TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Inga Konovalova

Andmetöötluse mooduli pandas õppematerjalide koostamine programmeerimise kursustele

Bakalaureusetöö (9 EAP)

Juhendaja: Eno Tõnisson

Tartu 2018

Andmetöötluse mooduli pandas õppematerjalide koostamine programmeerimise kursustele

Lühikokkuvõte:

Käesoleva bakalaureusetöö raames koostati andmetöötluse õppematerjalid Pythoni mooduli (ingl k *library*) pandase kohta kahele programmeerimise kursusele. Esimene neist oli magistriõppekava „Infotehnoloogia mitteinformaatikutele" üliõpilastele mõeldud kursus „Programmeerimine" ning teise näol oli tegemist kõigile kättesaadava MOOCina (ingl k *massive open online course*) läbiviidava e-kursusega „Programmeerimise alused II". Materjalide sobivust analüüsiti õppijate lahenduste, tulemuste ja nende poolt antud tagasiside põhjal. Antud bakalaureusetöös keskendutakse õppematerjalide valmimise protsessi kirjeldamisele ja andmetöötluse peatüki tagasisidele, mida saab hiljem kasutada olemasoleva töö täiustamiseks.

Võtmesõnad:

Programmeerimise kursused, andmetöötlus, õppematerjal, MOOC, moodul pandas, programmeerimiskeel Python

CERCS: P175 - Informaatika, süsteemiteooria

Creating Data Analysis with pandas Module Themed Study Materials for Programming Courses

Abstract:

The goal of this thesis is to create study materials about data analysis with Python module pandas for two different programming courses. First of the courses is for master's students from curriculum "Conversion Master in IT" and the second one is e-course "Introduction to programming II" in MOOC that is available for everyone. Suitability of materials was analysed according to the solutions, results and feedback of the students

from both courses. This thesis is focused on the process of creation of the study materials and feedback which will help to improve already implemented work.

Keywords:

Programming courses, data analysis, study material, MOOC, module pandas, programming language Python

CERCS: P175 - Informatics, systems theory

Sisukord

Sissejuhatus **5**

1 Kursustest ja andmetöötlusest **7**

1.1 Kursused 7

1.2 Andmetöötlus mooduliga pandas 8

2 Õppematerjalide valmimise protsess **11**

2.1 Sissejuhatus 11

2.2 Video 12

2.3 Konspektid courses-keskkonna lehel 16

2.4 Loenguslaidid 19

2.5 Andmetöötluse foorum 20

2.6 Kodune ülesanne 21

3 Lõpuprojektid **23**

3.1 Kursus „Programeerimine” 23

3.2 E-kursus „Programmeerimise alused II” 26

4 Tagasiside **30**

4.1 Kursuse „Programeerimine” õppematerjalide tagasiside 30

4.2 E-kursuse lõpuküsitluse tulemused 30

Kokkuvõte **34**

Lisad **37**

I. Courses-keskkonnas asuvad materjalid 37

II. Loenguslaidid 67

III. Litsents 75

Sissejuhatus

Päevast päeva kogutakse erinevates elu- või uurimisvaldkondades suurel hulgal andmeid. Näiteks salvestavad telefonid ning arvutid andmeid oma kasutajate kohta ja inimene ise oma treeninguid pulsikella või aktiivsusmonitori abiga. Kogutud informatsioon annab võimaluse teha paremaid otsuseid nii igapäevaelus, ettevõtluses kui ka teaduses. Ettevõte saab kasutada kogutud infot uute ja veelgi paremate toodete väljatöötamiseks. Aktiivsuse ning pulsisageduste andmeid saab inimene kasutada aga treeningkavade kokkupanekuks. Kogutav statistika annab aimu riigi arengutasemest, majanduse olukorrast ning võimaldab planeerida muudatusi. Üheks valdkonnaks, kus statistika kogumine ja analüüs võivad olla suureks abiks, on meditsiin. Arstid saavad andmete abil teada, mis konkreetset haigust kõige sagedamini põhjustab ning mida saaks ette võtta haiguse ennetamiseks. Samuti on oluline teada, milliste soodumustega patsiendid haigestuvad [1]. Selleks, et huvipakkuvast teemast korralikku ülevaadet saada ja midagi otsustada, tuleb algselt andmeid otstarbekalt ning mõistlikult abistavate tarkvaravahenditega töödelda. Andmete töötlemiseks on näiteks olemas programmeerimiskeel R, mis on arendatud andmete statistiliseks töötluseks ja tabelarvutussüsteem Microsoft Excel [2, 3]. Käesolev bakalaureusetöö on aga seotud Pythonit käsitletavate kursustega, seega keskendutakse Pythoni andmetöötluseks mõeldud moodulile pandas. Andmetöötluse vajalikkus motiveeris alustama andmetöötluse õpetamist programmeerimise kursustel ning eestikeelsete materjalide koostamist. Enne käesolevas töös vaatluse all olevate õppematerjalide valmimist puudusid autorile teadaolevalt nii põhjalikud eestikeelsed materjalid andmetöötluse kohta mooduliga pandas.

Pandase looja on Ameerika statistik, andmeteadlane, tarkvaraarhitekt ning ettevõtja Wes McKinney. Ta põhjendas pandase arendamise vajalikkust sooviga lihtsustada ning muuta produktiivsemaks nii enda kui ka teiste andmeteadlaste töö. Samuti on tõhus andmete töötlus oluline lisaks teadlastele ka tavainimestele. McKinney tahtis luua piisava koguse

vajaminevate funktsioonidega tarkvaravahendi, mille kasutamine oleks kergemini selgeks õpitav ja sobiv ka algaja arvutikasutaja jaoks. [4]

Antud töö eesmärgiks on luua andmetöötluse teemalised õppematerjalid „Infotehnoloogia mitteinformaatikutele" magistriõppekava aine „Programmeerimine" ning MOOCi e-kursuse „Programmeerimise alused II" tarbeks. Kursused on küll sarnased, aga teatud erinevustega, seega loodi kaks varianti õppematerjalidest andmetöötluse teemal. Õppematerjalide loomisel arvestati sellega, et iga inimene võib soovi korral andmetöötlusega tegeleda ja ka vajalik informatsioon huvipakkuva teema kohta on tihti kättesaadav. Õppijatele tutvustati veebilehekülgi ning mooduseid, kust ja kuidas andmeid saada ning lühidalt funktsioone, mida andmete töötlemisel rakendada saab. Materjalide kokkupanekul arvestati juba olemasolevate ingliskeelsete materjalide struktuuriga ning nende põhjal loodi eestikeelsed materjalid, mida senini nendel kursustel kasutatud ei ole.

Bakalaureusetöö koosneb neljast osast. Esimeses osas antakse ülevaade, millistele kursustele materjalid loodi ja milline oli nende kursuste ülesehitus ning osalejate arv. Lisaks räägitakse lühidalt andmetöötluseks mõeldud moodulist pandas. Teises osas kajastatakse õppematerjalide valmimise protsessi. Seetõttu, et õppematerjalid loodi kahele üksteisest natuke erinevale kursusele, tuuakse välja õppematerjalide omavahelised erinevused ja sarnasused. Samuti räägitakse sellest, kuidas materjalide erinevad osad valmisid ning miks tehti just sellised valikud. Kolmandas lõputöö osas kirjeldatakse mõlemal kursusel tehtud lõpuprojekte, nende seotust andmetöötlusega ja õppijate poolt valitud teemasid. Neljandas osas analüüsitakse õppijate antud tagasisidet ning üldmuljet, mis on jäänud õppijatele nii andmetöötlusest kui ka konkreetsetest õppematerjalidest. Valminud konspekt ja loenguslaidid on nähtavad on toodud bakalaureusetöö lisadena. Kasutatud kirjanduse loetelu on järjestatud viitamise järjekorras.

1 Kursustest ja andmetöötlusest

1.1 Kursused

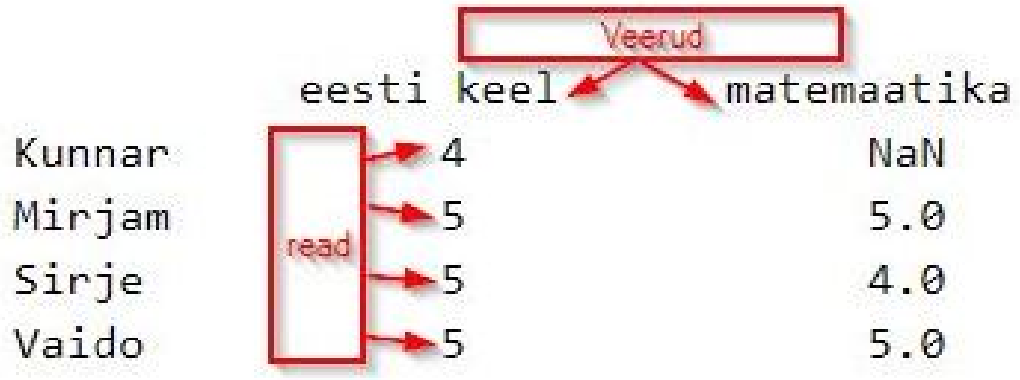
2017/2018. õppeaasta sügissemestril toimus magistriõppekava „Infotehnoloogia mitteinformaatikutele" kursus „Programmeerimine". Õppematerjale kasutati MOOCi kursuse „Programmeerimise alused" läbinud õppijatest moodustatud rühmas, mille osalejate arv oli 23. Õppetöö toimus põhiliselt Moodle'i vahendusel ning iga kahe nädala tagant loeti ka loenguid. Kursus toimus juba teist korda, kuid esimesel korral polnud pandasega andmetöötluse osa. Kursuse kava oli jaotatud viide ossa. Esimeses osas tuletati meelde materjale, mis said läbitud „Programmeerimise alused" e-kursusel, teemaks olid kahemõõtmeline järjend ja kahekordne tsükkel. Teises osas tutvuti andmestruktuuridega ning kolmandas osas viitamise ja muteerimisega. Kursuse eelviimases osas räägiti testimisest, silumisest ning rekursioonist. Viimasena mainitud osa alla kuulus ka andmetöötlus pandasega, mille jaoks valmisid õppematerjalid käesoleva bakalaureusetöö raames. Lõpupoole said õpilased tegeleda oma projekti kallal, mis pidi vastama teatud nõutele. Läbitavate teemade õpetamiseks kasutati loenguslaide, courses.cs.ut.ee (edaspidi courses-keskkond) lehel olevaid konspekte, kontrolltöid, koduseid ülesandeid ja teste. Courses-keskkonna näol on tegemist veebilehekülgega, mida kasutab Tartu ülikooli arvutiteaduse instituut. Seal asuvad erinevate kursuste kodulehed, kus on tihti kirjas kõik korraldusega seotud info ning sinna lisatud õppematerjalid kättesaadavad kõigile õppijatele.

„Programmeerimise alused II" e-kursus toimus 15. jaanuar - 11. märts 2018. aastal. See erines eelnevalt mainitud magistriõppekava kursusest sellepoolest, et õpetamine toimus vaid läbi Moodle keskkonna ja loenguid ei toimunud. Seega kasutatakse courses-keskkonna lehel olevaid konspekte ja teemade all olevaid kontrollülesandeid teadmiste kinnistamiseks. Kursusel osalejate arv oli 1064 ning käsitsi iga kontrollülesande hindamine ajamahukas, seega toimus põhiline ülesannete lahenduste hindamine automaatkontrol-

lide abiga. Käsitletavad teemad olid jaotatud kaheksasse ossa, millest kaks viimast olid seotud vaid projektiga. Nagu ka eelmainitud kursusel, olid teemadeks kahemõõtmeline järjend ning kahekordne tsükkel, kuid sellel kursusel olid need jaotatud kahe erineva nädala vahel. Kolmandas osas tutvuti andmestruktuuridega ning neljandas osas viitamise ja muteerimisega. Testimine, silumine, rekursioon I ja andmetöötlus pandasega kuulusid viiendasse ossa ning olid läbitavad, magistriõppekava kursusega sarnaselt, ühel ajal. Viimasesse projektieelsesse ossa kuulusid kordamine ja rekursioon II.

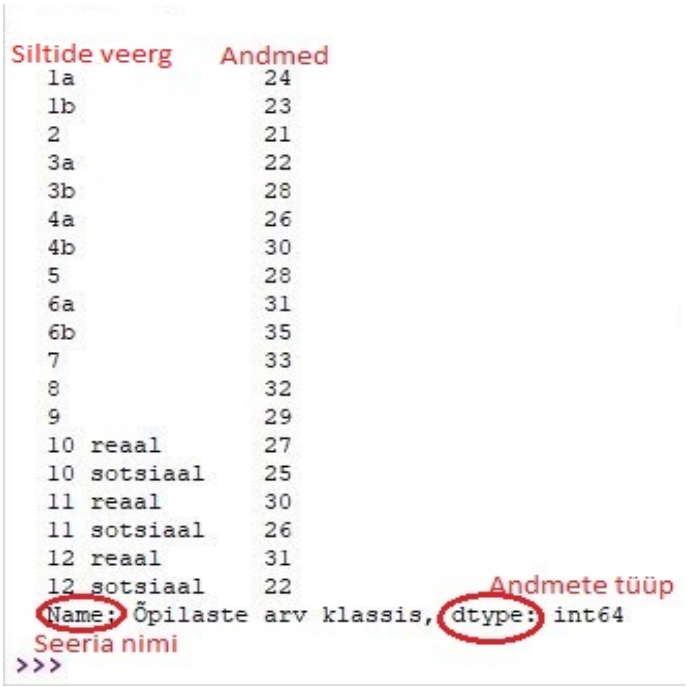
1.2 Andmetöötlus mooduliga pandas

Kõnealust moodulit kasutatakse laialdaselt andmete tõhusaks töötlemiseks. Samuti saab pandast kasutades teostada andmeanalüüsi ja koostöös mooduliga matplotlib andmeid ka visualiseerida. Pandase peamisteks andmestruktuurideks on Series, DataFrame ja Panel. Seetõttu, et pandase kohta pole varasemalt loodud ühtegi eestikeelset õppematerjali, tuli andmestruktuuride jaoks uued mõisted kasutusele võtta. Õppematerjalides ning edaspidi ka selles töös kasutame otsemugandusi nagu seeria (ingl k *Series*), andmefreim (ingl k *DataFrame*) ja paneel (ingl k *Panel*) [5] [6]. Andmefreim on kahemõõtmeline andmemassiiv, mida on lihtsam ette kujutada tabelina, sest selle struktuuri osad meenutavad ridasid ning veergusid. Iga rea eraldamiseks andmetest saab kasutada silte (ingl k *Label*), mis viitavad kindlale reale ning iga veeru eraldamiseks saab kasutada vastava veeru nime. [7] Järgnevas näites on siltideks nimed Kunnar, Mirjam, Sirje ja Vaido, mida kasutades saab tabelist kätte kindla rea. Veeru eraldamiseks tuleb kasutada ainete nimesid: eesti keel ja matemaatika.



