridade arv tabelis on sama, mis ridade arv indeksis

```
f.write("tüüpe kokku: "+str(len(sortdf.index))+"<br />\n")
```

Ridade ükshaaval läbi käimiseks sobib kordus. Pythonis on mugavalt kasutatav for-tsükkel. Käsklus range tekitab massiivi kõikidest arvudest nullist kuni etteantud arv miinus üheni (ehk sama palju elemente, kui see arv näitab), for-tsükli juurde antud muutuja käib need kõik ükshaaval läbi ning tsükli sisuosas saab seda järjekorranumbrit (nr) kasutada. Võetakse nii indeksist kui ainukesest tulbast vastav väärtus ja kuvatakse ekraanile.

```
for nr in range(len(sortdf.index)):
   f.write(sortdf.index[nr]+" - "+str(sortdf.iat[nr , 0])+"<br />\n")
```

Kood tervikuna

```
import urllib.request

aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()

from estnltk import Text
t=Text(tekst)
df=t.get.postag_descriptions.as_dataframe.groupby("postag_descriptions").postag_description
s.count().to_frame()
sortdf=df.sort_values("postag_descriptions", ascending=False)

f=open("kirjeldus4.html", "w")
f.write("<!doctype html>\n")
f.write("<html>\n")
f.write("  <head>\n")
f.write("    <title>Tekstide kirjeldus</title>\n")
f.write("    <meta charset='utf8' />\n")
```

```
f.write("  </head>\n")
f.write("  <body>\n")
f.write("    <h1>Kungla rahvas</h1>\n")
f.write("     Lausemärke: ")
f.write(str(df.at["lausemärk", "postag_descriptions"])+", \n")
f.write("levinumaid: "+sortdf.index[0]+" - " + str(sortdf.iat[0, 0])+"<br />\n")
f.write("tüüpe kokku: "+str(len(sortdf.index))+"<br />\n")
for nr in range(len(sortdf.index)):
   f.write(sortdf.index[nr]+" - "+str(sortdf.iat[nr , 0])+"<br />\n")
f.write("  </body>\n")
f.write("</html>\n")
f.close()
```

Valminud veebileht

ja selle kood

```
<!doctype html>
<html>
<head>
<title>Tekstide kirjeldus</title>
<meta charset='utf8' />
</head>
<body>
<h1>Kungla rahvas</h1>
Lausemärke: 11,
levinumaid: nimisõna - 34<br />
tüüpe kokku: 10<br />
nimisõna - 34<br />
tegusõna - 14<br />
lausemärk - 11<br />
sidesõna - 11<br />
määrsõna - 7<br />
asesõna - 3<br />
käändumatu omadussõna - 2<br />
omadussõna algvõrre - 2<br />
```

```
- 1<br />
kaassõna - 1<br />
</body>
</html>
```

Nagu näha, siis ühe sõna puhul on liik määramata ning selle puhul näidatakse, et sedagi on üks

Joonised

Pilt pidi vahel rääkima rohkem kui tuhat sõna. Üksiku joonise saab tabelarvutuskeskkonna abil ehk programmikoodist mugavamalt valmis. Igapäevaselt muutuvaid andmeid veebilehel esitades aga kuluvad jooniste koostamise programmikäsud igati ära. Siin näidetes kasutatakse Pythoni teeki matplotlib, koodilõigud käivitatakse käsurealt ning loodud joonised salvestatakse faili.

Mälupuhvris pilte hoida aitab Agg - siis saab need sealt hiljem faili salvestada. Mugavamaks ligipääsuks antakse matplotlib.pyplot teegi juurde pöördumiseks nimi plt - selline on sarnaste Pythoni rakenduste juures välja kujunenud tava.

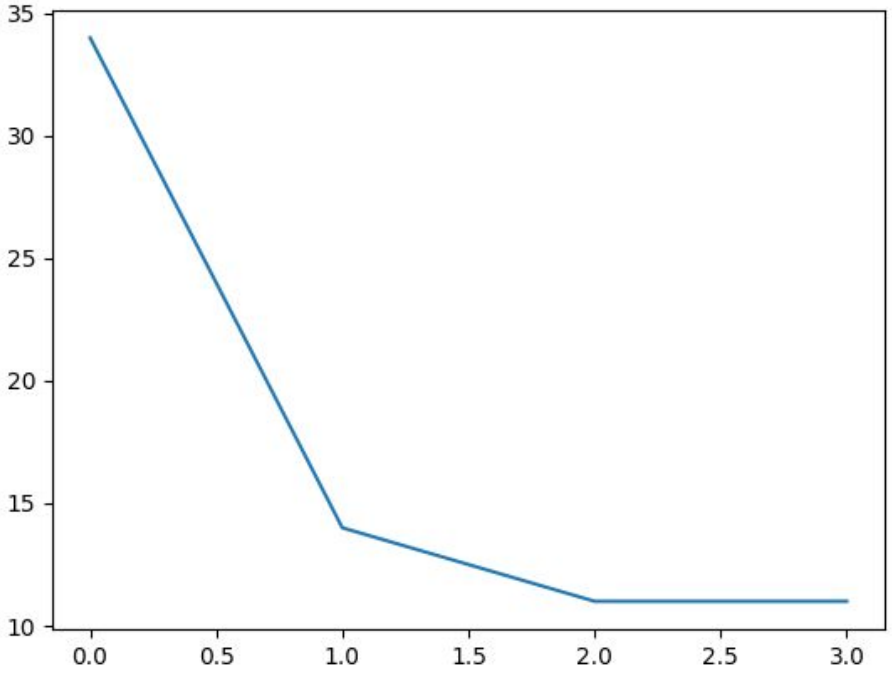
Tavaline plot-käsk loob massiivina ette antud arvudest joondiagrammi, mis siis järgmise savefig-käsuga faili salvestatakse.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

plt.plot([34, 14, 11, 11])
plt.savefig("joonis1.png")
```

Koodilõik käsurealt tööle

```
jaagup@praktika1 ~/public_html/2018/dt/15joonised $ python3.5 joonis1.py
```

ning võibki failis olevat joonist imetleda. Olgu veebilehte vaadates või muul moel.



Selgitustekstidega joonis

Sobivad sõnad joonisel õigetes kohtades aitavad selgemini kujutatut mõista. Järgnevalt eelnev joonis koos mõningate täiendustega.

```
plt.xticks([0, 1, 2, 3], ["nimisõna", "tegusõna", "sidesõna", "lausemärk"])
```

teatab, et x-telje kohtadele 0, 1, 2 ja 3 kirjutada selgituseks järgnevast massiivist tulevad sõnad.

Edasi tekstid telgedele ning joonise pealkiri

```
plt.xlabel("sõnaliik")
plt.ylabel("kogus")
plt.title("Sõnaliikide kogused")
```

Tekst joonise sees sobivasse kohta. Koordinaateljestik ikka sama - ehk siis praegusel juhul alustan teksti kohalt x=1.2 ning y=8

```
plt.text(1.2, 8, "sidesõnu ja lausemärke on ühepalju")
```

Vaikimisi kipub matplotlib telgede vahemiku näitama nii, et andmete erisus selgelt välja tuleb. Kuna tahan säilitada kasutajale võrdlust koguväärtuse ulatuses, siis määran, et y-telg algab nullist. Ka teistele suurustele siis silma järgi sobivad väärtused.

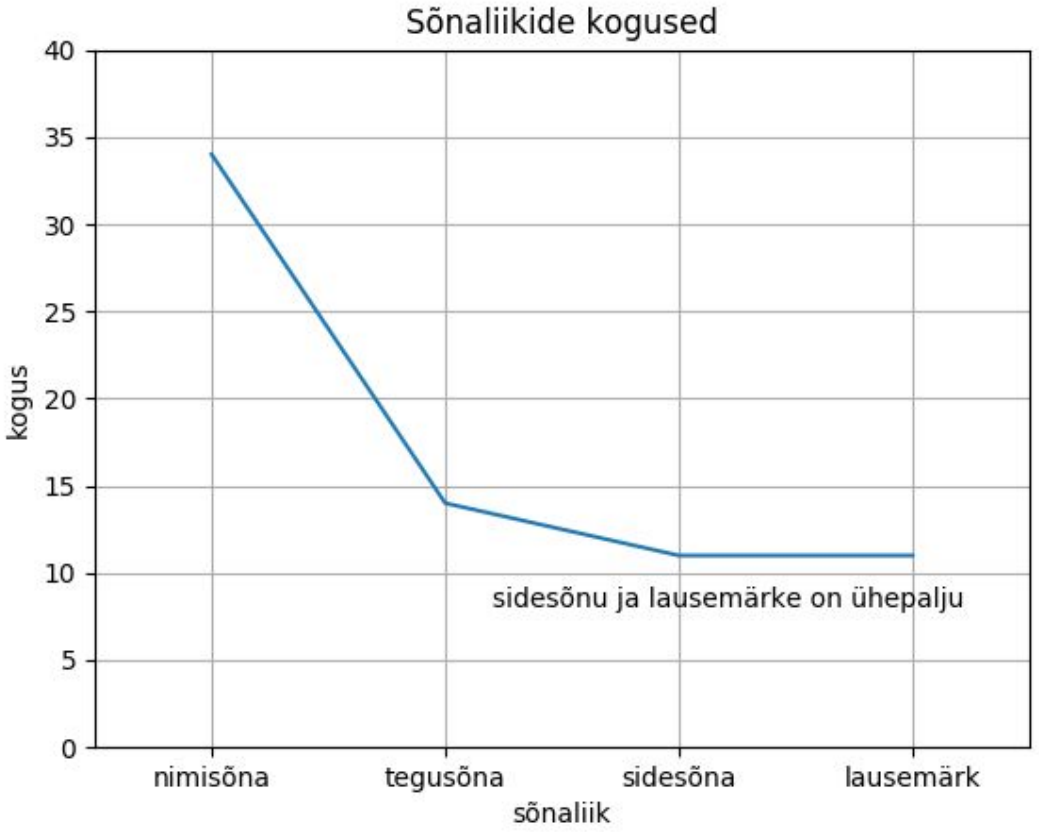
```
plt.axis([-0.5, 3.5, 0, 40]) #xmin, xmax, ymin, ymax
```

Joonise mugavamaks lugemiseks ruudustikukujulised abijooned

```
plt.grid(True)
```

Kood tervikuna

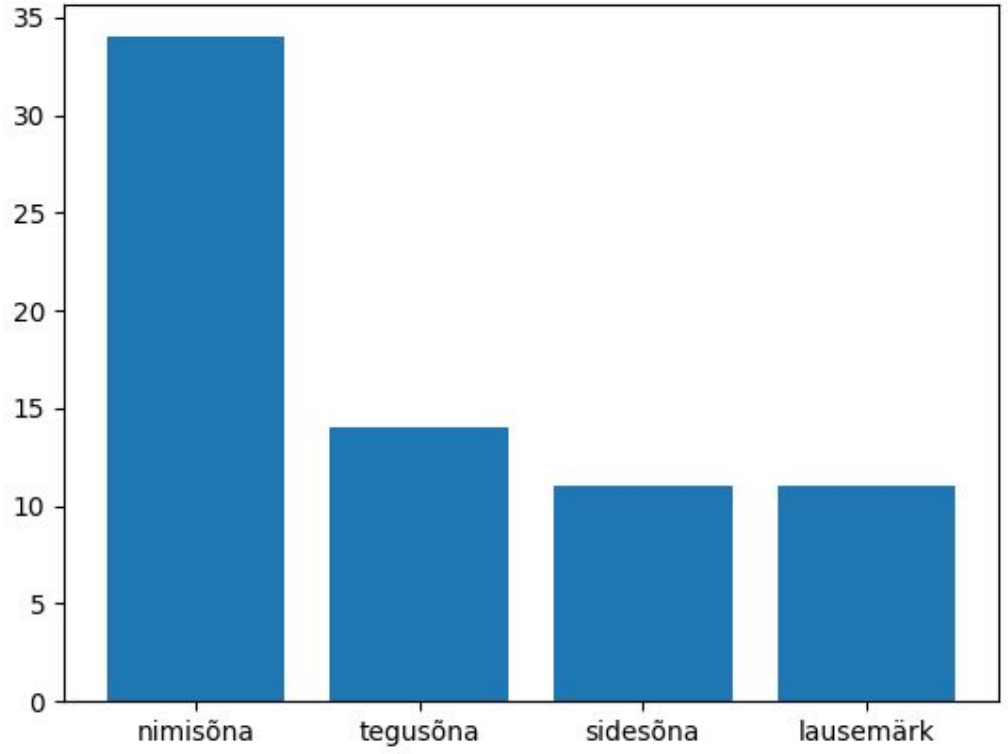
```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

plt.plot([34, 14, 11, 11])
plt.xticks([0, 1, 2, 3], ["nimisõna", "tegusõna", "sidesõna", "lausemärk"])
plt.xlabel("sõnaliik")
plt.ylabel("kogus")
plt.title("Sõnaliikide kogused")
plt.text(1.2, 8, "sidesõnu ja lausemärke on ühepalju")
plt.axis([-0.5, 3.5, 0, 40]) #xmin, xmax, ymin, ymax
plt.grid(True)
plt.savefig("joonis2.png")
```



Tulpdiagramm

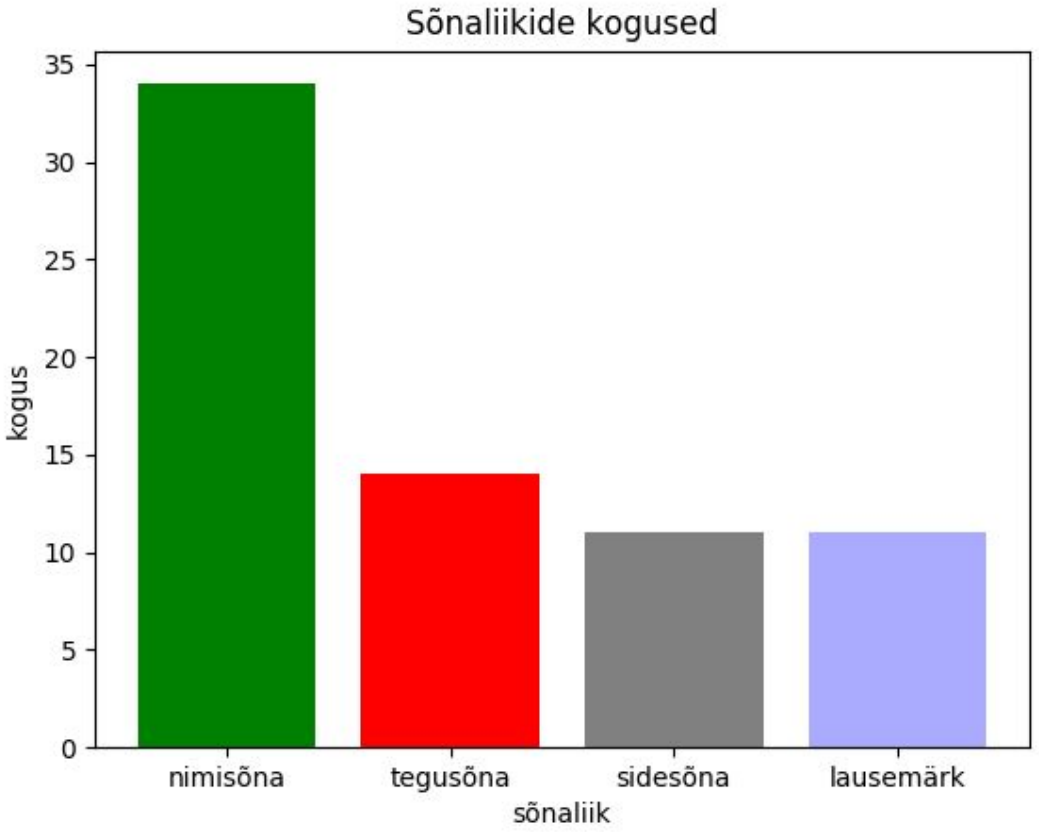
Võrreldes eelmisega käsuks bar.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

plt.bar([0, 1, 2, 3], [34, 14, 11, 11])
plt.xticks([0, 1, 2, 3], ["nimisõna", "tegusõna", "sidesõna", "lausemärk"])
plt.savefig("joonis3.png")
```



Juurde tulpade värvid ning eelnevale sarnaselt selgitustekstid

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
plt.bar([0, 1, 2, 3], [34, 14, 11, 11],
   color=["green", "red", "gray", "#AAAAFF"])
plt.xticks([0, 1, 2, 3], ["nimisõna", "tegusõna", "sidesõna", "lausemärk"])
plt.xlabel("sõnaliik")
plt.ylabel("kogus")
plt.title("Sõnaliikide kogused")
plt.savefig("joonis3.png")
```

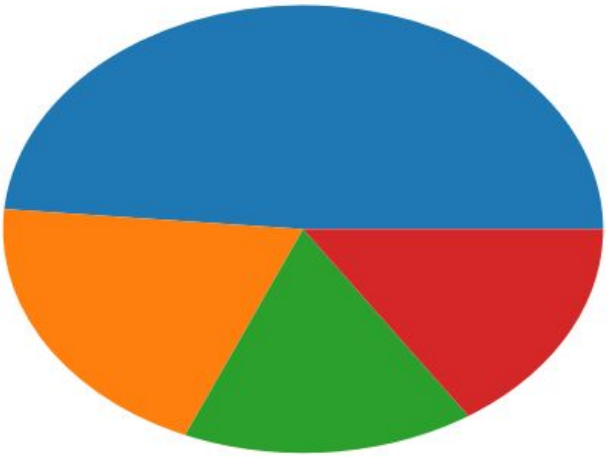


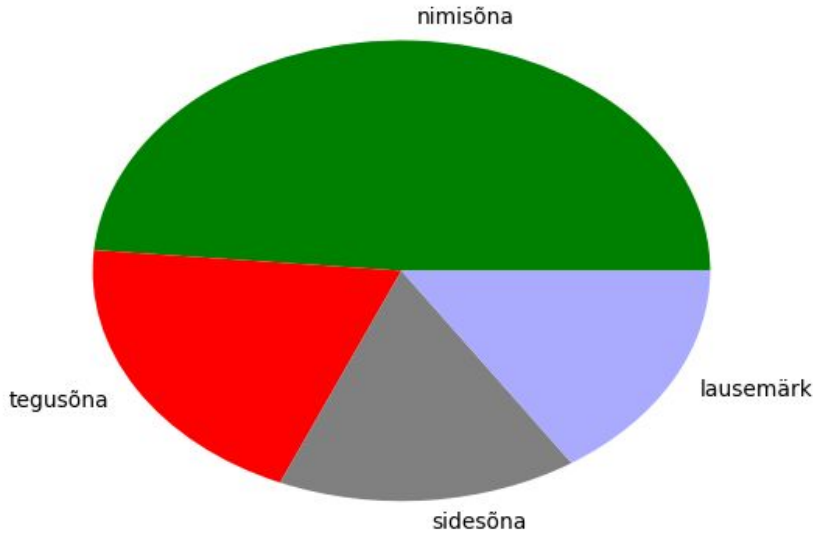
Sektordiagramm

Esimese puhul lihtsalt käsklus pie, kus värvid valib matplotlib ise. Järjest mitme joonise loomisel tuleb vahepeal puhver tühjendada - kui soovitakse puhtalt lehelt alustada. Juurde joonise sektorite värvid ning seletavad kirjad.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

plt.pie([34, 14, 11, 11])
plt.savefig("joonis4.png")

plt.clf() #clear figure
plt.pie([34, 14, 11, 11],
   labels=["nimisõna", "tegusõna", "sidesõna", "lausemärk"],
   colors=["green", "red", "gray", "#AAAAFF"])
plt.savefig("joonis4a.png")
```

Valminud joonised ise



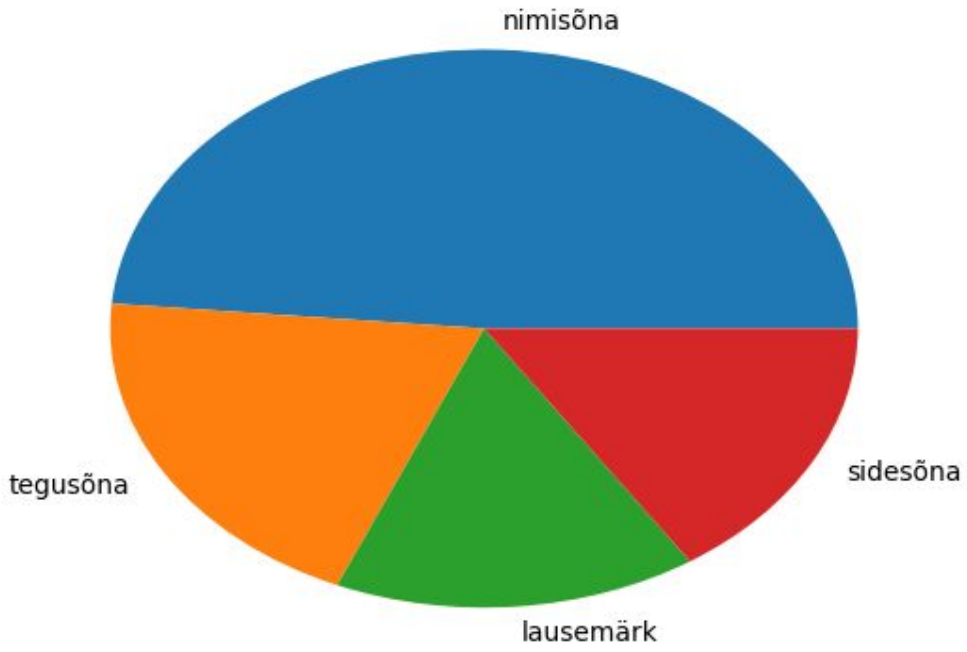


Joonis veebist loetud andmete põhjal

Enne joonise loomist tuleb andmed sobivalt ette valmistada. Praegusel juhul kuvatakse Kungla rahva laulust neli levinumat sõnaliiki kahanevas järjekorras, sektoritel sõnaliikide sildid juures.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonaliigid.txt")
levinumad=sagedused.sort_values(by="kunglarahvas", ascending=False).head(4)
print(levinumad)
plt.pie(levinumad.kunglarahvas, labels=levinumad.sõnaliik)
plt.savefig("joonis5.png")
```

Küs

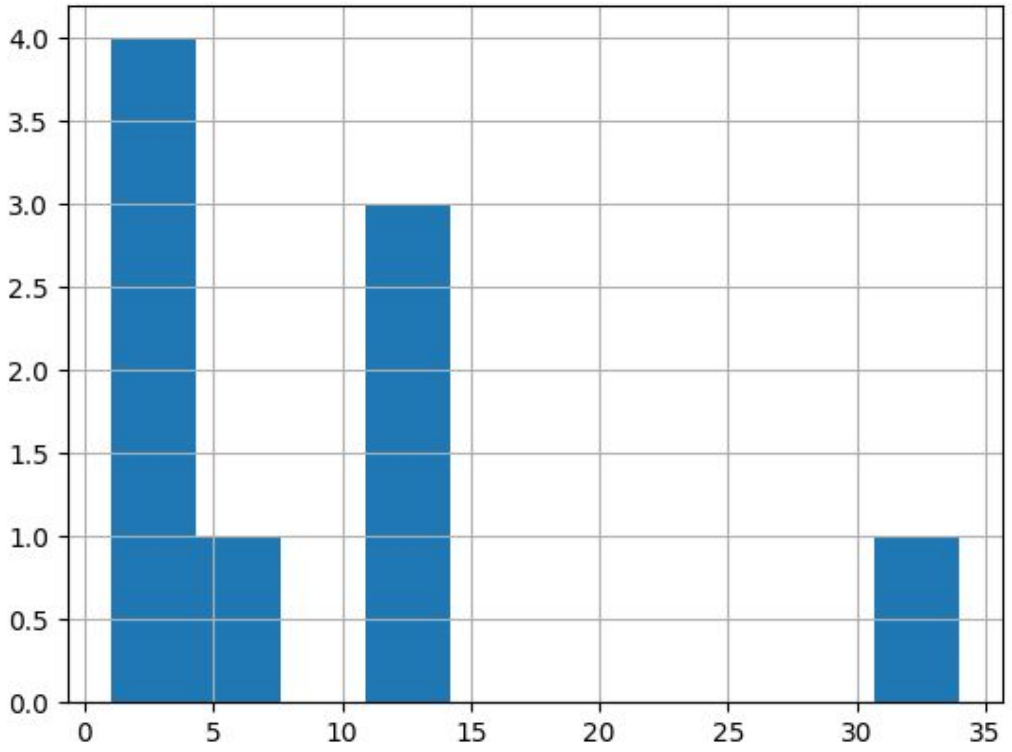


Käsklus DataFrame tulba küljes

Kui matplotlib sisse loetud, siis mõned joonise loomise käsklused on võimalik ka otse andmetulba küljest käivitada - nii võimalik vahel kood lühemana hoida. Näitena histogrammi loomine sõnaliikide sageduse jaotuse kohta

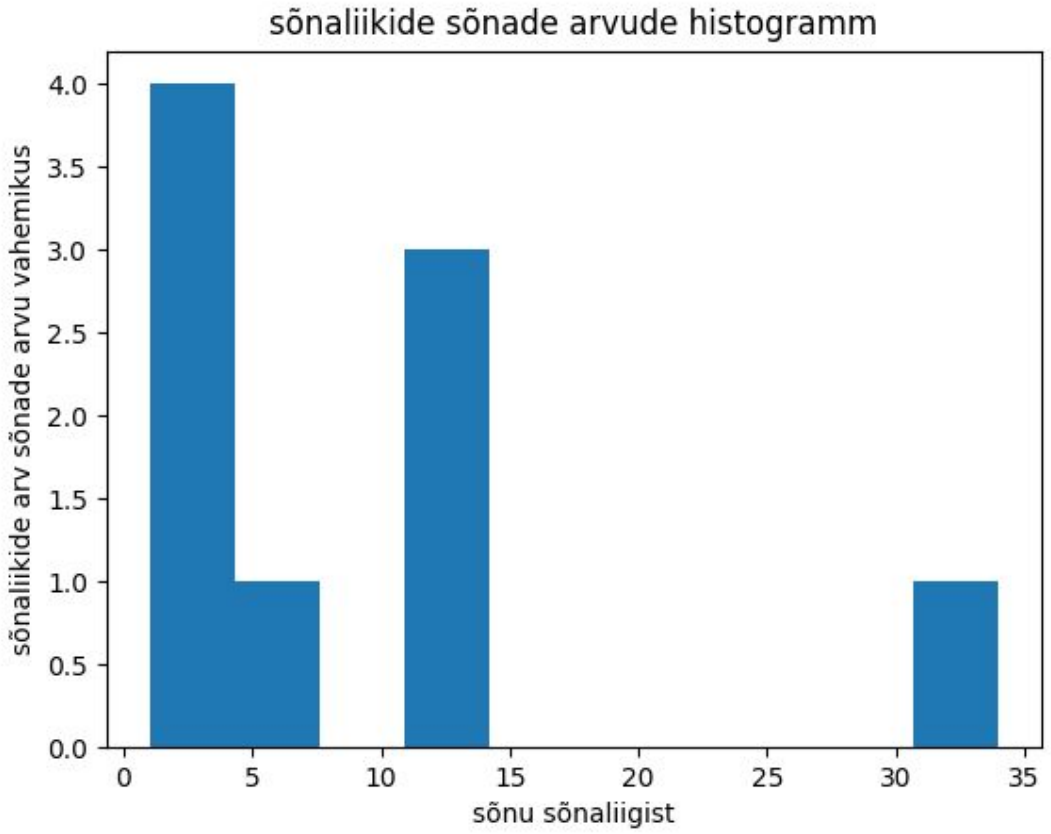
```
import matplotlib
matplotlib.use("Agg")
import pandas as pd
```

```
sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonaliigid.txt")
sagedused.kunglarahvas.hist()
matplotlib.pyplot.savefig("joonis6.png")
```



Võrdlusena sama käsklus välja kutsutuna plt-muutuja kaudu, paar selgitust lisaks

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonaliigid.txt")
plt.hist(sagedused.kunglarahvas)
plt.xlabel("sõnu sõnaliigist")
plt.ylabel("sõnaliikide arv sõnade arvu vahemikus")
plt.title("sõnaliikide sõnade arvude histogramm")
plt.savefig("joonis6a.png")
```



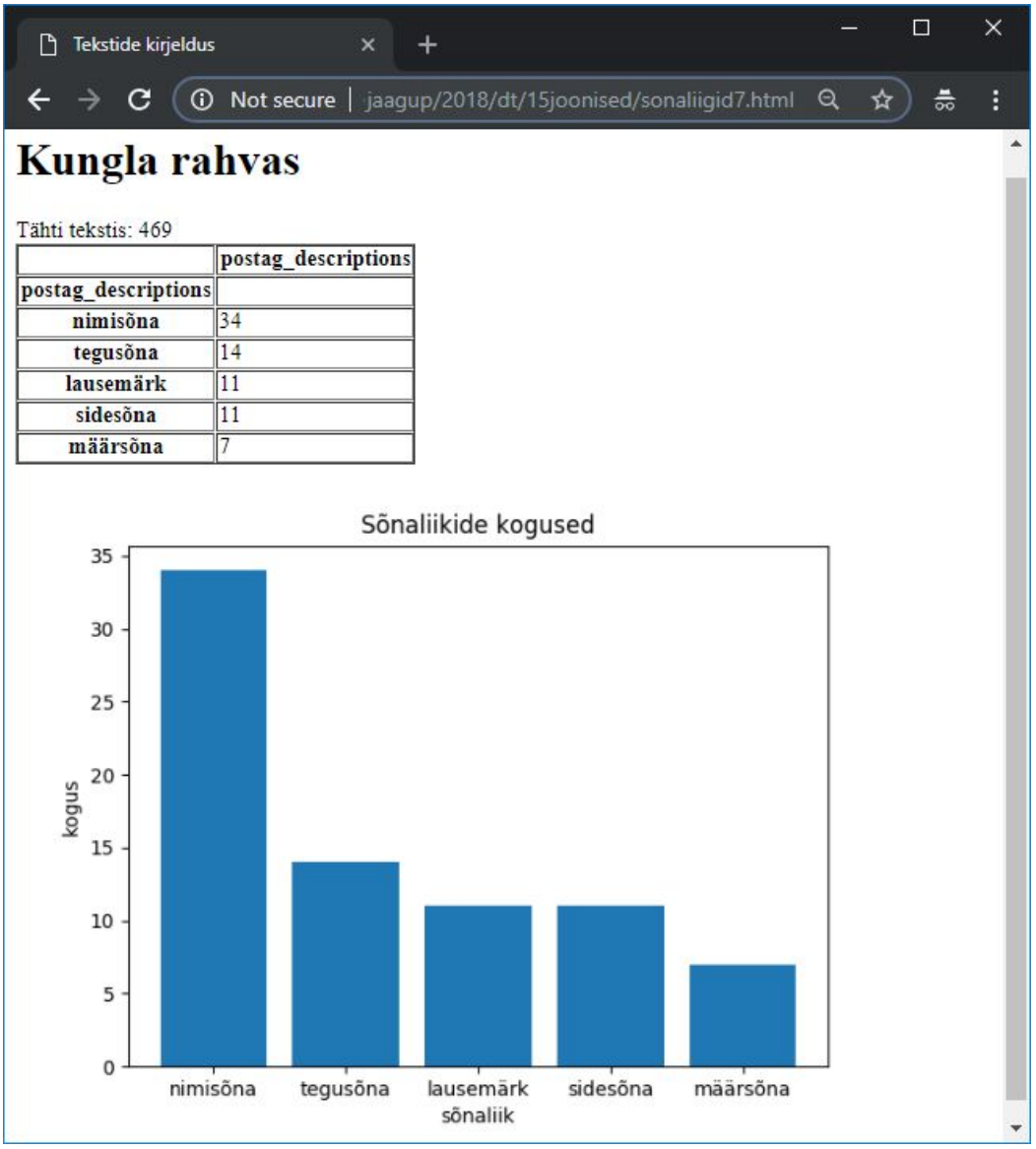
Veebilehe loomine andmete põhjal

Võrreldes eelnevaga arvutatakse andmed sisseloetavast tekstist estnltk abil, luuakse joonis ning selgitava tekstiga HTML-veebileht sinna juurde.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import urllib.request

aadress="http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas.txt"
tekst=urllib.request.urlopen(aadress).read().decode("utf8").lower()

from estnltk import Text
t=Text(tekst)
kogus=5
df=t.get.postag_descriptions.as_dataframe.groupby("postag_descriptions").postag_description
s.count().to_frame().sort_values(by="postag_descriptions", ascending=False).head(kogus)

plt.bar(range(kogus), df.postag_descriptions)
plt.xticks(range(kogus), df.index)
plt.xlabel("sõnaliik")
plt.ylabel("kogus")
plt.title("Sõnaliikide kogused")
```