Joonis 1. Kuvatõmmis konspektist - Andmefreim koos seletustega

Seeria näol on tegemist ühemõõtmelise andmestruktuuriga, mis tähistab andmefreimi ühte veergu ning milles hoitakse siltidega seostatud andmeid. See andmestruktuur on väga sarnane sõnastikuga, sest vajadusel võib iga elemendi sildi järgi kätte saada ning mainitud funktsionaalsus meenutab sõnastiku võtme ja väärtuse süsteemi.



Joonis 2. Kuvatõmmis konspektist - Seeria koos seletustega

Võrreldes teiste Pythonis kasutatavate andmestruktuuridega on pandase andmestruktuuridel palju funktsioone, mida saab andmete peal kiirelt ning lihtsalt rakendada - näiteks saab veergude ja ridadega teha aritmeetilisi tehteid, andmeid saab andmefreimis sorteerida ning vajaliku välja filtreerida.

2 Õppematerjalide valmimise protsess

2.1 Sissejuhatus

Järgnevalt on välja toodud kahe õppematerjalide komplekti erinevused ja sarnasused tabeli kujul.

Tabel 1. Õppematerjalide komplektide erinevused ja sarnasused kursustel „Programeerimine" ja „Programmeerimise alused II"

Õppematerjal	„Programeerimine"	„Programmeerimise alused II"
konspekt	lühem	mahukam
video	sama	sama
loeng	olemas	puudub
kodune ülesanne	olemas	modifitseeritud automaatkontrolli jaoks
foorum	puudub	olemas
lõpuprojekt	andmetöötlusega seotud	vaba teema

Magistriõppekava õppijate jaoks loodi konspekt courses-keskkonna lehele, õppevideo statistikaameti kodulehelt andmete allalaadimisest, kodune ülesanne raamatukogude teemal, lõpuprojekti nõuded ning kirjeldus ja loenguslaidid. Konspekt ja loenguslaidid koostati bakalaureusetöö autori poolt. Juhendaja tegeles siinkohal aktiivselt paranduste tegemisega ja nõu andmisega, nõu andsid ka teised käsitletavate kursuste õppejõud. Kursuse lõpus pidid õppijad välja mõtlema lõpuprojekti idee ja teostama selle. Projektide tingimused ning kirjeldus pandi paika ja räägiti läbi juhendaja ning kursuste teiste õppejõududega.

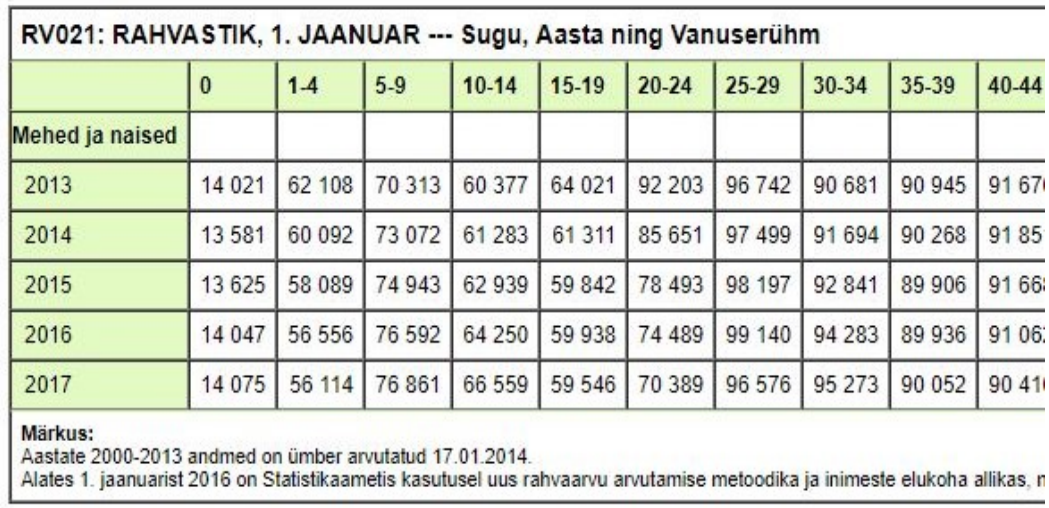
Mõlemal kursusel kasutati samat kodust ülesannet „Raamatukogud" ja videot andmete allalaadimise kohta. Nii video kui ka kodune ülesanne valmisid lõputöö autori poolt, kuid

koduse ülesande modifitseerimine automaatkontrolliga hindamise jaoks oli teostatud sama kursuse teiste õppejõudude poolt.

Lisaks eelmainitud õppematerjalidele, kontrolliti õppijate teadmisi e-kursusel ka testiküsimuste abil ja konspekti lugemisel oli jooksvalt võimalus vastata enesekontrolli küsimustele. Mainitud küsimused loodi juhendaja poolt. E-kursuse jaoks mõeldud materjalide täiustamiseks, ebaselgete kohtade seletamiseks ning vormistusvigadele viitamiseks, avati õppematerjalide avaldamise ning kasutamise ajaks Moodles foorum. Foorumi haldajaks, tekkinud küsimustele vastajaks ning materjalides paranduste tegijaks oli käesoleva lõputöö autor.

2.2 Video

Video koostamise eesmärgiks oli luua hõlpsasti jälgitav juhend selle kohta, kuidas statistikaameti kodulehelt andmeid sobival kujul alla laadida, programmis avada ning lihtsamal kujul töödelda. Kõnealuse video alguses sai jälgida, kuidas statistikaameti kodulehel erinevate kategooriate kaudu andmebaasi jõuda ja endale sobivate andmete tabel kokku panna. Lisaks seletati, kuidas valitud info õigel kujul alla laadida ja millised on üldse võimalikud failiformaadid. Näidisandmeteks kasutati tabelit „Rahvastiku jaotus soo ja vanuse järgi", mis laeti allas statistikaameti andmebaasist CSV kujul [8]. Mainitud failitüüp sai valitud tänu oma laialdasele kasutusele ning populaarsusele.



RV021: RAHVASTIK, 1. JAANUAR --- Sugu, Aasta ning Vanuserühm

	0	1-4	5-9	10-14	15-19	20-24	25-29	30-34	35-39	40-44
Mehed ja naised										
2013	14 021	62 108	70 313	60 377	64 021	92 203	96 742	90 681	90 945	91 67
2014	13 581	60 092	73 072	61 283	61 311	85 651	97 499	91 694	90 268	91 85
2015	13 625	58 089	74 943	62 939	59 842	78 493	98 197	92 841	89 906	91 66
2016	14 047	56 556	76 592	64 250	59 938	74 489	99 140	94 283	89 936	91 06
2017	14 075	56 114	76 861	66 559	59 546	70 389	96 576	95 273	90 052	90 41

Märkus:
Aastate 2000-2013 andmed on ümber arvutatud 17.01.2014.
Alates 1. jaanuarist 2016 on Statistikaametis kasutusel uus rahvaarvu arvutamise metoodika ja inimeste elukoha allikas, r

Joonis 3. Kuvatõmmis statistikaameti koduleheküljelt - õppevideos kasutatud andmed

Järgnevalt loeti andmed programmi koodis sisse ning tutvuti lühidalt töötlemise võimalustega. Näidisandmete peal rakendati mõningaid põhifunktsioone, mille kasutamist seletati ka konspektis. Läbitud materjali uuesti seletamine videos aitab teadmisi paremini omandada ning on vajadusel abiks nendele õppijatele, kellel oli probleeme konspektist arusaamisel. Videos õpetati, kuidas tutvuda sisseloetud andmetega (saada teada ridade või veergude arvu, tabelis olevate kirjete arvu, pealkirjasid ja silte), eemaldada ebavajalikud read või veerud ja rakendada aritmeetilist funktsiooni summa arvutamiseks.

		.1	0	1-4	5-9	10-14	15-19	\
0	Mehed ja naised	NaN	NaN	NaN	NaN	NaN	NaN	
1		2013.0	14021.0	62108.0	70313.0	60377.0	64021.0	
2		2014.0	13581.0	60092.0	73072.0	61283.0	61311.0	
3		2015.0	13625.0	58089.0	74943.0	62939.0	59842.0	
4		2016.0	14047.0	56556.0	76592.0	64250.0	59938.0	
5		2017.0	14075.0	56114.0	76861.0	66559.0	59546.0	

	20-24	25-29	30-34	...	55-59	60-64	65-69	\
0	NaN	NaN	NaN	...	NaN	NaN	NaN	
1	92203.0	96742.0	90681.0	...	88342.0	82220.0	60379.0	
2	85651.0	97499.0	91694.0	...	88964.0	82132.0	64621.0	
3	78493.0	98197.0	92841.0	...	89250.0	81921.0	69475.0	
4	74489.0	99140.0	94283.0	...	88728.0	83510.0	73809.0	
5	70389.0	96576.0	95273.0	...	89106.0	83645.0	75851.0	

	70-74	75-79	80-84	85-89	90-94	95-99	100 ja vanemad
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	64688.0	50293.0	37164.0	..	..	..	..
2	60720.0	52273.0	36746.0	..	..	..	..
3	57028.0	54557.0	36191.0	..	..	..	..
4	54248.0	55174.0	36430.0	22076	7422	1038	128
5	52762.0	56572.0	37131.0	22858	7983	1192	125

[6 rows x 24 columns]

Joonis 4. Kuvatõmmis Thonnyst - videos kasutatud andmed töötlemise eel

```
              0      1-4      5-9    10-14    15-19    20-24    25-29    30-34  \
Aasta
2013   14021.0  62108.0  70313.0  60377.0  64021.0  92203.0  96742.0  90681.0
2014   13581.0  60092.0  73072.0  61283.0  61311.0  85651.0  97499.0  91694.0
2015   13625.0  58089.0  74943.0  62939.0  59842.0  78493.0  98197.0  92841.0
2016   14047.0  56556.0  76592.0  64250.0  59938.0  74489.0  99140.0  94283.0
2017   14075.0  56114.0  76861.0  66559.0  59546.0  70389.0  96576.0  95273.0

          35-39    40-44      ...          55-59    60-64    65-69    70-74  \
Aasta                         ...
2013   90945.0  91676.0      ...        88342.0  82220.0  60379.0  64688.0
2014   90268.0  91851.0      ...        88964.0  82132.0  64621.0  60720.0
2015   89906.0  91668.0      ...        89250.0  81921.0  69475.0  57028.0
2016   89936.0  91062.0      ...        88728.0  83510.0  73809.0  54248.0
2017   90052.0  90416.0      ...        89106.0  83645.0  75851.0  52762.0

          75-79    80-84  85-89  90-94 95-99 100 ja vanemad
Aasta
2013   50293.0  37164.0     ..     ..    ..             ..
2014   52273.0  36746.0     ..     ..    ..             ..
2015   54557.0  36191.0     ..     ..    ..             ..
2016   55174.0  36430.0  22076   7422  1038            128
2017   56572.0  37131.0  22858   7983  1192           125

[5 rows x 22 columns]
Aasta
2013    1294645.0
2014    1288396.0
2015    1284170.0
2016    1285280.0
2017    1283477.0
dtype: float64
```

Joonis 5. Kuvatõmmis Thonnyst - videos kasutatud andmed töötlemise järel

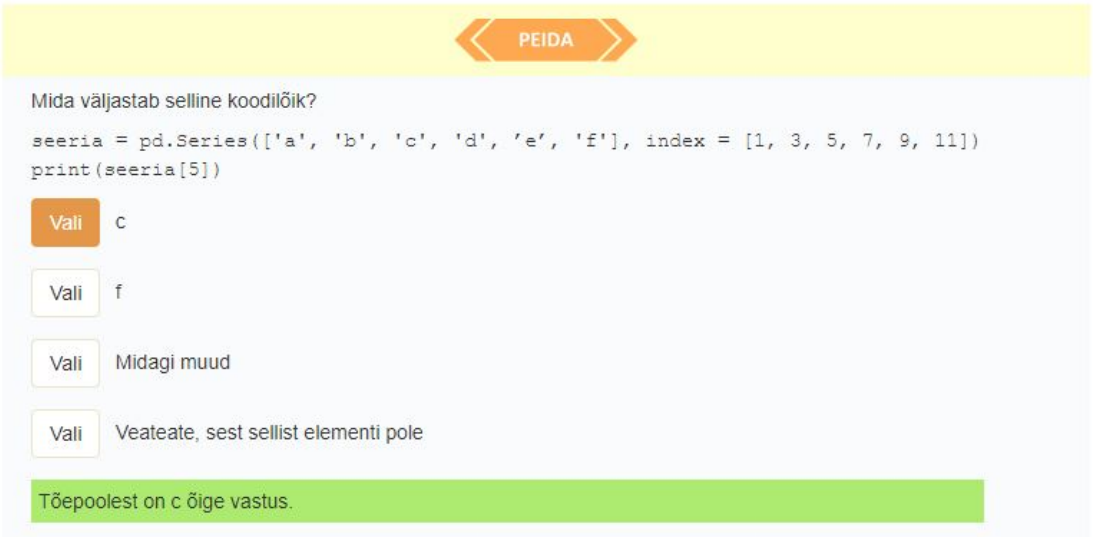
Video tegemiseks kasutati videotöötluse jaoks mõeldud tasuta tarkvara Ezvid Video Maker, mille abiga ühendati pealeloetud tekst ekraanisalvestustega [9]. Põhjusel, et autor polnud varem õppevideote tegemisega kokku puutunud, kulus enamus ajast suuliste õpetuste sisselugemisele. Keeruliseks osutus veatu ja häirivate faktoriteta video loomine ning alles mitmete katsete järel oli lõpptulemus piisavalt hea. Õppevideo on leitav Google Drive keskkonnas [10].

2.3 Konspektid courses-keskkonna lehel

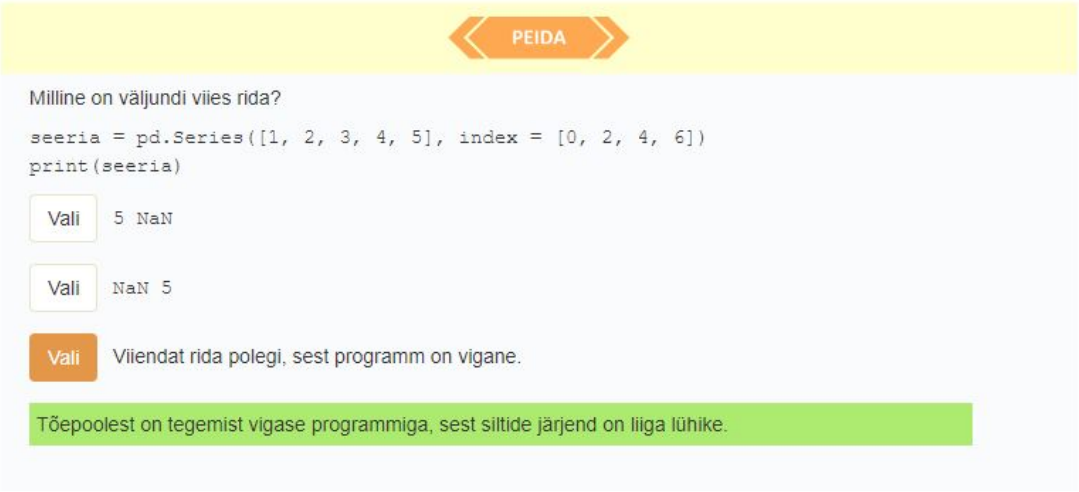
Sügissemestri magistriõppekava kursuse õppematerjalide komplekti üheks osaks sai loodud näitepõhine konspekt. Nagu eelnevalt mainitud, loeti kõnealuse kursuse raames ka loenguid, seega keskenduti konspekti kokkupanemisel sellele, et juba käsitletud teemad saaksid sügavamalt kaetud. Loengute lugejaks oli käesoleva lõputöö juhendaja. Konspektiks valiti näide, milles andmeid töödeldi ning seletati põhjalikumalt lahti, kuidas konkreetset operatsiooni teostati. Näites kasutati teatrite kohta käiva infoga tabelit, mis laeti alla juba varasemalt mainitud statistikaameti andmebaasist CSV faili kujul [8]. Sedasorti andmed said valitud oma arusaadavuse, mahukuse ja elulisuse järgi. Käesoleva bakalaureusetöö autori arvates oli oluline kasutada lihtsaid ja mahult mitte nii suuri andmeid, et õppijad näeksid funktsioonide rakendamisel koheselt vahet ning saaksid ka ise kontrollida saadud tulemuste tõesust.

Konspekt on jaotatud nelja suuremasse alapeatükki, millest igaüks seletas lähemalt lahti andmete töötlemisega seotud erinevad käsud. Õppematerjali struktuur ja teemade jaotus osadesse sarnanes suures osas videos kasutatule. Konspekti alguses kirjeldati andmete avamist ja kuvamist programmis ning näidati, milline on vahe tekstiredaktor *notepad*'iga avatud ja sisseloetud andmete visuaalsetel kujudel. Allalaetud ning programmis avatud andmed olid tabeli kujul, seega salvestati need andmefreimi. Järgnevalt näidati võimalusi sisseloetud infoga tutvumiseks. Selleks tutvustati õppijatele mitmeid erinevaid funktsioone, mille abiga sai kätte näiteks veergude pealkirju ja kindla arvu ridu tabeli algusest või lõpust. Infoga tutvumise järel rakendati veelgi funktsioone, mis määrasid näiteks kindla veeru siltide veeruks, eemaldasid andmeid ja muutsid veeru nime. Sissejuhatavate näidete järel kirjeldati lühidalt pandasega tihedalt seotud matplotlib mooduli kasutamist graafikute kuvamiseks. Andmetega tutvumiseks ja nende töötlemiseks kasutatud operatsioonid ning teadmised rakendamise kohta on lõputöö autori arvates heaks baasiks teiste funktsioonide kasutamisel ja pandasest aru saamisel.

Eelnevalt kirjeldatud näitepõhine konspekt võeti kasutusele ka e-kursuse õppematerjalina. Vaid internetis toimuva õppe tõttu tuli luua põhjalik ning piisavate selgitustega konspekt, mis annaks hea ülevaate ning millest saadud teadmised aitaksid lahendada kodust ülesannet või teha andmetöötluse teemal projekti. Näitepõhine konspekt oli e-kursuse jaoks mõeldud suure konspekti üheks eraldi peatükiks, mis kinnistas esimesest osast saadud teadmisi ja samas pakkus ka veidi teistsuguseid näiteid. Mõlemaid konspekte saab näha lisade I peatükis. Esimene osa konspektist koosnes mitmest väiksemast alapeatükist. Alustati installeerimisprotsessist ning seejärel tutvustati pandases kasutatavaid andmestruktuure ja nende funktsioone. Andmestruktuuride vaatlemisel üritati tuua paralleele teiste Pythonis kasutatavatega, toodi välja sarnasused ja põhjendused selle kohta, miks kasutada just pandase andmestruktuure. Selline võrdlus põhjendas autori arvates ära, miks pandas andmetöötluseks sobib ja aitas pandases kasutatavaid andmestruktuure paremini mõista. Näidetes kasutati väljamõeldud andmeid, mis olid piisavalt lihtsad selleks, et märgata iga funktsiooni rakendamise järel toimunud muutusi. Edasi keskenduti pikemalt ka sellele, kuidas üldse andmefreimi või seeriat luua, sest pandases saab seda teha mitmel erineval moel. Konspekti esimese osa peatükke lugedes said õppijad vastata ka enesekontrolli küsimustele. Neid saab näha järgnevatel piltidel.



Joonis 6. Kuvatõmmis enesekontrolli küsimusest e-kursuse jaoks mõeldud konspekti esimeses osas



Joonis 7. Kuvatõmmis enesekontrolli küsimusest e-kursuse jaoks mõeldud konspekti esimeses osas

Enesekontrolli küsimused on lõputöö autori arvates vajalikud seetõttu, et õppijad saavad saadud teadmisi koheselt rakendada ning sellisel moel neid paremini kinnistada. Samuti aitavad need tekstile paremini keskenduda, sest lugemise ajal hajunud tähelepanu koondub küsimusele vastamise järel tekstile paremini.

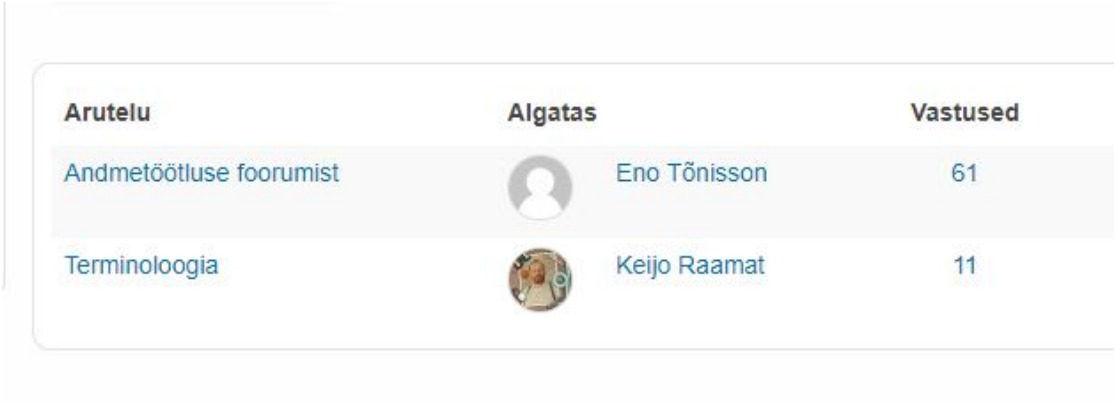
2.4 Loenguslaidid

Esitlus, mis magistriõppekava kursuse loengu jaoks loodi, oli lõputöö autori esimeseks loodud õppematerjaliks ning pidi andma kompaktse ülevaate sellest, mida pandas endast kujutab ja kuidas seda kasutada. See koosnes 25 slaidist, mis jaotati viie suurema teema vahel (kokkupandud slaidid on kättesaadavad lisade II peatükis). Esimesena räägiti üldiselt andmetöötlusest, selle tähtsusest ja kasutusvõimalustest. See osa esitlusest juhatas üldise teema hästi sisse ning andis aimu sellest, miks üldse andmete töötlust programmeerimise kursustel õpetatakse ning miks see tänapäeval tähtis on. Samuti püüti esimeses osas anda õppijatele edasi motivatsiooni andmetöötlusega tegelemiseks. Järgmisena oli vaatluse all moodul ise ja selle põhilised andmestruktuurid koos näidetega ning andmete sisselugemine programmi. Andmestruktuuride põhjalik tutvustamine annab vajalikud teadmised koduseks tööks ning on vajalik edaspidiste keerulisemate funktsioonide mõistmiseks. Seejärel tutvustati õppijatele erinevaid allikaid, kust andmeid saada võiks - näiteks räägiti statistikaameti kodulehest, hariduseteemalise statistika andmebaasist HaridusSILM ning mainiti ära ka see, et on võimalik allalaadida andmeid tänapäeval palju kasutatavatest aktiivsusmonitoridest või pulsikelladest. Võimalike allikate tutvustamine andmete saamiseks on oluline lõpuprojekti teostamiseks, sest seal tuleb õppijal valida andmed ja töödelda neid vastavalt ette antud nõuetele. Viimaks anti juhtnöörid pandase installimiseks Thonnys ja räägiti eelseisvast andmetöötluse teemalisest projektist. Juhendaja tegi esitluses omapoolsed parandused ja pidas valminud õppematerjali abiga loengu

andmetöötluse teemal.

2.5 Andmetöötluse foorum

Andmetöötluse teema näol oli tegemist millegi uuega, mille kohta ei leidu piisavalt eestikeelseid materjale. Kuna vastloodud õppematerjal oli samuti uus ja täiustamist vajav, avati õppematerjalide kasutusele võtmise ajaks e-kursuse õppijate jaoks foorum, kus õppijad said jooksvalt tekkivaid küsimusi küsida ja teha ettepanekuid paranduste tegemiseks konspektis. Mainitud foorum oli väga aktiivne ning igapäevaselt tekkis sinna nii ettepanekuid kui ka märkusi täpsustust vajavate osade kohta õppematerjalides. Sellise abivahendi kasutuselevõtt aitas käesoleva lõputöö autoril viia sisse muudatusi ning täiendada segaseks jäänud kohti konspektis.



Joonis 8. Kuvatõmmis andmetöötluse jaoks mõeldud foorumist

Rääkides foorumi kasutamisest üldiselt, tasub mainida, et see aitab parandada õppijate sooritust. On teada, et õppijad esitavad ainete foorumites rohkem läbimõeldud küsimusi, seega analüüsitakse saadud informatsioon paremini läbi ning teadmised kinnistuvad veelgi kindlamalt. Avalik arutelu probleemide üle pole kasulik ainult neile õppijatele, kes küsimusi esitavad ning aktiivselt kaasa räägivad. Tehtud uuringutest selgub, et ka

teemade lugemisega piirduvate õppijate sooritused paranevad. Eelnev informatsioon tugineb artiklile ja foorumite kasutamise kohta kirjutatud magistritööle [11, 12].

2.6 Kodune ülesanne

Koduses ülesandes kasutati õppijatele etteantud andmetena tabelit raamatukogude kohta. Andmetes kirjeldati Eesti raamatukogude arvu 2012-2016 aastatel, lugejate arvu kokku, lugejate arvu keskmiselt raamatukogu kohta, laenutusi lugeja kohta ning töötajate arvu. Ülesande eest oli võimalik saada 2 punkti. Magistriõppekava kursusel esitati kokku 18 lahendust. Enamus esitatud töid oli korrektselt lahendatud ning teenisid täispunktid. Järgnevalt on näha ülesande püstitus ja selles kasutatud andmed.

KU01: RAHVARAAMATUKOGUD --- Näitaja ning Aasta					
	2012	2013	2014	2015	2016
Raamatukogud	559	556	549	540	536
Lugejad, tuhat	383.2	377.7	368.1	363.4	351.3
Lugejaid keskmiselt raamatukogu kohta	686	679	670	673	655
Laenutusi keskmiselt lugeja kohta, arvestusüksust	30.5	29.6	29.1	28.6	28.5
Töötajad	1 554	1 544	1 528	1 471	1 442
Märkus: Rahvaarvuga seotud näitajate puhul on alates 2012. aastast kasutatud 2011. aasta rahvaloendusel põhinevat rahvaarvu, varasematel aastatel 2000. aasta rahvaloendusel põhinevat rahvaarvu.					

Joonis 9. Kuvatõmmis koduses ülesandes kasutatud andmetest

Ülesande püstitus:

1. Esimene veerg teha indeksite veeruks.
2. Lisada uus veerg „Keskmine”.
3. Eemaldada veerg pealkirjaga „2012”.

4. Arvutada ridade keskmised ja lisada need vastloodud veergu.

5. Väljastada ekraanile andmed tabeli kohta: veergude, ridade arv, veergude pealkirjad, ridade pealkirjad.

6. Esitada „Raamatukogud” rea andmed graafikul.

Nagu eelnevalt mainiti, muudeti e-kursuse jaoks kodune ülesanne automaatkontrollile sobivaks. Õppijatele anti samad andmed raamatukogude kohta, kuid programmi struktuur oli seekord paika pandud. Kirjutatud koodis pidi olema kaks funktsiooni („taienda_tabelit” ja „raamatukogud”) ning üks delegeeris osa oma tööst teisele. Andmete töötlemise punktid nagu veeru lisamine/eemaldamine, keskmise arvutamine jäid samaks, kuid eemaldati nõue, mille kohaselt tuleb ekraanile väljastada andmed tabeli kohta. Andmete esitamine graafikul oli seekord etteantud lisaülesandena, mida võis lahendada vabatahtlikult. Sellel kursusel hinnati ülesannet arvestatud/mittearvestatud kujul. Kokku esitati 421 lahendust, millest 9 said hindeks mittearvestatud, ülejäänud 412 lahendust said hindeks arvestatud.