```
plt.savefig("sonaliikide_sagedused.png")

f=open("sonaliigid7.html", "w")
f.write("<!doctype html>\n")
f.write("<html>\n")
f.write("  <head>\n")
f.write("    <title>Tekstide kirjeldus</title>\n")
f.write("    <meta charset='utf8' />\n")
f.write("  </head>\n")
f.write("  <body>\n")
f.write("    <h1>Kungla rahvas</h1>\n")
f.write("     Tähti tekstis: "+str(len(tekst))+"\n")
f.write(df.to_html())
f.write("    <img src='sonaliikide_sagedused.png' />")
f.write("  </body>\n")
f.write("</html>\n")
f.close()
```

Vaatamiseks ka leht ise



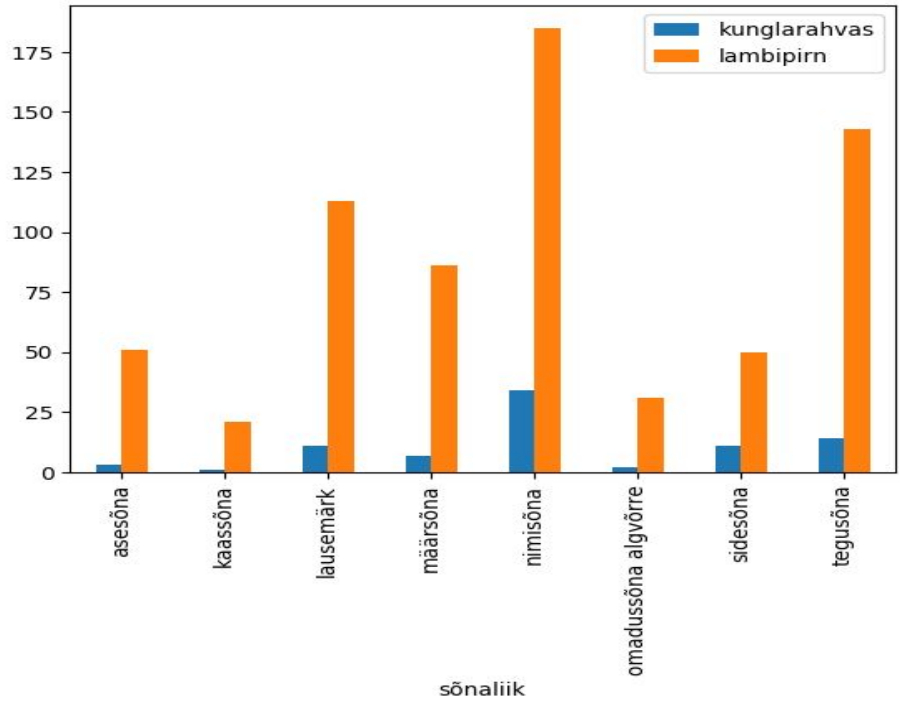
Mitme tunnusega tulpdiagramm

Sõnaliikide kaupa laulude kõrvutamisel aitab tulpdiagramm, kus iga sõnaliigi kohta kummagi laulu vastava sõnaliigi esinemissageduse tulp. Et sõnaliikide nimed alla kirjutataks, selleks tuleb sõnaliikide tulp indeksiks kopeerida. Faili salvestamise juures parameeter

bbox_inches='tight'

hoolitseb, et kõik parajasti ära mahuks ning et ka püsti keeratud tekstid ilusasti pildi sees oleksid.

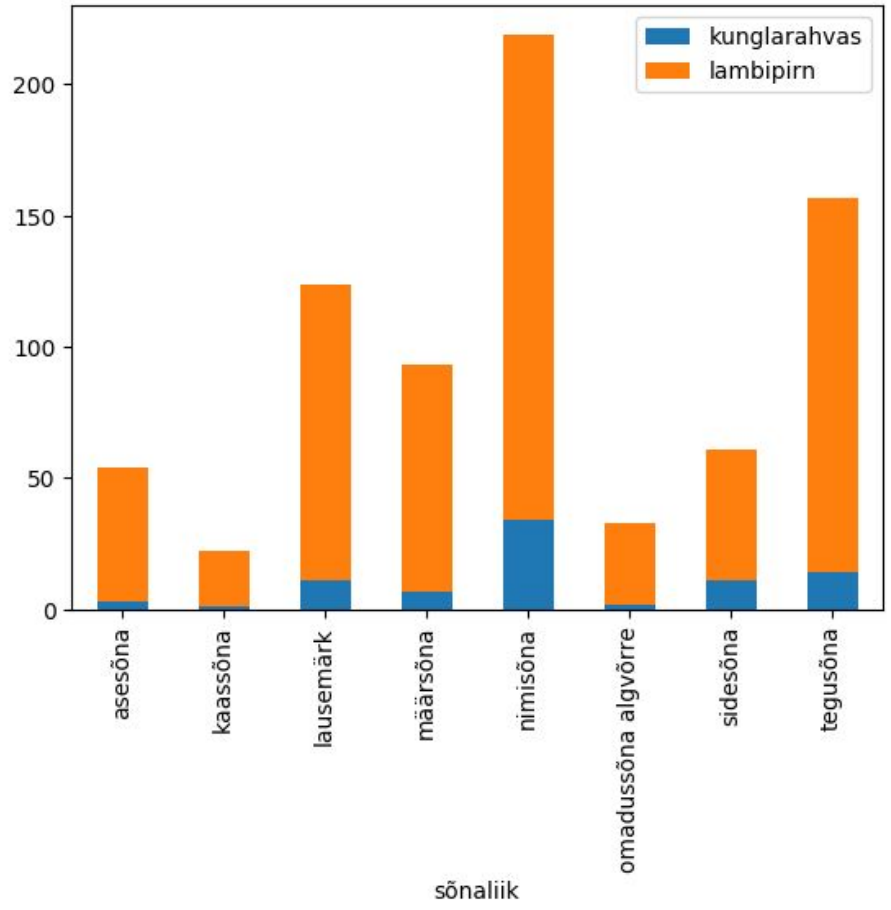
```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonaliigid.txt")
sagedused.index=sagedused.sõnaliik
sagedused.plot(kind="bar")
plt.savefig("joonis1.png", bbox_inches='tight')
```



Tulbad üksteise peal

Tulpade panekuks üksteise peale tuleb lisada joonistamisel parameeter stacked=True
Andmed esiotsa endiselt tähestiku järjekorras.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonalii
gid.txt")
sagedused.index=sagedused.sõnaliik
sagedused.plot(kind="bar", stacked=True)
plt.savefig("joonis1a.png", bbox_inches='tight')
```

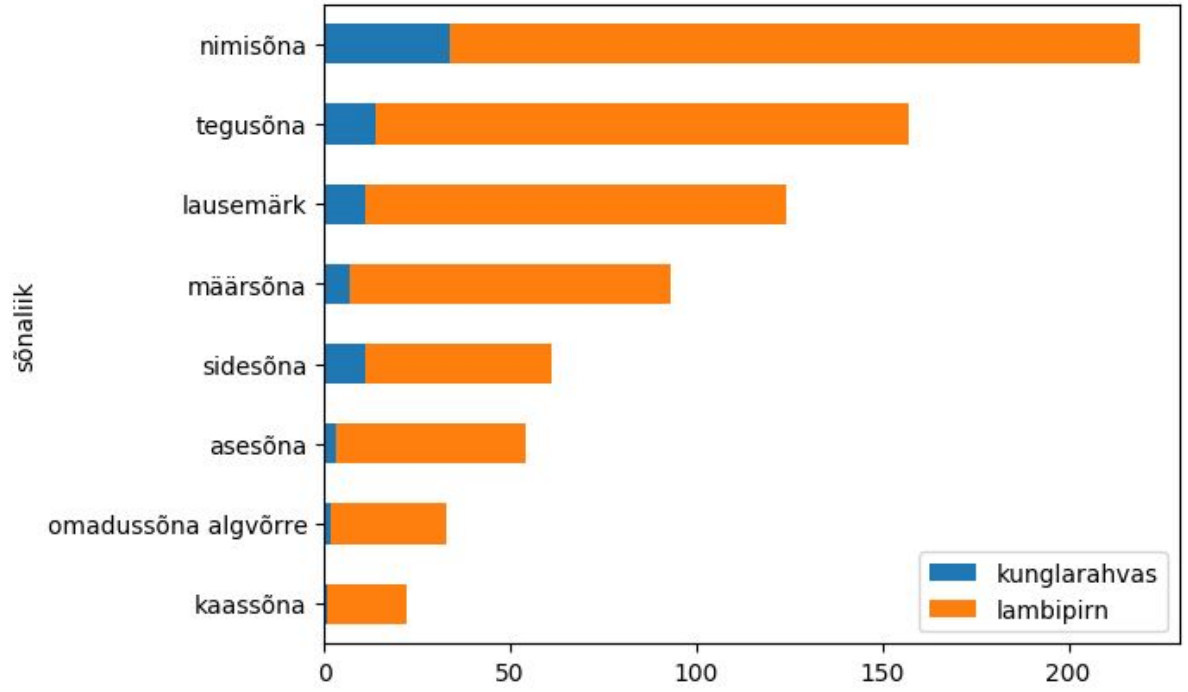


Järjestatud horisontaalsed tulbad

Sõnaliigi esinemissageduse summade järgi ritta seadmiseks kõigepealt arvutame assign-käsuga uue tulba summa tarbeks, järjestame andmed selle järgi, aga lõpuks ikka

valime tulbad, mille järgi kuvada - ehk siis kunglarahvas ja lambipirn. Langjoon \ assign-käskluse ja punkti järel lubab sama käsklust järgmisel real jätkata. Parameeter kind='barh' teatab, et tulbad tuleksid horisontaalselt.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonaliigid.txt")
sagedused.index=sagedused.sõnaliik
sagedused.assign(summa=sagedused.kunglarahvas+sagedused.lambipirn).\
  sort_values(by="summa")[["kunglarahvas", "lambipirn"]].plot(kind="barh", stacked=True)
plt.savefig("joonis1b.png", bbox_inches='tight')
```

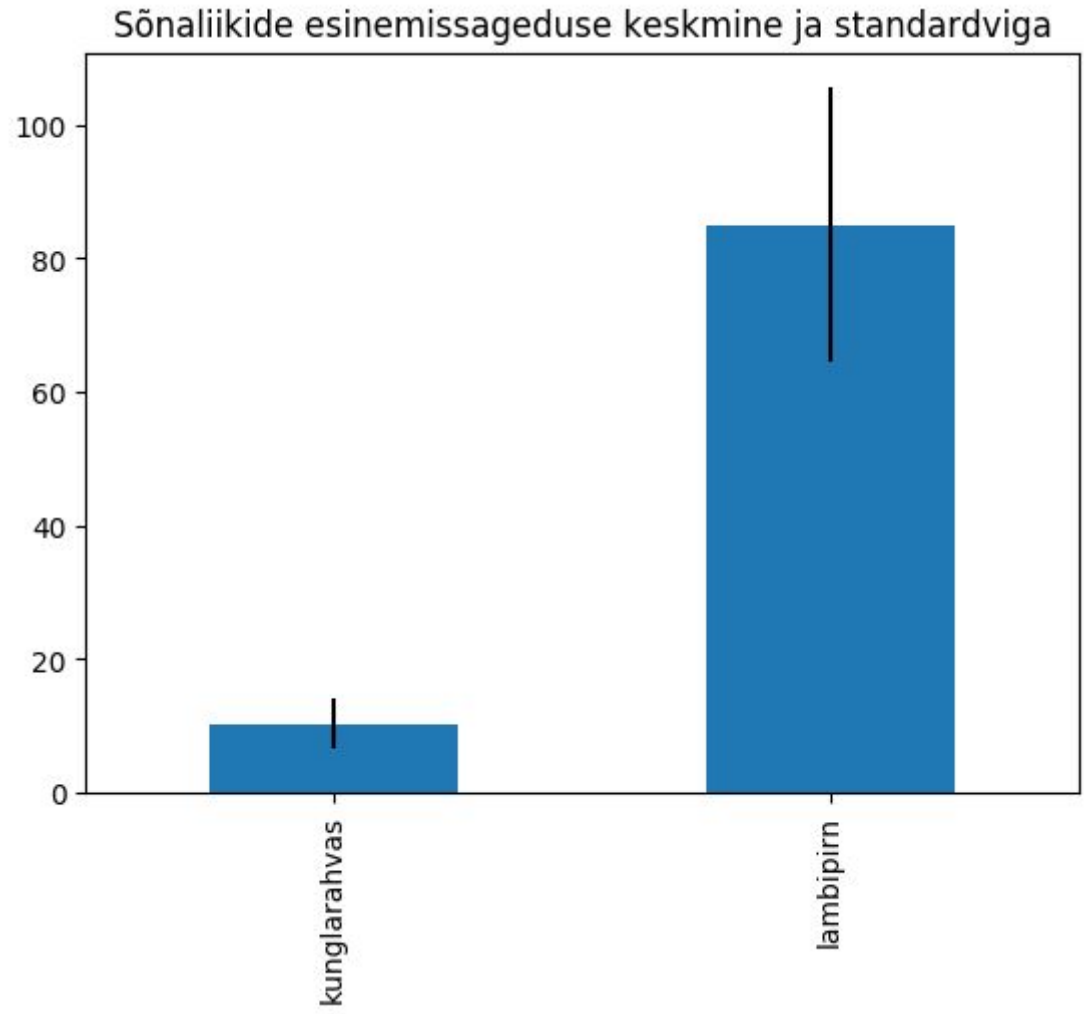


Standardviga tulpdiagrammil

Lisakse keskmise väärtusele näidatakse mõnigikord joonisel ka selle eeldatavat arvutusviga - standardviga, mis matemaatiliselt on arvude standardhälve jagatuna ruutjuurega mõõtmiste arvust. Keskmised arvutatakse ühe mean-käsklusega ning tekkinud Series-tüüpi objekt suudab ka ise oma küljes plot-käsu käivitada. Parameetriga yerr võib sinna kaasa anda standardvigade joonte jaoks tarvilikud andmed.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd
```

```
import math

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonalii
gid.txt")
keskmised=sagedused.mean()
standardvead=sagedused.std()/math.sqrt(len(sagedused))
keskmised.plot(kind="bar", yerr=standardvead,
  title="Sõnaliikide esinemissageduse keskmine ja standardviga")
plt.savefig("joonis1c.png", bbox_inches='tight')
```

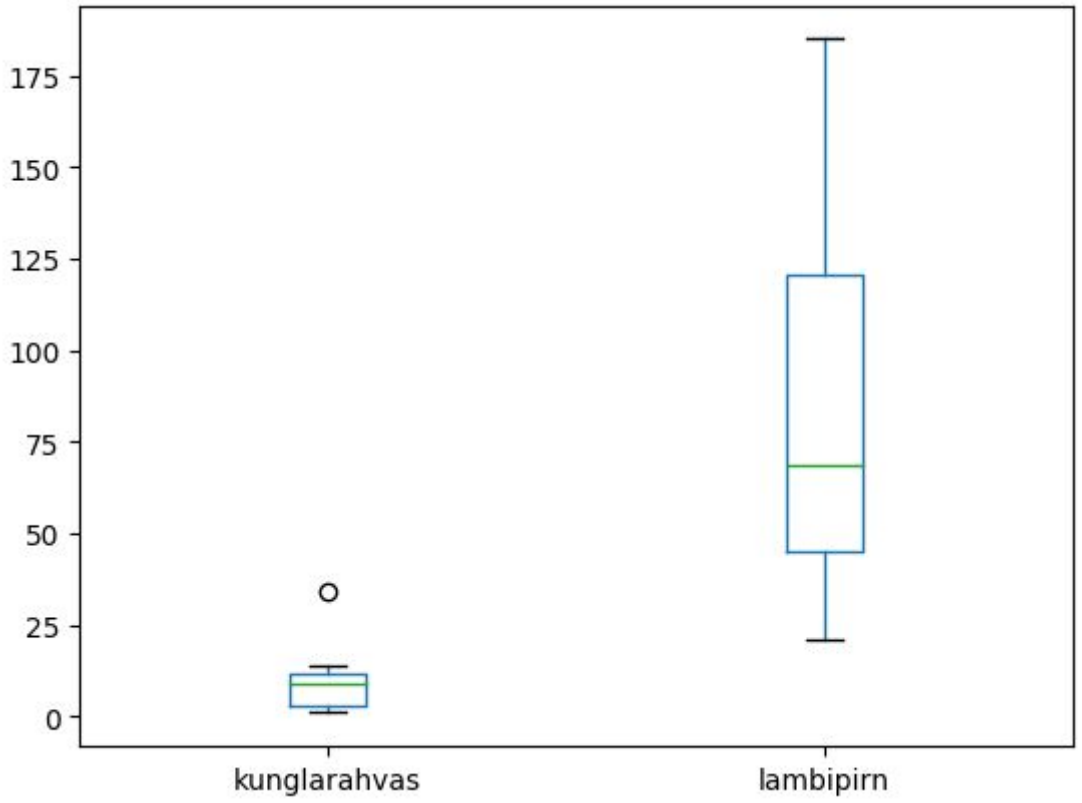


Karpdiagramm

Ehk vanemal ajal pidulikumalt väljendatud “karp ja vurrud” aitab andmete jaotust võimalikult hästi avada.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
```

```
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonaliigid.txt")
sagedused.plot(kind="box")
plt.savefig("joonis2.png", bbox_inches='tight')
```



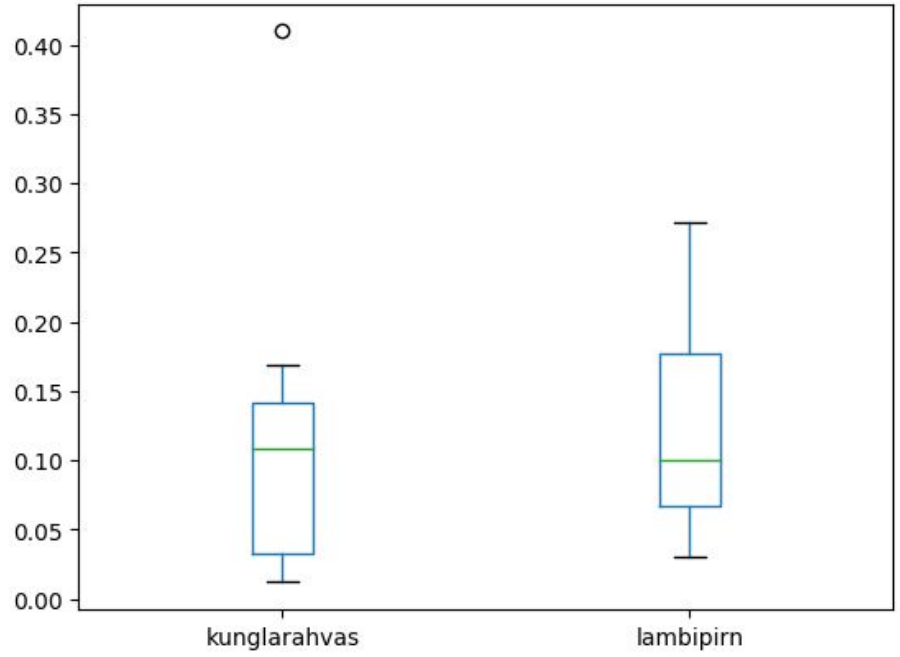
Seletus joonise juurde: Kuna tekstid on erineva pikkusega, siis ka sõnaliikide arvud on Kungla rahva sõnade juures märgatavalt väiksemad. Karbi keskel olev roheline joon kummagi teksti juures näitab mediaani - väärtust, millest pooled arvud on väiksemad ja pooled suuremad. Karbi ala- ja ülaserv on vastavalt 25% ja 75% piiriks. Vurruotstega on tähistatud alumine ja ülemine väärtus. Mummuke Kungla rahva kasti kohal näitab sõnaliiki, mis on ülejäänud jaotusest nõnda väljas, et see märgitakse eraldi ning ei paigutata üldise joonise sisse.

Karpdiagramm skaleeritud andmetega

Andmete võrreldavaks muutmiseks jagatakse siin sõnaliigi esinemissagedus vastavas laulus läbi laulu sõnaliikide arvude summaga, leitakse osakaal. Tehe tehakse vaid tulpade kunglarahvas ja lambipirn juures, sest need on arvulised - sõnaliigi nimetuse tulbaga sarnast tehet ette võttes oleks tulemuseks veateade.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonalii
gid.txt")
for tulp in ["kunglarahvas", "lambipirn"]:
  sagedused[tulp]=sagedused[tulp]/sagedused[tulp].sum()
print(sagedused)
sagedused.plot(kind="box")
plt.savefig("joonis2a.png", bbox_inches='tight')
```

Nüüd aga saab selgemini vaadata, kuidas lauludes arvud jaotuvad. Paistab, et Kungla rahva laulus on üks sõnaliik ülekaalukalt kaugel ja teisi vastavalt vähem, lambipirni jutu juures väärtused ühtlasemad

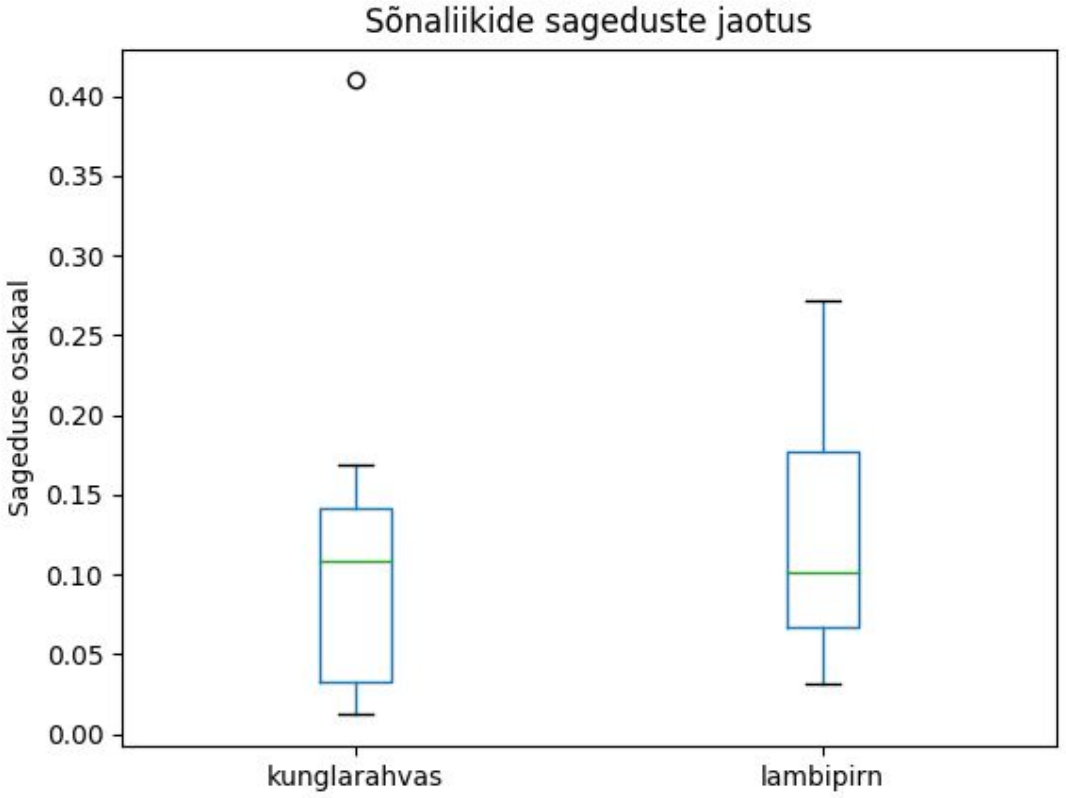


Automaatsemaks arvutuseks saab kõik tulbad tsükliga läbi käia ning siis otsustada funktsiooni is_numeric_dtype abil otsustada, kas tulbaga vastav tehe on võimalik. Funktsioon tuleb enne üleval ka importida.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd
from pandas.api.types import is_numeric_dtype

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonalii
gid.txt")
for tulp in sagedused.columns:
  if is_numeric_dtype(sagedused[tulp]):
    sagedused[tulp]=sagedused[tulp]/sagedused[tulp].sum()
ax=sagedused.plot(kind="box", title="Sõnaliikide sageduste jaotus")
```

```
ax.set_ylabel("Sageduse osakaal")
plt.savefig("joonis2b.png")
```

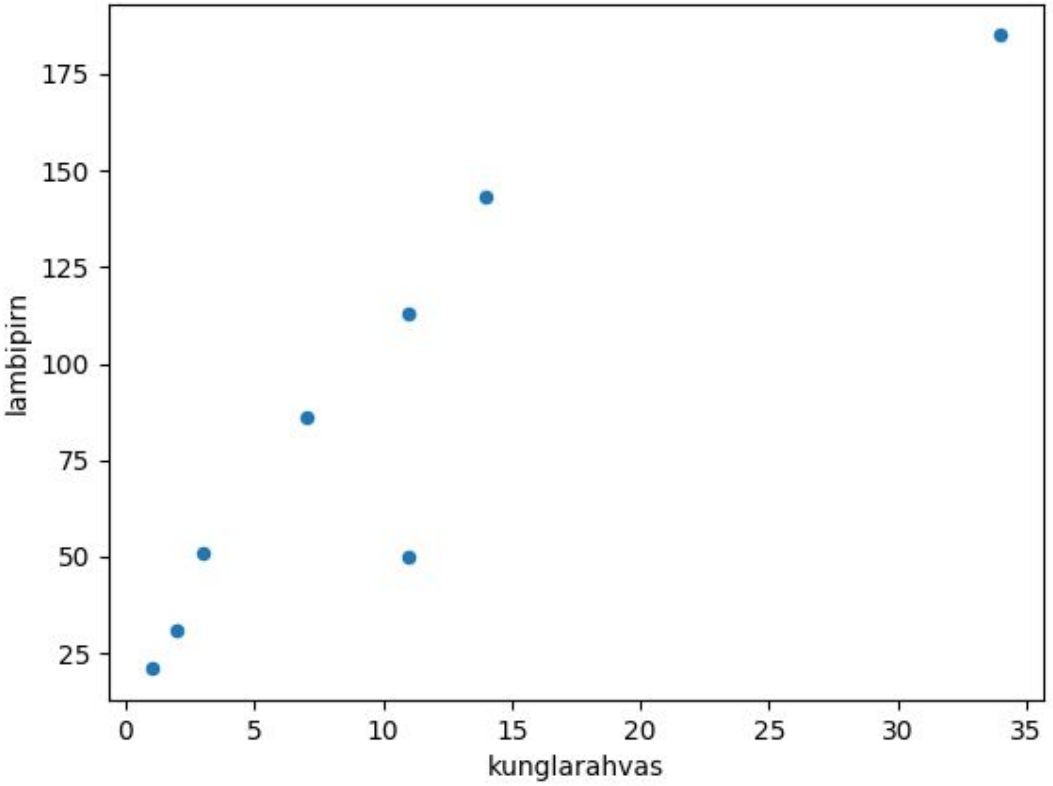
Joonis sarnane kui enne, pealkiri ja y-telje selgitus juures



XY-diagramm

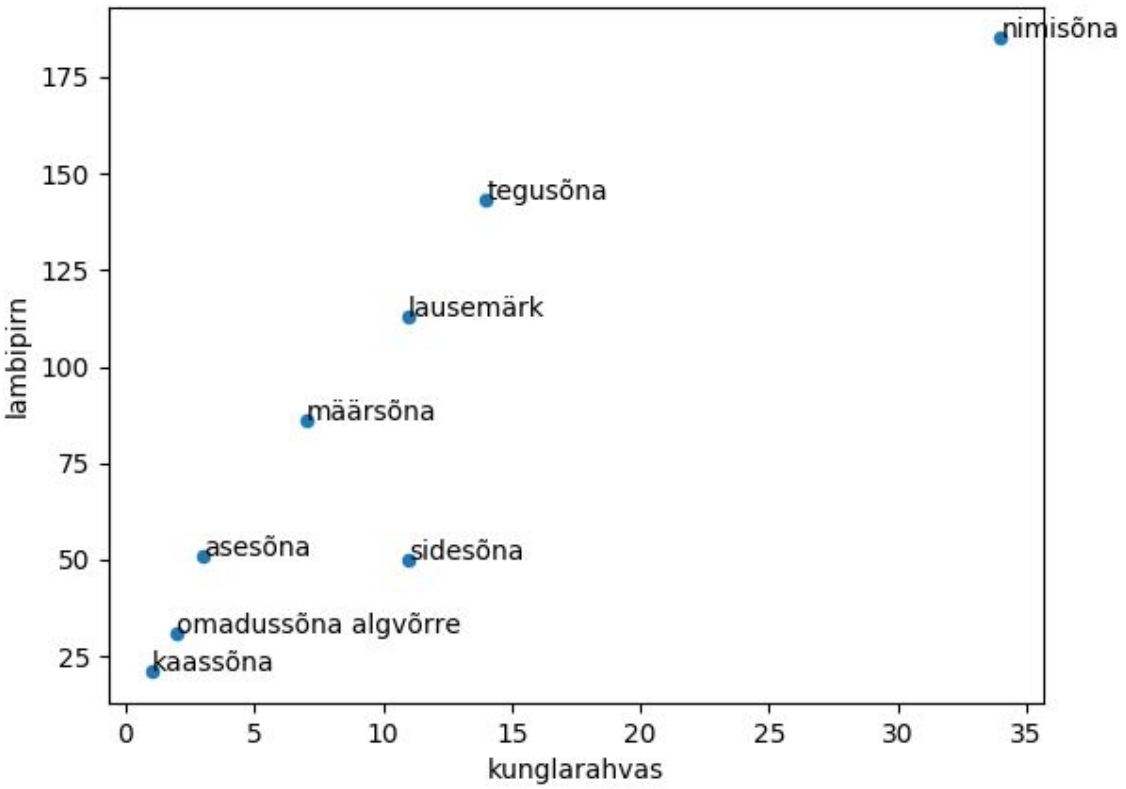
Kahe arvulise andmetulba võrdlemise juures tõenäoliselt levinuim diagramm - näitab, et kuidas iga mõõtmistulemus ühe ja teise telje suhtes paikneb. Siin käivitades tüübiks scatter

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonaliigid.txt")
sagedused.plot(kind="scatter", x="kunglarahvas", y="lambipirn")
plt.savefig("joonis3.png")
```



Tüüpide välja lugemiseks tuleb nad joonisele kuvada. Käsklusega annotate saab teksti paigutada soovitud kohta. Parameetriks kõigepealt tekst ise ning siis massiivina koordinaadid.

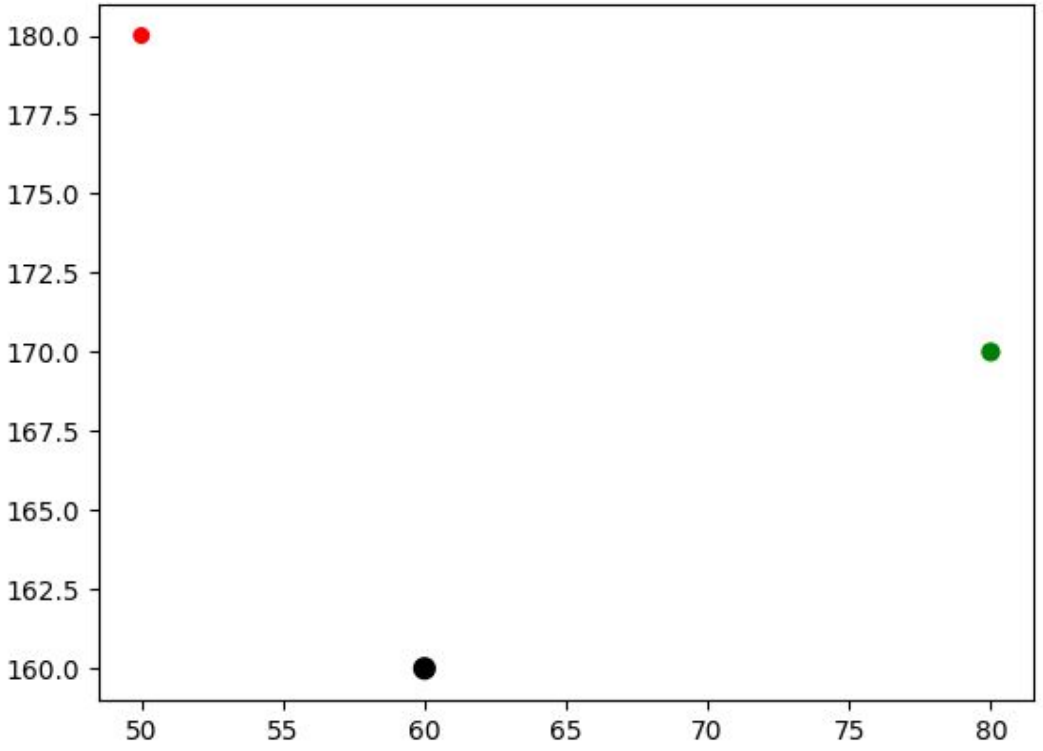
```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import pandas as pd

sagedused=pd.read_csv("http://www.tlu.ee/~jaagup/andmed/keel/kunglarahvas_lambipirn_sonalii
gid.txt")
ax=sagedused.plot(kind="scatter", x="kunglarahvas", y="lambipirn")
for nr in range(len(sagedused)):
  ax.annotate(sagedused.sõnaliik[nr],
    (sagedused.kunglarahvas[nr], sagedused.lambipirn[nr]))
plt.savefig("joonis3a.png")
```



Punktid ja joon ekraanil

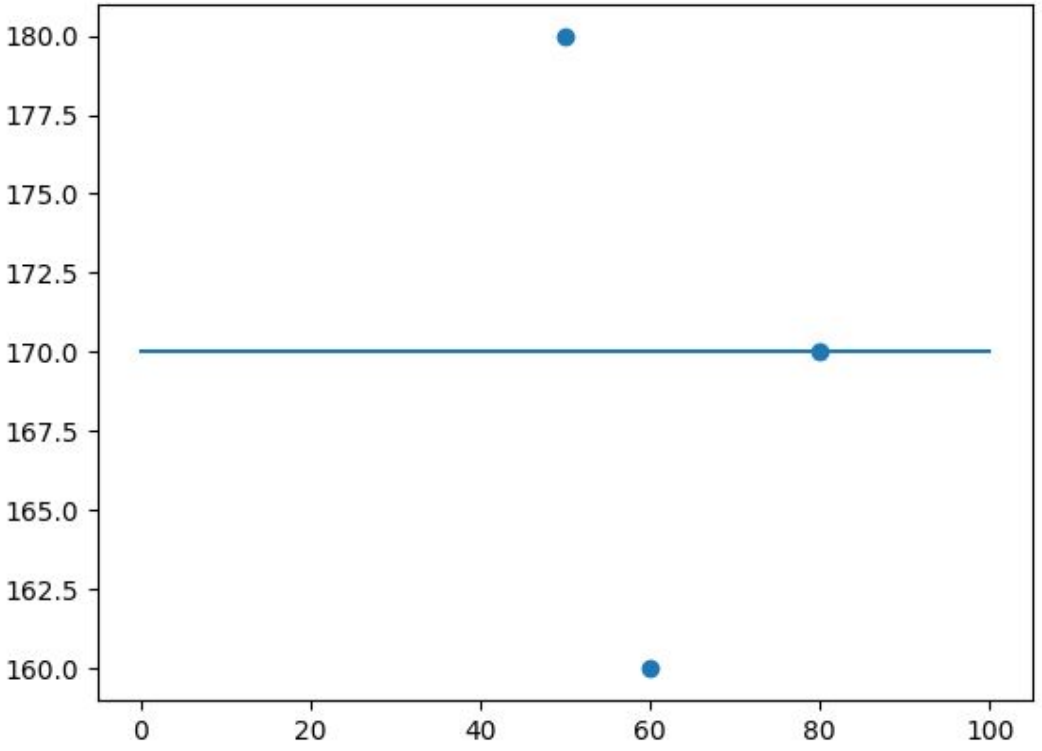
Käsklus scatter toimib ka otse pyplot-i objekti küljes, andmed sisse kahe massiivina. Võimalik määrata ka punktide värvi ja pindala.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

massid=[60, 80, 50]
pikkused=[160, 170, 180]
juuksevarvid=["black", "green", "red"]
kodusuurus=[60, 40, 30]
plt.scatter(massid, pikkused, c=juuksevarvid, s=kodusuurus)
plt.savefig("joonis3b.png")
```



Joone tõmbamiseks käsklus plot - esimese parameetrina x-ide massiiv, teisena y-ite massiv. Kolmandaks parameetriks olev miinusmärk teatab, et soovitakse joont. Enne seda võib lisada ka joone soovitava värvi, näiteks “r-” tähistab punast (red).

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

massid=[60, 80, 50]
pikkused=[160, 170, 180]
plt.scatter(massid, pikkused)
plt.plot([0, 100], [170, 170], "-")
plt.savefig("joonis3c.png")
```



Pythoni DataFrame andmete põhjal jooniste koostamise mitmesuguseid näiteid leiab lisaks aadressilt

https://pandas.pydata.org/pandas-docs/stable/visualization.html

SQL

Aastakümneid on andmete hoidmise ja päringute tegemise juures valitsevaks olnud relatsioonilised andmebaasid. Andmeid hoitakse tabelites. Samuti seotakse eri tüüpi andmed kokku viidetega tabelite vahel. Sealjuures levinuimaks andmete kirjeldamise ja päringukeeleks on SQL. Mõningad eripärad selles on vastavalt andmebaasiprogrammile, kuid põhioperatsioonid ikka sarnaselt ette võetavad.