Käesoleva bakalaureusetöö autori meelest oli tegemist jõukohase ülesandega, mille lahendamisel sai abi ülejäänud õppematerjalidest. Ülesande raskusaste tulenes loenguslaidide ja konspekti mahust. Eesmärgiks oli luua konspektis kasutatud näite põhjal ülesanne, mille lahenduste esituste järgi saaks vaadata, kas õppijad on materjalist piisavalt aru saanud.

3 Lõpuprojektid

3.1 Kursus „Programeerimine”

Magistriõppekava kursuse lõpuprojekti idee ja teostus pidid kindlasti seotud olema andmete töötlemisega. Esimeseks tähtajaks tuli õppijatel esitada lõpuprojekti teema kirjeldus, mille eest teeniti maksimaalselt 2 punkti ning hiljem üleslaadida ka projekt ise ja selle eest teeniti veel maksimaalselt 8 punkti.

Paika olid pandud järgmised nõuded:

1. Programm peab olema enda tehtud.
2. Orienteeruv töömaht lahendamisel 8 tundi inimese kohta. (Võis teha kahekesi)
3. Programm peab töötlema andmeid.
    - Andmed võivad olla pärit veebist või enda omad.
    - Andmed tuleb lugeda failist.
4. Programm peab kasutajalt midagi küsima.
5. Programm peab andmete põhjal midagi mõistlikku arvutama ja tulemuse ekraanile kuvama.

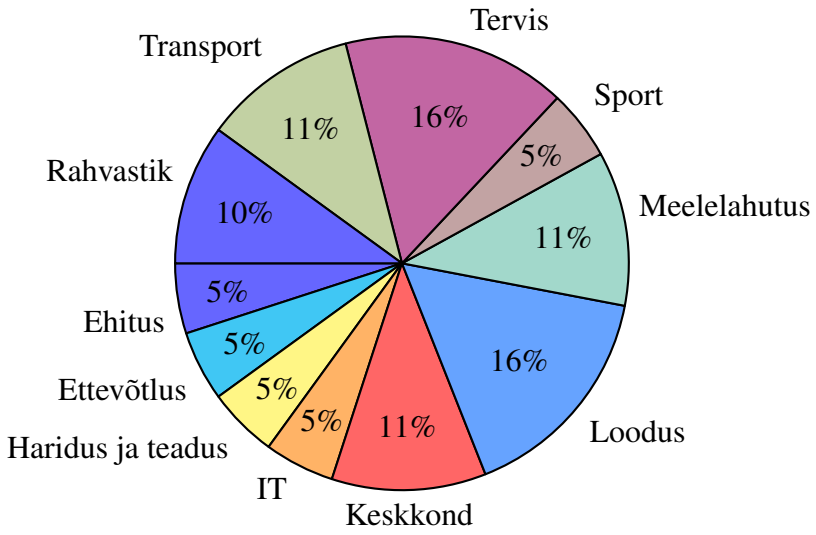
Lisaks tuli valida kaks nõuet neljast ja täita need:

- graafiline väljund
- filtreeritud andmete teise faili kirjutamine
- kasutaja valiku põhjal arvutuste tegemine
- iseseisva (ilma Pythonita käivituva) programmi kirjutamine

Kokku esitati 19 erinevat lõpuprojekti. Pea kõigi õppijate kirjutatud programmid olid seotud nende enda või tegevusvaldkonna andmetega. Esitatud projektid olid igaüks omamoodi ning neid oli huvitav avada ja katsetada. Järgnevalt on näha, kuidas tinglikult jaotusid lõpuprojektid autori poolt valitud erinevate teemade vahel.

Tabel 2. Lõpuprojektide tinglik jaotus autori poolt valitud erinevate teemade vahel

Lõpuprojekt	Teema, mille alla kuulub
Erinevate ühte tüüpi tsemendist tehtud betooni katsekehade uurimine	Ehitus
Vähki haigestumise ja suremise visuaalne võrdlus	Tervis
Unegraafik pulsikellast - oma andmed	Tervis
Unegraafik aktiivsusmonitorist - oma andmed	Tervis
Prantsusmaa ettevõtlusstatistika	Ettevõtlus
Murdekorpuse andmete analüüs	Haridus- ja teadus
Põtrade liikumisandmete analüüs	Loodus
Metsauuendamine	Loodus
X kromosoomi-spetsiifiliste kordusjärjestuste leidmine veise genoomist	Loodus
Rahvastik soo, vanuse ja asustusüksuste järgi	Rahvastik
Nimede populaarsus	Rahvastik
Olümpiaalade harrastajad maakonna, spordiala, soo ja vanuserühma järgi	Sport
Dubleeritud failide kustutaja kaustadest	IT
Videos olevate objektide tuvastus ja analüüs	Meelelahutus
TED Talk'ide kohta käiv informatsioon	Meelelahutus
Enamkasutatavate raskemetalliühendite ja toksiliste ainete kasutamine ettevõtetes	Keskkond
Kombineeritud ilmaprognoos erinevatelt teenusepakkujatelt	Keskkond
Kasutaja poolt sisestatud autode kogus riigis	Transport
Ülevaade Eesti ettevõtete tasulisest veoseveost, veoautodest	Transport



Joonis 10. Lõpuprojektide tinglik protsentuaalne jaotus autori poolt valitud erinevate teemade vahel

Kõik projektid loodi kasutades pandasele lisaks veel mitmeid muid mooduleid. Palju kasutati ka selliseid Pythoni võimalusi, mida antud kursusel ei õpetata. Oli näha, et õppijad nägid vaeva ning lõid enda jaoks huvitavaid ja kasutuskõlblikke programme.

3.2 E-kursus „Programmeerimise alused II”

E-kursuse „Programmeerimise alused II” lõpuprojekti tingimused olid sarnased magistriõppekava kursuse omadega. Programm võis, aga ei pidanud olema seotud andmetöötlusega ning selle loomiseks pidi minema orienteeruvalt 8 tundi. Lõpuprojekti eest oli igal õppijal võimalik saada kuni 15 punkti. Projekti kood oli 12 punkti väärtusega ja lisaks sellel tuli esitada projekti aruanne, mis andis 3 punkti.

Programmis pidid kindlasti mõistlikult kasutusel olema:

- valikulause (3 punkti),
- funktsioon (3 punkti).

Järgmistest pidid kasutusel olema vähemalt neli:

1. tsükkel (1 punkt),
2. kasutajalt millegi küsimine (1 punkt),
3. failist lugemine või faili kirjutamine (1 punkt),
4. järjed, ennik, hulk, sõnastik või pandase seeria (1 punkt),
5. kahemõõtmeline või mitmemõõtmeline andmestruktuur (nt kahemõõtmeline järjend, ennik või pandase andmefreim) (1 punkt),
6. lihtne graafika või graafiline kasutajaliides (nt kilpkonnagraafika, Tkinter, EasyGUI, pandas/matplotlib diagramm või graafik (1 punkt).

Projekti aruanne pidi sisaldama:

- projekti põhjalikku kirjeldust, kus on kirjas programmi eesmärk ja selgitus programmi üldisest tööst, vajadusel lühike kasutusjuhis;
- ülevaadet tööprotsessist - ajakulu, hinnang oma töö lõpptulemusele,
- selgitust ja/või näiteid, kuidas programmi osi eraldi ja oprgrammi tervikuna testisite ehk kuidas veendusite, et programm töötab korrektselt

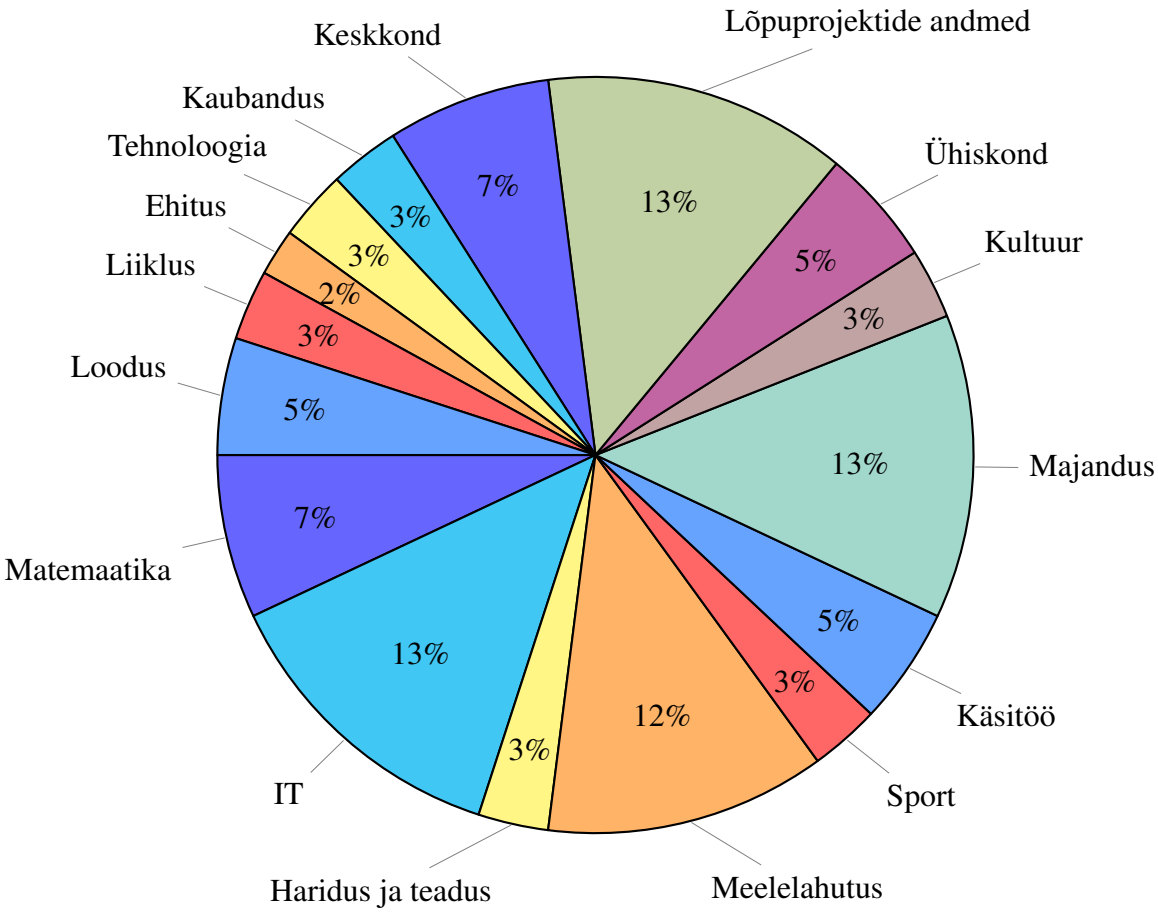
Selleks, et teha andmetöötluse valiku teinud õpilaste projektide kirjutamine huvitavamaks, avati pandase teemale järgneval nädalal 10 küsimust sisaldav küsimustik. Tulemused koondati kokku CSV faili ning muudeti õppijatele kättesaadavaks lõpuprojektide tegemise ajaks. Küsimustikku ning tulemusi haldas kursuse teine õppejõud.

Küsimused olid sellised:

1. Mis on Su eesnime esimene täht?
2. Sugu
3. Mis on Su lemmik tsitaat?
4. Mis on esimene sõna, mis tuleb pähe, kui loed sõna "PROGRAMMEERIMINE"?
5. Keda sooviksid enda programmeerimisõpetajaks?
6. Kuivõrd Sulle meeldib programmeerida?
7. Millist operatsioonisüsteemi tavaliselt kasutate?
8. Kas olete sel talvel suusatanud?
9. Kas olete sel talvel uisutanud?
10. Valige palun üks arv 1-9.

Projekte sai esitada avanenud õpikodades. Õpikoda on sarnane programmeerimisülesande automaatkontrollile, kuid automaatse kontrollimise asemel määratakse esitatud töö ühele juhuslikult valitud kaaskursuslasele hindamiseks. Iga töö esitaja peab ka ühte tööd hindama. Kokku esitati kaheksa õpikoja jooksul 367 projekti.

Bakalaureusetöö autor analüüsis juhusliku valiku teel 60 esitatud projekti. E-kursusel esitatud lõpuprojektid olid magistriõppekava kursusega võrreldes sama raskusastmega. Aruandeid lugedes, programmi avades ning katsetades oli näha, et õppijad on ideed põhjalikult läbimõelnud ning loonud rakendused, mis on kasutuskõlblikud. Järgneval sektordiagrammil on näha analüüsitud 60 projekti jaotus teemade vahel.



Joonis 11. Lõpuprojektide tinglik protsentuaalne jaotus autori poolt valitud teemade vahel

Nagu eelneval diagrammil näha, oli 13% õppijate poolt tehtud projektidest seotud nende andmetega, mida jooksvalt kursuse jooksul küsimustiku abiga koguti. Kuigi andmetöötluse temaatika polnud kohustuslik, kasutasid 31 õppijat 60st pandast ja selle võimalusi ning tegelesid andmete töötlemisega. Lõputöö autori arvates mõjutas huvi pandase kasutamise vastu see, et ette olid antud andmed, mida koguti õppijate enda kohta. Kuigi kogutud andmed oli kõigi õppijate jaoks samad, leidis igaüks siiski neis midagi uut ja lõi seetõttu teistest erineva projekti. Osad pandast kasutanud õppijad tõid aruannetes välja,

et kasutavad seda just seetõttu, et moodul tundub praktilises mõttes vajalik ja huvitav.