Andmete loomine

Andmebaasiühenduse katsetamiseks kuluvad ära sisulised andmed, millega midagi ette võtta. Siin anname ESTNLTK Pythoni paketi Text-tüüpi objektile ette kirjandusklassikast tuntud lause ning küsime lause elementide sõnaliigid. Kus rakendus ei oska kindlat vastet pakkuda, seal jääb väärtus tühjaks. Kus tegemist kirjavahemärgi ehk lausemärgiga, siis nii kirjutataksegi.

```
jaagup@praktika1 ~/public_html/2018/dt/17sql $ python3.5
Python 3.5.1+ (default, Mar 30 2016, 22:46:26)
[GCC 5.3.1 20160330] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from estnltk import Text
>>> t=Text("Kui Arno isaga koolimajja jõudis, olid tunnid juba alanud.")



>>> t.get.word_texts.postag_descriptions.as_dataframe
    word_texts postag_descriptions
0          Kui
1         Arno           pärisnimi
2        isaga            nimisõna
3   koolimajja            nimisõna
4       jõudis            tegusõna
5            ,           lausemärk
6         olid            tegusõna
7       tunnid            nimisõna
8         juba            määrsõna
9       alanud
10           .           lausemärk
```

Väike andmestik olemas. Pythonist välja

```
>>> exit()
```

ning olemegi taas käsureal, et SQLiga tutvust teha

```
jaagup@praktika1 ~/public_html/2018/dt/17sql $
```

Andmebaasi loomine

Suures ametlikus ettevõttes võib andmebaasi loomise taotlus käia mitu sammu läbi, kuni siis otsustatakse, et millise mahu ja õigustega baas millise lahenduse tarbeks luuakse. Siinses testserveris aga olemas harjutuskonto, mille kaudu võimalik omale kasutamiseks baas luua ning seal sees andmetega toimetada. Haldamise utiliidiks mysqladmin, testkasutaja nimega dh18, parooliga dh18praktika. Vastavale kasutajale on antud õigus hallata baase, mis algavad eesliitega dh18_ . Näites luuakse baas nimega dh18_jaagup

```
jaagup@praktika1 ~/public_html/2018/dt/17sql $ mysqladmin -udh18 -pdh18praktika create
dh18_jaagup
```

Baasi loomine on ühekordne käsklus, järgmistel sisenemistel baas juba olemas.

Andmebaasi sisenemine

Baasis käskluste käivitamiseks tuleb baasi siseneda. Abiks utiliit mysql, samad kasutajanimi ja parool ning baasi nimi

```
jaagup@praktika1 ~/public_html/2018/dt/17sql $ mysql -udh18 -pdh18praktika dh18_jaagup
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 273895
Server version: 10.0.24-MariaDB-7 Ubuntu 16.04

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Edasi ootab käsuviip juba käske andmebaasile

MariaDB [dh18_jaagup]>

Andmetabeli loomine

Tabeli loomise käsklus on mugav eraldi tekstiredaktori aknas valmis kirjutada ning siis SQL-i käsureale kopeerida. Sellisel puhul kui ka pikema käskluse juures vigu peaks sisse tulema, saab need algses kohas ära parandada ning siis uuesti kopeerida.

Tabeli loomiseks käsklus CREATE TABLE. SQLi omad käsud kirjutatakse suurte tähtedega, muud osad väikestega. Selline tava on keele sisse ammusest ajast jäänud ning võimaldab suurema käskluse korral mugavamalt haarata, et kus midagi paikneb. Tabeli nimeks kevadelause - siin näites toimetamegi vaid selle ühe lausega.

Iga andmetulba kohta tuleb öelda selle nimi ja tüüp. Tungivalt soovituslik on lisada ühele tulbale omadus PRIMARY KEY ehk primaarvõti - nii hoolitseb andmebaasirakendus, et sinna ei pandaks korduvaid väärtusi ning sealtkaudu on võimalik vastavale tabeli reale kindlalt viidata. Tabearvutusprogrammides näiteks on olemas selgelt eristatav reanumber, siin aga mitte. Ning primaarvõti täidab suurelt osalt vastavat ülesannet.

Täisarvutüübiks INT (sõnast integer), NOT NULL teatab, et väärtus on kohustuslik. Tüüp VARCHAR (variable character) näitab, et tegemist on tekstiga, sulgudes kirjas selle lubatav maksimumpikkus

```
CREATE TABLE kevadelause(
   sonanr INT NOT NULL PRIMARY KEY,
   sona VARCHAR(20),
   sonaliik VARCHAR(20)
);
```

Kopeerime koodi käsuviiba juurde. Vastuseks saame Query OK, ehk käsklus õnnestus

```
MariaDB [dh18_jaagup]> CREATE TABLE kevadelause(
    ->    sonanr INT NOT NULL PRIMARY KEY,
    ->    sona VARCHAR(20),
    ->    sonaliik VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

Andmete sisestamine

Andmete lisamiseks käib käsklus INSERT. Lihtsamal juhul sisestus tulpade järjekorras. Praegusel juhul siis kõigepealt sõna järjekorranumber, siis sõna ise ning edasi sõnaliik. Vastuseks, et rida lisati. Inimlikkuse huvides alustame lause sõnade loendamist ühest.

```
INSERT INTO kevadelause VALUES (1, 'Kui', '');
Query OK, 1 row affected (0.00 sec)
```

Edasi sarnased sisestuskäsud ka andmestike teiste ridade kohta.

```
INSERT INTO kevadelause VALUES (2, 'Arno', 'pärisnimi');
INSERT INTO kevadelause VALUES (3, 'isaga', 'nimisõna');
INSERT INTO kevadelause VALUES (4, 'koolimajja', 'nimisõna');
INSERT INTO kevadelause VALUES (5, 'jõudis', 'tegusõna');
INSERT INTO kevadelause VALUES (6, ',', 'lausemärk');
INSERT INTO kevadelause VALUES (7, 'olid', 'tegusõna');
INSERT INTO kevadelause VALUES (8, 'tunnid', 'nimisõna');
INSERT INTO kevadelause VALUES (9, 'juba', 'määrsõna');
INSERT INTO kevadelause VALUES (10, 'alanud', '');
INSERT INTO kevadelause VALUES (11, '.', 'lausemärk');
```

Andmete päring

Tulemuste nägemiseks käsklus SELECT. Tärn ütleb, et soovime näha kõikide tulpade sisu

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause;
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
+--------+------------+-------------+
|      1 | Kui        |             |
|      2 | Arno       | pärisnimi   |
|      3 | isaga      | nimisõna    |
|      4 | koolimajja | nimisõna    |
|      5 | jõudis     | tegusõna    |
|      6 | ,          | lausemärk   |
|      7 | olid       | tegusõna    |
|      8 | tunnid     | nimisõna    |
|      9 | juba       | määrsõna    |
|     10 | alanud     |             |
|     11 | .          | lausemärk   |
+--------+------------+-------------+
11 rows in set (0.00 sec)
```

Filtreerimine vastavalt sõnaliigile

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause WHERE sonaliik='nimisõna';
+--------+------------+-----------+
| sonanr | sona       | sonaliik  |
+--------+------------+-----------+
|      3 | isaga      | nimisõna  |
|      4 | koolimajja | nimisõna  |
|      8 | tunnid     | nimisõna  |
+--------+------------+-----------+
```

Soovitud liikide loetelu

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause WHERE sonaliik IN ('nimisõna',
'tegusõna');
+--------+------------+-----------+
| sonanr | sona       | sonaliik  |
+--------+------------+-----------+
|      3 | isaga      | nimisõna  |
|      4 | koolimajja | nimisõna  |
|      5 | jõudis     | tegusõna  |
|      7 | olid       | tegusõna  |
|      8 | tunnid     | nimisõna  |
+--------+------------+-----------+
5 rows in set (0.02 sec)
```

või siis sama tulemus käskude tingimuste ühendamise teel

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause WHERE sonaliik='nimisõna' OR
sonaliik='tegusõna';
+--------+------------+-----------+
| sonanr | sona       | sonaliik  |
```

```
+--------+------------+-----------+
|      3 | isaga      | nimisõna  |
|      4 | koolimajja | nimisõna  |
|      5 | jõudis     | tegusõna  |
|      7 | olid       | tegusõna  |
|      8 | tunnid     | nimisõna  |
+--------+------------+-----------+
5 rows in set (0.00 sec)
```

Tulemuste järjestamine

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause ORDER BY sona;
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
+--------+------------+-------------+
|      6 | ,          | lausemärk   |
|     11 | .          | lausemärk   |
|     10 | alanud     |             |
|      2 | Arno       | pärisnimi   |
|      3 | isaga      | nimisõna    |
|      5 | jõudis     | tegusõna    |
|      9 | juba       | määrsõna    |
|      4 | koolimajja | nimisõna    |
|      1 | Kui        |             |
|      7 | olid       | tegusõna    |
|      8 | tunnid     | nimisõna    |
+--------+------------+-------------+
11 rows in set (0.04 sec)
```

Sõnad kahanevas järjekorras, näidatakse esimesed 5 vastust

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause ORDER BY sona DESC LIMIT 5;
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
+--------+------------+-------------+
|      8 | tunnid     | nimisõna    |
|      7 | olid       | tegusõna    |
|      1 | Kui        |             |
|      4 | koolimajja | nimisõna    |
|      9 | juba       | määrsõna    |
+--------+------------+-------------+
5 rows in set (0.00 sec)
```

Harjutus

- Kuvage andmed sõnaliikide järjekorras
- Kuvage nimisõnad tähestiku järjekorras

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause ORDER BY sonaliik;
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
```

```
+--------+------------+-------------+
|      1 | Kui        |             |
|     10 | alanud     |             |
|      6 | ,          | lausemärk   |
|     11 | .          | lausemärk   |
|      9 | juba       | määrsõna    |
|      4 | koolimajja | nimisõna    |
|      3 | isaga      | nimisõna    |
|      8 | tunnid     | nimisõna    |
|      2 | Arno       | pärisnimi   |
|      5 | jõudis     | tegusõna    |
|      7 | olid       | tegusõna    |
+--------+------------+-------------+
11 rows in set (0.00 sec)
```

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause WHERE sonaliik='nimisõna' ORDER BY sona;
+--------+------------+-----------+
| sonanr | sona       | sonaliik  |
+--------+------------+-----------+
|      3 | isaga      | nimisõna  |
|      4 | koolimajja | nimisõna  |
|      8 | tunnid     | nimisõna  |
+--------+------------+-----------+
3 rows in set (0.00 sec)
```

Lause sõnad alates kuuendast

```
MariaDB [dh18_jaagup]> SELECT sonanr, sona FROM kevadelause WHERE sonanr >=6;
+--------+--------+
| sonanr | sona   |
+--------+--------+
|      6 | ,      |
|      7 | olid   |
|      8 | tunnid |
|      9 | juba   |
|     10 | alanud |
|     11 | .      |
+--------+--------+
6 rows in set (0.00 sec)
```

Sõnad koos sõnapikkustega - tähtede arvu leidmiseks käsklus LENGTH

```
MariaDB [dh18_jaagup]> SELECT sona, length(sona) FROM kevadelause;
+------------+--------------+
| sona       | length(sona) |
+------------+--------------+
| Kui        |            3 |
| Arno       |            4 |
| isaga      |            5 |
| koolimajja |           10 |
| jõudis     |            7 |
| ,          |            1 |
| olid       |            4 |
| tunnid     |            6 |
| juba       |            4 |
| alanud     |            6 |
```

```
| .          |            1 |
+------------+--------------+
11 rows in set (0.06 sec)
```

Sõnad kahanevalt pikkuse järjekorras. Tulba ümber nimetamine AS abil

```
MariaDB [dh18_jaagup]> SELECT sona, length(sona) AS pikkus FROM kevadelause
    ->   ORDER BY pikkus DESC;
+------------+--------+
| sona       | pikkus |
+------------+--------+
| koolimajja |     10 |
| jõudis     |      7 |
| alanud     |      6 |
| tunnid     |      6 |
| isaga      |      5 |
| Arno       |      4 |
| olid       |      4 |
| juba       |      4 |
| Kui        |      3 |
| ,          |      1 |
| .          |      1 |
+------------+--------+
11 rows in set (0.00 sec)
```

Agregaatfunktsioonid

Ehk siis ridade kaupa kokku arvutamise käsklused. Kõige lihtsam on kokku lugeda tabeli ridade arv

```
MariaDB [dh18_jaagup]> SELECT COUNT(*) FROM kevadelause;
+----------+
| COUNT(*) |
+----------+
|       11 |
+----------+
1 row in set (0.02 sec)
```

Nimisõnadega ridade arv tabelis

```
MariaDB [dh18_jaagup]> SELECT COUNT(*) FROM kevadelause WHERE sonaliik='nimisõna';
+----------+
| COUNT(*) |
+----------+
|        3 |
+----------+
1 row in set (0.00 sec)
```

Ridade arvud sõnaliikide kaupa

```
MariaDB [dh18_jaagup]> SELECT sonaliik, COUNT(*) FROM kevadelause GROUP BY sonaliik;
+-------------+----------+
| sonaliik    | COUNT(*) |
```

```
+-------------+----------+
|             |        2 |
| lausemärk   |        2 |
| määrsõna    |        1 |
| nimisõna    |        3 |
| pärisnimi   |        1 |
| tegusõna    |        2 |
+-------------+----------+
6 rows in set (0.00 sec)
```

Suurim sõna järjekorranumber

```
MariaDB [dh18_jaagup]> SELECT MAX(sonanr) FROM kevadelause;
+-------------+
| MAX(sonanr) |
+-------------+
|          11 |
+-------------+
1 row in set (0.00 sec)
```

Harjutus

- Leidke iga sõnaliigi kohta suurim sõnanumber
- Leidke iga sõnaliigi kohta suurim ja vähim sõnanumber

```
MariaDB [dh18_jaagup]> SELECT MAX(sonanr), sonaliik FROM kevadelause GROUP BY sonaliik;
+-------------+-------------+
| MAX(sonanr) | sonaliik    |
+-------------+-------------+
|          10 |             |
|          11 | lausemärk   |
|           9 | määrsõna    |
|           8 | nimisõna    |
|           2 | pärisnimi   |
|           7 | tegusõna    |
+-------------+-------------+
6 rows in set (0.00 sec)
```

```
MariaDB [dh18_jaagup]> SELECT MIN(sonanr) as esimene, MAX(sonanr) AS viimane, sonaliik FROM
kevadelause GROUP BY sonaliik;
+---------+---------+-------------+
| esimene | viimane | sonaliik    |
+---------+---------+-------------+
|       1 |      10 |             |
|       6 |      11 | lausemärk   |
|       9 |       9 | määrsõna    |
|       3 |       8 | nimisõna    |
|       2 |       2 | pärisnimi   |
```

```
|       5 |       7 | tegusõna    |
+---------+---------+-------------+
6 rows in set (0.00 sec)
```

Sõnade rühmitamine

MySQL-i käsklus GROUP_CONCAT paneb rühma sõnad kõik ühte, vaikimisi komaga eraldatud loetellu. Nii saab tulemusi mugavalt ühe tervikuna näha

```
MariaDB [dh18_jaagup]> SELECT sonaliik, GROUP_CONCAT(sona) FROM kevadelause GROUP BY
sonaliik;
+-------------+--------------------------+
| sonaliik    | GROUP_CONCAT(sona)       |
+-------------+--------------------------+
|             | Kui,alanud               |
| lausemärk   | ,,.                      |
| määrsõna    | juba                     |
| nimisõna    | koolimajja,isaga,tunnid  |
| pärisnimi   | Arno                     |
| tegusõna    | jõudis,olid              |
+-------------+--------------------------+
6 rows in set (0.00 sec)
```

Andmete muutmine

Alustuseks näha tabeli sisu

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause;
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
+--------+------------+-------------+
|      1 | Kui        |             |
|      2 | Arno       | pärisnimi   |
|      3 | isaga      | nimisõna    |
|      4 | koolimajja | nimisõna    |
|      5 | jõudis     | tegusõna    |
|      6 | ,          | lausemärk   |
|      7 | olid       | tegusõna    |
|      8 | tunnid     | nimisõna    |
|      9 | juba       | määrsõna    |
|     10 | alanud     |             |
|     11 | .          | lausemärk   |
+--------+------------+-------------+
11 rows in set (0.00 sec)
```

Real oleva väärtuse muutmiseks sobib käsklus UPDATE. WHERE piiranguga tasub ära määrata, et millise rea kohta muutus käib.

```
MariaDB [dh18_jaagup]> UPDATE kevadelause SET sonaliik='teadmata' WHERE sonanr=1;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [dh18_jaagup]> SELECT * FROM kevadelause;
```

```
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
+--------+------------+-------------+
|      1 | Kui        | teadmata    |
|      2 | Arno       | pärisnimi   |
|      3 | isaga      | nimisõna    |
|      4 | koolimajja | nimisõna    |
|      5 | jõudis     | tegusõna    |
|      6 | ,          | lausemärk   |
|      7 | olid       | tegusõna    |
|      8 | tunnid     | nimisõna    |
|      9 | juba       | määrsõna    |
|     10 | alanud     |             |
|     11 | .          | lausemärk   |
+--------+------------+-------------+
11 rows in set (0.00 sec)
```

Andmete kustutamiseks käsklus DELETE

```
MariaDB [dh18_jaagup]> DELETE FROM kevadelause WHERE sonanr=1;
Query OK, 1 row affected (0.00 sec)
```

Siis näha, et sõna järjekorranumbriga 1 on lahkunud

```
MariaDB [dh18_jaagup]> SELECT * FROM kevadelause;
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
+--------+------------+-------------+
|      2 | Arno       | pärisnimi   |
|      3 | isaga      | nimisõna    |
|      4 | koolimajja | nimisõna    |
|      5 | jõudis     | tegusõna    |
|      6 | ,          | lausemärk   |
|      7 | olid       | tegusõna    |
|      8 | tunnid     | nimisõna    |
|      9 | juba       | määrsõna    |
|     10 | alanud     |             |
|     11 | .          | lausemärk   |
+--------+------------+-------------+
10 rows in set (0.00 sec)
```

Andmete säilimiseks sama lause taas INSERT käsu abil tagasi

```
MariaDB [dh18_jaagup]> INSERT INTO kevadelause VALUES (1, 'Kui', '');
Query OK, 1 row affected (0.01 sec)

MariaDB [dh18_jaagup]> SELECT * FROM kevadelause;
+--------+------------+-------------+
| sonanr | sona       | sonaliik    |
+--------+------------+-------------+
|      1 | Kui        |             |
|      2 | Arno       | pärisnimi   |
|      3 | isaga      | nimisõna    |
|      4 | koolimajja | nimisõna    |
|      5 | jõudis     | tegusõna    |
|      6 | ,          | lausemärk   |
|      7 | olid       | tegusõna    |
|      8 | tunnid     | nimisõna    |
```

```
|      9 | juba       | määrsõna    |
|     10 | alanud     |             |
|     11 | .          | lausemärk   |
+--------+------------+-------------+
11 rows in set (0.00 sec)
```

Tegevuse lõpetuseks andmebaasist välja

```
MariaDB [dh18_jaagup]> exit
Bye
```

Näited keelekorpuse andmetega

Veebis asuva faili alla tõmbamiseks sobib käsklus wget koos faili aadressiga

```
jaagup@praktika1 ~/public_html/2018/oma/11 $ wget
http://www.tlu.ee/~jaagup/andmed/keel/korpus/keelekorpus.sql.zip
```

Käsklus ühel real, lihtsalt siia praegu ei mahtunud muidu. Siis ekraanilt näha, kuidas alla laadimine edeneb

```
--2018-11-06 10:07:50--  http://www.tlu.ee/~jaagup/andmed/keel/korpus/keelekorpus.sql.zip
Resolving www.tlu.ee (www.tlu.ee)... 193.40.239.30
Connecting to www.tlu.ee (www.tlu.ee)|193.40.239.30|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 92848399 (89M) [application/zip]
Saving to: ‘keelekorpus.sql.zip’

keelekorpus.sql.zip 100%[===================>]  88,55M  8,29MB/s    in 11s

2018-11-06 10:08:01 (8,05 MB/s) - ‘keelekorpus.sql.zip’ saved [92848399/92848399]
```

Pakitud fail lahti. Läheb küll mitu korda suuremaks, aga siis käsud sees, millega uude baasi andmed saata

```
jaagup@praktika1 ~/public_html/2018/oma/11 $ unzip keelekorpus.sql.zip
Archive:  keelekorpus.sql.zip
  inflating: keelekorpus.sql.txt
```

Eraldi baas ehk tabelite komplekt keeleandmete jaosk - baas nimega dh18_keel

```
jaagup@praktika1 ~/public_html/2018/oma/11 $ mysqladmin -udh18 -pdh18praktika create
dh18_keel
```

Andmed failist käskudena sisse. SQL-laused on võimalik andmebaasile ka nõnda käsurealt saata.

```
jaagup@praktika1 ~/public_html/2018/oma/11 $ mysql -udh18 -pdh18praktika dh18_keel <
keelekorpus.sql.txt
```

Käskude ükshaaval andmiseks ise baasi sisse

```
jaagup@praktika1 ~ $ mysql -udh18 -pdh18praktika dh18_keel
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 273951
Server version: 10.0.24-MariaDB-7 Ubuntu 16.04

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [dh18_keel]>
```

Võõra baasiga tutvumiseks tasub kõigepealt vaadata tabelite loetelu

```
MariaDB [dh18_keel]> SHOW TABLES;
+----------------------+
| Tables_in_dh18_keel  |
+----------------------+
| dokarvud             |
| dokmeta              |
| doksonaliigid        |
| elukohad             |
| haridustasemed       |
| keeled               |
| keeletasemed         |
| korpusenimed         |
| ngram1               |
| ngram2               |
| ngram3               |
| ngram4               |
| ngram5               |
| sonaliikide_lyhendid |
| taustad              |
| tekstityybid         |
| vanusetasemed        |
+----------------------+
17 rows in set (0.00 sec)
```

Edasi saab juba ükshaaval tabelite sisse minna

```
MariaDB [dh18_keel]> SELECT * FROM sonaliikide_lyhendid;
+-------------+---------------------------+
| liigilyhend | liigikirjeldus            |
+-------------+---------------------------+
| A           | omadussõna algvõrre       |
| C           | omadussõna keskvõrre      |
| D           | määrsõna                  |
| G           | käändumatu omadussõna     |
| H           | pärisnimi                 |
```

```
| I           | hüüdsõna                  |
| J           | sidesõna                  |
| K           | kaassõna                  |
| N           | põhiarvsõna               |
| O           | järgarvsõna               |
| P           | asesõna                   |
| S           | nimisõna                  |
| U           | omadussõna ülivõrre       |
| V           | tegusõna                  |
| X           | verbi juurde kuuluv sõna  |
| Y           | lühend                    |
| Z           | lausemärk                 |
+-------------+---------------------------+
17 rows in set (0.00 sec)
```

Tabeli struktuuri kirjeldamiseks käsklus EXPLAIN. Sealt näeb, millised on tulpade nimed ja tüübid

```
MariaDB [dh18_keel]> EXPLAIN ngram1;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| tekstikood | varchar(50)  | NO   | PRI | NULL    |       |
| sona       | varchar(255) | NO   |     | NULL    |       |
| ngram1     | char(1)      | YES  | MUL | NULL    |       |
| alguskoht  | int(11)      | NO   | PRI | NULL    |       |
| suurtahega | varchar(50)  | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Edasi juba tabeli sisu ise. LIMIT-käsuga piirang peale, et suur kogus silme eest kirjuks ei ajaks.

```
MariaDB [dh18_keel]> SELECT * FROM ngram1 LIMIT 3;
+----------------------+---------+--------+-----------+------------+
| tekstikood           | sona    | ngram1 | alguskoht | suurtahega |
+----------------------+---------+--------+-----------+------------+
| doc_100636852915_item | suvel   | S      |         1 | Suvel      |
| doc_100636852915_item | ma      | P      |         2 | ma         |
| doc_100636852915_item | lugesin | V      |         3 | lugesin    |
+----------------------+---------+--------+-----------+------------+
3 rows in set (0.00 sec)
```

Nagu tulbad näitavad, siis iga sõna juures on kirjas teksti kood, et millisest tekstist võetud ning mitmenda sõnana. Sõnaliigi lühend väljas eraldi tulbas nimega ngram1. Et tabeli nimi ja tulba nimi kokku langevad, see õnneks segadust ei põhjusta.

Sõnaliigi lühendile inimkeelse seletuse kõrvale panekuks aitab tabelite sidumine JOIN-käskluse abil

```
SELECT * FROM ngram1
  JOIN sonaliikide_lyhendid ON ngram1.ngram1=sonaliikide_lyhendid.liigilyhend
```

Käivitasin päringu, aga see jäi pikemaks ajaks mõtlema. Meenutades selgus, et kolme miljoni sõna ühendamine ja välja kuvamine ongi suuremat sorti ettevõtmine.

```
MariaDB [dh18_keel]> SELECT * FROM ngram1
   ->   JOIN sonaliikide_lyhendid ON ngram1.ngram1=sonaliikide_lyhendid.liigilyhend
   -> ;
^CCtrl-C -- query killed. Continuing normally.
ERROR 1317 (70100): Query execution was interrupted
```

Vajutasin CTRL+C ning katkestasin päringu. Käivitasin uuesti ning lisasin LIMIT-käsu abil, et soovin vaid viit rida tutvumiseks näha. See tuleb ruttu

```
MariaDB [dh18_keel]> SELECT * FROM ngram1   JOIN sonaliikide_lyhendid ON
ngram1.ngram1=sonaliikide_lyhendid.liigilyhend LIMIT 5;

+----------------------+---------------+--------+-----------+---------------+-------------+----------------------+
| tekstikood           | sona          | ngram1 | alguskoht | suurtahega    | liigilyhend | liigikirjeldus       |
+----------------------+---------------+--------+-----------+---------------+-------------+----------------------+
| doc_100636852915_item | huvitav       | A      |        15 | huvitav       | A           | omadussõna algvõrre  |
| doc_100636852915_item | väikesest     | A      |        20 | väikesest     | A           | omadussõna algvõrre  |
| doc_100636852915_item | sõbralik      | A      |        31 | sõbralik      | A           | omadussõna algvõrre  |
| doc_100636852915_item | tähelepanelik | A      |        33 | tähelepanelik | A           | omadussõna algvõrre  |
| doc_100636852915_item | väikeses      | A      |        44 | väikeses      | A           | omadussõna algvõrre  |
+----------------------+---------------+--------+-----------+---------------+-------------+----------------------+
```

Tabeli ühendamisel pannakse tagumise tabeli vastavad veerud esimese tabeli omadele järele. Esimese tabeli viimane tulp on “suurtahega”, talle järgneb teise tabeli esimene tulp nimega “liigilyhend”. Nagu näha, siis tulbad ngram1 ja liigilyhend on samasuguse väärtusega nii nagu ka päringus kirjutatud. Ning kuna liigilühendite tabelis on iga lühendit ühekordselt (nagu ka sealne primaarvõti nõuab), siis saabki esimese tabeli reale konkreetse vaste.

Soovides ainult hetkel vajalikke tulpasid, tuleb nende nimed tärni asemel kirjutada päringusse

```
MariaDB [dh18_keel]> SELECT sona, liigilyhend, liigikirjeldus FROM ngram1   JOIN
sonaliikide_lyhendid ON ngram1.ngram1=sonaliikide_lyhendid.liigilyhend LIMIT 5;
+---------------+-------------+----------------------+
| sona          | liigilyhend | liigikirjeldus       |
+---------------+-------------+----------------------+
| huvitav       | A           | omadussõna algvõrre  |
| väikesest     | A           | omadussõna algvõrre  |
| sõbralik      | A           | omadussõna algvõrre  |
| tähelepanelik | A           | omadussõna algvõrre  |
| väikeses      | A           | omadussõna algvõrre  |
+---------------+-------------+----------------------+
5 rows in set (0.00 sec)
```

Tekstide võrdlus

Küsin tabelist ngram1 kaks esimest erinevat tekstikoodi

```
MariaDB [dh18_keel]> SELECT DISTINCT tekstikood FROM ngram1 ORDER BY tekstikood LIMIT 2;
+-----------------------+
| tekstikood            |
+-----------------------+
| doc_100636852915_item |
| doc_100636852916_item |
+-----------------------+
```

Mitme sõna andmed on esimese teksti juures

```
MariaDB [dh18_keel]> SELECT COUNT(*) FROM ngram1 WHERE tekstikood='doc_100636852915_item';
+----------+
| COUNT(*) |
+----------+
|      215 |
+----------+
```

Sealtkaudu saab iga sõnaliigi kohta arvutada selle osakaalu tekstis - jagame vastava sõnaliigi koguarvu sõnade arvuga tekstis, mis praegu on 215

```
MariaDB [dh18_keel]>   SELECT ngram1, COUNT(*) / 215 AS osakaal FROM ngram1
    ->  WHERE tekstikood='doc_100636852915_item'
    ->  GROUP BY ngram1;
+--------+---------+
| ngram1 | osakaal |
+--------+---------+
| A      |  0.1163 |
| D      |  0.0837 |
| H      |  0.0140 |
| J      |  0.0884 |
| K      |  0.0233 |
| N      |  0.0140 |
| P      |  0.0791 |
| S      |  0.2512 |
| V      |  0.1628 |
| Z      |  0.1674 |
+--------+---------+
```

Sõnade koguarvu võib ka otse arvutamise käigus eraldi alampäringuga sulgude sees arvutada

```
 SELECT ngram1, COUNT(*) /
   (SELECT COUNT(*) FROM ngram1 WHERE tekstikood='doc_100636852915_item')
  AS osakaal FROM ngram1
 WHERE tekstikood='doc_100636852915_item'
 GROUP BY ngram1


+--------+---------+
| ngram1 | osakaal |
+--------+---------+
| A      |  0.1163 |
| D      |  0.0837 |
| H      |  0.0140 |
| J      |  0.0884 |
| K      |  0.0233 |
| N      |  0.0140 |
```

```
| P       |  0.0791 |
| S       |  0.2512 |
| V       |  0.1628 |
| Z       |  0.1674 |
+--------+---------+
```

Või siis veel põhjalikum moodus, kus tuleb tekstikood kirjutada vaid ühte kohta. Sisemine päring võtab tekstikoodi välimisest päringust. Et tabeli nimedega segadust ei tekiks, siis ühel juhul antakse tabeli ajutiseks nimeks AS-i abil sisemine, teisel v2limine.

```
  SELECT ngram1, COUNT(*) /
   (SELECT COUNT(*) FROM ngram1 AS sisemine WHERE  sisemine.tekstikood=v2limine.tekstikood)
  AS osakaal FROM ngram1 AS v2limine
 WHERE tekstikood='doc_100636852916_item'
 GROUP BY ngram1
```

```
+--------+---------+
| ngram1 | osakaal |
+--------+---------+
| A       |  0.0417 |
| D       |  0.0500 |
| H       |  0.0333 |
| J       |  0.1000 |
| K       |  0.0083 |
| N       |  0.0250 |
| O       |  0.0083 |
| P       |  0.1167 |
| S       |  0.2583 |
| V       |  0.1833 |
| Z       |  0.1750 |
+--------+---------+
```

Kahe teksti võrdlemiseks kõigepealt sõnaliikide sagedused kummaski tekstis

```
MariaDB [dh18_keel]> SELECT ngram1, COUNT(*) FROM ngram1
    ->  WHERE tekstikood='doc_100636852915_item'
    ->  GROUP BY ngram1;
+--------+----------+
| ngram1 | COUNT(*) |
+--------+----------+
| A      |       25 |
| D      |       18 |
| H      |        3 |
| J      |       19 |
| K      |        5 |
| N      |        3 |
| P      |       17 |
| S      |       54 |
| V      |       35 |
| Z      |       36 |
+--------+----------+
10 rows in set (0.00 sec)
```

Teisel tekstis vahe vaid üks number tekstikoodis

```
MariaDB [dh18_keel]> SELECT ngram1, COUNT(*) FROM ngram1
    ->  WHERE tekstikood='doc_100636852916_item'
    ->  GROUP BY ngram1;
+--------+----------+
| ngram1 | COUNT(*) |
+--------+----------+
| A      |        5 |
| D      |        6 |
| H      |        4 |
| J      |       12 |
| K      |        1 |
| N      |        3 |
| O      |        1 |
| P      |       14 |
| S      |       31 |
| V      |       22 |
| Z      |       21 |
+--------+----------+
11 rows in set (0.00 sec)
```

Nüüd kaks päringut kokku ja JOINi abil tulbad kõrvuti

```
SELECT * FROM
(SELECT ngram1, COUNT(*) AS kogus15 FROM ngram1
 WHERE tekstikood='doc_100636852915_item'
 GROUP BY ngram1) AS tabel1
  JOIN
  (SELECT ngram1, COUNT(*) AS kogus16 FROM ngram1
 WHERE tekstikood='doc_100636852916_item'
 GROUP BY ngram1) AS tabel2
  ON tabel1.ngram1=tabel2.ngram1;

+--------+---------+--------+---------+
| ngram1 | kogus15 | ngram1 | kogus16 |
+--------+---------+--------+---------+
| A      |      25 | A      |       5 |
| D      |      18 | D      |       6 |
| H      |       3 | H      |       4 |
| J      |      19 | J      |      12 |
| K      |       5 | K      |       1 |
| N      |       3 | N      |       3 |
| P      |      17 | P      |      14 |
| S      |      54 | S      |      31 |
| V      |      35 | V      |      22 |
| Z      |      36 | Z      |      21 |
+--------+---------+--------+---------+
```

Kuna tekstid on tõenäoliselt mõnevõrra erineva pikkusega, siis sobivam võrrelda sõnaliikide osakaale kui absoluutarve. Nii tuleb mõlema tabeli andmete juurde jagamine tekstis olevate sõnade üldarvuga

```
 SELECT tabel1.ngram1, tabel1.osakaal AS osakaal15,
   tabel2.osakaal AS osakaal16 FROM
(SELECT ngram1, COUNT(*) /
```

```
   (SELECT COUNT(*) FROM ngram1 AS sisemine WHERE
sisemine.tekstikood=v2limine.tekstikood)
  AS osakaal FROM ngram1 AS v2limine
 WHERE tekstikood='doc_100636852915_item'
 GROUP BY ngram1) AS tabel1
  JOIN
  (SELECT ngram1, COUNT(*) /
   (SELECT COUNT(*) FROM ngram1 AS sisemine WHERE
sisemine.tekstikood=v2limine.tekstikood)
  AS osakaal FROM ngram1 AS v2limine
 WHERE tekstikood='doc_100636852916_item'
 GROUP BY ngram1) AS tabel2
  ON tabel1.ngram1=tabel2.ngram1;
```

```
+--------+-----------+-----------+
| ngram1 | osakaal15 | osakaal16 |
+--------+-----------+-----------+
| A      |    0.1163 |    0.0417 |
| D      |    0.0837 |    0.0500 |
| H      |    0.0140 |    0.0333 |
| J      |    0.0884 |    0.1000 |
| K      |    0.0233 |    0.0083 |
| N      |    0.0140 |    0.0250 |
| P      |    0.0791 |    0.1167 |
| S      |    0.2512 |    0.2583 |
| V      |    0.1628 |    0.1833 |
| Z      |    0.1674 |    0.1750 |
+--------+-----------+-----------+
```

Eripärade välja selgitamiseks võib osakaalu ühes tekstis jagada osakaaluga teises tekstis - siis on näha, mitu korda kummaski vastavat sõnaliiki rohkem on. Vastuse järgi paistab, et pärisnimede (H) tekstist koodilõpuga 15 moodustab vaid 0.42 osa nende esinemistest tekstis koodilõpuga 16. Absoluutarvud samas 3 ja 4 nagu eelnevast päringust paistab. Kaassõnade osakaal esimeses tekstis jällegi üle kahe korra suurem, absoluutarvud 5 ja 1.

```
 SELECT tabel1.ngram1, tabel1.osakaal AS osakaal15,
   tabel2.osakaal AS osakaal16, tabel1.osakaal/tabel2.osakaal AS suhe FROM
(SELECT ngram1, COUNT(*) /
   (SELECT COUNT(*) FROM ngram1 AS sisemine WHERE
sisemine.tekstikood=v2limine.tekstikood)
  AS osakaal FROM ngram1 AS v2limine
 WHERE tekstikood='doc_100636852915_item'
 GROUP BY ngram1) AS tabel1
  JOIN
  (SELECT ngram1, COUNT(*) /
   (SELECT COUNT(*) FROM ngram1 AS sisemine WHERE
sisemine.tekstikood=v2limine.tekstikood)
  AS osakaal FROM ngram1 AS v2limine
 WHERE tekstikood='doc_100636852916_item'
 GROUP BY ngram1) AS tabel2
  ON tabel1.ngram1=tabel2.ngram1
  ORDER BY suhe;
```

```
+--------+-----------+-----------+------------+
| ngram1 | osakaal15 | osakaal16 | suhe       |
```

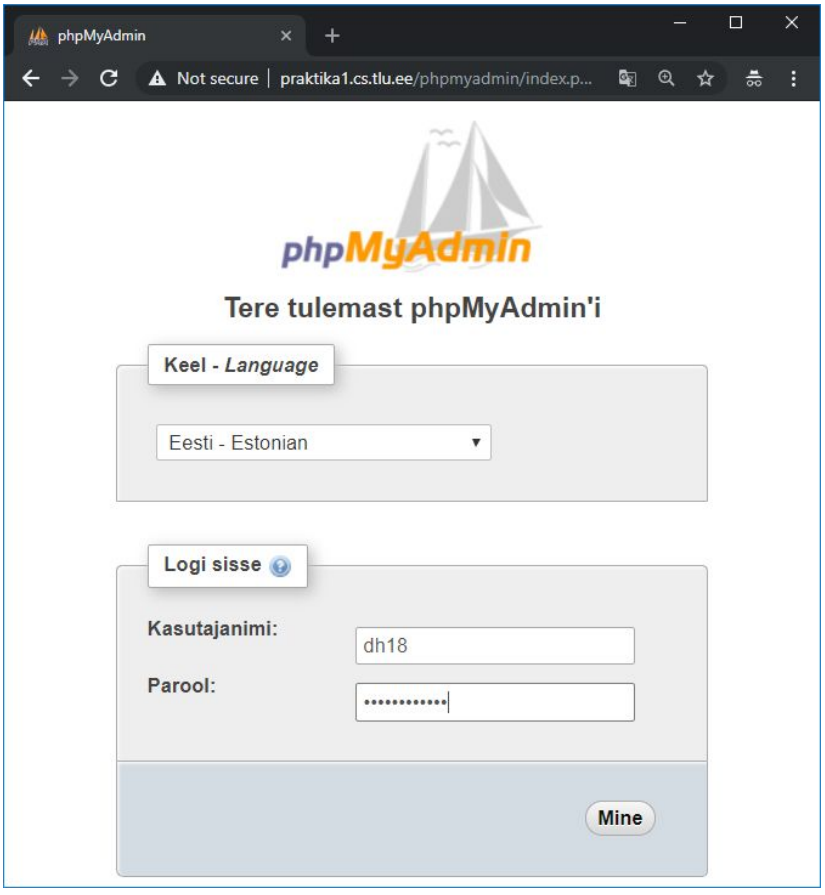
```
+--------+-----------+-----------+------------+
| H      |    0.0140 |    0.0333 | 0.42042042 |
| N      |    0.0140 |    0.0250 | 0.56000000 |
| P      |    0.0791 |    0.1167 | 0.67780634 |
| J      |    0.0884 |    0.1000 | 0.88400000 |
| V      |    0.1628 |    0.1833 | 0.88816148 |
| Z      |    0.1674 |    0.1750 | 0.95657143 |
| S      |    0.2512 |    0.2583 | 0.97251258 |
| D      |    0.0837 |    0.0500 | 1.67400000 |
| A      |    0.1163 |    0.0417 | 2.78896882 |
| K      |    0.0233 |    0.0083 | 2.80722892 |
+--------+-----------+-----------+------------+
```

Pikemate järelduste tegemiseks tuleb andmeid lähemalt uurida. Kirjeldavana aga on vähemasti, mida konkreetsel juhul kusagil mitu korda parajasti rohkem leiab. Kuna praegu näha vaid sõnaliigid, mis mõlemas tekstis olemas, siis pääseme nulliga jagamisel tekkivast määramatusest - mõnikord aga tuleb ka sellega arvestada.

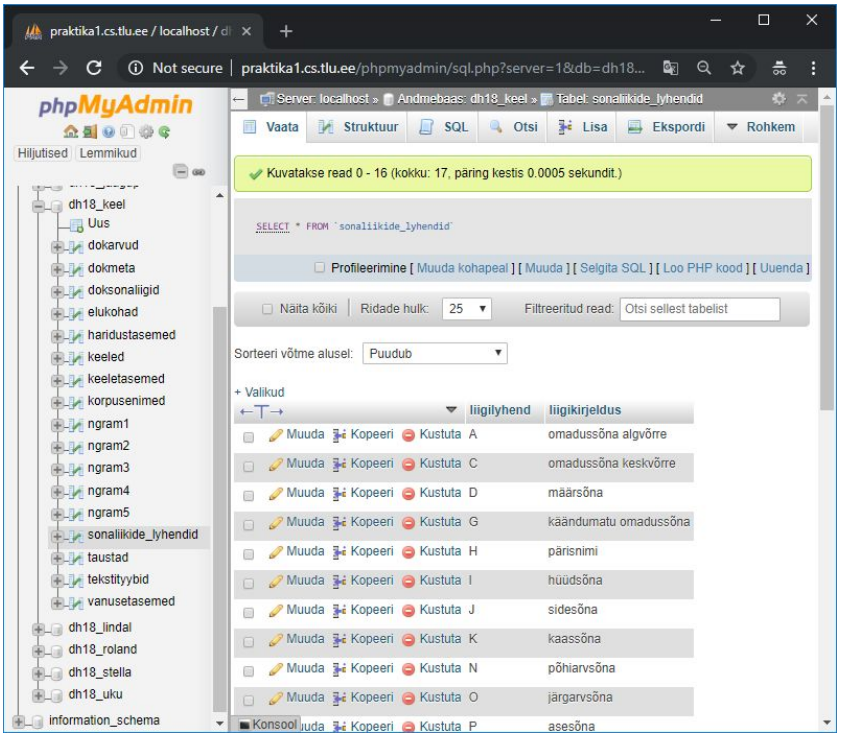
PhpMyAdmin

Näited siiani tekstiakna kaudu tehtud. Mõnikord on aga andmebaasile mugavam või ka ainus võimalus ligi pääseda veebi või muu graafilise liidese kaudu. Üheks selliseks on phpMyAdmin

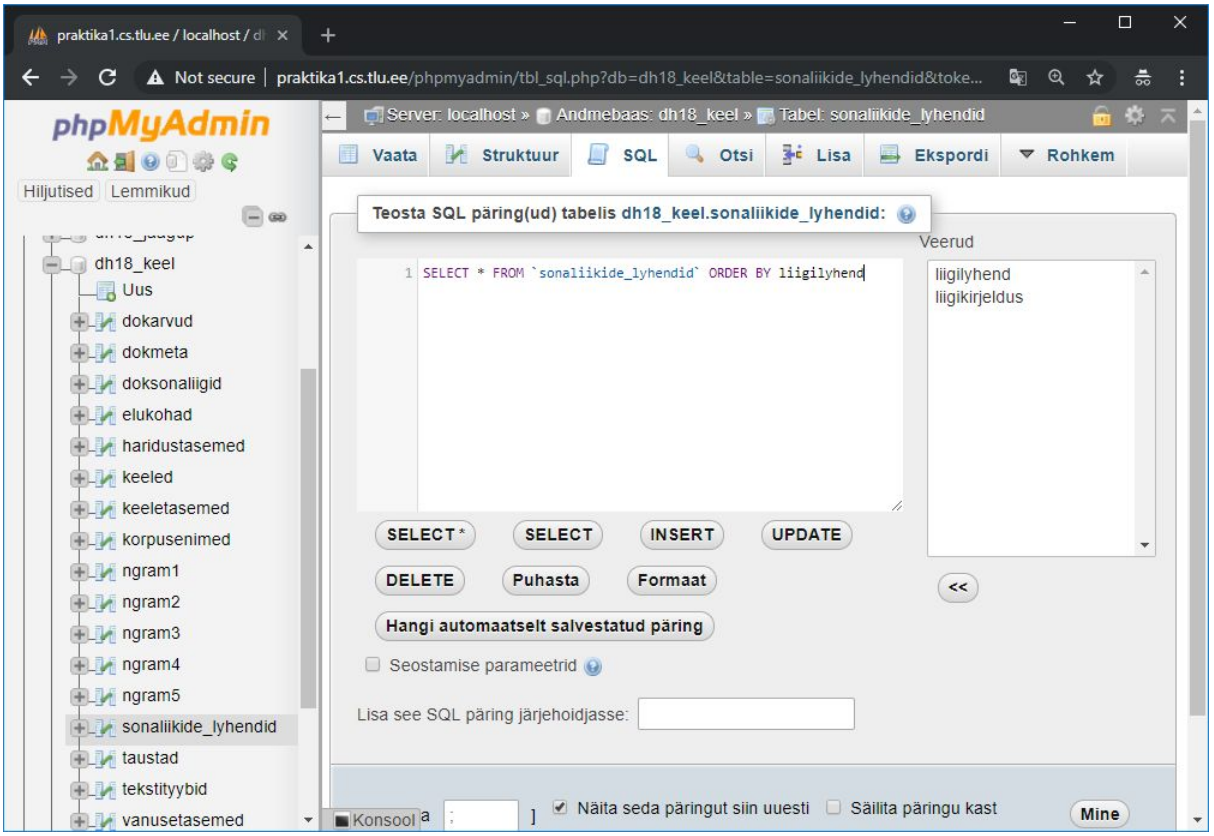
Kui lahendus installitud ja kättesaadav, siis tuleb kasutajatunnuse ja parooliga siseneda

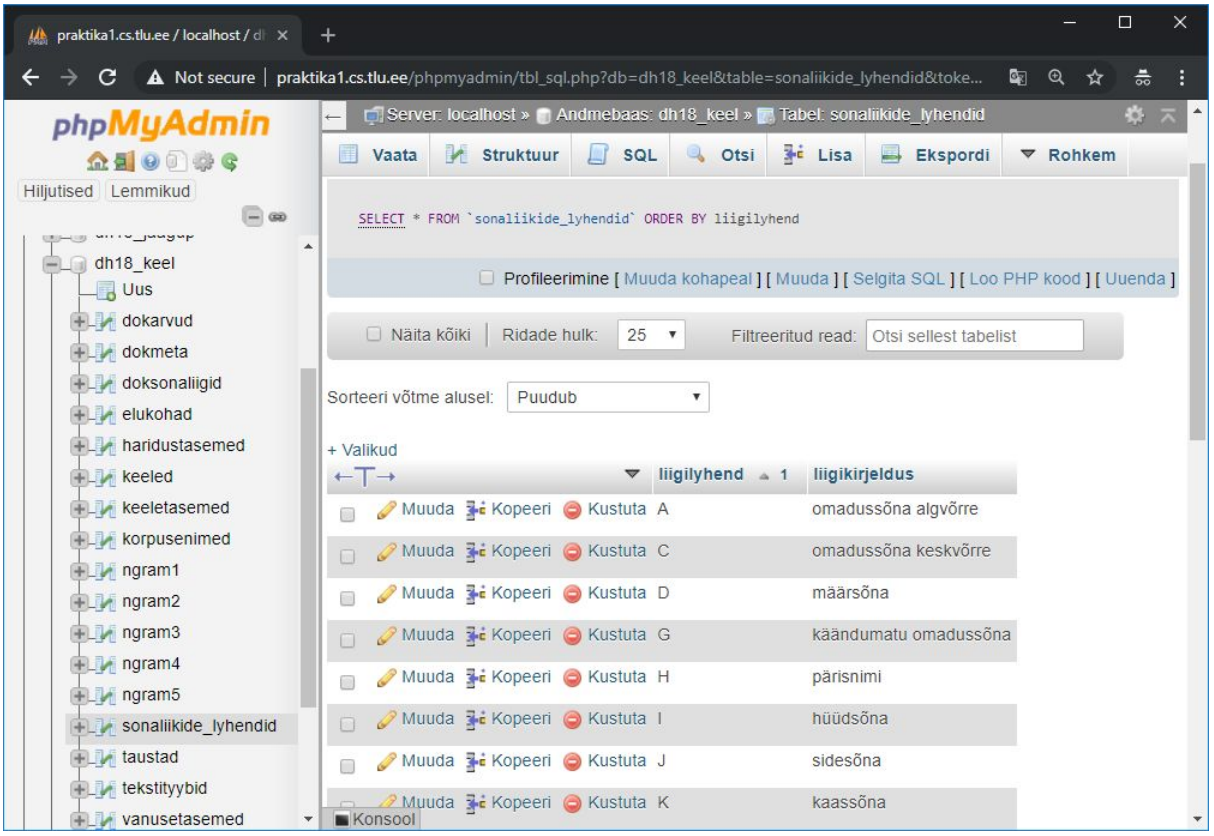


Pärast sisenemist näha kättesaadava andmebaasid, sealsed tabelid ning valitud tabeli sisu



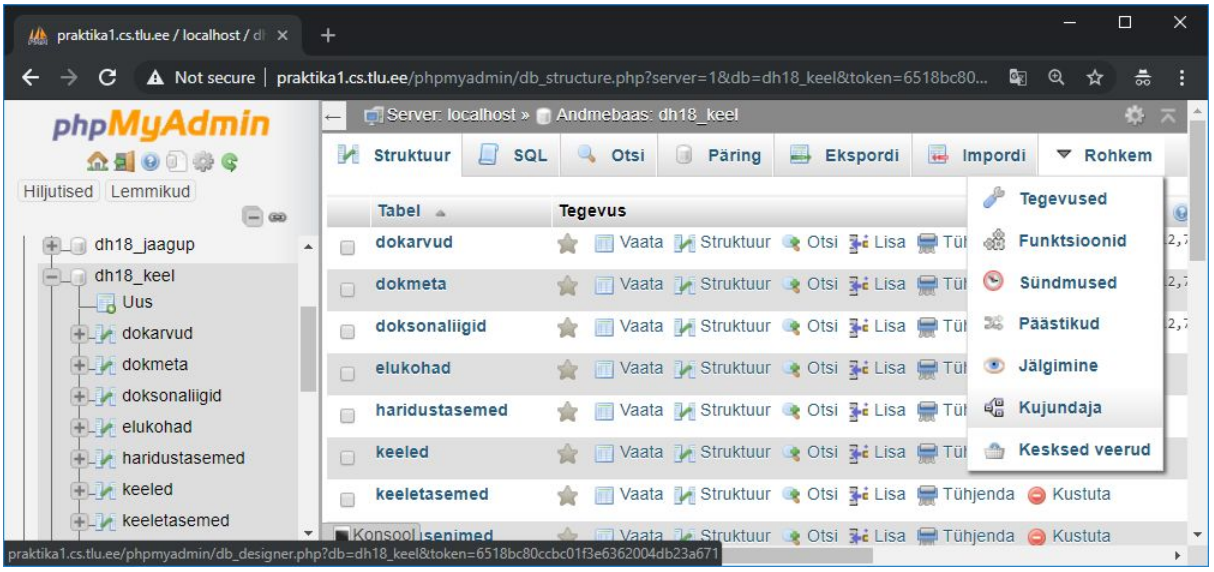
Ülevalt vastava valiku alt on võimalik sisestada SQL-käsklusi ning vaadata nende töö tulemusi



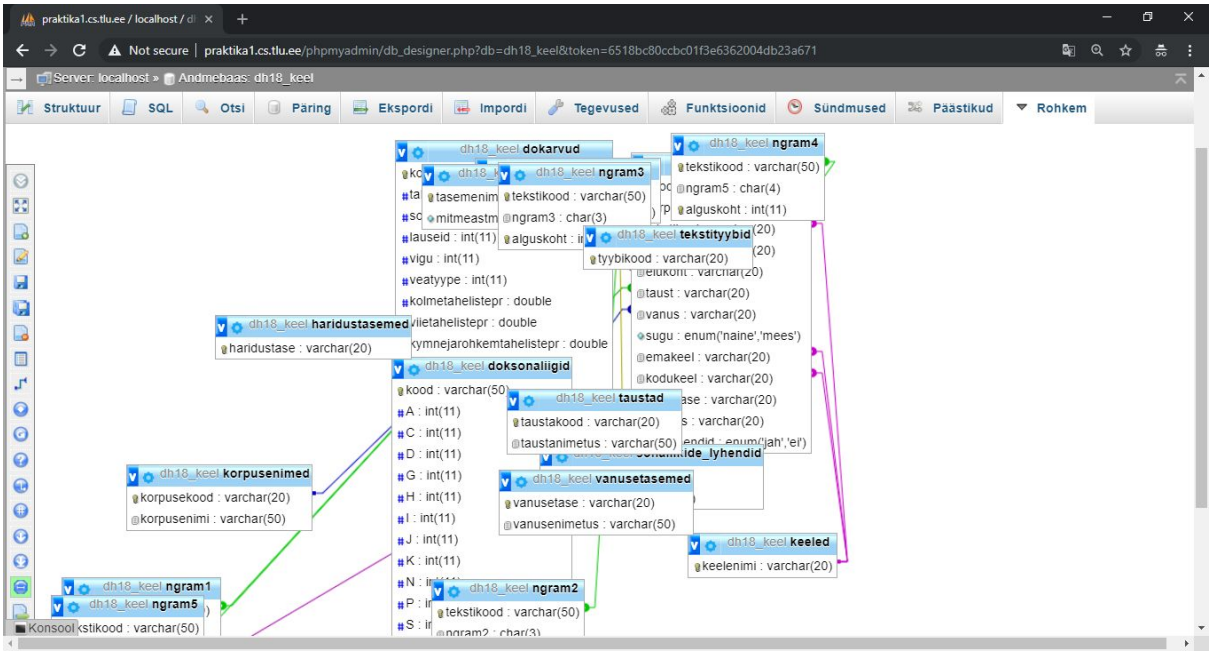


Andmebaasiskeem

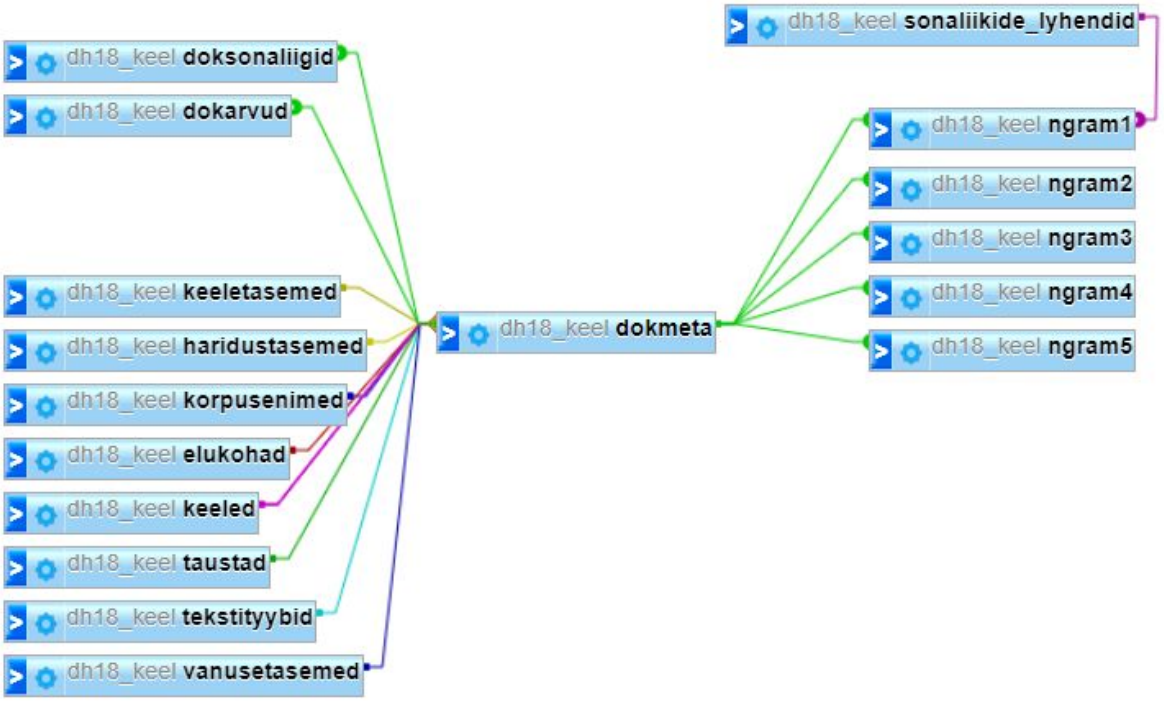
Suuremast andmestikust ülevaate saamiseks on kasulik andmebaasiskeem. Selle vaatamiseks on phpMyAdmini juures olemas kujundaja/designer. Vastava valikuni jõudmiseks tuleb kõigepealt vasakmenüüs valida vastav andmebaas ning siis saab ülamenüüst lisavalikute juurest kätte kujundaja.



Esmasel pilgul satuvad tabeli suhteliselt juhuslikult üle ekraani. Jooned nende vahel näitavad, et milliste tabelite millised tulbad teistega seotud on.



Veidi hiirega tõstmist ja ristumiste vähendamist ning pilti on juba tunduvalt meeldivam vaadata

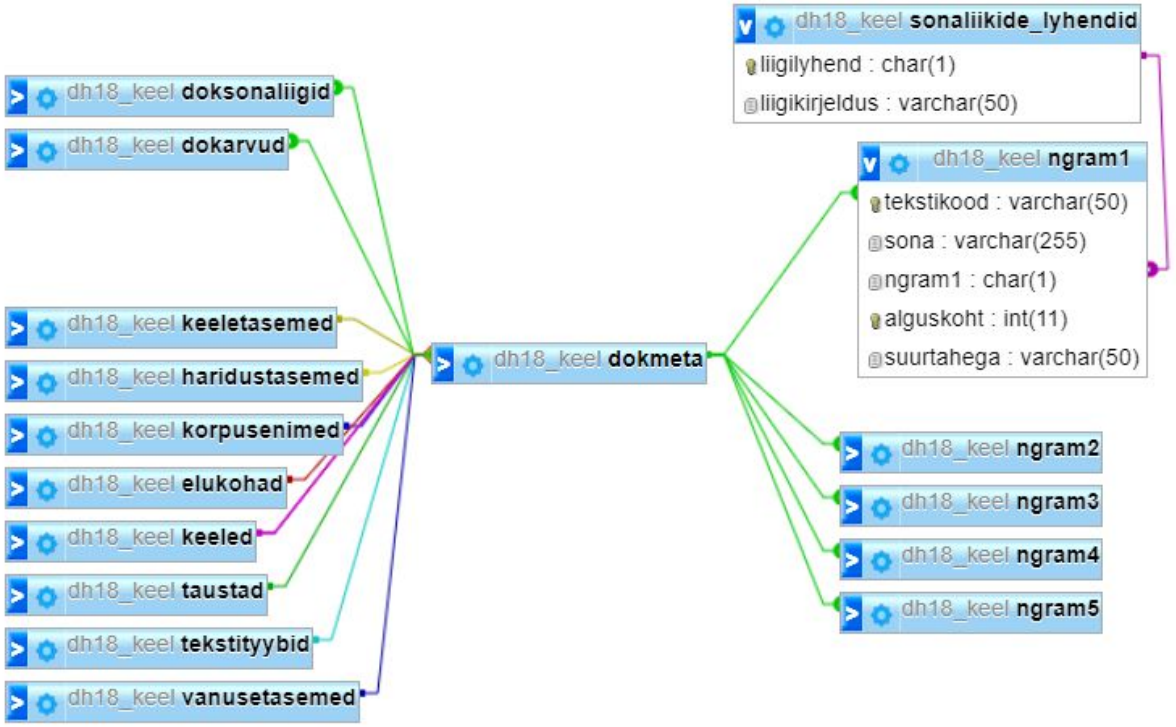


Nagu paistab, on siinse andmebaasi keskseks tabeliks dokmeta. Sellega üheks seotud grupiks on eri pikkusega ngramid, teiseks seltskonnaks dokumentide metaandmete

võimaluste loetelud - keeletasemed, haridustasemed jm. Eraldi veel arvulise andmed dokumentide sõnaliikide sageduste ja muude arvuliste andmete kohta ja veidi eraliseisva abitabelina sõnaliikide kirjeldused.

Kui serveri konfiguratsioon võimaldab, siis saab tabelite paigutuse ka eraldi lehele salvestada, et sea tulevikus taas võimalik vaadata oleks.

Tabeli nimede alt pääseb sisse vaatama - et millised tulbad tabelites on ning millised tulbad omavahel seotud.



Sõnaliigipaaride sageduste võrdlus

Lähemat infot tabelite kohta võib küsida käsuga EXPLAIN. Siit paistavad tulpade nimed ja tüübid

```
MariaDB [dh18_keel]> EXPLAIN ngram2;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| tekstikood | varchar(50) | NO   | PRI | NULL    |       |
| ngram2     | char(3)     | NO   |     | NULL    |       |
| alguskoht  | int(11)     | NO   | PRI | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Näitele selgust juurde annab ka tegelike andmete vaatamine

```
MariaDB [dh18_keel]> SELECT * FROM ngram2 LIMIT 3;
+-----------------------+--------+-----------+
| tekstikood            | ngram2 | alguskoht |
+-----------------------+--------+-----------+
| doc_100636852915_item | SP     |         1 |
| doc_100636852915_item | PV     |         2 |
| doc_100636852915_item | VD     |         3 |
+-----------------------+--------+-----------+
3 rows in set (0.00 sec)
```

Võrdlusena kõrvale teksti sõnad koos sõnaliikidega. Paaridesse lihtsalt koondatakse kaks järjestikust sõnaliiki. Nende juurest saavad välja kooruma hakata keelekasutusmustrid

```
MariaDB [dh18_keel]> SELECT * FROM ngram1 LIMIT 4;
+-----------------------+---------+--------+-----------+------------+
| tekstikood            | sona    | ngram1 | alguskoht | suurtahega |
+-----------------------+---------+--------+-----------+------------+
| doc_100636852915_item | suvel   | S      |         1 | Suvel      |
| doc_100636852915_item | ma      | P      |         2 | ma         |
| doc_100636852915_item | lugesin | V      |         3 | lugesin    |
| doc_100636852915_item | läbi    | D      |         4 | läbi       |
+-----------------------+---------+--------+-----------+------------+
4 rows in set (0.01 sec)
```

Tabelite sidumise abil paigutame ngram2 tabeli andmete kõrvale vastava järjekorranumbriga sõna samast tekstist tabelist ngram1

```
SELECT ngram2.tekstikood, ngram2.ngram2, ngram2.alguskoht, ngram1.suurtahega
 FROM ngram2
  JOIN ngram1 ON
    ngram2.tekstikood=ngram1.tekstikood AND ngram2.alguskoht=ngram1.alguskoht
  LIMIT 3;
```

```
+-----------------------+--------+-----------+------------+
| tekstikood            | ngram2 | alguskoht | suurtahega |
+-----------------------+--------+-----------+------------+
| doc_100636852915_item | SP     |         1 | Suvel      |
| doc_100636852915_item | PV     |         2 | ma         |
| doc_100636852915_item | VD     |         3 | lugesin    |
+-----------------------+--------+-----------+------------+
3 rows in set (0.01 sec)
```

Võrdlusena juurde paari tagumise sõna andmed, ehk siis algses tekstis ühe võrra suurema järjekorranumbriga sõnad

```
SELECT ngram2.tekstikood, ngram2.ngram2, ngram2.alguskoht, ngram1.suurtahega
 FROM ngram2
  JOIN ngram1 ON
    ngram2.tekstikood=ngram1.tekstikood AND
         ngram2.alguskoht+1=ngram1.alguskoht
  LIMIT 3;
```

```
+-----------------------+--------+-----------+------------+
| tekstikood            | ngram2 | alguskoht | suurtahega |
+-----------------------+--------+-----------+------------+
| doc_100636852915_item | SP     |         1 | ma         |
| doc_100636852915_item | PV     |         2 | lugesin    |
| doc_100636852915_item | VD     |         3 | läbi       |
+-----------------------+--------+-----------+------------+
3 rows in set (0.00 sec)
```

Kahe järjestikuse JOINi abil õnnestub mõlemad sõnad sõnaliigipaarile külge haakida ning paistab välja, et millised sõnad siis tegelikult selle tähepaari all peidus on

```
SELECT ngram2.tekstikood, ngram2.ngram2, ngram2.alguskoht,
   abitabel1.suurtahega AS sona1,  abitabel2.suurtahega AS sona2
 FROM ngram2
  JOIN ngram1 AS abitabel1 ON
    ngram2.tekstikood=abitabel1.tekstikood AND
         ngram2.alguskoht=abitabel1.alguskoht
  JOIN ngram1 AS abitabel2 ON
    ngram2.tekstikood=abitabel2.tekstikood AND
         ngram2.alguskoht+1=abitabel2.alguskoht
  LIMIT 3;
```

```
+-----------------------+--------+-----------+---------+---------+
| tekstikood            | ngram2 | alguskoht | sona1   | sona2   |
+-----------------------+--------+-----------+---------+---------+
| doc_100636852915_item | SP     |         1 | Suvel   | ma      |
| doc_100636852915_item | PV     |         2 | ma      | lugesin |
| doc_100636852915_item | VD     |         3 | lugesin | läbi    |
+-----------------------+--------+-----------+---------+---------+
```

Dokumentide metaandmed

Õppijakeele korpusele andmeid kogudes märgiti tekstidele võimalusel juurde andmed teksti autori kohta. Tulbad näha järgnevas päringus

```
MariaDB [dh18_keel]> EXPLAIN dokmeta;
+-------------+----------------------+------+-----+---------+-------+
| Field       | Type                 | Null | Key | Default | Extra |
+-------------+----------------------+------+-----+---------+-------+
| kood        | varchar(50)          | NO   | PRI | NULL    |       |
| korpus      | varchar(20)          | NO   | MUL | NULL    |       |
| tekstikeel  | varchar(20)          | YES  | MUL | NULL    |       |
| tekstityyp  | varchar(20)          | YES  | MUL | NULL    |       |
| elukoht     | varchar(20)          | YES  | MUL | NULL    |       |
| taust       | varchar(20)          | YES  | MUL | NULL    |       |
| vanus       | varchar(20)          | YES  | MUL | NULL    |       |
| sugu        | enum('naine','mees') | YES  |     | NULL    |       |
| emakeel     | varchar(20)          | YES  | MUL | NULL    |       |
| kodukeel    | varchar(20)          | YES  | MUL | NULL    |       |
| keeletase   | varchar(20)          | YES  | MUL | NULL    |       |
| haridus     | varchar(20)          | YES  | MUL | NULL    |       |
| abivahendid | enum('jah','ei')     | YES  |     | NULL    |       |
+-------------+----------------------+------+-----+---------+-------+
```

Näitena esimese nelja rea väärtused. Kuna tulpasid on andmete juures palju, siis tekst murtakse

```
MariaDB [dh18_keel]> SELECT * FROM dokmeta limit 4;
+-----------------------+-----------+------------+------------+---------+-------+--------+-
------+---------+----------+-----------+---------+-------------+
| kood                  | korpus    | tekstikeel | tekstityyp | elukoht | taust | vanus  |
sugu  | emakeel | kodukeel | keeletase | haridus | abivahendid |
+-----------------------+-----------+------------+------------+---------+-------+--------+--
-----+---------+----------+-----------+---------+-------------+
| doc_100636852915_item | cFOoRQekA | eesti      | essee      | idaviru | op    | kuni18 |
naine | vene    | vene     | B         | pohi    | ei          |
| doc_100636852916_item | cFOoRQekA | eesti      | muu        | idaviru | op    | kuni18 |
naine | vene    | vene     | B         | pohi    | ei          |
| doc_100636852917_item | cFOoRQekA | eesti      | essee      | idaviru | op    | kuni18 |
naine | vene    | vene     | B         | pohi    | ei          |
| doc_1010138197_item   | cFOoRQekA | eesti      | muu        | tallinn | ylop  | kuni26 |
naine | vene    | vene     | A         | kesk    | ei          |
+-----------------------+-----------+------------+------------+---------+-------+--------+-
------+---------+----------+-----------+---------+-------------+
4 rows in set (0.01 sec)
```

Vaid teksti koodi, autori emakeele ja teksti keele näeb siis, kui need tulba eraldi välja küsida

```
MariaDB [dh18_keel]> SELECT kood, emakeel, tekstikeel FROM dokmeta LIMIT 4;
+-----------------------+---------+------------+
| kood                  | emakeel | tekstikeel |
+-----------------------+---------+------------+
| doc_100636852915_item | vene    | eesti      |
| doc_100636852916_item | vene    | eesti      |
| doc_100636852917_item | vene    | eesti      |
```

```
| doc_1010138197_item   | vene    | eesti      |
+-----------------------+---------+------------+
4 rows in set (0.00 sec)
```

Teksti autorite emakeelte sagedused kahanevas järjekorras - suuremad eespool. NULL loetelu alguses tähendab, et rohkem kui pooltel juhtudel pole teksti autori emakeel andmebaasis märgitud

```
MariaDB [dh18_keel]> SELECT emakeel, COUNT(*) FROM dokmeta GROUP BY emakeel ORDER BY
COUNT(*) DESC;
+-----------+----------+
| emakeel   | COUNT(*) |
+-----------+----------+
| NULL      |     8576 |
| vene      |     3184 |
| soome     |      391 |
| eesti     |      229 |
| inglise   |      119 |
| saksa     |       86 |
| muud      |       81 |
| leedu     |       24 |
| ukraina   |       17 |
| ungari    |        6 |
| poola     |        3 |
| rootsi    |        2 |
| lati      |        2 |
| valgevene |        1 |
| jidis     |        1 |
| katalaani |        1 |
| sloveenia |        1 |
+-----------+----------+
17 rows in set (0.01 sec)
```

Kuna andmestikus on märgatavalt ka venekeelseid tekste, siis eestikeelsete tekstide autorite emakeelte leidmiseks tuleb ka teksti keel eraldi ära määrata

```
MariaDB [dh18_keel]> SELECT emakeel, COUNT(*) FROM dokmeta WHERE tekstikeel='eesti' GROUP
BY emakeel ORDER BY COUNT(*) DESC;
+-----------+----------+
| emakeel   | COUNT(*) |
+-----------+----------+
| NULL      |     8270 |
| vene      |     2814 |
| soome     |      391 |
| eesti     |      229 |
| inglise   |      119 |
| saksa     |       86 |
| muud      |       81 |
| leedu     |       24 |
| ukraina   |       17 |
| ungari    |        6 |
| poola     |        3 |
| lati      |        2 |
| rootsi    |        2 |
| jidis     |        1 |
| sloveenia |        1 |
| valgevene |        1 |
| katalaani |        1 |
```

```
+-----------+----------+
17 rows in set (0.07 sec)
```

Sõnaliikide paarid

Tekstide metaandmed ning tekstide sõnaliikide andmed aitab omavahel kokku ühendada tabeleid siduv käsklus JOIN. Siin näites esimesed paarid vene emakeelega autorite sõnaliigipaaridest.