4 Tagasiside

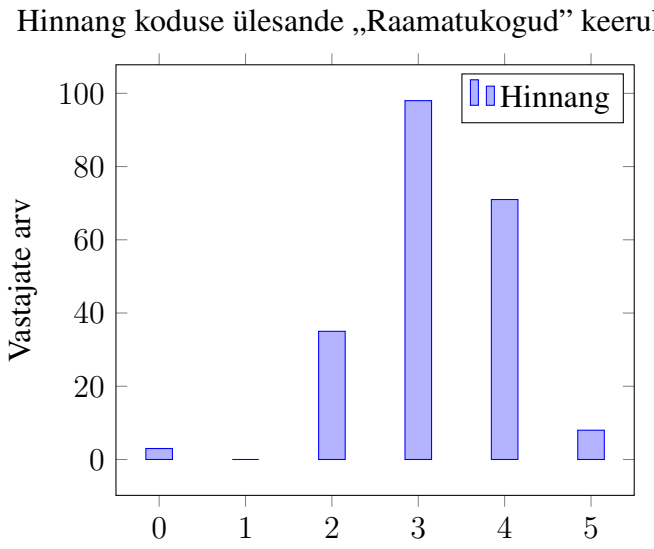
4.1 Kursuse „Programeerimine" õppematerjalide tagasiside

Kursusel „Programeerimine" õppematerjalide kasutamise ajal avatud tagasiside küsimustikule vastas 6 õppijat. Vastused küsimusele „Mis oli hästi?" tõid välja põhiliselt kolm arvamust. Mainiti, et õppematerjalidesse lisatud video oli hästi tehtud ja kergesti jälgitav. Samuti meeldis õppijatele materjal, sest see oli lihtne ja piisavalt detailsete selgitustega. Õppijate arvates on pandas huvitav Pythoni moodul ja seda on kindlasti tarvis ka edaspidi üliõpilastele tutvustada.

Küsimus „Mida saaks veel paremini teha?" tõi välja mõned täiustamist vajavad kohad õppematerjalides. Konspektis sooviti näha rohkem põhjalikumaid näiteid arvutuslike funktsioonide rakendamiste kohta (nt keskmised, summad jm) ja kontrollülesandeid. Lisaks arvati, et võiks teha sarnaseid õppevideosid ka teiste olulisemate pandase osade kohta. Olemasoleva video täiendamiseks soovitati lisada seletused pandase levinumate käskude kohta.

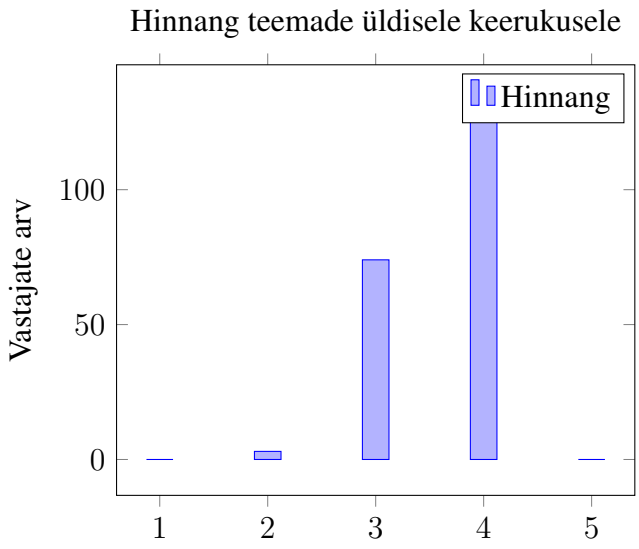
4.2 E-kursuse lõpuküsitluse tulemused

E-kursuse õppematerjalide jaoks eraldi tagasiside küsimustikku ei loodud. Analüüsiti vastuseid, mis on pärit kursuse üldisest lõpuküsitlusest. Lõpuküsitlusele vastas 215 õppijat. Esimesena tuli õppijatel hinnata 5. nädala koduse ülesande „Raamatukogud" keerukust. Järgnevalt näete esitatud küsimuse vastuste kokkuvõtet:



Joonis 12. Hinnang koduse ülesande „Raamatukogud” keerukusele

Õppijad vastasid ka küsimusele kursuse 5. ja 6. nädala teemade üldise keerukuse kohta. Nagu järgnevalt tulpdiagrammilt näha, oli andmetöötluse koduse ülesande keerukus õppijate arvates väiksem kui kursuse mainitud teemade üldine keerukus. Seega oli ülesanne pigem lihtsam kui keerulisem teiste läbitud teemade ülesannetest, kuid leidus õppijaid, kelle jaoks oli see kursuse kõige keerulisem osa. Selliseid õppijaid oli küsitlusele vastanutest 16.



Joonis 13. Hinnang 5. ja 6. nädala teemade üldisele keerukusele

Lisaks keerukusele hindasid õppijad ka teemade huvitavust. 30% õppijatest mainis enda vastustes, kas pandase moodulit, „Raamatukogud" ülesannet või matplotlib moodulit. Osa õppijatest ei toonud välja ühtegi lemmikut, vaid ütlesid, et kõik teemad või ülesanded olid huvitavad. Üllatavalt suure huviga andmetöötluse osa vastu kaasnes õppijate soov, et olemasolevaid materjale täiendatakse. Osad vastused küsimusele „Millised teemad/ülesanded/küsimused vajaksid Teie arvates muutmist?" olid muutmata kujul sellised:

- „pandas, matplotlib. viimast oleks võinud rohkem olla kaetud teemas ja ka ilma pandaseta."
- „Andmetöötlusest võiks põhjalikumalt rääkida. Väga huvitav teema, kuid projekti tehes jäi neist teadmistest puudu."
- „Andmetöötluse õppematerjale võiks veel natuke täiendada"
- „Pandase mooduli teema. Teema peatükk oli küll pikk, kuid võrreldes teistega, oleksin Pandase võimalustest soovinud rohkem teada, ja nende teadmiste varal

vastavalt ka keerulisemaid ülesandeid lahendada.”

- „Raamatukogude ül. jäi väga segaseks, ei saanud sellest kuidagi aru. Ehk peaks seda ülesannet selgemalt püstitama.”
- „Andmetöötlust võiks veel põhjalikumalt käsitleda.”
- „pandase juures võiks olla selgitatud ka kuidas saab kasutada arvutamiseks tabeli konkreetset rea/veeru väärtust ühekaupa”

Nagu eelnevatest vastustest näha, sooviti olemasolevate õppematerjalide põhjalikumaks muutmist ning koduse ülesande modifitseerimist. Õppijate arvates võiksid õppematerjalid olla abiks ka keerulisemate ülesannete või lõpuprojekti tegemisel.

Kokkuvõte

Antud bakalaureusetöö eesmärgiks oli luua õppematerjalid, mis annaksid programmeerimise kursuste õppijatele aimu Pythoni mooduli pandas võimaluste kohta andmete töötlemisel. Loodi kaks erinevat õppematerjalide komplekti, mis võeti kasutusele vastavalt magistriõppekava kursusel „Programmeerimine" ning MOOCi e-kursusel „Programmerimise alused II". Õppematerjalid valmisid käesoleva lõputöö autori ja samade kursuste teiste õppejõudude koostööl.

Magistiriõppekava kursuse jaoks mõeldud õppematerjalide komplekti kuulusid loenguslaidid, õppevideo andmete allalaadmisest ja töötlemisest, näitepõhine konspekt, kodune ülesanne ning lõpuprojekti kirjeldus. E-kursuse õppematerjalide komplektis oli mahukam ning põhjalikum konspekt, eelnevalt mainitud õppevideo, kodune ülesanne ja lõpuprojekti kirjeldus. Lisaks tegutses ka andmetöötluse teemaline foorum, milles õppijad aitasid kaasa vigade leidmisel, küsisid küsimusi ning tegid ettepanekuid õppematerjalide täiustamiseks.

Kogutud tagasiside põhjal võib öelda, et uudne andmetöötluse teema pakkus õppijatele huvi ning nad sooviksid selle kohta veelgi põhjalikumaid ning täiendatud õppematerjale. Hetkel olemasolevad õppematerjalid on mahult sobitatud õppeks mõeldud perioodi pikkusega, kuid tulevikus on võimalik täiendatud varianti kasutada ka näiteks täiesti eraldiseisva teemana kahe nädala vältel. Samuti võib soovi korral luua uue programmeerimise kursuse ainult andmetöötluse kohta käivate teemade põhjalikumaks läbimiseks.

Viidatud kirjandus

[1] Marr, B. How Big Data Is Changing Healthcare. *Forbes*. 2015
https://bit.ly/2JV9Bn1 (06.05.2018)

[2] What is R?
https://www.r-project.org/about.html (06.04.2018)

[3] The Editors of Encyclopaedia Britannica. Microsoft Excel. *Encyclopaedia Britannica*. 2018.
https://www.britannica.com/technology/Microsoft-Excel (06.05.2018)

[4] Kopf, D. Meet the man behind the most important tool in data science. *Quartz*. 2017.
https://qz.com/1126615 (20.03.2018)

[5] L. Liikane, M. Kesa. *Arvutisõnastik*.
https://bit.ly/2ro76Ck (06.05.2018)

[6] Cybernetica AS. STANDARDIPÕHINE TARKVARATEHNIKA SÕNASTIK.
https://stats.cyber.ee/term/4020-andmefreim (06.05.2018)

[7] pandas: powerful Python data analysis toolkit
https://pandas.pydata.org/pandas-docs/stable/ (14.02.2018)

[8] Statistikaameti kodulehekülg.
https://www.stat.ee/ (24.11.2017)

[9] Ezvid For Windows: The Easiest Screen Recorder And Video Editor.
https://www.ezvid.com/ezvid_ for _windows (24.11.2017)

[10] Õppevideo andmete allalaadimisest ja töötlemisest.
https://bit.ly/2FOzaDV (24.11.2017)

[11] Cheng, C. K., Dwayne, E. P., Collimore L-M., Joordens, S. Assessing the effectiveness of a voluntary online discussion forum on improving students' course performance. 2010.
https://bit.ly/2riZzWc (23.04.2018)

[12] Reponen, H.-L. FOORUMITE KASUTAMINE VABA JUURDEPÄÄSUGA E-KURSUSTEL KURSUSE „PROGRAMMEERIMISEST MAALÄHEDASELT“ NÄITEL. TÜ haridusteaduste instituudi magistritöö. 2017.
https://bit.ly/2HQzEPW (06.05.2018)

Lisad

I. Courses-keskkonnas asuvad materjalid

5.4 ANDMETÖÖTLUS MOODULIGA PANDAS

Materjal on veel toorevõitu. Palun teatage parandusettepanekutest vastavas foorumis!!!

Sissejuhatus

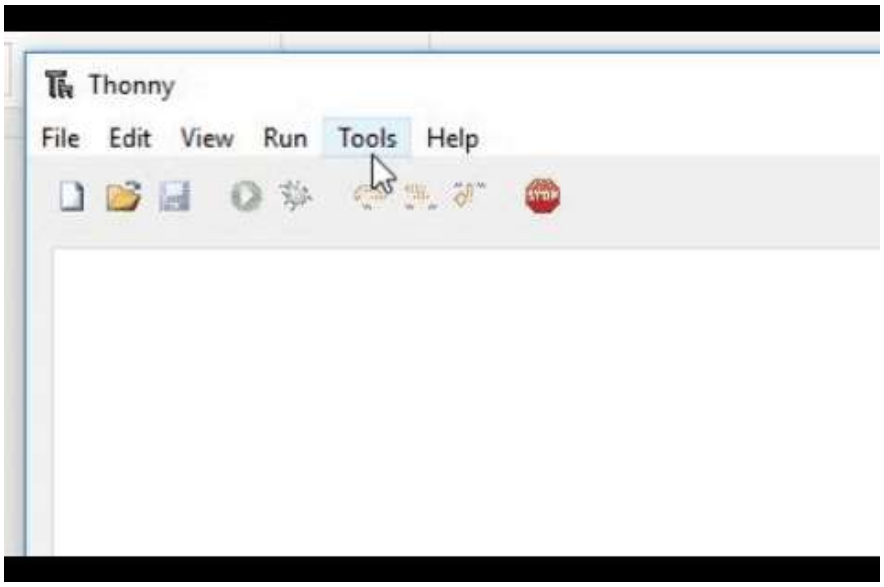
Elame ajastul, kus kogutakse hulgaliselt erinevaid andmeid. Ühelt poolt sisaldab see näiteks ohte privaatsusele, teiselt poolt aga annab võimalusi õigemaid otsuseid teha. Otsustamiseks või ka lihtsalt millestki parema ülevaate saamiseks tuleb andmeid mõistlikult töödelda. Andmetöötluseks on väga erinevaid vahendeid, millest käesoleval kursusel kasutame Pythoni moodulit Pandas. Selle nädala materjalides tutvustatakse Pandast ja vaadeldakse andmeid teatrikülastuste kohta. Ülesanne on aga raamatukogude kohta. Andmed on saadud statistikaameti veebilehelt, kus on tohutult palju erinevaid andmeid. Valmis andmeid on võimalik saada ka mitmetest muudest kohtadest. Samuti on võimalik ise andmeid koguda, näiteks pulsikella abil. Natuke etteruttavalt võib öelda, et 7. ja 8. nädalal tehtav projekt võib põhineda andmetöötlusel. Seega on need materjalid ka projekti ettevalmistuseks.

Kuna Pandase kasutamise kohta eriti palju eestikeelseid materjale pole ja pole ka väljakujunenud eestikeelset terminoloogiat, on materjalides kasutatud mõningaid otsemugandusi, nt *seeria* (Series), *andmefreim* (DataFrame). Andmetöötluse materjalid on seotud Inga Konovalova bakalaureusetööga. Tema hoiab silma peal ka vastaval foorumil ning kohendab soovituste põhjal materjale.

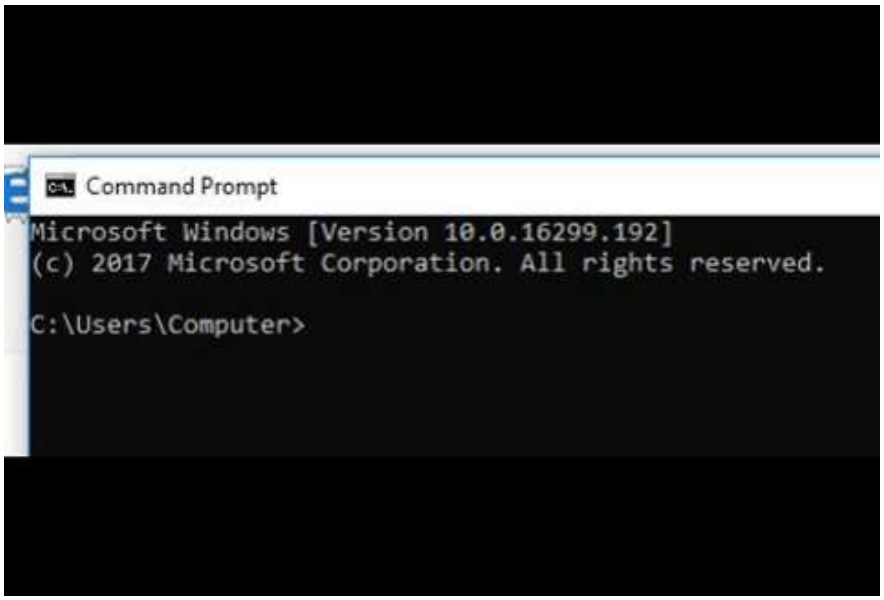
Installeerimine

Kursusel *Programmeerimise alused II* kasutavad paljud osalejad Pythonis programmide kirjutamiseks keskkonda Thonny. Vaatamegi, kuidas Thonnys installeerida Pythoni andmetöötluseks mõeldud moodul Pandas.

Pandase paigaldamiseks tuleb Thonny ülamenüüst valida Tools --> Manage packages ja otsingusse kirjutada *Pandas* ning vajutada nuppu Install.



Kui te ei kasuta Thonnyt, siis võib Pandase installeerimine olla mõnevõrra keerulisem. Windowsi käsurealt saab Pandase installeerida käsuga `py -m pip install pandas`

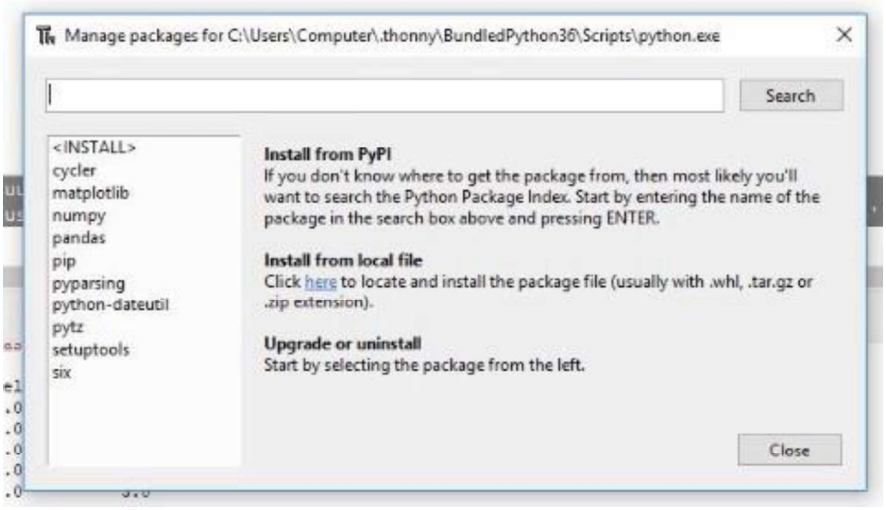


Jupyter notebookis saab Pandase installeerida reaga: `!pip install pandas`. Lisainfot installeerimise kohta saab pandase veebilehelt.

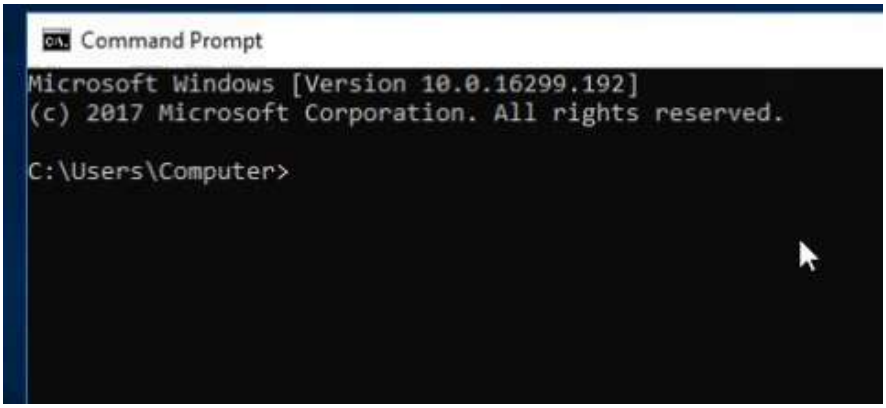
Koos Pandasega installitakse ka moodul NumPy. Tegelikult saaks pelgalt NumPy abiga andmeid töödelda, kuid Pandases on rohkem funktsioone ning andmete töötlemine on mugavam. Mooduliga NumPy saab soovi korral tutvuda siin.

Graafikute tegemiseks läheb vaja moodulit Matplotlib, mille installeerimine on sama nagu Pandase oma:

Thonnys paigaldamiseks tuleb ülamenüüst valida Tools --> Manage packages ja otsingusse kirjutada *matplotlib* ning vajutada nuppu Install.



Windowsi käsurealt saab Matplotlib installeerida käsuga py -m pip install matplotlib.



Jupyter notebookis tuleks kasutada koodirida: !pip install matplotlib.

Andmestruktuurid

Pandase kohta leidub väga vähe eestikeelseid materjale ja terminoloogia on lünklik. Kasutame siin materjalides ingliskeelseid termineid ja nende otsetõlkeid: *seeria*, *andmefreim* ja *paneel*.

Pandases on andmete hoidmiseks 3 põhilist andmestruktuuri:

- Series - seeria
- DataFrame - andmefreim
- Panel - paneel

Enimkasutatav andmestruktuur on andmefreim ning ka nendes materjalides keskendutakse sellele. Esmalt aga tutvustatakse seeriaga.

Seeria (*Series*)

Seeria (Series) on ühemõõtmeline, sarnastest andmetest koosnev, muutumatu suurusega andmestruktuur. Sellest võib mõelda ka kui kindla suurusega sõnastikust, milles igale võtmele vastab kindel väärtus.

Erinevus sõnastiku ja seeria vahel

Struktuuri poolest meenutab seeria kõige rohkem sõnastikku. Andmeid hoitakse siltidega seostatuna ja vajadusel võib kindla elemendi sildi järgi kätte saada. See meenutab sõnastiku võtme ja väärtuse süsteemi.

Sõnastiku puhul

```
isikud_sonastik = {'Võistleja nr 1': 'Mari', 'Võistleja nr
2': 'Joonas', 'Võistleja nr 3': 'Kati'}
print(isikud_sonastik)
```

saame

```
{'Võistleja nr 1': 'Mari', 'Võistleja nr 2': 'Joonas', 'Võistleja nr
3': 'Kati'}
```

Seeria puhul

```
import pandas as pd
isikud_seeria = pd.Series({'Võistleja nr 1': 'Mari', 'Võistleja nr
2': 'Joonas', 'Võistleja nr 3': 'Kati'})
print(isikud_seeria)
```

aga

```
Võistleja nr 1    Mari
Võistleja nr 2    Joonas
Võistleja nr 3    Kati
dtype: object
```

Sõnastikku tasub kasutada juhul, kui on soov andmeid võtmete ja väärtuste paaridena hoida ning pole vajadust edaspidise põhjalikuma andmete töötluse järele. Andmetöötluse jaoks sobib paremini seeria, sest sellel on rohkem funktsioone (aritmeetilised tehted, sorteerimine, filtreerimine jne). Tegevused toimuvad ka efektiivsemalt ning kood on lihtsam.

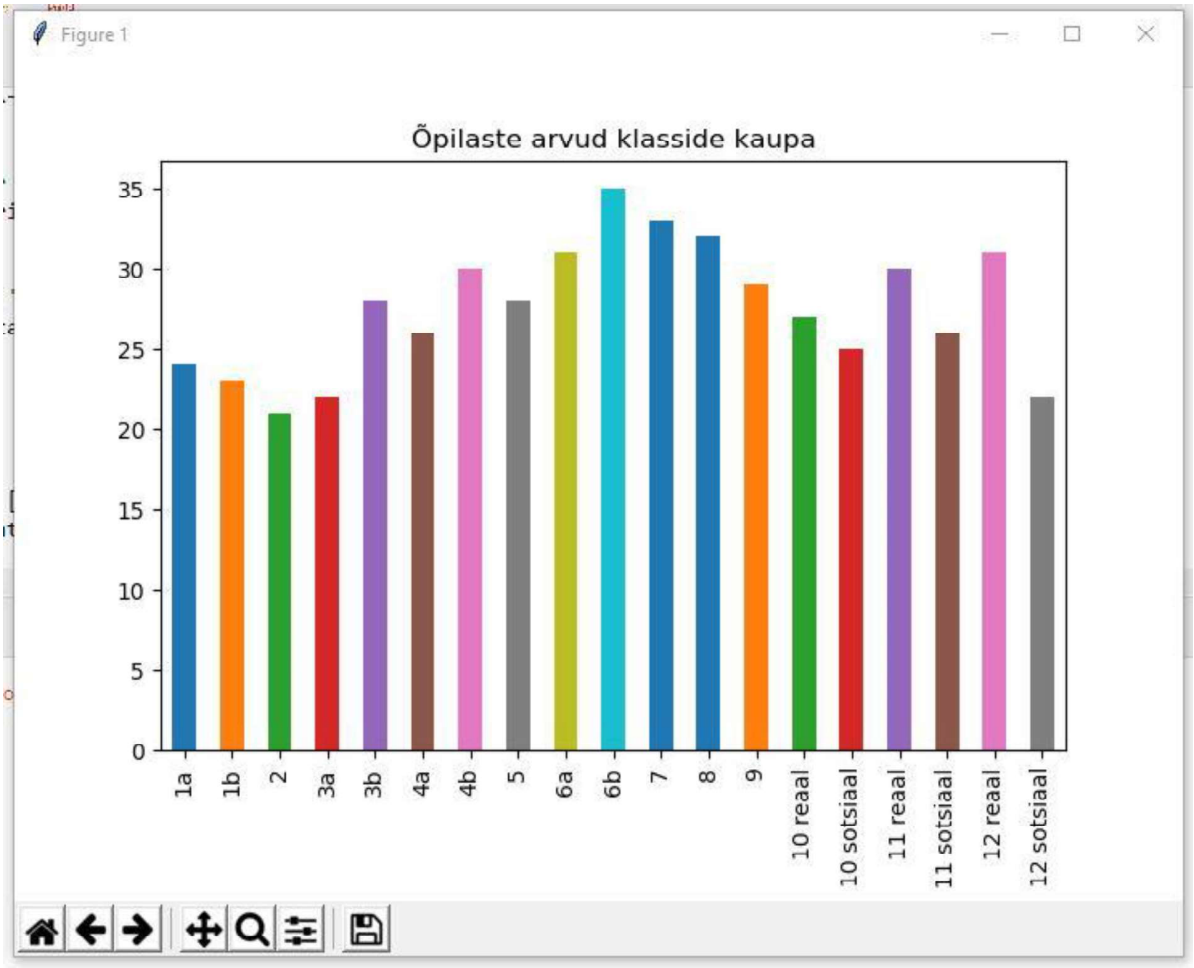
Näitena vaadeldakse seeriat, milles hoitakse õpilaste arvusid klasside kaupa.

```
import pandas as pd
import matplotlib.pyplot as plot

klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22],
index =
['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7', '8',
 '9', '10 reaal', '10 sotsiaal', '11 reaal', '11 sotsiaal', '12
reaal', '12 sotsiaal'], name = 'Õpilaste arv klassis')
print(klassid)
```

Seeria puhul loetakse veeruks vaid andmete veergu ja seal asuvaid andmeid töödeldakse. Siltide veerg on vajalik konkreetsetele andmetele viitamiseks.

Illustreerime ülaltoodud andmeid graafikul.

```
klassid.plot.bar(title = 'Õpilaste arvud klasside kaupa')
plot.show()
```



Täpsemalt saab graafikute loomise ja kujundamise kohta infot materjalide lõpuosas esitatud näitest ja näiteks ingliskeelsest materjalist.

Sarnasus järjendi ja seeria vahel

Seeria on teatud mõttes sarnane tavalise järjendiga. Järgmises näites on püütud järjendile eriti sarnane seeria tekitada.

```
import pandas as pd
järjend = ['a', 'b', 'c', 'd']
print(järjend)
print('Element indeksiga 0 järjendis on:', järjend[0])
```

Ekraanile ilmub siis

```
['a', 'b', 'c', 'd']
Element indeksiga 0 järjendis on: a
```

Anname ka seerias samad indeksid

```
seeria = pd.Series(['a', 'b', 'c', 'd'], index = [0, 1, 2, 3])
print(seeria)
print('Element sildiga 0 seerias on:', seeria[0])
```

Nüüd saame ekraanile

```
0   a
1   b
2   c
3   d
dtype: object
Element sildiga 0 seerias on: a
```

Enesetest:

Mida väljastab selline koodilõik?

```
seeria = pd.Series(['a', 'b', 'c', 'd', 'e', 'f'],
index = [1, 3, 5, 7, 9, 11])
print(seeria[5])
```

Vali c

Vali f

Vali Midagi muud

Vali Veateate, sest sellist elementi pole

Nii seeria kui järjendi puhul on konkreetsele elemendile lisatud viide, mille järgi saab vajadusel elemendi kätte. Järjendis on elementidel järjekorranumbrid ehk indeksid `0, 1, 2, ..., n`. Seeria puhul on lisaks indeksitele tegemist siltidega (ingl *label*), mille kasutaja saab ise määrata. Sildid ei pruugi olla alati järjestikused ja isegi mitte arvud. Klasside õpilaste arvu näites polnud näiteks arvud.

Uue seeria loomine

Uue Series-tüüpi andmestruktuuri saab luua järgmiselt.

```
pandas.Series(data, index, dtype, copy)
```

Kõiki parameetreid ei pea defineerimisel kasutama, mingi parameetri ärajätmisel kasutatakse selle vaikimisi määratud väärtust. Kui näiteks ära jätta `data`, kuid lisada siltide järjend, siis genereeritakse seeria, mille igale sildile vastab spetsiaalne väärtus `NaN` (*Not a Number*). Lisaks tabelis olevatele parameetritele võib kasulikuks osutuda ka `name`, mille lisamisel saab seeriale määrata nime.