```
  SELECT ngram2.ngram2 FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene'
   LIMIT 10;
```

```
+--------+
| ngram2 |
+--------+
| SP     |
| PV     |
| VD     |
| DS     |
| SJ     |
| JS     |
| SZ     |
| ZJ     |
| JS     |
| SZ     |
+--------+
```

Rühmitades saab kätte, et millist sõnaliigipaari kui palju esineb

```
  SELECT ngram2.ngram2, COUNT(*) FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10;
```

```
+--------+----------+
| ngram2 | COUNT(*) |
+--------+----------+
| SZ     |   126756 |
| SS     |   124720 |
| ZS     |    57566 |
| SV     |    44944 |
| YS     |    37804 |
| PV     |    35655 |
| VS     |    35072 |
| AS     |    31634 |
| ZP     |    30287 |
| PS     |    27224 |
+--------+----------+
```

Kuna baasis tekste aga mitmes keeles, siis tasub lisaks autorite vene emakeelele juurde märkida, et uuritakse eestikeelseid tekste. Nagu näha, siis levinuima paari sagedus on niimoodi ligi kolmandiku jagu väiksem, samuti järjestus mõnevõrra teistsugune. Esimeses loetelus neljandal kohal olev SV tõusis vaid eestikeelseid tekste arvestades teisele kohale. Ülemises tabelis juhtiva SZ-iga sageduselt peaaegu võrdne SS on vaid eestikeelsete tekstide puhul SZ-ist rohkem kui kaks korda väiksema sagedusega.

```
  SELECT ngram2.ngram2, COUNT(*) FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10;
```

```
+--------+----------+
| ngram2 | COUNT(*) |
+--------+----------+
| SZ     |    90053 |
| SV     |    44944 |
| SS     |    41562 |
| ZS     |    37216 |
| PV     |    35655 |
| VS     |    35072 |
| AS     |    31634 |
| ZP     |    30287 |
| PS     |    27224 |
| VZ     |    26317 |
+--------+----------+
```

Sõnaliigipaaride sageduste võrdlus keeleti

Soome emakeelega autorite sõnaliigipaare paistab olema 120000

```
MariaDB [dh18_keel]> SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON
dokmeta.kood=ngram2.tekstikood WHERE dokmeta.emakeel='soome' AND
dokmeta.tekstikeel='eesti';
+----------+
| COUNT(*) |
+----------+
|   120630 |
+----------+
```

Vene emakeelega autorite omi aga ligi miljon

```
MariaDB [dh18_keel]> SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON
dokmeta.kood=ngram2.tekstikood WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti';
+----------+
| COUNT(*) |
+----------+
|   944848 |
+----------+
```

Järelikult otse arvuline võrdlemine ei sobi. Küll aga saab võrrelda levinumate sõnaliigipaaride järjestust või osakaale. Viimase nüüd ette võtamegi. Jagame sõnaliigipaaride sageduse vastava keele sõnaliigipaaride koguarvuga.

```
  SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti') AS veneosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10;
```

Sealt näha, et levinuma paari - SZ - esinemisosakaal on 0.095 ehk peaaegu 10%

```
+--------+-------------+
| ngram2 | veneosakaal |
+--------+-------------+
| SZ     |      0.0953 |
| SV     |      0.0476 |
| SS     |      0.0440 |
| ZS     |      0.0394 |
| PV     |      0.0377 |
| VS     |      0.0371 |
| AS     |      0.0335 |
| ZP     |      0.0321 |
| PS     |      0.0288 |
| VZ     |      0.0279 |
+--------+-------------+
```

Harjutus

- Leidke soome emakeelega inimeste levinumad sõnaliigipaaride osakaalud

Lahenduses tuleb sobivatel kohtadel keel ära vahetada

```
     SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='soome' AND dokmeta.tekstikeel='eesti') AS soomeosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='soome' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10;
```

Paistab, et soomekeelsete tekstide puhul näiteks üleval neljandal kohal olev järgnevus ZS jääb sootuks esikümnest välja. Kas see erinevus on juhuslik, sõltub valitud tekstidest või vastava emakeele mõjudest on juba eraldi uurimisküsimus, esmane võrdlus aga annab märku, et sealt on põhjust midagi lähemalt vaadata.

```
+--------+--------------+
| ngram2 | soomeosakaal |
+--------+--------------+
| SZ     |       0.0754 |
| SV     |       0.0501 |
| SS     |       0.0450 |
| PV     |       0.0433 |
| VS     |       0.0389 |
| AS     |       0.0338 |
| ZP     |       0.0333 |
| VD     |       0.0326 |
| PS     |       0.0300 |
| VA     |       0.0240 |
+--------+--------------+
10 rows in set (0.29 sec)
```

Vahel tekib kahtlusi, et kas ikka arvutatakse õigesti. Osakaalude puhul on võimalik päringus kõik leitud osakaalud kokku lugeda ja vaadata, et kas tuleb ligikaudu kokku arv 1 ehk 100%.

```
  SELECT SUM(veneosakaal) FROM
   (  SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti') AS veneosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
) AS tabel1;
```

Sellise vastuse päring ka praegu annab

```
+------------------+
| SUM(veneosakaal) |
+------------------+
|           1.0000 |
+------------------+
1 row in set (1.30 sec)
```

Python ja MySQL

Alustuseks ja meeldetuletuseks päring otse SQLi abil

```
MariaDB [dh18_keel]> SELECT * FROM sonaliikide_lyhendid;
+-------------+---------------------------+
| liigilyhend | liigikirjeldus            |
+-------------+---------------------------+
| A           | omadussõna algvõrre       |
| C           | omadussõna keskvõrre      |
| D           | määrsõna                  |
| G           | käändumatu omadussõna     |
| H           | pärisnimi                 |
| I           | hüüdsõna                  |
| J           | sidesõna                  |
| K           | kaassõna                  |
| N           | põhiarvsõna               |
| O           | järgarvsõna               |
| P           | asesõna                   |
| S           | nimisõna                  |
| U           | omadussõna ülivõrre       |
| V           | tegusõna                  |
| X           | verbi juurde kuuluv sõna  |
| Y           | lühend                    |
| Z           | lausemärk                 |
+-------------+---------------------------+
17 rows in set (0.00 sec)
```

Samadele andmetele Pythoni kaudu pöördumise näide. Esimesel real import-käsuga sisse MySQLi moodul. Edasi vaja luua ühendus Pythoni ja andmebaasi vahel. Parameeter host=”localhost” tähendab, et Python ja andmebaas on samas masinas, muul juhul tuleks sinna masina aadress. Edasi kasutajanimi, parool ja andmebaasi nimi. Pandas-paketis olemas mugav käsklus sql-päringu vastuse lugemiseks dataframe objekti ning edasi võib seal juba Pandase vahenditega majandada.

```
import mysql.connector as sql
import pandas as pd

yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
df=pd.read_sql("SELECT * FROM sonaliikide_lyhendid", yhendus)
print(df)
```

Koodilõik tööle ning tulemused käes

```
jaagup@praktika1 ~/public_html/2018/dt/21pysql $ python3.5 sql1.py
   liigilyhend            liigikirjeldus
0            A       omadussõna algvõrre
1            C      omadussõna keskvõrre
2            D                  määrsõna
3            G     käändumatu omadussõna
4            H                 pärisnimi
```

```
5           I                  hüüdsõna
6           J                  sidesõna
7           K                  kaassõna
8           N               põhiarvsõna
9           O               järgarvsõna
10          P                   asesõna
11          S                  nimisõna
12          U       omadussõna ülivõrre
13          V                  tegusõna
14          X  verbi juurde kuuluv sõna
15          Y                    lühend
16          Z                 lausemärk
```

Programm on mõnigikord põhjust vaid siis koostada, kui vastus sõltub kasutaja sisendist - muul puhul saadakse tulemus ühekordselt mugavamalt ka muul moel kätte. Kasutajalt rea jagu andmete küsimiseks sobib käsklus input - parameetrina saab anda arvutipoolse küsimuse teksti. Kuna SQL-is kipuvad kasutaja sisestatud ülakomad segadusi tekitama, siis if-käsu abil kontrollin, et kui ülakoma (programmikoodi tõttu paigutatud jutumärkide vahele) on sisestatud tekstis muutujas nimega tunnus, siis trükitakse, et tegemist keelatud sümboliga ning katkestatakse programmi töö.

Sobiva sisendi korral paigutatakse sisestatud tunnus SQL-lausesse liigilühendi võrdluseks. Ülakomad tunnusetekstil ümber, et päringulause andmebaasi tarbeks korrektne oleks. Kui andmeid vastuseks ei tulnud, siis järelikult vastava liigilühendiga liigikirjeldust tabelis polnud, muidu saab selle ainukese vastusena kätte - liigikirjelduse tulba väärtuste seast kohalt number 0.

```
import mysql.connector as sql
import pandas as pd

tunnus=input("Millisele tunnusele vastet soovid? ")
if "'" in tunnus:
  print("Keelatud sümbol: ' ")
  exit()
yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
df=pd.read_sql("SELECT liigikirjeldus FROM sonaliikide_lyhendid "+
           "WHERE liigilyhend='"+tunnus+"'", yhendus)
#print(df)

if len(df)==0: print(tunnus +" puudub")
else: print(tunnus +" on "+df["liigikirjeldus"].values[0])
```

Mõned katsetused loodud koodiga. Esimese proovina tüüpiline SQLi abil sissemurdmise katsetus (SQL injection), kus püütakse lisaks A-tüüpi liigilühendi vastele küsimisele ka elukohtade tabel baasist maha kustutada. Programm aga vastab, et ülakoma on sisendis keelatud ning päringuga rohkem edasi ei tegele.

```
jaagup@praktika1 ~/public_html/2018/dt/21pysql $ python3.5 sql2.py
Millisele tunnusele vastet soovid? A'; DELETE FROM elukohad; --
Keelatud sümbol: '
```

Kui küsitakse vastet tähele A, siis saab teada, et see on omadussõna algvõrre.

```
jaagup@praktika1 ~/public_html/2018/dt/21pysql $ python3.5 sql2.py
Millisele tunnusele vastet soovid? A
A on omadussõna algvõrre
```

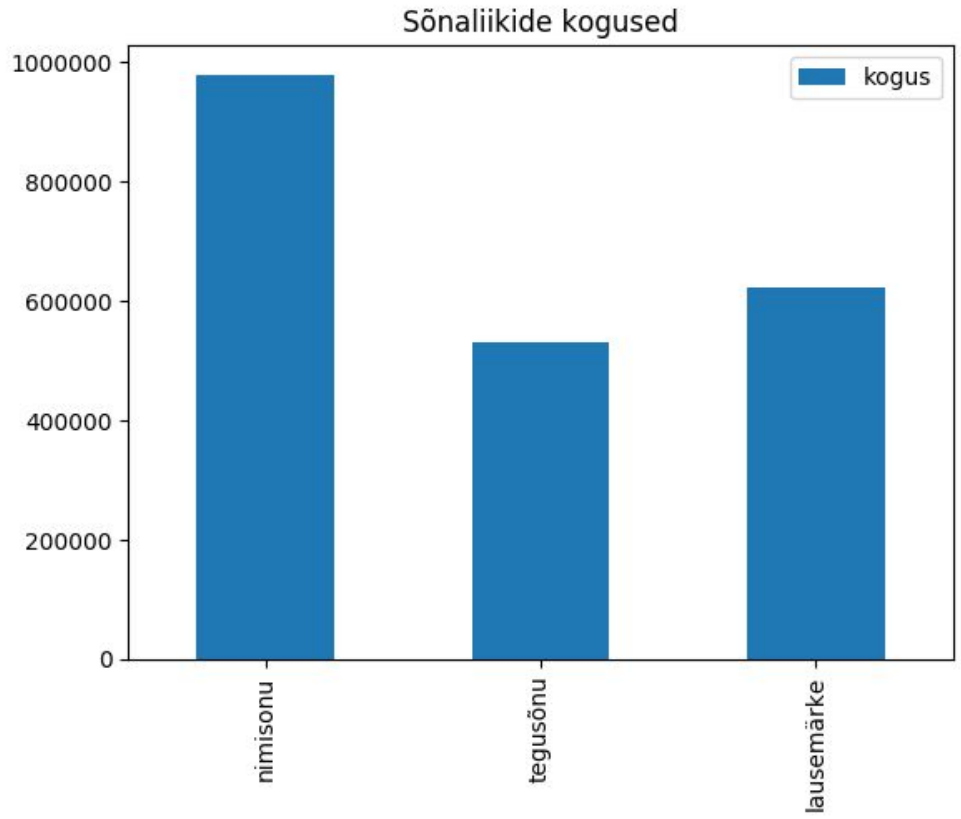
Tähte ASD lühendite seas ei leidu ning nii teataksegi viisakalt, et ASD puudub.

```
jaagup@praktika1 ~/public_html/2018/dt/21pysql $ python3.5 sql2.py
Millisele tunnusele vastet soovid? ASD
ASD puudub
```

Joonis SQL-tabelist tulnud andmete põhjal

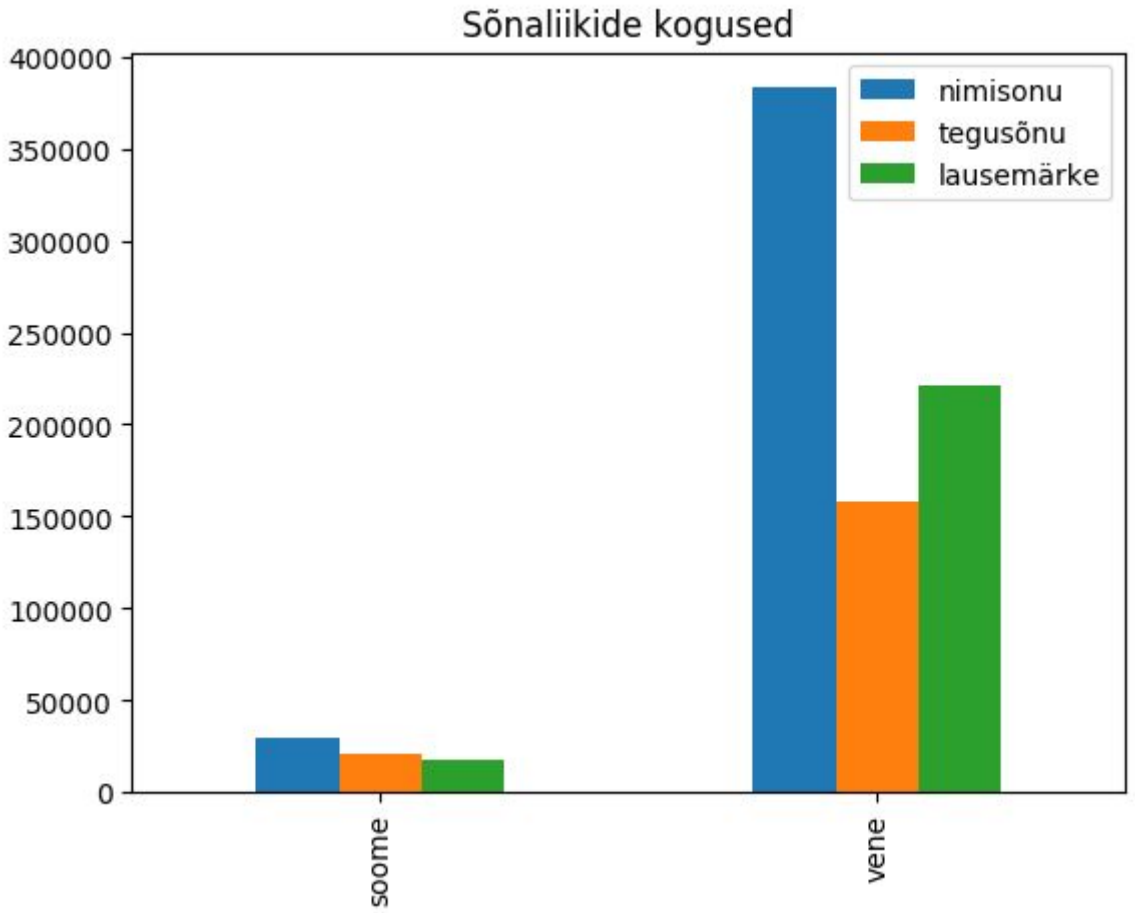
Päringust saame vastuseks nimisõnade, tegusõnade ja lausemärkide koguarvu. Kui sobivad teegid imporditud, siis suudab DataFrame sellest joonise koostada. Päringu tulemusena olid kõik vastused ühel real. Käsk df.T (transponse) aitab tabeli read ja veerud vahetada ning nii tuleb vastuseks viisakas tulpdiagramm. Ilma pööramata olnuks tulbad kõrvuti ühes plokis koos.

```
import mysql.connector as sql
import pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt


yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
df=pd.read_sql("""SELECT SUM(S) AS nimisonu, SUM(V) AS tegusõnu,
    SUM(Z) AS lausemärke FROM doksonaliigid""", yhendus)
#print(df)
pooratud=df.T
pooratud.columns=["kogus"]
pooratud.plot(kind="bar")
plt.title("Sõnaliikide kogused")
plt.savefig("joonis1.png", bbox_inches="tight")
```



Kui uuritavaid keeli mitu, siis sobib ka pööramata tabeli põhjal tehtud joonis. Iga keele kohta tuleb rida vastuste tabelis ning oma tulpade komplekt. Käsklus range aitab luua arvud alates nullist tulpade asukohtade järjekorranumbriteks, kõrval df[“emakeel”].values paneb sinna vastavad väärtused.

```
import mysql.connector as sql
import pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt


yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
df=pd.read_sql("""SELECT SUM(S) AS nimisonu, SUM(V) AS tegusõnu,
    SUM(Z) AS lausemärke, emakeel
         FROM doksonaliigid
         JOIN dokmeta ON doksonaliigid.kood=dokmeta.kood
          WHERE emakeel IN ("Vene", "Soome")
          GROUP BY emakeel""", yhendus)
print(df)

df.plot(kind="bar")
plt.xticks(range(len(df["emakeel"].values)), df["emakeel"].values)
plt.title("Sõnaliikide kogused")
plt.savefig("joonis1.png", bbox_inches="tight")
```



Sõnaliikide paarid

Meeldetuletuseks lause vene emakeelega autorite eestikeelsete tekstide hulgas sõnaliikide paaride osakaalude leidmiseks

```
  SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti') AS veneosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10;
```

```
+--------+-------------+
| ngram2 | veneosakaal |
+--------+-------------+
```

```
| SZ     |      0.0953 |
| SV     |      0.0476 |
| SS     |      0.0440 |
| ZS     |      0.0394 |
| PV     |      0.0377 |
| VS     |      0.0371 |
| AS     |      0.0335 |
| ZP     |      0.0321 |
| PS     |      0.0288 |
| VZ     |      0.0279 |
+--------+-------------+
10 rows in set (0.00 sec)
```

Sama lause tööle Pythoni käivitatud SQLi kaudu ning tulemus väljastatuna veebilehele.

```
yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
df=pd.read_sql("""SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti') AS veneosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10""", yhendus)

f=open("vastus1.html", "w")
f.write("<html><body>")
f.write(df.to_html())
f.write("</body></html>")
f.close()
```

Programm tööle

```
jaagup@praktika1 ~/public_html/2018/dt/22pysql $ python3.5 ylevaade1.py
```

Tekkis juurde HTML-fail

```
jaagup@praktika1 ~/public_html/2018/dt/22pysql $ ls
vastus1.html  ylevaade1.py
jaagup@praktika1 ~/public_html/2018/dt/22pysql $ more vastus1.html

<html><body><table border="1" class="dataframe">
  <thead>
    <tr style="text-align: right;">
      <th></th>
      <th>ngram2</th>
      <th>veneosakaal</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>0</th>
      <td>SZ</td>
```

```
      <td>0.0953</td>
    </tr>
```

….
Fail brauseris

	ngram2	veneosakaal
0	SZ	0.0953
1	SV	0.0476
2	SS	0.0440
3	ZS	0.0394
4	PV	0.0377
5	VS	0.0371
6	AS	0.0335
7	ZP	0.0321
8	PS	0.0288
9	VZ	0.0279

Sama päring eraldi autorite kolme emakeele kohta. Iga keele puhul sõnaliikide paarid järjestatuna suhteliste sageduste järgi. Kuna SQL-lause Pythoni koodis läheb üle mitme rea, siis on mugavam sinna ümber panna kolmekordsed jutumärgid, et reavahetus teksti sees oleks lubatud.

```
import mysql.connector as sql
import pandas as pd

yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
f=open("vastus2.html", "w")
f.write("<html><body>")
emakeeled=["vene", "soome", "saksa"]
for emakeel in emakeeled:
 f.write("<h2>"+emakeel+"</h2>")
 df=pd.read_sql("""SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti') AS keeleosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10""", yhendus)
 f.write(df.to_html())

f.write("</body></html>")
f.close()
```

Pilt väljundist

vene

	ngram2	keeleosakaal
0	SZ	0.0953
1	SV	0.0476
2	SS	0.0440
3	ZS	0.0394
4	PV	0.0377
5	VS	0.0371
6	AS	0.0335
7	ZP	0.0321
8	PS	0.0288
9	VZ	0.0279

soome

	ngram2	keeleosakaal
0	SZ	0.0754
1	SV	0.0501
2	SS	0.0450

Harjutus

- Küsi Pythoni abil tabelist “keeled” kõik väärtused, kuva välja.
- Püüa nad kätte saada massiivina, trüki tsükli abil välja
- Koosta HTML-leht, kus näha iga keele puhul kümme levinumat sõnaliigipaari koos suhteliste sagedustega

Pandas-paketi read_sql võimaldab päringu tulemuse korraga dataframe sisse lugeda ja pärast ühe käsuna välja trükkida. Nii ka siin. Lõppu veel näide keelte kätte saamisest lihtsa listina

```
import mysql.connector as sql
import pandas as pd

yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
dfkeeled=pd.read_sql("SELECT keelenimi FROM keeled", yhendus)
print(dfkeeled)
print(dfkeeled["keelenimi"].values.tolist())
```

```
jaagup@praktika1 ~/public_html/2018/dt/22pysql $ python3.5 ylevaade3.py
    keelenimi
0       eesti
1      hebrea
2     inglise
3       jidis
4   katalaani
5        lati
6       leedu
7        muud
8       poola
9   prantsuse
10     rootsi
11      saksa
12  sloveenia
13      soome
14     tšehhi
15    ukraina
16     ungari
17  valgevene
18       vene
['eesti', 'hebrea', 'inglise', 'jidis', 'katalaani', 'lati', 'leedu', 'muud', 'poola',
'prantsuse', 'rootsi', 'saksa', 'sloveenia', 'soome', 'tšehhi', 'ukraina', 'ungari',
'valgevene', 'vene']
```

Edasi saab loetelust võetud keeled tsükli abil läbi käia ning iga keele kohta sobiva väljundi kirjutada nii nagu varasemates näidetes.

```
import mysql.connector as sql
import pandas as pd

yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
dfkeeled=pd.read_sql("SELECT keelenimi FROM keeled", yhendus)
emakeeled=dfkeeled["keelenimi"].values.tolist()

f=open("vastus4.html", "w")
f.write("<html><body>")

for emakeel in emakeeled:
 f.write("<h2>"+emakeel+"</h2>")
 df=pd.read_sql("""SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti') AS keeleosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti'
```

```
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10""", yhendus)
 f.write(df.to_html())

f.write("</body></html>")
f.close()
```

eesti

	ngram2	keeleosakaal
0	SZ	0.0706
1	SS	0.0585
2	AS	0.0519
3	SV	0.0500
4	VS	0.0301
5	ZD	0.0270
6	PS	0.0270
7	VZ	0.0264
8	ZJ	0.0261
9	VD	0.0258

hebrea

	ngram2	keeleosakaal

inglise

	ngram2	keeleosakaal
0	SZ	0.0870
1	PV	0.0742

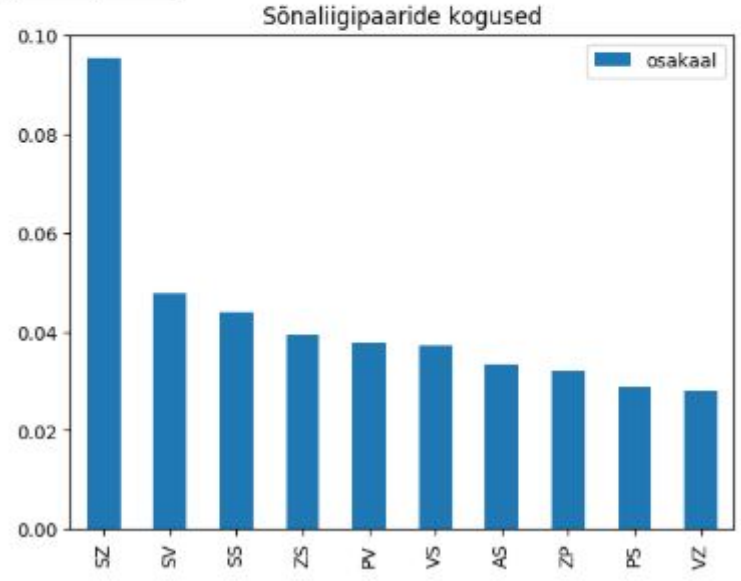
Joonis genereeritud veebilehel

Joonise loomise oskuse saab siduda veebilehe kokkupanekuga. Pythoni abiga tekitatakse kettale pildifail ning veebilehe img-elemendist viidatakse sinna

```
import mysql.connector as sql
import pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt


yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
df=pd.read_sql("""SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti') AS osakaal
        FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10""", yhendus)

f=open("vastus5.html", "w")
f.write("<html><body>")
f.write(df.to_html())
df.plot(kind="bar")
plt.xticks(range(len(df["ngram2"].values)), df["ngram2"].values)
plt.title("Sõnaliigipaaride kogused")
plt.savefig("joonis5.png", bbox_inches="tight")
f.write("<img src='joonis5.png' />")
f.write("</body></html>")
f.close()
```

Tabel ja joonis veebilehel

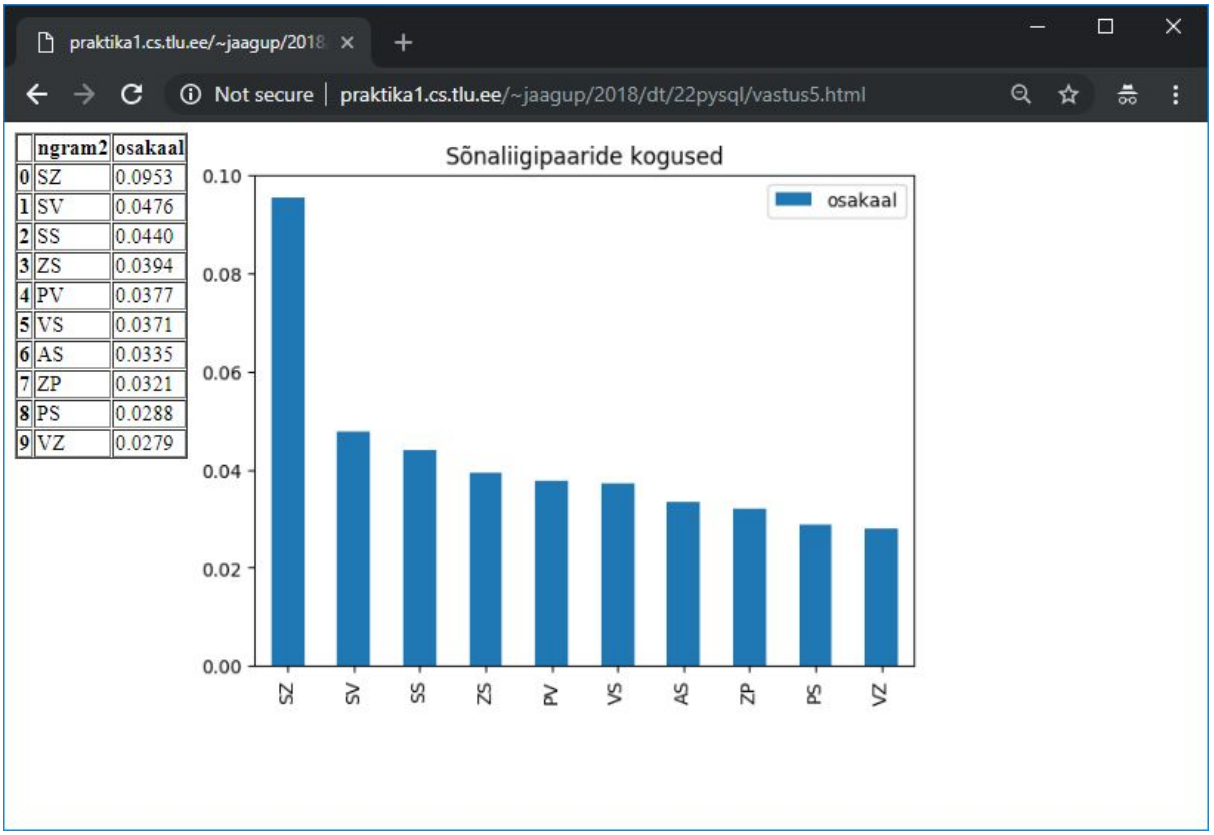
	ngram2	osakaal
0	SZ	0.0953
1	SV	0.0476
2	SS	0.0440
3	ZS	0.0394
4	PV	0.0377
5	VS	0.0371
6	AS	0.0335
7	ZP	0.0321
8	PS	0.0288
9	VZ	0.0279



Lisades veebilehtede div-elementidele laadikäsklused float:left, õnnestub tabel ja pilt lehel kõrvu paigutada

```
import mysql.connector as sql
import pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt

yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
df=pd.read_sql("""SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti') AS osakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='vene' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10""", yhendus)

f=open("vastus5.html", "w")
f.write("<html><body>")
f.write("<div style='float: left'>"+df.to_html()+"</div>")
df.plot(kind="bar")
plt.xticks(range(len(df["ngram2"].values)), df["ngram2"].values)
plt.title("Sõnaliigipaaride kogused")
```

```
plt.savefig("joonis5.png", bbox_inches="tight")
f.write("<div style='float: left'><img src='joonis5.png' /></div>")
f.write("</body></html>")
f.close()
```



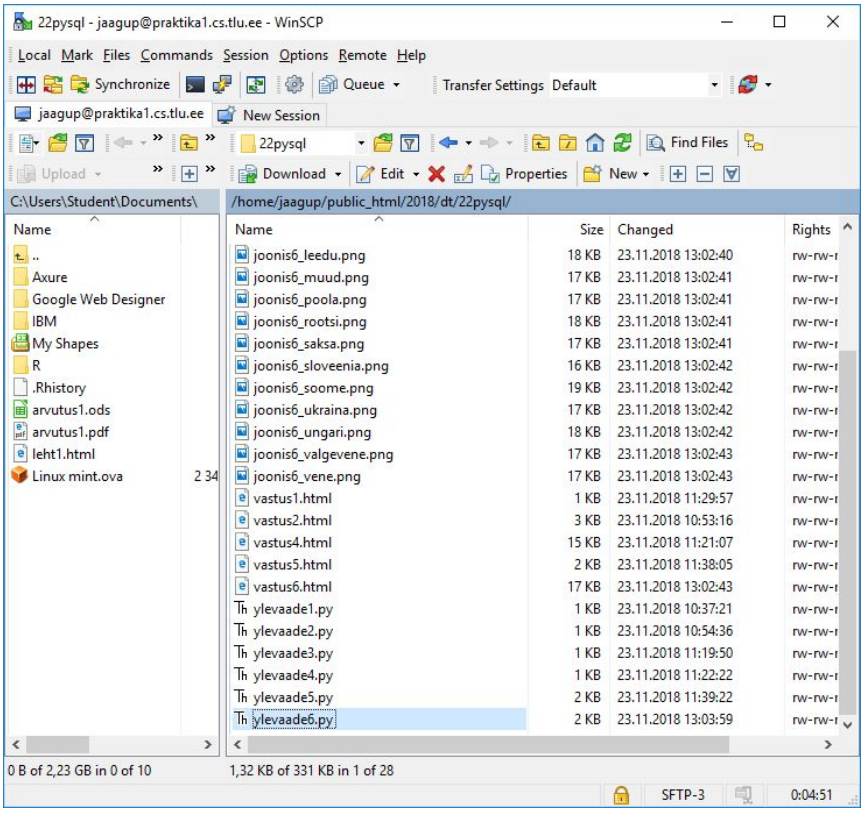
	ngram2	osakaal
0	SZ	0.0953
1	SV	0.0476
2	SS	0.0440
3	ZS	0.0394
4	PV	0.0377
5	VS	0.0371
6	AS	0.0335
7	ZP	0.0321
8	PS	0.0288
9	VZ	0.0279

Andmed mitme emakeele kohta

Keeled saab keeletabelist tsükliga kätte. Edasi siis sobib iga keele kohta sagedustabel küsida. Kuna jooniseid rohkem, siis nad vähemasti sama eesliitega, et kataloogi päris segamini ei ajaks