Järgmine tabel kirjeldab, mis tüüpi väärtused igale parameetrile saab anda ja mida need tähendavad.

Parameeter	Selgitus
`data`	Andmed võivad olla järjendi, sõnastiku või konstandi (üks element) kujul. Näiteks `pandas.Series({'Mari': '12', 'Karmo': '15', 'Malle': '20'}, ...)` või `pandas.Series(['punane', 'kollane', 'sinine'], ...)` või `pandas.Series('a', ...)`
`index`	Siltide järjend. Vaikimisi määratakse siltideks arvud 0, 1, 2,..., n-1, kus n on elementide arv parameetriga `data` lisatud andmestruktuuris.
`dtype`	Andmete tüüp, mida seerias hoitakse. Vaikimisi tuvastatakse tüüp antud andmetest. Näiteks kui hoitakse seerias ujukomaarve, siis vaikimisi määratakse tüübiks `float`.
`copy`	Määrab, kas andmetest, mida kasutatakse, tehakse koopia. Vaikimisi väärtus `false`.

Lisaks tabelis olevatele parameetritele, võib kasulikuks osutuda ka `name`, mille lisamisel saab seeriale määrata nime.

Vaatame ülaltoodud klasside ja õpilaste näidet uuesti ning analüüsime selle seeria loomist:

```
klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22], #andmed ehk data
index =
['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7', '8',
 '9', '10 reaal', '10 sotsiaal', '11 reaal', '11 sotsiaal', '12
reaal', '12 sotsiaal'], #sildid
name = 'Õpilaste arv klassis') #seeriale antud nimi
```

Siltide arv (antud juhul klasside arv) peab olema võrdne parameetriga `data` antud andmestruktuuri elementide arvuga ehk õpilaste arvude arvuga. Antud juhul on meil tegemist 19 erineva klassiga ning järelikult ka õpilaste arvusid peab olema andmete järjendis 19. Parameetri `name` abil on seeriale lisatud nimeks “Õpilaste arv klassis”. Muudele parameetritele määrati väärtused vaikimisi.

Loodud seeria näeb ekraanile väljastatuna välja niisugune.

```
1a              24
1b              23
2               21
3a              22
3b              28
4a              26
4b              30
5               28
6a              31
6b              35
7               33
8               32
9               29
10 reaal        27
10 sotsiaal     25
11 reaal        30
11 sotsiaal     26
12 reaal        31
12 sotsiaal     22
Name: Õpilaste arv klassis, dtype: int64
```

Andmete tüübiks on vaikimisi määratud `int64`, sest õpilaste arvude näol on tegemist täisarvudega.

Enesetest:

Milline on väljundi viies rida?

```
seeria = pd.Series([1, 2, 3, 4, 5], index = [0, 2, 4,
6])
print(seeria)
```

Vali `5 NaN`

Vali `NaN 5`

Vali Viiendat rida polegi, sest programm on vigane.

Erandjuhuna saab seeria luua nii, et loomisel on lisatud vaid üks väärtus ja näiteks 4 silti. Sel juhul korratakse seda ühte väärtust 4 korda ning luuakse 4 sama väärtusega elementi.

```
import pandas as pd
seeria = pd.Series(7, index = [0, 2, 4, 6])
print(seeria)
```

Ekraanile saame

```
0   7
2   7
4   7
6   7
dtype: object
```

Sildid võivad olla arvud, sõned või mingit muud mittemuteeritavat tüüpi.

```
seeria = pd.Series([7, 'Tartu Ülikool', 3.14, -
1789710578, 'Informaatika!'], index = ['A', '2', 'C', '4', 'E'])
print(seeria)
```

Väljund

```
A    7
2    Tartu Ülikool
C    3.14
4    -1789710578
E    Informaatika!
dtype: object
```

Ülaltoodud näidetes loodi seeriad kasutades järjendit või konstanti (ühte elementi). Lisatavad andmed võivad olla ka sõnastikus.

```
import pandas as pd
dict_isikud = {'Võistleja nr 1' : 'Mari','Võistleja nr
2': 'Joonas','Võistleja nr 3': 'Kati'}
seeria = pd.Series(dict_isikud)
print(seeria)
```

Väljund

```
Võistleja nr 1    Mari
Võistleja nr 2    Joonas
Võistleja nr 3    Kati
dtype: object
```

Selleks, et tutvuda osaga võimalustest, mida seeria kasutamine andmete töötlemiseks annab, võtame ette juba tuttava klasside õpilaste arvude näite.

```
import pandas as pd

klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22],
index =
['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7', '8',
 '9', '10 reaal', '10 sotsiaal', '11 reaal', '11 sotsiaal', '12
reaal', '12 sotsiaal'], name = 'Õpilaste arv klassis')


print(klassid)

1a             24
1b             23
2              21
3a             22
3b             28
4a             26
4b             30
5              28
6a             31
6b             35
7              33
8              32
9              29
10 reaal       27
10 sotsiaal    25
11 reaal       30
11 sotsiaal    26
12 reaal       31
12 sotsiaal    22
Name: Õpilaste arv klassis, dtype: int64
```

Oletame, et meil on vaja eraldada ainult algklasside õpilaste arvud. Seda saab teha kahel moel, kasutades andmete asukohta seerias või silte. Asukoha järgi saab andmeid eraldada seetõttu, et lisaks siltidele, on igal elemendil olemas ka indeks, mis tähistab numbriliselt elemendi asukohta seerias. Nagu ka järjendis, on indeksiteks arvud `0, 1, 2, ..., n-1`, kus n tähistab elementide arvu seerias.

```
import pandas as pd

klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22],
index =
['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7', '8',
 '9', '10 reaal', '10 sotsiaal', '11 reaal', '11 sotsiaal', '12
reaal', '12 sotsiaal'], name = 'Õpilaste arv klassis')
algklassid =
klassid[['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b']]
print(algklassid)
```

Ja teistmoodi

```
import pandas as pd

klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22],
index =
['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7', '8',
 '9', '10 reaal', '10 sotsiaal', '11 reaal', '11 sotsiaal', '12
reaal', '12 sotsiaal'], name = 'Õpilaste arv klassis')
algklassid = klassid[0:10]
print(algklassid)
```

Tulemus tuleb mõlemal juhul täpselt sama.

```
1a    24
1b    23
2     21
3a    22
3b    28
4a    26
4b    30
5     28
6a    31
6b    35
Name: Õpilaste arv klassis, dtype: int64
```

Järgmisena vaatame, mitu klassi kokku on meil seerias, selleks kasutame `.size` tunnust.

```
import pandas as pd

klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22],
index =
['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7', '8',
 '9', '10 reaal', '10 sotsiaal', '11 reaal', '11 sotsiaal', '12
reaal', '12 sotsiaal'], name = 'Õpilaste arv klassis')
print(klassid.size)
```

Väljastatakse `19`.

Tihti on vaja andmeid mingis kindlas järjekorras sorteerida, sorteerime klassid õpilaste arvude järgi kahanevas järjekorras funktsiooni `.sort_values()` abil. Samuti saab sorteerida ka siltide järgi funktsiooni `.sort_index()` abil.

```
import pandas as pd

klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22],
index =
['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7', '8',
 '9', '10 reaal', '10 sotsiaal', '11 reaal', '11 sotsiaal', '12
reaal', '12 sotsiaal'], name = 'Õpilaste arv klassis')
klassid_kahanevalt = klassid.sort_values(ascending=False)
print(klassid_kahanevalt)
6b             35
7              33
8              32
12 reaal       31
6a             31
11 reaal       30
4b             30
9              29
3b             28
5              28
10 reaal       27
11 sotsiaal    26
4a             26
10 sotsiaal    25
1a             24
1b             23
3a             22
12 sotsiaal    22
2              21
Name: Õpilaste arv klassis, dtype: int64
```

Mõnikord on vaja lisada seeriasse andmeid juurde. Vaatame, kuidas saab kahe seeria andmed ühendada. Lisame juurde õpetajate arvu, õpilaste arvu kokku ja klasside arvu kokku. Selleks loome uue `Series`-tüüpi andmestruktuuri, mille andmeteks on järjend õpetajate arvuga, õpilaste arvuga kokku ja klasside arvuga kokku ning mille siltideks on 'Õpetajate arv', 'Õpilaste arv kokku' ja 'Klasside arv kokku'. Õpilaste arvu kokku saame kasutades `.sum()` funktsiooni klasside seeria peal ning klasside arvu kokku leiame atribuudiga `.size`, mida kasutasime ka varem.

```
import pandas as pd

klassid =
pd.Series([24, 23, 21, 22, 28, 26, 30, 28, 31, 35, 33, 32, 29, 27, 2
5, 30, 26, 31, 22],
index=['1a', '1b', '2', '3a', '3b', '4a', '4b', '5', '6a', '6b', '7'
, '8', '9', '10 reaal', '10 sotsiaal', '11 reaal', '11
sotsiaal', '12 reaal', '12 sotsiaal'], name='Õpilaste arv klassis')
uus_seeria = pd.Series([25, klassid.sum(), klassid.size],
index=['Õpetajate arv', 'Õpilaste arv kokku','Klasside arv kokku'])
klassid = klassid.append(uus_seeria)
print(klassid)
```

Tulemuseks on uuendatud klasside seeria. Oluline on meelde jätta, et funktsioonide rakendamisel seeria peal, tuleb uus ja modifitseeritud muutuja alati uuesti omistada, kas siis uuele muutujale või kirjutada vana seeria üle.

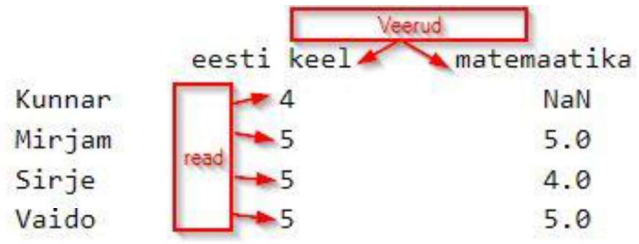
```
1a                    24
1b                    23
2                     21
3a                    22
3b                    28
4a                    26
4b                    30
5                     28
6a                    31
6b                    35
7                     33
8                     32
9                     29
10 reaal              27
10 sotsiaal           25
11 reaal              30
11 sotsiaal           26
12 reaal              31
12 sotsiaal           22
Õpetajate arv         25
Õpilaste arv kokku   523
Klasside arv kokku    19
dtype: int64
```

Andmefreim (`DataFrame`)

Juba seeria abil saab andmeid teatud määral töödelda ja illustreerida. Veelgi põnevamaid võimalusi pakub andmefreim (`DataFrame`), mis kujutab endast põhimõtteliselt tabelit. Seeria tähistab andmefreimis (`DataFrame`) ühte veergu. Andmefreimis võivad olla erinevat tüüpi andmed ja selle suurust saab muuta. Ridadele ja veergudele saab rakendada aritmeetilisi funktsioone - näiteks saab veergusid liita, lahutada, korrutada ja jagada ning arvutada rea või veeru kaupa selle maksimumi, keskmist jpm.

Esimese näitena vaatame andmefreimi, milles on õpilaste nimed ning eesti keele ja matemaatika hinded.

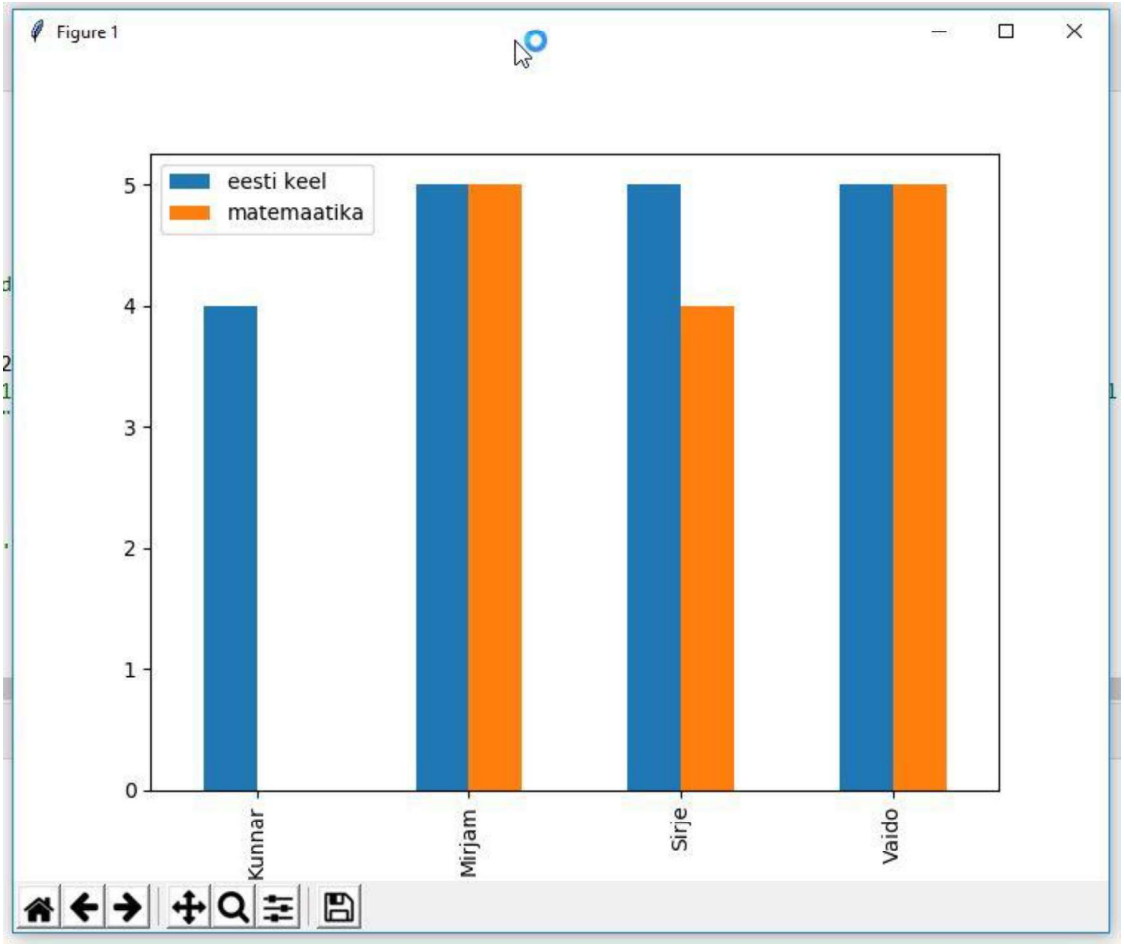
```
import pandas as pd
import matplotlib.pyplot as plot
opilased = { 'matemaatika': pd.Series([5, 4, 5], index =
['Mirjam', 'Sirje', 'Vaido']),
'eesti keel': pd.Series([5, 5, 5, 4], index =
['Mirjam', 'Sirje', 'Vaido', 'Kunnar'])}
opilased_tabel = pd.DataFrame(opilased)
print(opilased_tabel)
```



Andmefreimis ei loeta siltide veergu ja veergude pealkirjasid ridade ja veergude hulka. Erinevalt seeriast, vastab siin igale sildile üks rida. See koosneb põhimõtteliselt pealkirjaga seeriatest (veergudest), millel on ühised sildid.