```
import mysql.connector as sql
import pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt


yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
dfkeeled=pd.read_sql("SELECT keelenimi FROM keeled", yhendus)
emakeeled=dfkeeled["keelenimi"].values.tolist()

f=open("vastus6.html", "w")
f.write("<html><body>")
```

```
for emakeel in emakeeled:

 df=pd.read_sql("""SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti') AS keeleosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10""", yhendus)
 if(len(df)>0):
  f.write("<h2 style='clear: left; padding-top: 70px'>"+emakeel+"</h2>")
  f.write("<div style='float: left'><img src='joonis6_"+emakeel+".png' /></div>")
  f.write("<div style='float: left; padding-top: 50px'>"+df.to_html()+"</div>")
  df.plot(kind="bar")
  plt.xticks(range(len(df["ngram2"].values)), df["ngram2"].values)
  plt.title("Sõnaliigipaaride kogused")
  plt.savefig("joonis6_"+emakeel+".png", bbox_inches="tight")
f.write("</body></html>")
f.close()
```



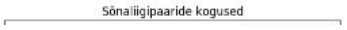
eesti

	1gram2	keeleosakaal
0	SZ	0.0706
1	SS	0.0585
2	AS	0.0519
3	SV	0.0500
4	VS	0.0301
5	ZD	0.0270
6	PS	0.0270
7	VZ	0.0264
8	ZJ	0.0261
9	VD	0.0258

inglise

	1gram2	keeleosakaal
0	SZ	0.0870
1	PV	0.0742
2	ZP	0.0582
3	VS	0.0530
4	SS	0.0418
5	SV	0.0341
6	PS	0.0302
7	ZS	0.0291
8	ZH	0.0233
9	VD	0.0222

jidis



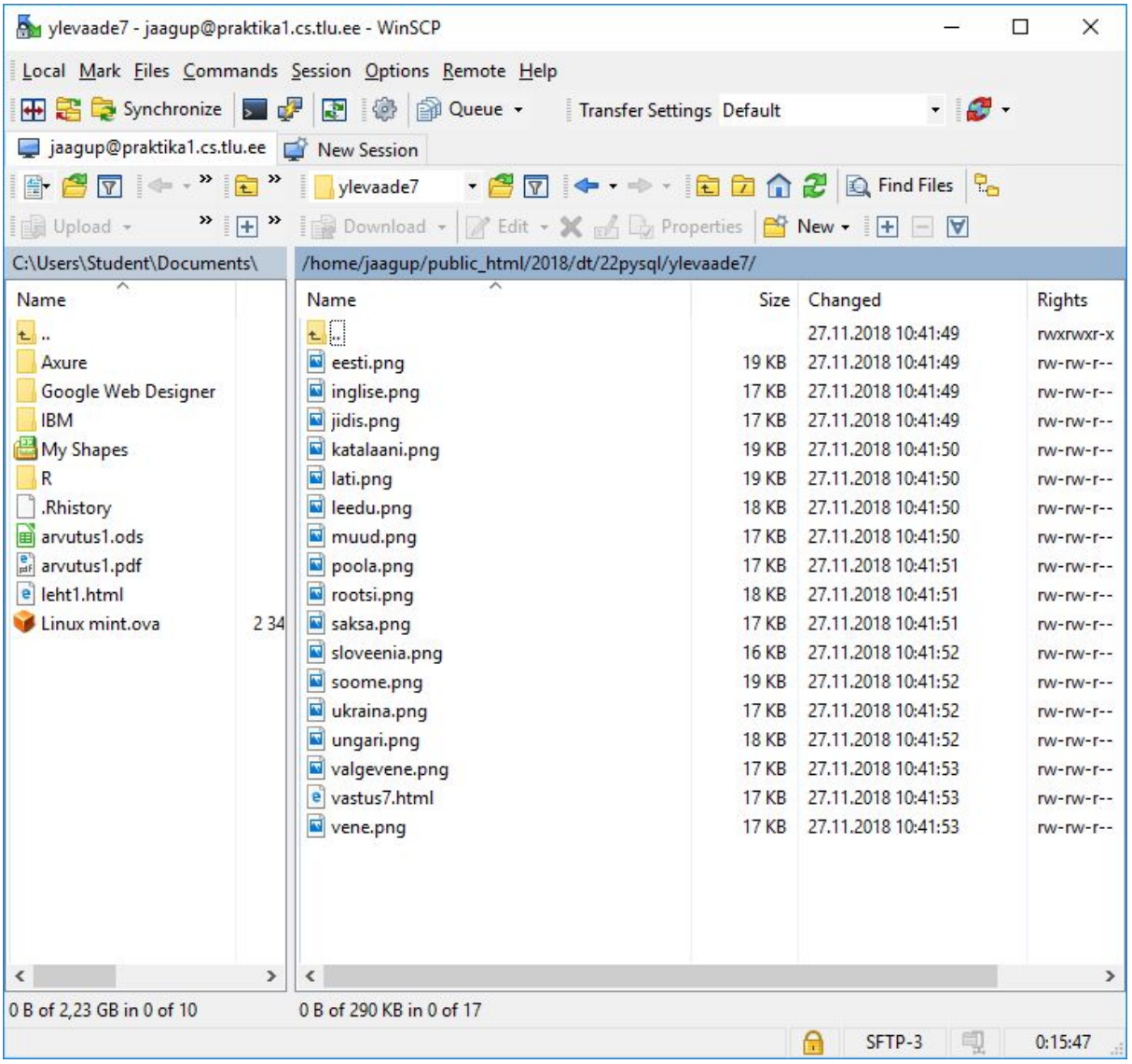
Vastused omaette kataloogis

Üksiku mitme pildiga lehe puhul mahuvad need veel kuidagi sobivalt ära. Kui aga samasse kataloogi peaks hulk piltidega veebilehti jõudma, siis varsti on silme eest juba üsna kirju. Nii sobib HTML-fail + pildid/joonised omaette kataloogi paigutada. Programmikoodi paindlikkuse huvides salvestatakse kataloogi nimi eraldi muutujas. Kaldkriips ka lõppu, et seda eraldajat ei peaks hiljem juurde tekitama hakkama.

Operatsioonisüsteemiga suhtlemiseks tuleb juurde importida moodul os. Käsklus os.path.exists kontrollib, et kas vastav kataloog juba olemas. Kui mitte, siis luuakse. HTML-faili loomisel peab asukoha määrama vastava kataloogi sisse, samuti piltide loomisel. HTML-failist piltidele viidates aga ei tasu enam kataloogi nime juurde lisada, sest omavahel on HTML ja png-failid samas kataloogis

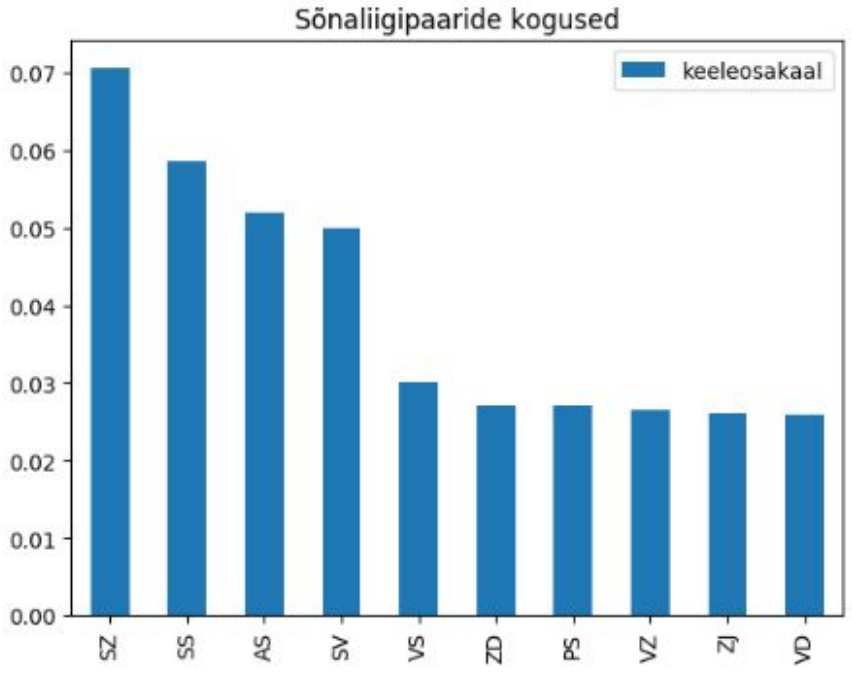
```
import mysql.connector as sql
import pandas as pd
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import os
```

```
kataloog="ylevaade7/"
if not os.path.exists(kataloog):
  os.mkdir(kataloog)

yhendus=sql.connect(host="localhost",
    user="dh18", password="dh18praktika", database="dh18_keel")
dfkeeled=pd.read_sql("SELECT keelenimi FROM keeled", yhendus)
emakeeled=dfkeeled["keelenimi"].values.tolist()

f=open(kataloog+"vastus7.html", "w")
f.write("<html><body>")

for emakeel in emakeeled:
 df=pd.read_sql("""SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti') AS keeleosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel='"""+emakeel+"""' AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10""", yhendus)
 if(len(df)>0):
  f.write("<h2 style='clear: left; padding-top: 70px'>"+emakeel+"</h2>")
  f.write("<div style='float: left'><img src='"+emakeel+".png' /></div>")
  f.write("<div style='float: left; padding-top: 50px'>"+df.to_html()+"</div>")
  df.plot(kind="bar")
  plt.xticks(range(len(df["ngram2"].values)), df["ngram2"].values)
  plt.title("Sõnaliigipaaride kogused")
  plt.savefig(""+kataloog+emakeel+".png", bbox_inches="tight")
f.write("</body></html>")
f.close()
```



Tulemuseks taas lehestik, kus keelte kaupa tabelid ja joonised näha

eesti



	ngram2	keeleosakaal
0	SZ	0.0706
1	SS	0.0585
2	AS	0.0519
3	SV	0.0500
4	VS	0.0301
5	ZD	0.0270
6	PS	0.0270
7	VZ	0.0264
8	ZJ	0.0261
9	VD	0.0258

inglise

PHP

Eelnevate näidete järgi Pythoni abil veebilehtede loomine täiesti toimib. Ka saab Pythoni programme veebis tööle panna nii CGI-liidese kui ka Django või Flaski veebiserveri kaudu. Samas need moodused enamasti siiski eeldavad oma serveri seadistamist, mis tehnilise ja turvapoole pealt parasjagu keerukas ettevõtmine. Veebis hiljemalt sajandivahetusest alates on väiksema ja keskmise suurusega lehestike puhul levinud abiliseks PHP, mille ülessättimiseks leiab 2018. aasta seisuga nii tasuta kui mõneeurose kuutasuga kohti. Heal juhul piisab vaid vajalike failide ning ehk ka andmebaasi õigesse kohta kopeerimisest ning töö võibki alata.

Tutvustuseks lihtne kaht arvu kokku liitev veebileht, mille abil kontrollida, et serveris php töötab. Faili laiendiks üldjuhul vajalik .php

```
<!doctype html>
<html>
  <head>
    <title>PHP katsetus</title>
  </head>
  <body>
   Arvutuse tulemus:
          <?php echo 3+2;  ?>
  </body>
</html>
```

Väljund:

Arvutuse tulemus: 5

ehk siis 3 ning 2 liideti serveri pool kokku ning kliendini jõudis vaid tulemus.

Päring andmebaasist

Sarnaselt Pythonile tuleb ka siin määrata, et kus masinas asub soovitud andmebaas, mis on sealsed kasutajanimi ja parool ning millise nimega baasiga soovitakse suhelda. Siin näites kasutatakse PHP teeki nimega mysqli (MySQL Improved), mis hiljem võimaldab ettevalmistatud päringute abil hoolitseda, et veebist tulevad pahatahtlikud sisestused andmebaasile suuremat kahju ei saaks teha. Käsuga prepare valmistatakse päring ette, bind_result määrab, kuhu muutujatesse paigutatakse saabuvad vastused. Hiljem while-tsüklis iga fetch-käsklusega tõstetakse vastusetabeli ühe rea tulpade andmed bind_result-käsu abil määratud muutujatesse ning neid saab lehe kuvamisel pruukida

Lehe lõpus pannakse andmebaasiühendus kinni. Enamasti sulgub see ka ise millalgi automaatselt, aga kui lehtedel juhtub palju kasutajaid olema, siis võib kergemini juhtuda, et parasjagu pole vaba ühendust käepärast.

Kood tervikuna:

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
  $kasklus=$yhendus->prepare(
    "SELECT liigilyhend, liigikirjeldus FROM sonaliikide_lyhendid");
  $kasklus->bind_result($lyhend, $kirjeldus);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Sõnaliikide andmed</title>
  </head>
  <body>
    <h1>Sõnaliikide lühendite loetelu</h1>
       <?php
          while($kasklus->fetch()){
                 echo "$lyhend - $kirjeldus<br />\n";
          }
       ?>
  </body>
</html>
<?php $yhendus->close(); ?>
```

ja töö tulemus:

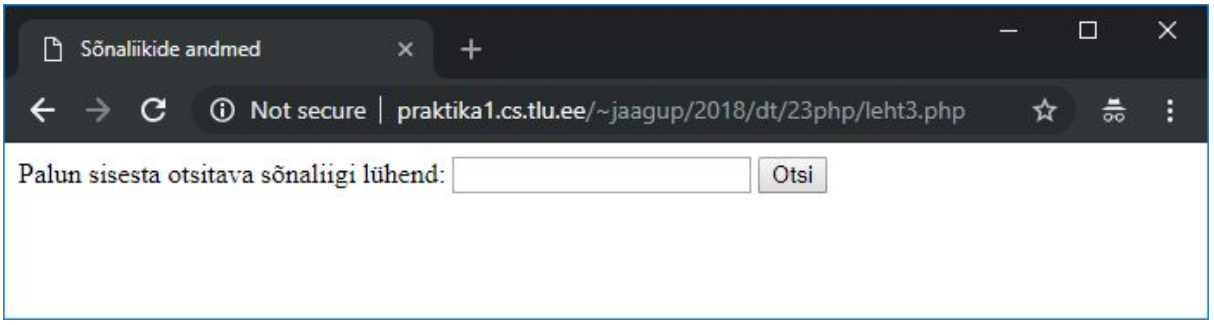
Sõnaliikide lühendite loetelu
A - omadussõna algvõrre
C - omadussõna keskvõrre
D - määrsõna
G - käändumatu omadussõna
H - pärisnimi
I - hüüdsõna
J - sidesõna
K - kaassõna
N - põhiarvsõna
O - järgarvsõna
P - asesõna
S - nimisõna
U - omadussõna ülivõrre
V - tegusõna
X - verbi juurde kuuluv sõna
Y - lühend
Z - lausemärk

Sisestus kasutajalt

PHPst on suurelt jaolt kasu just seetõttu, et siis on võimalik vastavalt kasutaja valikutele või sisestatud andmetele paigutada lehele sobivat sisu. Sisestuselemendid pannakse üldjuhul form-elemendi sisse, kuhu siis action-parameetriga määratase ära, millise faili juurde sisestatud andmed saadetakse. Tekstisisestusväli on input-element atribuudiga type=”text”. Elemendi nime järgi saab pärast vastuvõtval lehel inimese sisestatud andmed välja küsida. Andmete teele saatmiseks on submit-nupp

```
<!doctype html>
<html>
  <head>
     <title>Sõnaliikide andmed</title>
  </head>
  <body>
     <form action="leht3a.php">
           Palun sisesta otsitava sõnaliigi lühend:
              <input type="text" name="lyhend" />
              <input type="submit" value="Otsi" />
        </form>
  </body>
</html>
```

Leht avaneb, sinna saab sisestada soovitud lühendi



Otsi-nupule vajutades püütakse avada uus, action-parameetriga määratud leht. Kuna seda veel ei ole, siis annab Apache veebiserver veateate. Küll aga on aadressirealt näha, et püütakse avada näidatud lehte ning kaasa antakse ka inimese sisestatud lühend.

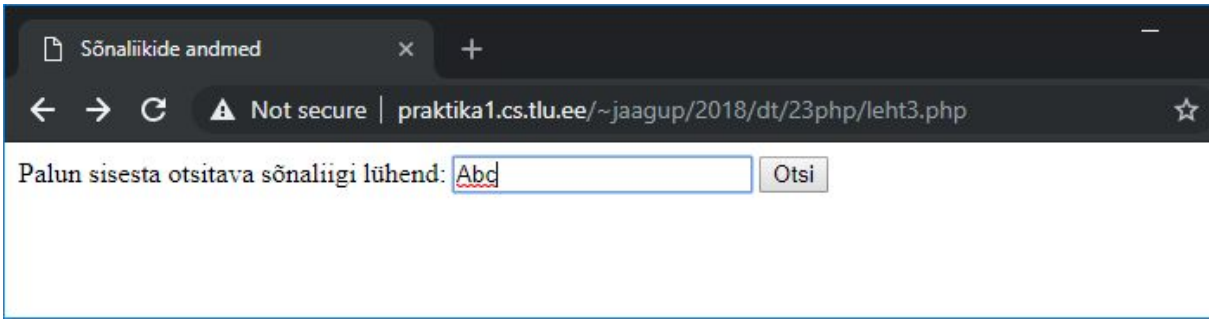
Andmete kinni püüdmiseks tuleb vastav leht valmis teha nime all leht3a.php

SQL-lausesse veebist saabuvate andmete sisestamiseks tuleb sinna kõigepealt panna küsimärk. Käsu bind_param abil asendatakse küsimärk sobiva väärtusega. Sedakorda võetakse uuritav lühend aadressirealt. Selle kättesaamiseks aitab muutuja $_REQUEST[“lyhend”]. All kontrollitakse if-i abil: kui baasis on lühendile vastus olemas, siis kuvatakse vastus välja, muidu antakse puudumisest teada.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
  $kasklus=$yhendus->prepare(
    "SELECT liigilyhend, liigikirjeldus
          FROM sonaliikide_lyhendid WHERE liigilyhend=?");
  $kasklus->bind_param("s", $_REQUEST["lyhend"]);
  $kasklus->bind_result($lyhend, $kirjeldus);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Sõnaliikide andmed</title>
  </head>
  <body>
    <h1>Sõnaliikide lühendite loetelu</h1>
       <?php
          if($kasklus->fetch()){
                 echo "$lyhend - $kirjeldus<br />\n";
          } else{
                 echo "$_REQUEST[lyhend] puudub baasist";
          }
       ?>
  </body>
</html>
<?php $yhendus->close(); ?>
```

Leitud lühendi vaste kuvatakse ekraanil

Abc pole aga sobiv lühend, ning nii tuleb ka vastav teade



Sõnaliigipaarid vastavalt emakeelele

Eelnevas peatükis küsiti emakeelele vastavad levinumad sõnaliigipaarid Pythoni abil. Siin pannakse sama lause tööle PHP kaudu ning töö tulemust on võimalik ekraanil otse vaadata. Kuna emakeel läheb päringusse sisse kahte kohta, siis tuleb ka lausesse kaks küsimärki, bind_param-käsu juurde kahel korral s-täht (andmetüüp string) ning kahel korral võetakse aadressiribalt emakeele väärtus.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
  $kasklus=$yhendus->prepare(
    "SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel=? AND dokmeta.tekstikeel='eesti') AS keeleosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel=? AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10
");
  $kasklus->bind_param("ss", $_REQUEST["emakeel"], $_REQUEST["emakeel"]);
  $kasklus->bind_result($ngram2, $osakaal);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Sõnaliikide andmed</title>
  </head>
  <body>
```

```
    <h1>Levinumad sõnaliigipaarid</h1>
       <table>
       <?php
          while($kasklus->fetch()){
                 echo "<tr><td>$ngram2</td><td>$osakaal</td></tr>\n";
          }
       ?>
       </table>
  </body>
</html>
<?php $yhendus->close(); ?>
```

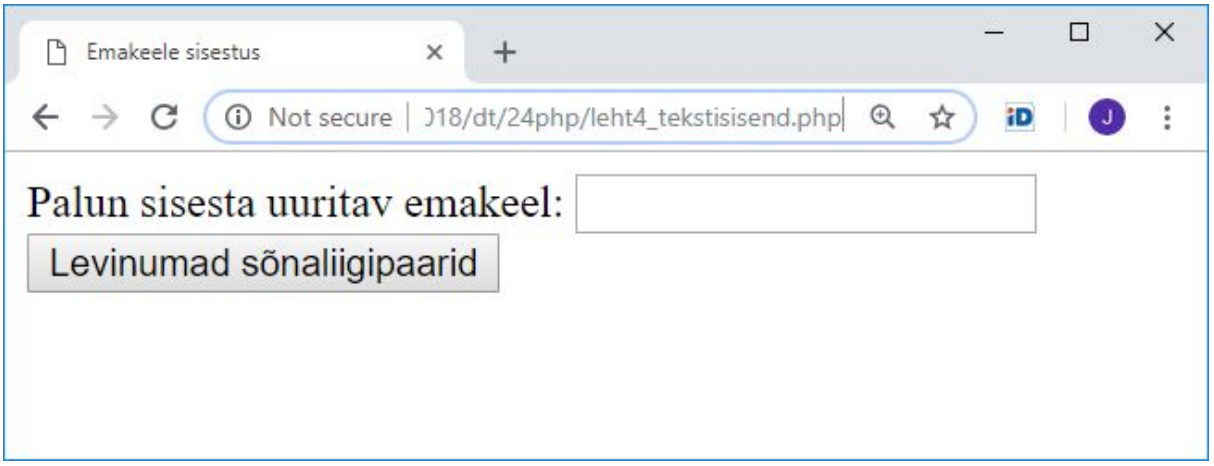
Harjutus

- Koostage leht nimega leht4_tekstisisend.php , kus kasutaja saab tekstivälja sisestada soovitud keele, see suunatakse aadressile leht4.php ning näeb vastava keele levinumaid sõnaliigipaare.
- Vastava keele tekstide puudumisel antakse sellest teada

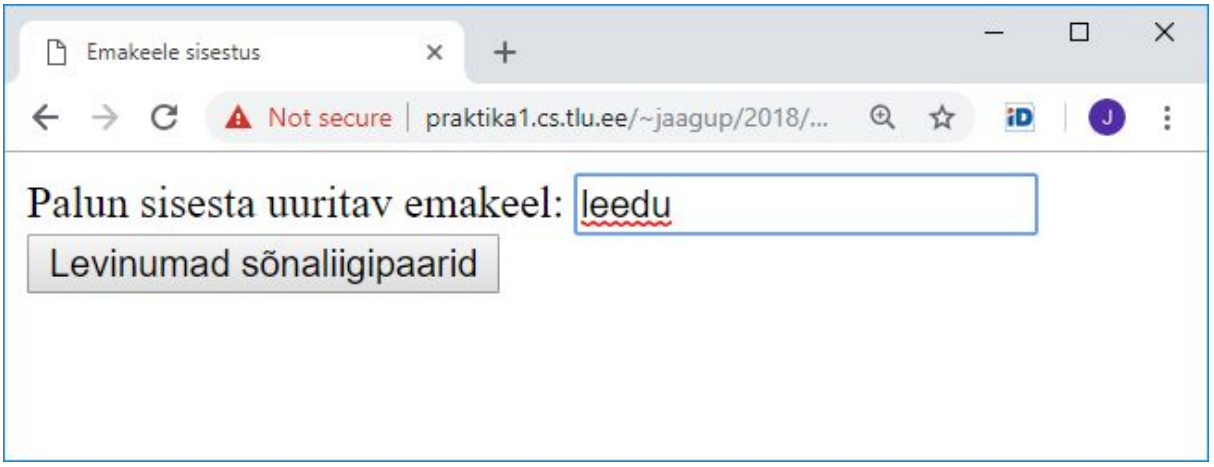
Lahenduseks eelnenule sarnane tekstisisestusleht. Elemendi form parameeter action määrab aadressi, kuhu saadetakse andmed. Kuna leht4.php ootab parameetrit nimega emakeel, siis tuleb tekstiväljale ka vastav nimi panna, nii jõuavad andmed kohale

```
<!doctype html>
<html>
  <head>
    <title>Emakeele sisestus</title>
  </head>
  <body>
    <form action="leht4.php">
          Palun sisesta uuritav emakeel:
          <input type="text" name="emakeel" />
          <input type="submit" value="Levinumad sõnaliigipaarid" />
       </form>
  </body>
</html>
```

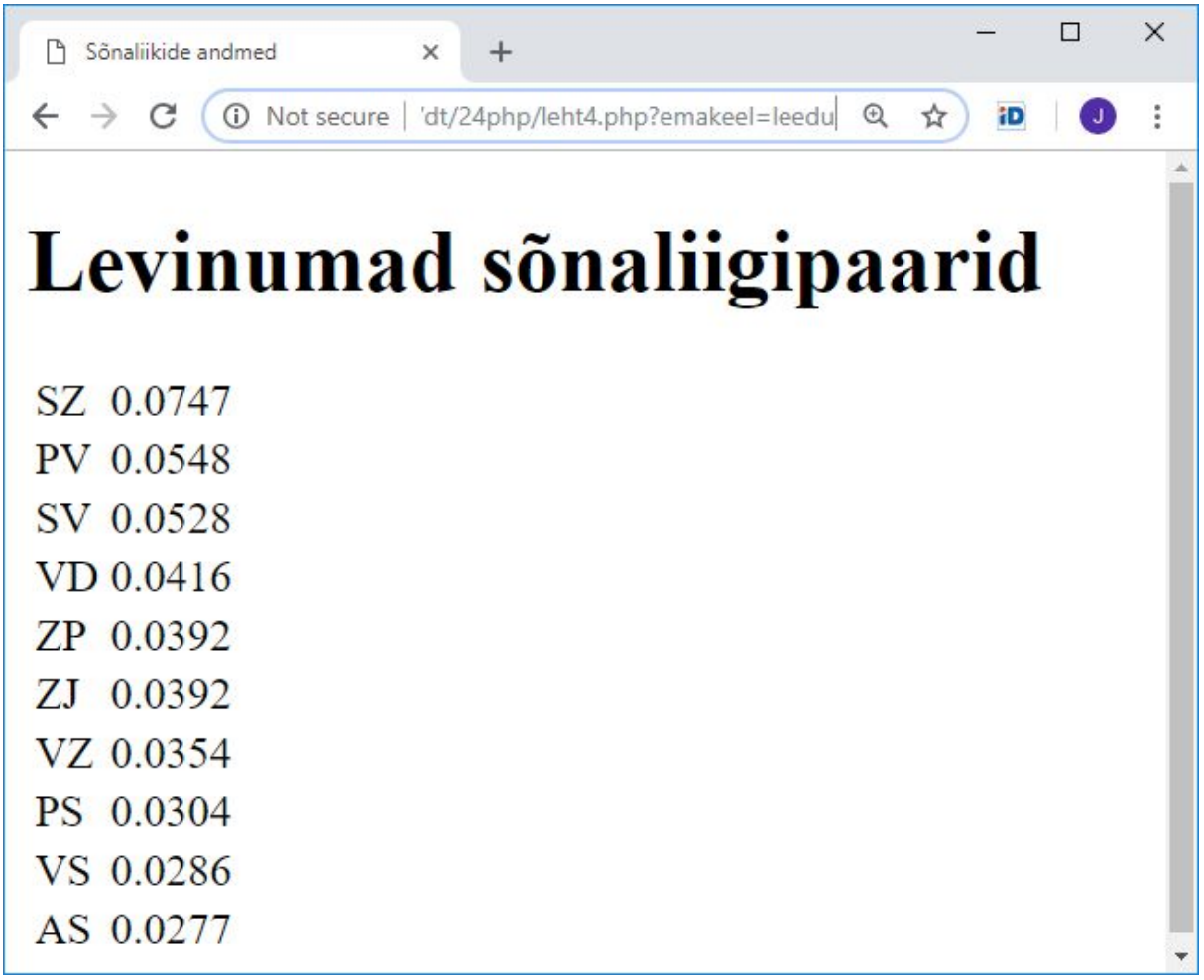
Leht tühjana



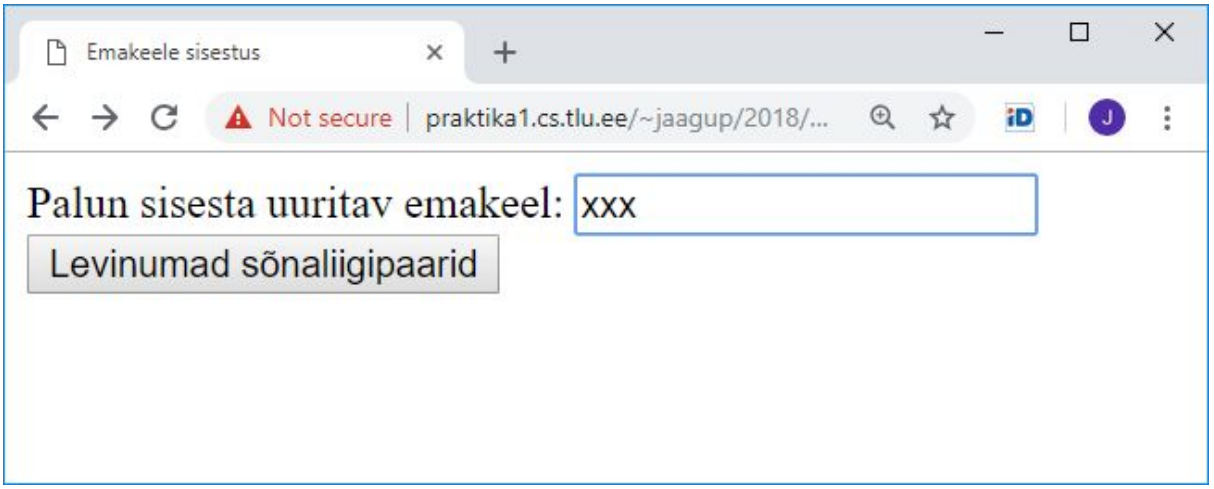
Sisestatud keelega



Keel jõudis uuele lehele kohale ning sealt anti sobivad vasted



Kui sisendiks panna olematu keel



siis esialgu on sõnaliigipaaride all vasteks tühjus

Eelnevalt sõnaliigilühendite otsimisel sai mugavalt if-i abil teada anda, et vaste puudumisel teatatakse sellest. Tsüklil while ei ole else osa, siis tuleb muutujate abil kavaldada. Siin näites pannakse $loendur algul nulliks, iga rea kuvamisel suurendatakse seda ühe võrra. Hiljem saab kontrollida, et kui loendur on endiselt 0, siis pole midagi kuvada ning selle asemel saab muud teadet näidata.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
  $kasklus=$yhendus->prepare(
    "SELECT ngram2.ngram2,
    COUNT(*)/(SELECT COUNT(*) FROM ngram2 JOIN dokmeta ON dokmeta.kood=ngram2.tekstikood
WHERE dokmeta.emakeel=? AND dokmeta.tekstikeel='eesti') AS keeleosakaal
       FROM dokmeta
   JOIN ngram2 ON dokmeta.kood=ngram2.tekstikood
   WHERE dokmeta.emakeel=? AND dokmeta.tekstikeel='eesti'
   GROUP BY ngram2
   ORDER BY COUNT(*) DESC
   LIMIT 10
");
  $kasklus->bind_param("ss", $_REQUEST["emakeel"], $_REQUEST["emakeel"]);
  $kasklus->bind_result($ngram2, $osakaal);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Sõnaliikide andmed</title>
  </head>
  <body>
    <h1>Levinumad sõnaliigipaarid</h1>
       <table>
       <?php
          $loendur=0;
          while($kasklus->fetch()){
                 echo "<tr><td>$ngram2</td><td>$osakaal</td></tr>\n";
                 $loendur=$loendur+1;
          }
       ?>
       </table>
       <?php
         if($loendur==0){
```

```
                echo "Keeles $_REQUEST[emakeel] pole sõnaliigipaare";
            }
        ?>
  </body>
</html>
<?php $yhendus->close(); ?>
```

Nii ka vastuseks saab teate

Sisestus rippmenüüst

Valik nimega select on HTMLis mugav sisestuselement. Alamelementidega option saab soovitud valikud ette anda, inimene valib ning tulemus saadetakse vormi action-parameetriga määratud aadressil

```
<!doctype html>
<html>
  <head>
    <title>Emakeele sisestus</title>
  </head>
  <body>
    <form action="leht4.php">
          Palun sisesta uuritav emakeel:
          <select name="emakeel">
            <option>soome</option>
            <option>vene</option>
            <option>leedu</option>
          </select>
          <input type="submit" value="Levinumad sõnaliigipaarid" />
        </form>
  </body>
</html>
```

Meie leht4.php oskab saadetud emakeele väärtust lugeda ning vastavad sõnaliigipaarid kuvada.

Keeli võib aga palju olla, neid võib ajapikku lisanduda ning selleks pole mugav veebilehte alati eraldi muutma hakata. Keelte loetelu tasub vaadata andmebaasist

```
jaagup@praktika1 ~ $ mysql -udh18 -pdh18praktika dh18_keel
```

Tabelite loetelu, sealt paistab tabel keeled

```
MariaDB [dh18_keel]> SHOW TABLES;
+----------------------+
| Tables_in_dh18_keel  |
+----------------------+
| dokarvud             |
| dokmeta              |
| doksonaliigid        |
| elukohad             |
| haridustasemed       |
| keeled               |
| keeletasemed         |
```

```
| korpusenimed         |
| ngram1               |
| ngram2               |
| ngram3               |
| ngram4               |
| ngram5               |
| sonaliikide_lyhendid |
| taustad              |
| tekstityybid         |
| vanusetasemed        |
```

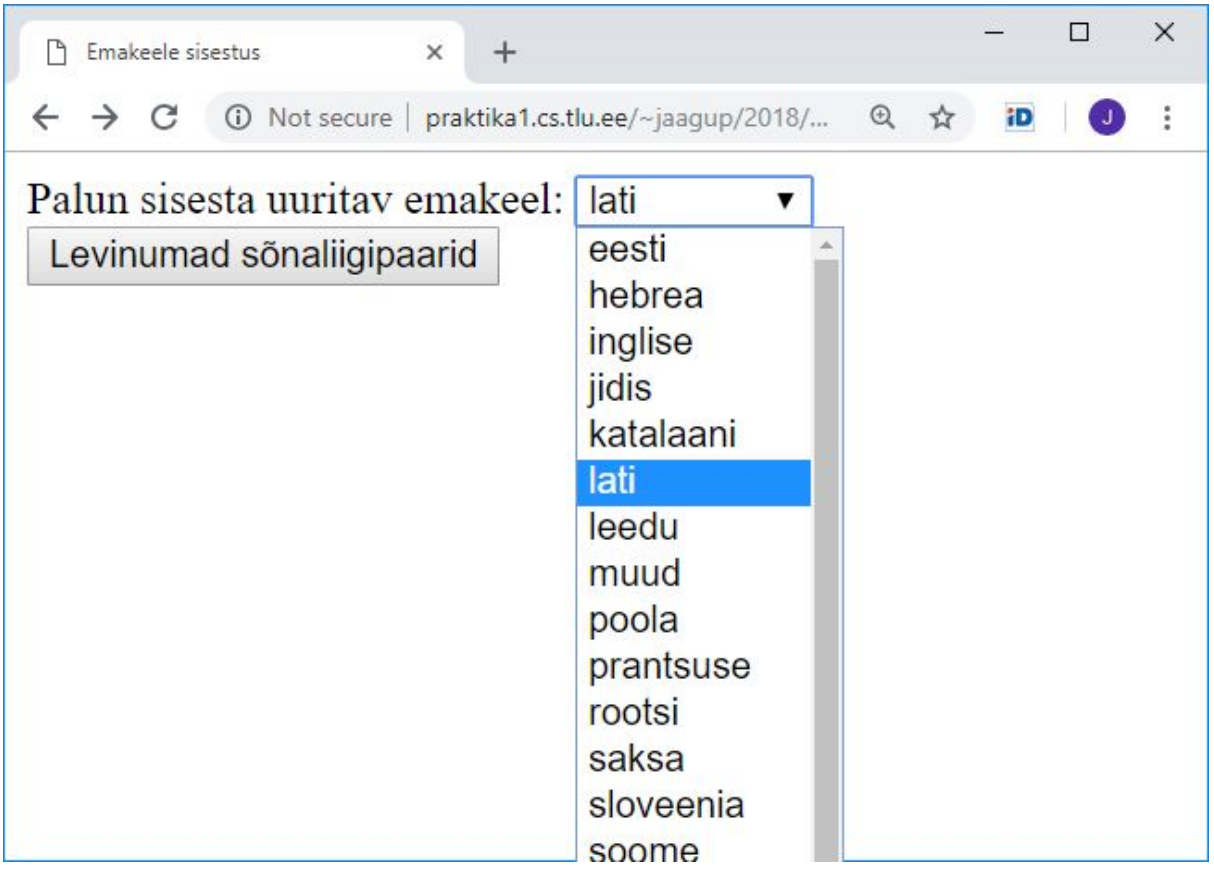
Selles omakorda on keelte nimed

```
MariaDB [dh18_keel]> SELECT * FROM keeled;
+-----------+
| keelenimi |
+-----------+
| eesti     |
| hebrea    |
| inglise   |
| jidis     |
| katalaani |
| lati      |
| leedu     |
| muud      |
| poola     |
| prantsuse |
| rootsi    |
| saksa     |
| sloveenia |
| soome     |
| tšehhi    |
| ukraina   |
| ungari    |
| valgevene |
| vene      |
+-----------+
19 rows in set (0.00 sec)
```

Päringuga saab keelte nimed tabelist kätte ning võib tsükli abil kuvada veebilehele rippmenüü sisse

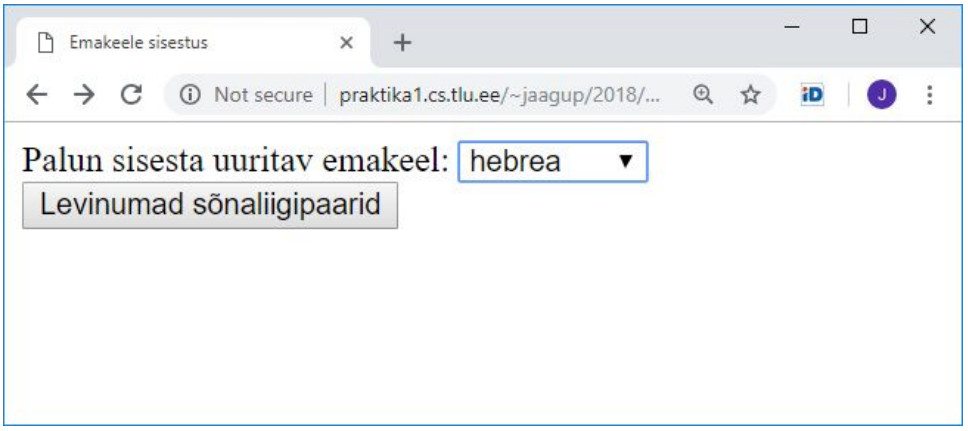
```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
  $kasklus=$yhendus->prepare("SELECT keelenimi FROM keeled");
  $kasklus->bind_result($keelenimi);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Emakeele sisestus</title>
  </head>
  <body>
    <form action="leht4.php">
          Palun sisesta uuritav emakeel:
          <select name="emakeel">
```

```
        <?php
                while($kasklus->fetch()){
                        echo "<option>$keelenimi</keelenimi>\n";
                }
              ?>
          </select>
          <input type="submit" value="Levinumad sõnaliigipaarid" />
       </form>
  </body>
</html>
```



Valitule vastavad sõnaliigipaarid:

Selgus aga, et mõne keele puhul sõnaliigipaarid puuduvad, ehkki keel oli võimalusena loetelus olemas.



Nii võib sobivad keeled võtta mitte keelte üldtabelist, vaid tekstide loetelu tabelist, kus näha, kui palju igas keeles tekste on

```
SELECT emakeel, COUNT(*) AS kogus FROM dokmeta
 GROUP BY emakeel ORDER BY COUNT(*) DESC;
```

```
+-----------+-------+
| emakeel   | kogus |
+-----------+-------+
| NULL      |  8576 |
| vene      |  3184 |
| soome     |   391 |
| eesti     |   229 |
| inglise   |   119 |
| saksa     |    86 |
| muud      |    81 |
| leedu     |    24 |
| ukraina   |    17 |
| ungari    |     6 |
| poola     |     3 |
| rootsi    |     2 |
| lati      |     2 |
| valgevene |     1 |
| jidis     |     1 |
| katalaani |     1 |
| sloveenia |     1 |
+-----------+-------+
17 rows in set (0.03 sec)
```

Täiendatud päring - näidatakse vaid neid ridu, kus emakeel on määratud ning kus vastavas keeles on tekste rohkem kui üks

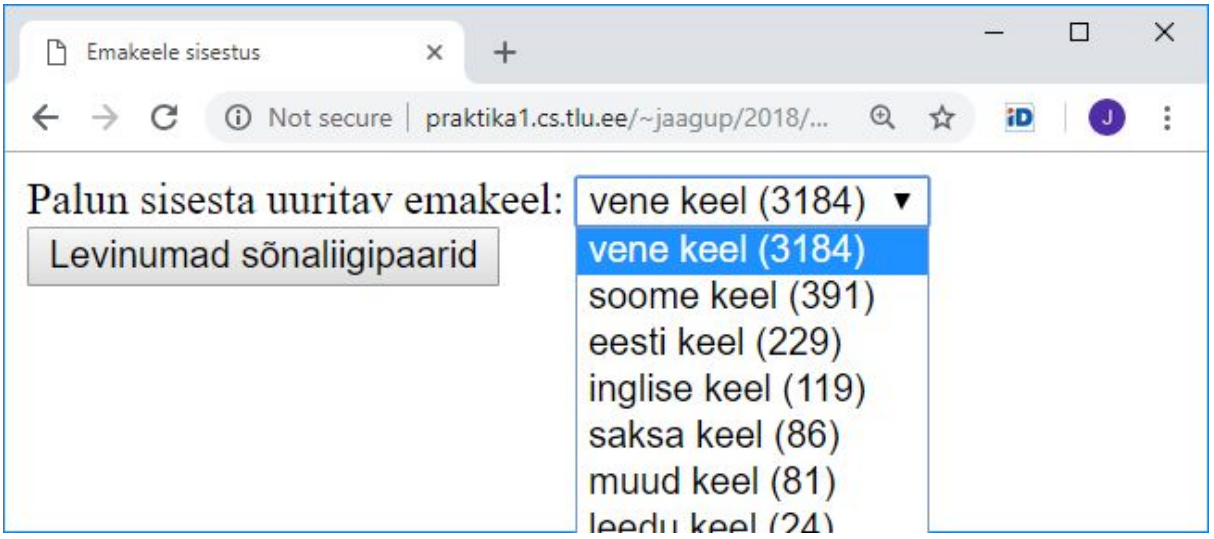
```
SELECT emakeel, COUNT(*) AS kogus FROM dokmeta
 WHERE emakeel IS NOT NULL
 GROUP BY emakeel
 HAVING COUNT(*)>1
 ORDER BY COUNT(*) DESC;
```

```
+---------+-------+
| emakeel | kogus |
+---------+-------+
| vene    |  3184 |
| soome   |   391 |
| eesti   |   229 |
| inglise |   119 |
| saksa   |    86 |
| muud    |    81 |
| leedu   |    24 |
| ukraina |    17 |
| ungari  |     6 |
| poola   |     3 |
| rootsi  |     2 |
| lati    |     2 |
+---------+-------+
12 rows in set (0.00 sec)
```

Päringus järjestatakse emakeeled tekstide arvu sageduse järjekorras. Sõnaliigipaare näitav leht tahab sisendiks keele nime väikeste tähtedega, rippmenüüs võib kasutajale aga kuvada ka midagi muud. Mis serverisse saadetakse vastaval real, tuleb kirjutada parameetri option väärtuseks.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
  $kasklus=$yhendus->prepare(
  "SELECT emakeel, COUNT(*) AS kogus FROM dokmeta
   WHERE emakeel IS NOT NULL
   GROUP BY emakeel
   HAVING COUNT(*)>1
   ORDER BY COUNT(*) DESC;");
  $kasklus->bind_result($keelenimi, $kogus);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Emakeele sisestus</title>
  </head>
  <body>
    <form action="leht4.php">
          Palun sisesta uuritav emakeel:
          <select name="emakeel">
         <?php
                 while($kasklus->fetch()){
                   echo "<option value='$keelenimi'>$keelenimi keel ($kogus)
                     </option>\n";
                 }
               ?>
          </select>
          <input type="submit" value="Levinumad sõnaliigipaarid" />
       </form>
  </body>
</html>
```

Nii näeb juba valikul, kui suure valimiga on millise keele puhul tegemist



Vaste kätte nagu ikka

Sisestatu säilitamine lehel

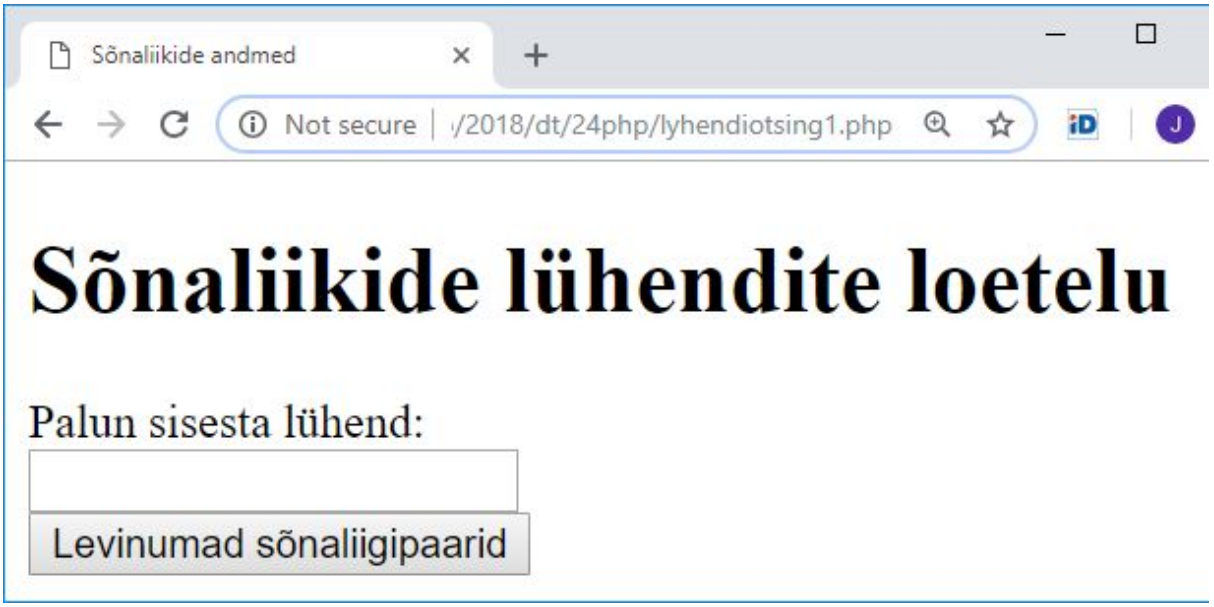
Lehe avamisel hakatakse toimetustega üldjuhul otsast peale. Kui aga parameetreid on rohkem, võib neid olla tülikas määrata - eriti juhul, kui märgatav osa neist võiks samaks jääda. Siin näites pannakse sisestusvorm ning kuvamise koht samale lehele ning kuvatakse lehe päringu vasteteks uuel avamisel välja esialgne sõnaliigi lühend. Kui saabuvate andmete hulgas on aadressirealt sisestatud lühend, siis läheb see muidu tühja muutuja “sisu” väärtuseks ning hiljem sealtkaudu tekstivälja väärtuseks

```
$sisu="";
if(isSet($_REQUEST["lyhend"])){$sisu=$_REQUEST["lyhend"];}
echo "<input type='text' name='lyhend' value='$sisu' /><br />";
```

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
```

```
  $kasklus=$yhendus->prepare(
    "SELECT liigilyhend, liigikirjeldus
          FROM sonaliikide_lyhendid WHERE liigilyhend=?");
  $kasklus->bind_param("s", $_REQUEST["lyhend"]);
  $kasklus->bind_result($lyhend, $kirjeldus);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Sõnaliikide andmed</title>
  </head>
  <body>
    <h1>Sõnaliikide lühendite loetelu</h1>
       <?php
          if($kasklus->fetch()){
                 echo "$lyhend - $kirjeldus<br />\n";
          } else if(isSet($_REQUEST["lyhend"])){
                 echo "$_REQUEST[lyhend] puudub baasist";
          }
       ?><br />
       <form action="?">
       Palun sisesta lühend:<br />
       <?php
          $sisu="";
          if(isSet($_REQUEST["lyhend"])){$sisu=$_REQUEST["lyhend"];}
          echo "<input type='text' name='lyhend' value='$sisu' /><br />";
       ?>
          <input type="submit" value="Levinumad sõnaliigipaarid" />

       </form>
  </body>
</html>
<?php $yhendus->close(); ?>
```

Esmasel avamisel on sisestuskast tühi



Edasi aga kuvatakse aadressireal saadetud lühend sisestuskasti

Sessioonimuutuja

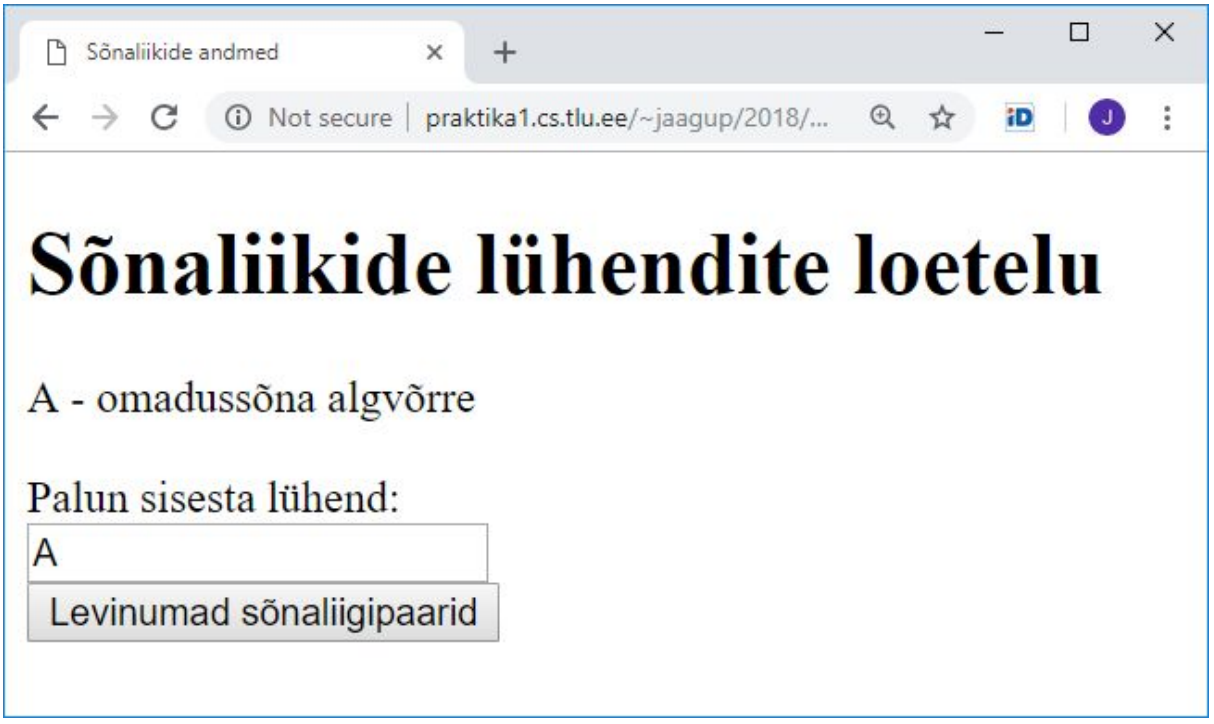
Aadressiribal andmete järelvedamine toimib ainult sama lehe piires. Mõnikord soovitakse kasutaja eelistusi ka mitme lehe peal kasutada või siis lehele tagasi tulles meenutada. Selliste kohtade juures aitab sessioonimuutuja - brauseri lahtioleku ajal kasutajaga seotuna serverisse salvestatud andmed. Nendega tegelemiseks tuleb koodi algusesse paigutada käsklus session_start(), mille tulemusena muutub kättesaadavaks muutuja $_SESSION. Kui selles on juba soovitud lühend salvestatud, siis pannakse see muutujasse $sisu, mida hiljem tekstivälja väärtuse kuvamisel kasutatakse. Kui juhtub ka aadressiribalt väärtus tulema, siis

sellega kirjutatakse mõlemad muutujad üle, nii et järgmisel korral on uus väärtus sessioonist võtta.

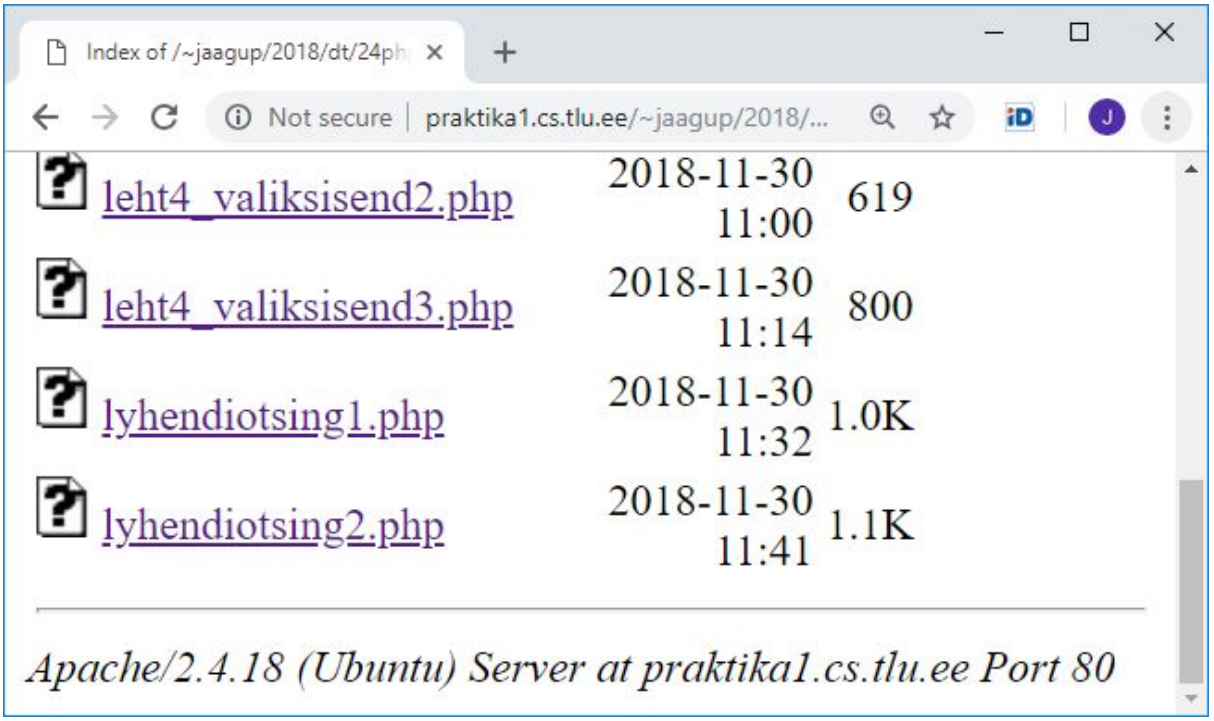
```
<?php
  session_start();
  $sisu="";
  if(isSet($_SESSION["lyhend"])){$sisu=$_SESSION["lyhend"];}
  if(isSet($_REQUEST["lyhend"])){
           $sisu=$_REQUEST["lyhend"];
              $_SESSION["lyhend"]=$_REQUEST["lyhend"];
  }

  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_keel");
  $kasklus=$yhendus->prepare(
    "SELECT liigilyhend, liigikirjeldus
          FROM sonaliikide_lyhendid WHERE liigilyhend=?");
  $kasklus->bind_param("s", $_REQUEST["lyhend"]);
  $kasklus->bind_result($lyhend, $kirjeldus);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
     <title>Sõnaliikide andmed</title>
  </head>
  <body>
    <h1>Sõnaliikide lühendite loetelu</h1>
       <?php
          if($kasklus->fetch()){
                 echo "$lyhend - $kirjeldus<br />\n";
          } else if(isSet($_REQUEST["lyhend"])){
                 echo "$_REQUEST[lyhend] puudub baasist";
          }
       ?><br />
       <form action="?">
       Palun sisesta lühend:<br />
       <?php
          echo "<input type='text' name='lyhend' value='$sisu' /><br />";
       ?>
          <input type="submit" value="Levinumad sõnaliigipaarid" />

       </form>
  </body>
</html>
<?php $yhendus->close(); ?>
```

Algul on tekstiväli tühi

Sisestatud väärtus ilmub sinna uuesti



Vahepeal võib vaadata näiteks eraldi lehena failide loetelu



Kui taas tagasi tulla, siis on lühend kastis endiselt olemas

Andmete lisamine

Andmeid küsivat ja analüüsivat rakendust on kõige ohutum teha - pole vaja mõelda, et kuhu lisanduvad andmed panna ning kas keegi pahatahtlik võiks serveri nendega ära ummistada.

Kui aga kasutajalt tulevaid andmeid edasi töödelda on vaja, siis salvestamisest ei pääse. Näites siseneme andmebaasi

```
mysql -udh18 -pdh18praktika dh18_jaagup
```

ning loome tabeli võrreldavate tekstide tarbeks. Igale tekstile poolkohustuslik automaatne id - nii on võimalik selle järgi teksti hiljem kindlalt eristada. Seletused tabeliloomislause sõnade juurde. Tulbad; id, autor ja sisu. INT = integer = täisarv, NOT NULL nõuab, et väärtus oleks kindlasti olemas. AUTO_INCREMENT = isesuurenev, ehk siis baas paneb automaatselt järgmise arvu väärtuseks. PRIMARY KEY teatab, et selle tulba järgi viidatakse tabeli reale. VARCHAR ehk muutuva pikkusega tekst, praegusel juhul pikkus kuni 50 sümbolit. TEXT tüübina tähistab suhteliselt piiramatu pikkusega teksti, piiriks pigem andmebaasi mahutavus.

```
CREATE TABLE vordlustekstid(
   id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   autor VARCHAR(50),
   sisu TEXT
);
```

Baasist käsu käivitamisel vastus, et tabeli loomine õnnestus

```
Query OK, 0 rows affected (0.01 sec)
```

Näitrida sisse. DEFAULT laseb baasil enesel ID-numbri panna.

```
INSERT INTO vordlustekstid VALUES (DEFAULT, 'Siim', 'Tere tulemast!');
```

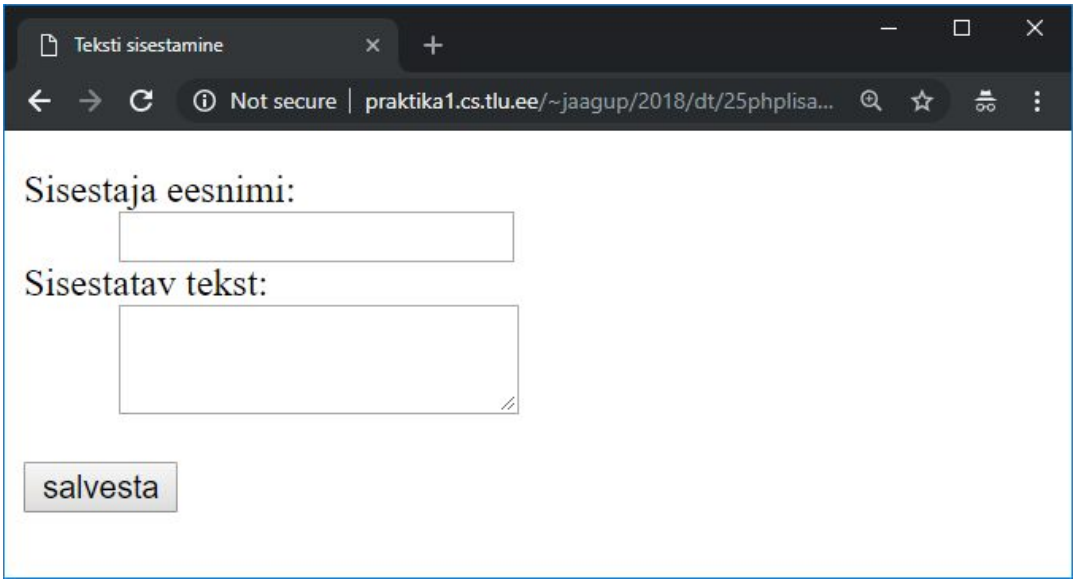
Kontroll, et andmed jõudsid kohale

```
MariaDB [dh18_jaagup]> SELECT * FROM vordlustekstid;
+----+-------+-----------------+
| id | autor | sisu            |
+----+-------+-----------------+
|  1 | Siim  | Tere tulemast! |
+----+-------+-----------------+
1 row in set (0.00 sec)
```

Sisestamiseks on vaja andmed kõigepealt kasutajalt kätte saada. Tekstiväli (input type=”text”) ning tekstiala (textarea) andmete kirjutamiseks. Mugavale paigutusele aitab kaasa definition list (dl) võtme (definition term, dt) ja väärtusega (definition data, dd). All submit-nupp andmete teele saatmiseks vormi action-parameetris näidatud aadressil

```
<!doctype html>
<html>
  <head>
    <title>Teksti sisestamine</title>
  </head>
  <body>
    <form action="salvestusleht.php">
        <dl>
          <dt>Sisestaja eesnimi:</dt>
          <dd><input type="text" name="eesnimi" /></dd>
```

```
          <dt>Sisestatav tekst:</dt>
          <dd><textarea name="tekstisisu" ></textarea></dd>
        </dl>
        <input type="submit" value="salvesta" />
      </form>
  </body>
</html>
```

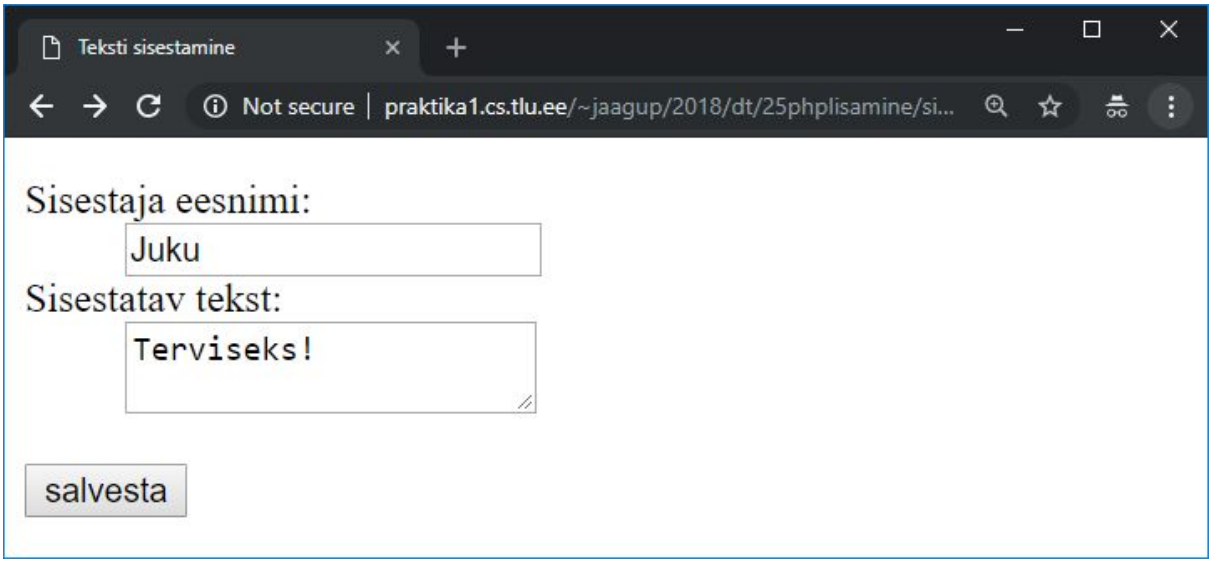


salvestusleht.php

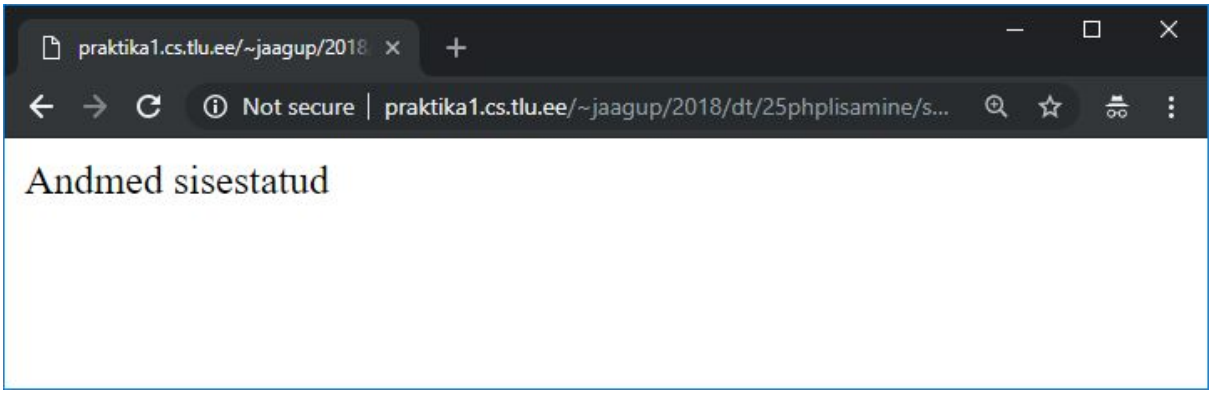
Lehe ülesandeks on saabuvad andmed baasi kirjutada. Taas kõigepealt küsimärgid lausesse ning hiljem parameetrite kaudu andmed sinna asemele, et ei tekiks võimalust veebist pahatahtlikke käsklusi teele saata. Lehe lõppu kasutajale teade sisestatud andmete kohta, et ta lihtsalt tühja valget lehte ei näeks.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
  $kasklus=$yhendus->prepare("INSERT INTO vordlustekstid VALUES (DEFAULT, ?, ?)");
  $kasklus->bind_param("ss", $_REQUEST["eesnimi"], $_REQUEST["tekstisisu"]);
  $kasklus->execute();
  $yhendus->close()
?>
Andmed sisestatud
```

Sisestusleht



ning salvestusleht



Ning kontroll, et andmed kohale jõudsid

```
MariaDB [dh18_jaagup]> SELECT * FROM vordlustekstid;
+----+-------+----------------+
| id | autor | sisu           |
+----+-------+----------------+
|  1 | Siim  | Tere tulemast! |
|  2 | Juku  | Terviseks!     |
+----+-------+----------------+
2 rows in set (0.00 sec)
```

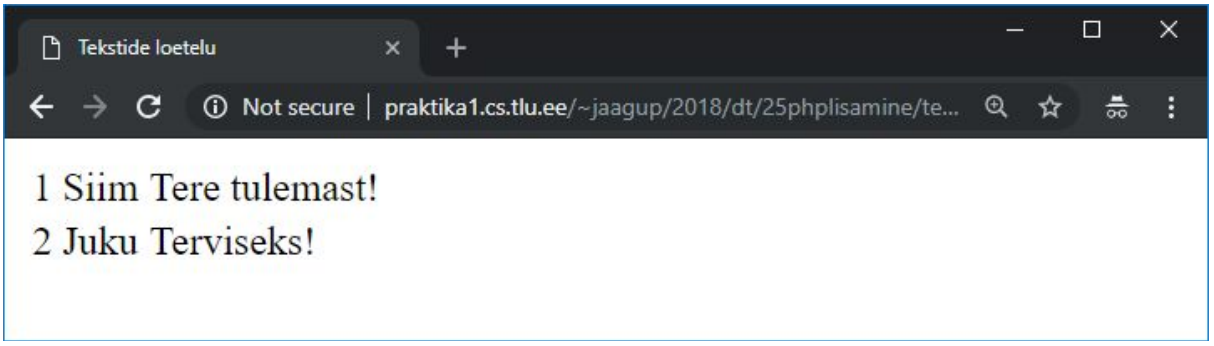
Andmete vaatamine

Päringu tulemuste nägemine varasemast tuttav. Juures on käsklused htmlspecialchars ning nl2br (newline to break). Esimene neist asendab tekstis sisalduvad HTML-i erisümbolid nende kuvamiseks brauseris sobivate koodidega. Teine asendab tekstisisesed reavahetused HTML-koodis mõjuvate reavahetustega.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
```

```
  $kasklus=$yhendus->prepare("SELECT id, autor, sisu FROM vordlustekstid");
  $kasklus->bind_result($id, $autor, $sisu);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Tekstide loetelu</title>
  </head>
  <body>
     <table>
          <?php
           while($kasklus->fetch()){
             echo "<tr><td>$id</td><td>".htmlspecialchars($autor)."</td><td>".
                  nl2br(htmlspecialchars($sisu))."</td></tr>";
              }
          ?>
        </table>
  </body>
</html>
```

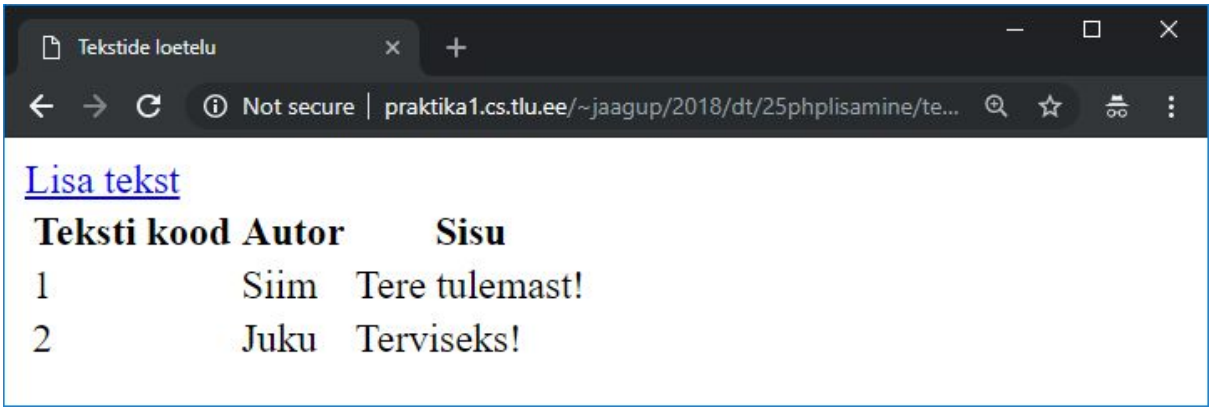
Loodud leht:



Lehele juurde mõned täiendused. Üles viide sisestuslehele, et saaks mugavamalt uut teksti lisada. Tabelile juurde veergude pealkirjad. Vastavaks tulbanimeks th - table head

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
  $kasklus=$yhendus->prepare("SELECT id, autor, sisu FROM vordlustekstid");
  $kasklus->bind_result($id, $autor, $sisu);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Tekstide loetelu</title>
  </head>
  <body>
     <a href="sisestusleht.php">Lisa tekst</a>
     <table>
          <tr>
            <th>Teksti kood</th><th>Autor</th><th>Sisu</th>
          </tr>
          <?php
           while($kasklus->fetch()){
             echo "<tr><td>$id</td><td>".htmlspecialchars($autor)."</td><td>".
                  nl2br(htmlspecialchars($sisu))."</td></tr>";
```

```
            }
        ?>
      </table>
  </body>
</html>
```



Seisestuslehe vormile juurde method=”POST”. Vaikimisi meetodi nimi on GET ning selle puhul edastatakse andmed aadressirea kaudu. Seal aga on pikkuspiirang ning pikemate tekstide tagumine pool läheb lihtsalt kaduma. POSTi puhul aga on lubatud andmemahud märgatavalt suuremad.

sisestusleht.php

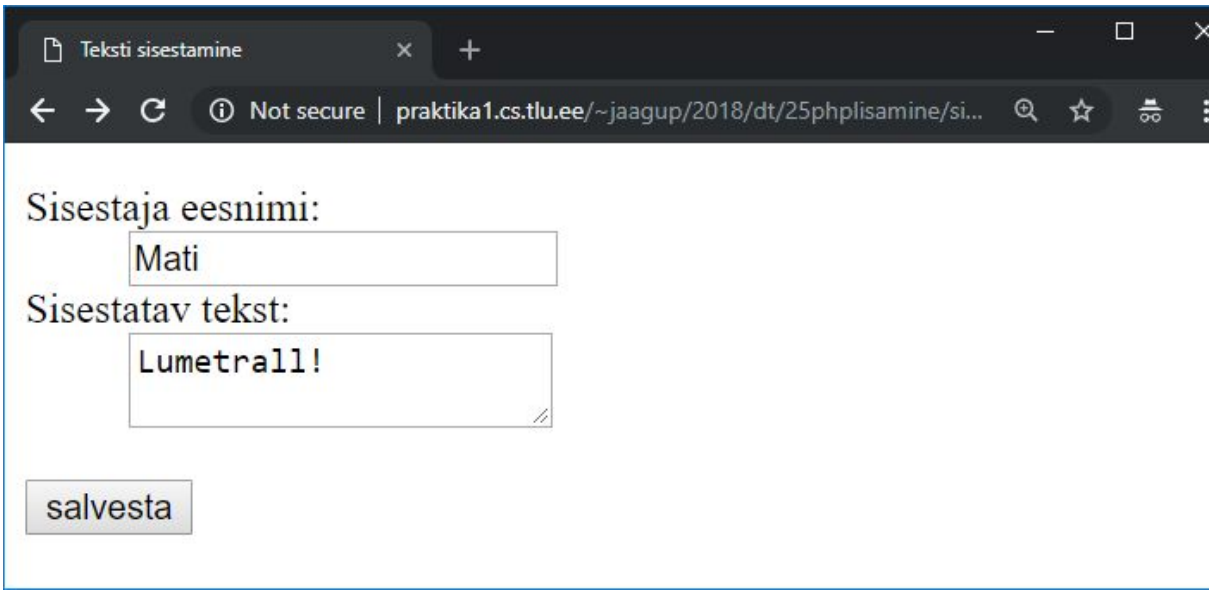
```
<!doctype html>
<html>
  <head>
    <title>Teksti sisestamine</title>
  </head>
  <body>
    <form action="salvestusleht.php" method="POST">
        <dl>
          <dt>Sisestaja eesnimi:</dt>
          <dd><input type="text" name="eesnimi" /></dd>
          <dt>Sisestatav tekst:</dt>
          <dd><textarea name="tekstisisu" ></textarea></dd>
        </dl>
        <input type="submit" value="salvesta" />
       </form>
  </body>
</html>
```

Andmete salvestamisel tasub kontrollida, et kas need üldse saadeti. Kui kogemata vajutati tühja välja puhul enterit või ärasaatmisnuppu, siis on parem tühjust baasi mitte panna. PHP koodi viimane rida annab käskluse veebilehitseja edasi suunata lehele nimega tekstideloetelu.php. Nii on see salvestusleht.php ajutine koht, mida kasutajale otseselt ei näidatagi, mis aga salvestustöö enese kanda võtab.

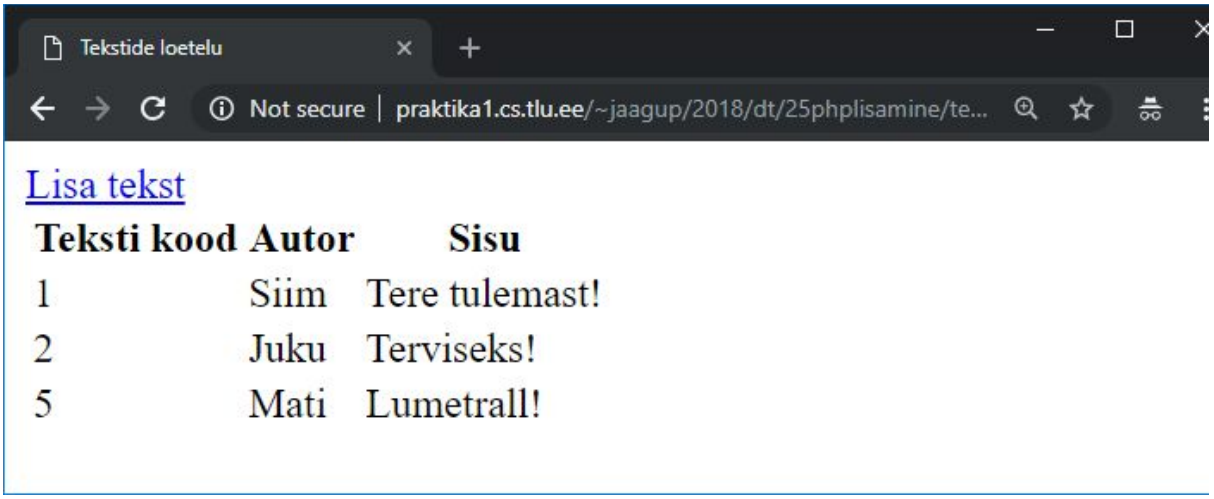
```
<?php
  if(empty($_REQUEST["eesnimi"]) or empty($_REQUEST["tekstisisu"])){
         echo "Autor või sisu puudub";
```

```
        exit();
  }
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
  $kasklus=$yhendus->prepare("INSERT INTO vordlustekstid VALUES (DEFAULT, ?, ?)");
  $kasklus->bind_param("ss", $_REQUEST["eesnimi"], $_REQUEST["tekstisisu"]);
  $kasklus->execute();
  $yhendus->close();
  header("Location: tekstideloetelu.php");
?>
```

Kasutaja paneb sisestuselehele andmed



ning näeb selle järel kohe juba tekstide loetelu



Tekstide võrdlemine

Teksti üks lihtne omadus on tema tähtede arv, mille SQL-i abil saab kätte käsuga LENGTH

```
MariaDB [dh18_jaagup]> SELECT id, autor, LENGTH(sisu) FROM vordlustekstid;
+----+-------+--------------+
| id | autor | LENGTH(sisu) |
+----+-------+--------------+
|  1 | Siim  |           14 |
|  2 | Juku  |           10 |
|  5 | Mati  |           10 |
+----+-------+--------------+
3 rows in set (0.00 sec)
```

Sama lause kaudu saab tekstide pikkused ka lehele kuvada.

Mõnigikord juhtub sisestusi, millest tahetakse loobuda. Muutmisest lihtsam moodus on eelmine tekst kustutada ning uus lisada. Kustutamiseks koostatakse viide

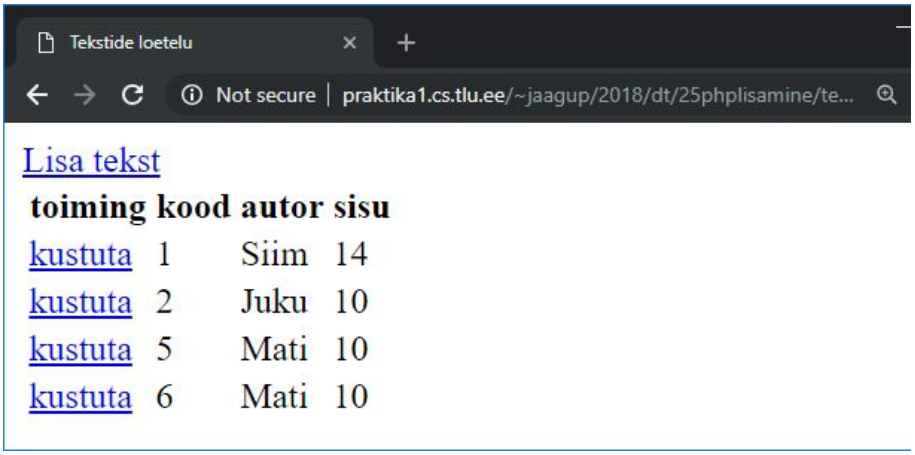
```
<a href='?kustutus=$id'>kustuta</a>
```

mis siis küsimärgi kaudu näitab samale lehele, kaasa pannakse teksti id.

Lehe ülaservas kontrollitakse, et kui kustutatava teksti id-number on määratud, siis käivitatakse DELETE SQL-lause vastava id-ga ning tekst kaob tabelist.

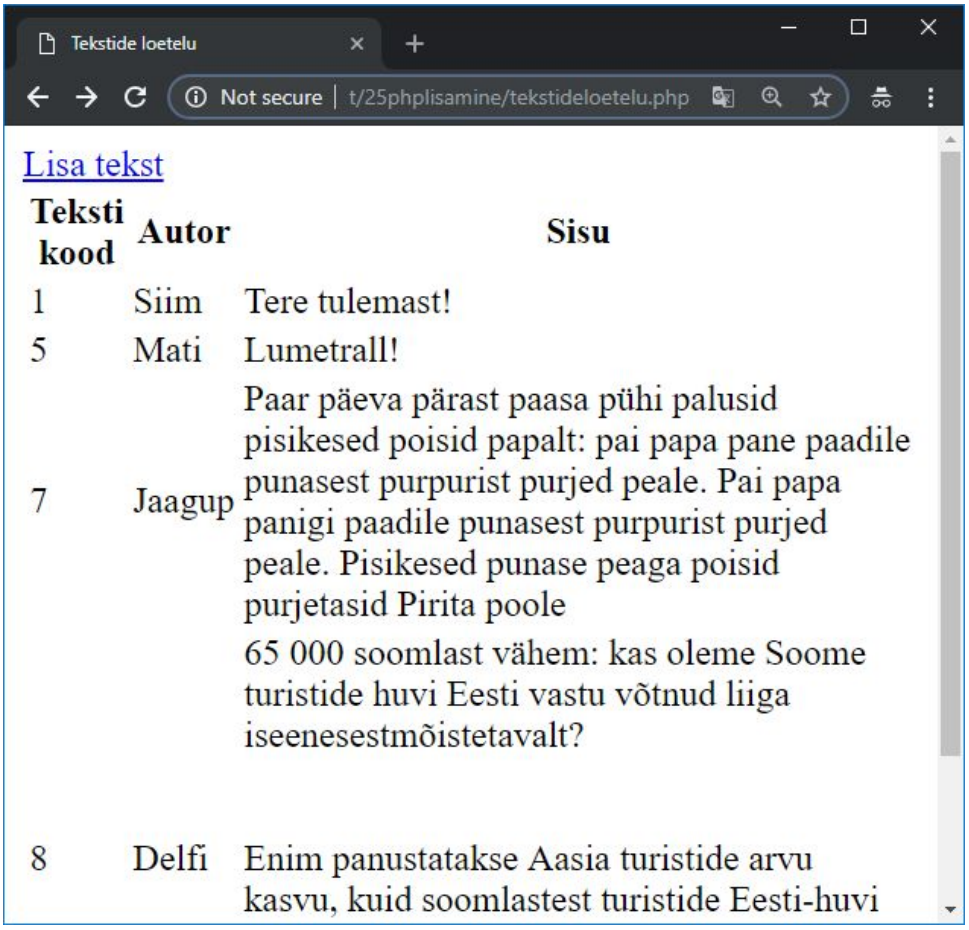
```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
  if(isSet($_REQUEST["kustutus"])){
         $kasklus=$yhendus->prepare("DELETE FROM vordlustekstid WHERE id=?");
         $kasklus->bind_param("i", $_REQUEST["kustutus"]);
         $kasklus->execute();
  }
  $kasklus=$yhendus->prepare("SELECT id, autor, LENGTH(sisu) FROM vordlustekstid");
  $kasklus->bind_result($id, $autor, $tahtedearv);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Tekstide loetelu</title>
  </head>
  <body>
     <a href="sisestusleht.php">Lisa tekst</a>
     <table>
          <tr>
            <th>toiming</th><th>kood</th><th>autor</th><th>sisu</th>
          </tr>
          <?php
           while($kasklus->fetch()){
             echo "<tr><td><a href='?kustutus=$id'>kustuta</a></td><td>$id</td><td>".
                   htmlspecialchars($autor)."</td><td>$tahtedearv</td></tr>";
            }
          ?>
        </table>
  </body>
</html>
```

Nagu näha, on Matilt algul kaks teksti



pärast kustutamist aga ainult üks.

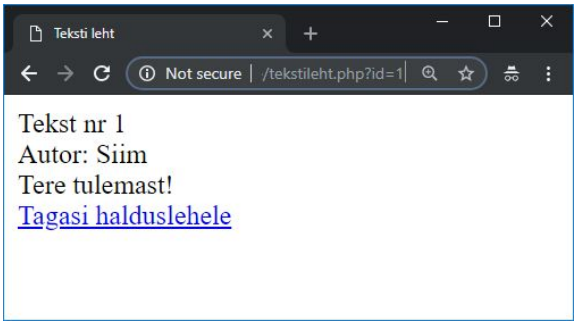
Paar veidi pikemat teksti juurde



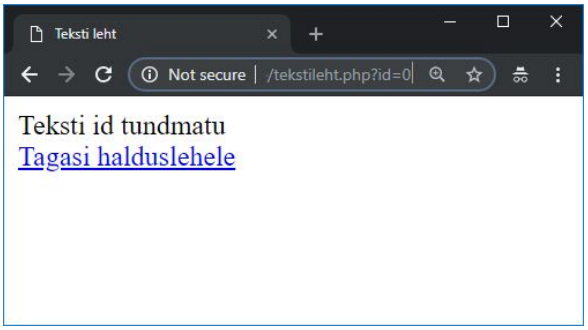
Ning juurde leht eraldi teksti vaatamiseks. Kui aadressireale pannakse vastava teksti id, siis näidatakse seda teksti.

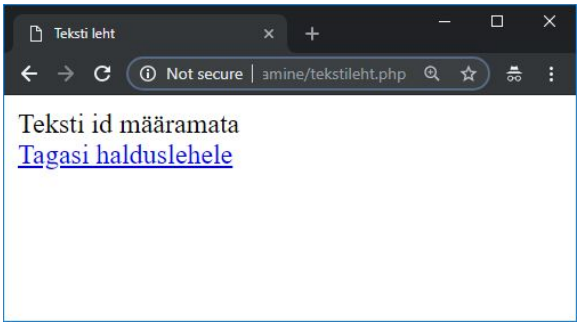
```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
?>
<!doctype html>
<html>
  <head>
    <title>Teksti leht</title>
  </head>
  <body>
      <?php
          if(isSet($_REQUEST["id"])){
        $kasklus=$yhendus->prepare(
          "SELECT id, autor, sisu FROM vordlustekstid WHERE id=?");
              $kasklus->bind_param("i", $_REQUEST["id"]);
              $kasklus->bind_result($id, $autor, $sisu);
              $kasklus->execute();
              if($kasklus->fetch()){
                     echo "Tekst nr $id<br />";
                  echo "Autor: ".htmlspecialchars($autor)."<br />\n";
                  echo nl2br(htmlspecialchars($sisu));
                     $kasklus->close();
              } else {
                 echo "Teksti id tundmatu";
              }
```

```
          } else {
                 echo "Teksti id määramata";
          }
         ?><br />
      <a href="tekstidehaldus.php">Tagasi halduslehele</a>
  </body>
</html>
<?php
  $yhendus->close();
?>
```



Juures on aga ka võimalused, et vastava id-ga teksti ei leidu või on id sootuks määramata

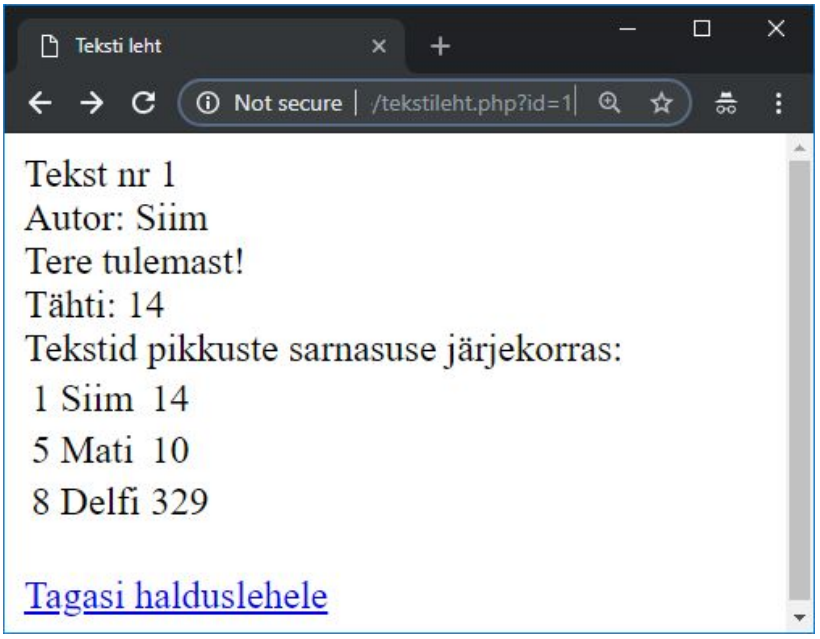




Juurde mõningane statistika teksti kohta ning tekstide hulgast sarnase pikkusega tekstide otsing. Loetelus näitatakse teised tekstid järjestatuna tähtede arvu erinevuse järgi vaadeldava tekstiga.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
?>
<!doctype html>
<html>
  <head>
    <title>Teksti leht</title>
  </head>
  <body>
      <?php
          if(isSet($_REQUEST["id"])){
        $kasklus=$yhendus->prepare(
          "SELECT id, autor, sisu FROM vordlustekstid WHERE id=?");
              $kasklus->bind_param("i", $_REQUEST["id"]);
              $kasklus->bind_result($id, $autor, $sisu);
              $kasklus->execute();
              if($kasklus->fetch()){
```

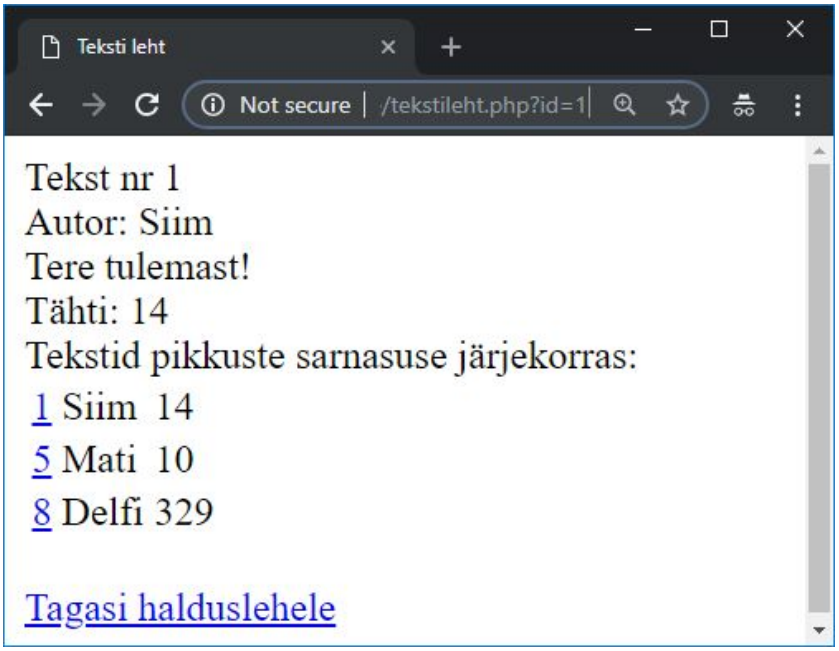
```
                    echo "Tekst nr $id<br />";
                 echo "Autor: ".htmlspecialchars($autor)."<br />\n";
                 echo nl2br(htmlspecialchars($sisu))."<br />";
                    $tahtedearv=strlen($sisu);
                    echo "Tähti: $tahtedearv<br />";
                    $kasklus->close();
                    $kasklus=$yhendus->prepare("SELECT id, autor, LENGTH(sisu) AS pikkus
                      FROM vordlustekstid
                      ORDER BY ABS(LENGTH(sisu)-?)");
                    $kasklus->bind_param("i", $tahtedearv);
                    $kasklus->bind_result($id, $autor, $pikkus);
                    $kasklus->execute();
                    echo "Tekstid pikkuste sarnasuse järjekorras: <table>";
                    while($kasklus->fetch()){
                     echo
"<tr><td>$id</td><td>".htmlspecialchars($autor)."</td><td>$pikkus</td></tr>";
                    }
                    echo "</table>";
              } else {
                 echo "Teksti id tundmatu";
              }
           } else {
                 echo "Teksti id määramata";
           }
          ?><br />
      <a href="tekstidehaldus.php">Tagasi halduslehele</a>
  </body>
</html>
<?php
  $yhendus->close();
?>
```

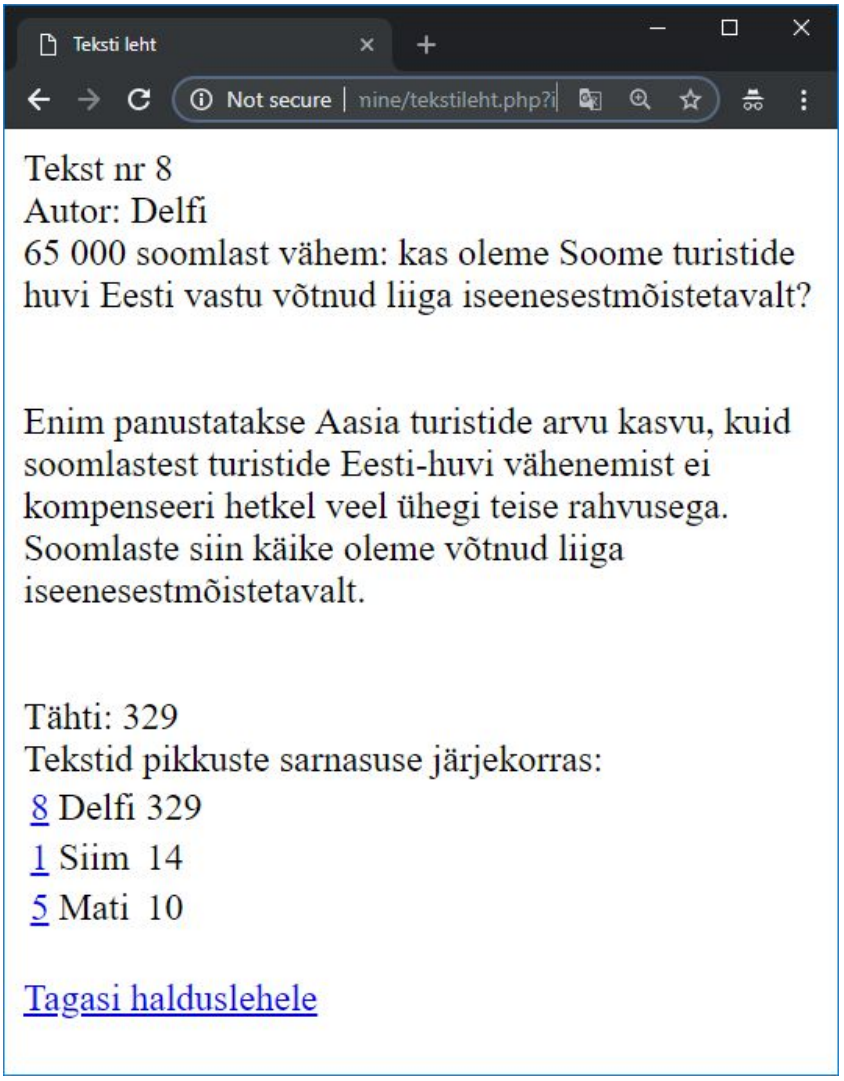


Harjutus

- Vajutades teksti numbrile, kuvatakse sama tekstileht vastava numbriga tekstiga

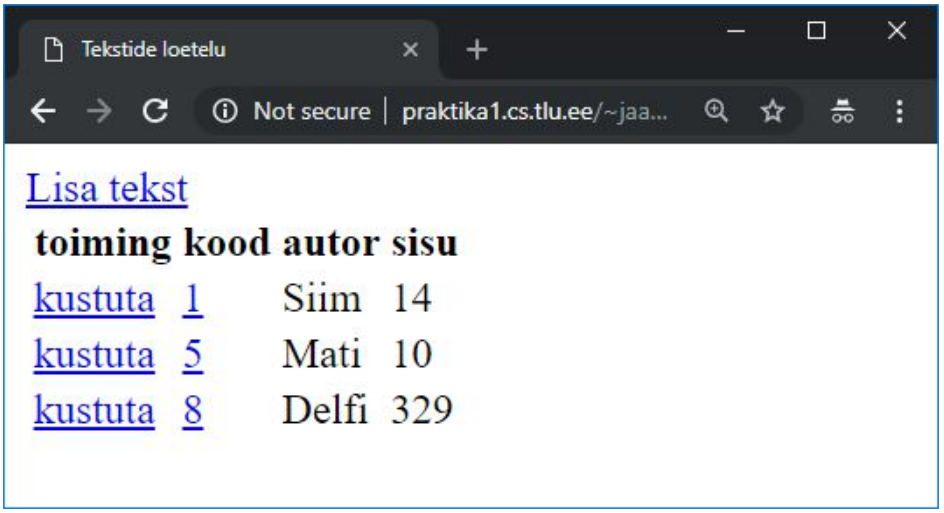
```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
?>
<!doctype html>
<html>
  <head>
    <title>Teksti leht</title>
  </head>
  <body>
      <?php
          if(isSet($_REQUEST["id"])){
        $kasklus=$yhendus->prepare(
          "SELECT id, autor, sisu FROM vordlustekstid WHERE id=?");
               $kasklus->bind_param("i", $_REQUEST["id"]);
               $kasklus->bind_result($id, $autor, $sisu);
               $kasklus->execute();
               if($kasklus->fetch()){
                      echo "Tekst nr $id<br />";
                   echo "Autor: ".htmlspecialchars($autor)."<br />\n";
                   echo nl2br(htmlspecialchars($sisu))."<br />";
                      $tahtedearv=strlen($sisu);
                      echo "Tähti: $tahtedearv<br />";
                      $kasklus->close();
                      $kasklus=$yhendus->prepare("SELECT id, autor, LENGTH(sisu) AS pikkus
                        FROM vordlustekstid
                        ORDER BY ABS(LENGTH(sisu)-?)");
                      $kasklus->bind_param("i", $tahtedearv);
                      $kasklus->bind_result($id, $autor, $pikkus);
                      $kasklus->execute();
                      echo "Tekstid pikkuste sarnasuse järjekorras: <table>";
                      while($kasklus->fetch()){
                       echo "<tr><td><a
href='?id=$id'>$id</a></td><td>".htmlspecialchars($autor)."</td><td>$pikkus</td></tr>";
                      }
                      echo "</table>";
               } else {
                  echo "Teksti id tundmatu";
               }
          } else {
                  echo "Teksti id määramata";
          }
         ?><br />
      <a href="tekstidehaldus.php">Tagasi halduslehele</a>
  </body>
</html>
<?php
  $yhendus->close();
?>
```


Teksti leht
Not secure | /tekstileht.php?id=1
Tekst nr 1
Autor: Siim
Tere tulemast!
Tähti: 14
Tekstid pikkuste sarnasuse järjekorras:
1 Siim 14
5 Mati 10
8 Delfi 329
Tagasi halduslehele


Teksti leht
Not secure | nine/tekstileht.php?i
Tekst nr 8
Autor: Delfi
65 000 soomlast vähem: kas oleme Soome turistide huvi Eesti vastu võtnud liiga iseenesestmõistetavalt?
Enim panustatakse Aasia turistide arvu kasvu, kuid soomlastest turistide Eesti-huvi vähenemist ei kompenseeri hetkel veel ühegi teise rahvusega. Soomlaste siin käike oleme võtnud liiga iseenesestmõistetavalt.
Tähti: 329
Tekstid pikkuste sarnasuse järjekorras:
8 Delfi 329
1 Siim 14
5 Mati 10
Tagasi halduslehele

Juurde kustutamise võimalus. Kui vastavat viidet vajutatakse, siis saadetakse vastav id-number serverisse ning selle numbriga teksti kustutatakse.

```
<?php
  $yhendus=new mysqli("localhost", "dh18", "dh18praktika", "dh18_jaagup");
  if(isSet($_REQUEST["kustutus"])){
         $kasklus=$yhendus->prepare("DELETE FROM vordlustekstid WHERE id=?");
         $kasklus->bind_param("i", $_REQUEST["kustutus"]);
         $kasklus->execute();
  }
  $kasklus=$yhendus->prepare("SELECT id, autor, LENGTH(sisu) FROM vordlustekstid");
  $kasklus->bind_result($id, $autor, $tahtedearv);
  $kasklus->execute();
?>
<!doctype html>
<html>
  <head>
    <title>Tekstide loetelu</title>
  </head>
  <body>
     <a href="sisestusleht.php">Lisa tekst</a>
     <table>
          <tr>
            <th>toiming</th><th>kood</th><th>autor</th><th>sisu</th>
          </tr>
          <?php
           while($kasklus->fetch()){
             echo "<tr><td><a href='?kustutus=$id'>kustuta</a></td>".
                     "<td><a href='tekstileht.php?id=$id'>$id</a></td><td>".
                   htmlspecialchars($autor)."</td><td>$tahtedearv</td></tr>";
              }
          ?>
        </table>
  </body>
</html>
```




Teksti leht
Not secure
imine/tekstileht.php?id=5
Tekst nr 5
Autor: Mati
Lumetrall!
Tähti: 10
Tekstid pikkuste sarnasuse järjekorras:
5 Mati 10
1 Siim 14
8 Delfi 329
Tagasi halduslehele

Kordamisküsimused

Otsimine ja asendamine tekstiredaktoriga - võimaluste ja tehnikate näiteid. Veebilehelt kopeeritud andmete puhastamine. Naise- ja mehenimede leidmise näide.

2x2 andmetabel - võimalikud arvutused ja esitused nende andmete põhjal. Rõhuasetused seotuna lauseehitustega.

Regulaaravaldised. Kasutusvaldkonnad ja näited. Otsimine, asendamine ning andmete korjamine. Veel näiteid.

Andmete avaldamine veebis. FTP/SCP, kasutajatunnus, parool, asukoht serveris, aadress veebis. Veebiruumi teenusepakkujad, hinnad ja võimalused. Sisevõrgu serverile ligipääs, tunnel.

Linuxi käsurida. Käsud pwd, cd, ls, mkdir, more, cat, echo, pico, wc, head, tail, grep, sort, cut, paste, bc. Operaatorid >, >> ja <.

Pythoni programmeerimiskeel. Käivitamine, võimalused, näited. Tähed tekstist, sõnad tekstist, split. Muutujad, omistamine, väärtuse muutmine. Tingimuslause. Massiivi loomine ja läbi käimine - filtreerimine ja muutmine. Collections ja counter.

Regulaaravaldised Pythonis - andmete eraldamine, findall. Käsud sum, max, map, list. Hulgad - otsimine, ühisosa, vahe, ühend - nion, intersection, difference, XOR.

Tekstifailid Pythoni keeles - lugemine, kirjutamine ja lisamine.

Andmehaldusteek Pandas. Andmete sisselugemine. Käsud head, tail, min, max, mean. Tulpade poole pöördumine, järjestamine tulba järgi, filtreerimine. Andmete küsimine massiivina. Tulba lisamine ja eemaldamine. Rühmitamine ja tehted rühmadega, andmed tagasi dataframeks. Väljund csv-faili.

Eesti keele analüüsiteek estnltk. Lause sõnade andmete leidmine - lemmad, sõnaliigid. Andmete lugemine failist ja veebist.

Andmetabelite ühendamine, merge, viited tulpade vahel, left, right, inner ja outer join. Tühjade lahtrite muundamine. Suhteliste osakaalude leidmine, järjestamine nende järgi.

Tekstide võrdlemine erisuguste näitajate alusel. Tähed, sõnad, laused, tähepaarid, sõnaliigipaarid - tehnilised näited nende kohta.

Veebilehtede genereerimine vastavalt kasutaja parameetritele. Tabel ja joonis veebilehel. Jooniste koostamine pandas + matplotlib teekide abil. Selgitavad tekstid joonisel.

Joondiagramm, tulpdiagramm, sektordiagramm, karpdiagramm, xy-diagramm, histogramm, mitme tunnuse kujutamine joonisel.

SQL ja relatsioonilised andmebaasid. Andmebaasi loomine, tabeli loomine, andmete sisestamine, muutmine, kustutamine, päringud. Primaarvõti, võõrvõti, andmebaasiskeem. Andmetüübid INT, DOUBLE, VARCHAR, TEXT. Grupeerimine, agregaatfunktsioonid, näited. Keelekorpuse andmebaas, päringute näited. Tekstide ja tekstirühmade võrdlemise näiteid. SQL-andmebaasi poole pöördumine Pythoni kaudu. Ühenduse loomine, päringu edastamine, tulemuste vastuvõtt.

PHP ja veebirakenduse loomine. Sisend kasutajalt, väljundi kuvamine. Ühendus SQL-andmebaasiga, tulemuste esitamine. PHP kaudu andmete lisamine SQL-andmebaasi, andmeid järjestava ja võrdleva ülevaatelehe loomine.