Illustreerime ülaltoodud andmeid graafikul:

```
opilased_tabel.plot.bar()
plot.show()
```



Andmefreimi saab luua järgmiselt:

```
pandas.DataFrame(data, index, columns, dtype, copy)
```

Ka andmefreimi puhul ei pea kõiki parameetreid defineerimisel kasutama, parameetri ärajätmisel kasutatakse selle vaikimisi määratud väärtust.

Parameeter	Selgitus
data	Andmed võivad olla n-dimensioonilise andmemassiivi, sõnastiku, konstandi, järjendi või teise andmefreimi kujul. Näiteks: pandas.DataFrame([['Ruudi', 12], ['Malle', 10]], ...)
index	Ridade siltide järjend, pikkus peab olema võrdne ridade arvuga. Vaikimisi määratakse arvud 0, 1, 2, ..., n-1, kus n on ridade arv.
columns	Veergude siltide järjend, pikkus peab olema võrdne veergude arvuga. Vaikimisi määratakse arvud 0, 1, 2, ..., m-1, kus m on veergude arv.

dtype	Andmetüüp, mida DataFrames olevatele andmetele määratakse. Vaikimisi järeldatakse andmetest.
copy	Määrab, kas andmetest, mida kasutatakse, tehakse koopia. Vaikimisi väärtus False.

Vaatame ülaltoodud klasside ja õpilaste näidet uuesti ning analüüsime selle seeria loomist:

```
import pandas as pd
opilased = { 'matemaatika': pd.Series([5, 4, 5], index =
['Mirjam', 'Sirje', 'Vaido']),
'eesti keel': pd.Series([5, 5, 5, 4], index =
['Mirjam', 'Sirje', 'Vaido', 'Kunnar'])}
opilased_tabel = pd.DataFrame(opilased)
print(opilased_tabel)
```

Sisendiks antakse sõnastik, mille võtmed on veeru pealkirjad ja vastavateks väärtusteks kaks seeriat (kaks veergu). Seeriate siltideks on õpilaste nimed ning andmeteks õpilaste poolt saadud hinded. Pange tähele, et mõlemas seerias on õpilaste arv võrdne hinnete arvuga, kuid lisatavad seeriad pole sama pikad. Muudele parameetritele määrati väärtused vaikimisi.

Rohkem näiteid andmefreimide loomisest:

Tühja andmefreimi loomine:

```
import pandas as pd
tühi = pd.DataFrame()
print(tühi)
```

Väljund:

```
Empty DataFrame
Columns: []
Index: []
```

Andmefreimi loomine järjendi põhjal:

Tegemist on järjendite järjendiga, mis antakse ette andmefreimi andmetena. Columns parameetriks antud järjend määrab ära veergude pealkirjad. Andmed jaotatakse veergudesse vastavalt nende paigutusele järjendites, iga järjend on eraldi rida ja sisemiste järjendite elemendid asuvad erinevates veergudes. Määratud on ka andmefreimis arvuna antud vanuste andmetüüp, selleks on float.

```
import pandas as pd
isikud = [['Juss',10],['Illar',12],['Mari-Liis',13]]
andmestruktuur = pd.DataFrame(isikud, columns = ['Nimi','Vanus'],
dtype=float)
print(andmestruktuur)
```

```
     Nimi         Vanus
0    Juss         10.0
1    Illar        12.0
2    Mari-Liis    13.0
dype: object
```

Andmefreimi loomine sõnastiku põhjal:

Sarnaselt ülaltoodud näitele on siingi sõnastiku võtmed veergude pealkirjadeks. Konkreetsetele võtmetele vastavad järjendid on aga veergudes väärtusteks. Andmefreimi loomisel on etteantud ka siltide järjend.

```
import pandas as pd
voistlejad = {'Nimi' :
['Malle', 'Kalle', 'Jaanus', 'Pille'], 'Vanus' : [24, 34, 51, 47]}
voistlejad_tabel = pd.DataFrame(voistlejad, index = ['Nr. 1', 'Nr.
2', 'Nr. 3', 'Nr. 4'])
print(voistlejad_tabel)
```

Väljund:

```
         Nimi    Vanus
Nr. 1    Malle     24
Nr. 2    Kalle     34
Nr. 3    Jaanus    51
Nr. 4    Pille     47
```

Andmefreimi (DataFrame) loomine sõnastikust, milles on seeriad:

Ülaltoodud näide õpilastest ja hinnetest sobib hästi näiteks, kus andmefreim luuakse sõnastikust, milles on seeriad.

```
import pandas as pd
opilased = { 'matemaatika' : pd.Series([5, 4, 5],
index=['Mirjam', 'Sirje', 'Vaido']),
'eesti keel' : pd.Series([5, 5, 5, 4], index =
['Mirjam', 'Sirje', 'Vaido', 'Kunnar'])}
opilased_tabel = pd.DataFrame(opilased)
print(opilased_tabel)
```

Väljund:

```
        eesti keel     matemaatika
Kunnar           4           NaN
Mirjam           5           5.0
Sirje            5           4.0
Vaido            5           5.0
```

Matemaatika veerus oli andmeid vaid 3 inimese kohta, seega lisati Kunnari matemaatika hindeks `NaN` ehk siis "pole arv" väärtus (ingl *not a number*). Sellest tulenevalt muudeti selle veeru väärtused ujukomaarvudeks (ingl *float*), sest `NaN` väärtust loetakse ujukomaarvuks.

Edasi tutvume mõnede andmefreimi võimalustega ja selleks on jällegi hea kasutada õpilaste ja hinnete näidet.

```
import pandas as pd
opilased = { 'matemaatika': pd.Series([5, 4, 5], index =
['Mirjam', 'Sirje', 'Vaido']),
'eesti keel': pd.Series([5, 5, 5, 4], index =
['Mirjam', 'Sirje', 'Vaido', 'Kunnar'])}
opilased_tabel = pd.DataFrame(opilased)
print(opilased_tabel)
```

Oletame, et pandi välja veel mitme aine hinded ning vaja oleks lisada ka need tabelisse.

```
import pandas as pd
opilased = { 'matemaatika': pd.Series([5, 4, 5], index =
['Mirjam', 'Sirje', 'Vaido']),
'eesti keel': pd.Series([5, 5, 5, 4], index =
['Mirjam', 'Sirje', 'Vaido', 'Kunnar'])}
opilased_tabel = pd.DataFrame(opilased)
print(opilased_tabel)
```

Algse tabeli kuju:

```
            eesti keel  matemaatika
Kunnar           4           NaN
Mirjam           5           5.0
Sirje            5           4.0
Vaido            5           5.0
```

Selleks loome sama põhimõtte järgi uue andmefreimi ning ühendame `.join()` funktsiooni abil kaks andmefreimi omavahel.

```
import pandas as pd
lisa = { 'inglise keel' : pd.Series([5, 4, 5, 5, 3], index =
['Mirjam', 'Sirje', 'Vaido','Karmen','Kunnar']),
             'kehaline kasvatus' : pd.Series([5, 5, 5, 4, 5],
index = ['Mirjam', 'Sirje', 'Vaido', 'Karmen', 'Kunnar'])}

lisa_tabel = pd.DataFrame(lisa)
```

Uute andmetega tabeli kuju:

```
             inglise keel  kehaline kasvatus
Mirjam              5                   5
Sirje               4                   5
Vaido               5                   5
Karmen              5                   4
Kunnar              3                   5
```

Näeme, et lisatavas tabelis on rohkem õpilasi, seega peame ühendama algse õpilaste tabeli uue tabeliga, et kõik nimed oleksid olemas ka uues tabelis.

```
import pandas as pd
opilased = { 'matemaatika': pd.Series([5, 4, 5], index =
['Mirjam', 'Sirje', 'Vaido']),
'eesti keel': pd.Series([5, 5, 5, 4], index =
['Mirjam', 'Sirje', 'Vaido', 'Kunnar'])}
opilased_tabel = pd.DataFrame(opilased)

lisa = { 'inglise keel' : pd.Series([5, 4, 5, 5, 3], index =
['Mirjam', 'Sirje', 'Vaido','Karmen','Kunnar']),
             'kehaline kasvatus' : pd.Series([5, 5, 5, 4, 5],
index = ['Mirjam', 'Sirje', 'Vaido', 'Karmen', 'Kunnar'])}
lisa_tabel = pd.DataFrame(lisa)

uus_tabel = lisa_tabel.join(opilased_tabel)
print(uus_tabel)
```

Uue tabeli kuju:

```
             inglise keel  kehaline kasvatus  eesti keel  matemaatika
Mirjam              5                   5           5.0          5.0
Sirje               4                   5           5.0          4.0
Vaido               5                   5           5.0          5.0
Karmen              5                   4           NaN          NaN
Kunnar              3                   5           4.0          NaN
```

Lähenemas on õppeaasta lõpp ja lisanduvad ka viimased üksikud hinded, mis tuleb tabelisse lisada. Lisame hinded ühekaupa õigele kohale. Esimese parameetriga määrame aine (veeru), mille hinnet soovime muuta ning teise parameetriga õpilase nime (rea):

```
uus_tabel['matemaatika']['Karmen'] = 3
uus_tabel['eesti keel']['Karmen'] = 5
uus_tabel['matemaatika']['Kunnar'] = 3
```

Kuna keskmisest hindest oleneb ekskursioonile minek, soovime välja arvutada keskmised hinded ja lisada need uude veergu "Keskmine hinne". Lisame uue veeru ja arvutame õpilaste keskmised hinded. Õige rea valimiseks kasutame tunnust `.loc` ning keskmise arvutamiseks reas funktsiooni `.mean()`:

```
uus_tabel['Keskmine hinne'] =
pd.Series([uus_tabel.loc['Sirje'].mean(),
uus_tabel.loc['Mirjam'].mean(),
                                        uus_tabel.loc['Vaido'].mean(
), uus_tabel.loc['Karmen'].mean(), uus_tabel.loc['Kunnar'].mean()],
index = ['Sirje', 'Mirjam', 'Vaido', 'Karmen', 'Kunnar'])
```

Pane tähele, et uude veergu lisatavad keskmised peavad olema siltide järjendis olevate siltidega samas järjekorras, siis saab õige hinne lisatud õige õpilase juurde.

Tabel, milles on puuduvad hinded ja keskmised:

```
        inglise keel  kehaline kasvatus  eesti keel  matemaatika  \
Mirjam             5                  5         5.0          5.0
Sirje              4                  5         5.0          4.0
Vaido              5                  5         5.0          5.0
Karmen             5                  4         5.0          3.0
Kunnar             3                  5         4.0          3.0
```

Järgmises peatükis leiate veelgi näiteid andmetöötluse jaoks kasutatavatest andmefreimi funktsioonidest.

5.5 TEATRIKÜLASTUSTE NÄIDE

Materjal on veel toorevõitu. Palun teatage parandusettepanekutest vastavas foorumis!!!

Sissejuhatus

Eelmises peatükis tutvustati Pandase põhilisi andmestruktuure - seeriat (`Series`) ja andmefreimi (`DataFrame`). Nüüd tegutseme andmefreimi abil põhjalikumalt tegelike andmetega. Kui siin on vaatluse all teatrikülastused, siis ülesanne tuleb raamatukogude kohta.

Senini oleme vaadanud olukordi, kus me defineerime mõnes teises andmestruktuuris asuvate algandmete järgi andmefreimi (`DataFrame`). Selleks, et säästa end andmete ümberkirjutamisest, saab andmeid ka otse failist andmefreimi (`DataFrame`) laadida.

Siin näete lühikest videot sellest, kuidas andmeid statistikaametist alla laadida ning neid töödelda.

Andmete saamine failist

Kasutades andmetöötluseks Pythoni moodulit Pandas, saab andmed lihtsalt sisse lugeda näiteks csv-failist. Võtame näiteks faili KU086.csv, mille sisu näeb tavalise tekstiredaktoriga (nt notepad) avades välja selline:

```
 ;2010;2011;2012;2013;2014;2015;2016
Teatrite arv;29;34;41;41;37;49;46
Lavastused;417;464;487;490;511;550;540
..uuslavastused;173;190;203;186;196;216;196
Etendused;4593;5012;5678;5803;6010;6434;6573
Vaatajad, tuhat;899.9;1008.3;1143.0;1090.7;1047.1;1146.6;1186.0
Teatrisk?igud 1000 elaniku
kohta;671.5;752.5;864.1;827.5;796.6;872.2;901.4
```

Ärgem laskem end hetkel segada moondunud ä-tähest. Hiljem tegeleme ka sellega.

```
import pandas as pd
andmed = pd.read_csv('KU086s.csv', delimiter=';')
print(andmed)
```

Muutuja `delimiter` väärtus näitab, milline eraldaja on andmeid sisaldavas failis määratud, antud juhul on tegemist semikooloniga. Seda, missugust eraldajat kasutatakse, saab näiteks statistikaameti andmebaasis valida andmete salvestamisel, kuid vajadusel võib andmefaili teisele kujule ka ümber teisendada.

Funktsiooni `read_csv` kasutamisel on veel mitmeid muid parameetreid, mida võib vaja minna. Nendega saab lähemalt tutvuda siin.

Saame väljundiks:

```
                                                  2012    2013    2014  \
0                                       Raamatukogud   559.0   556.0   549.0
1                                     Lugejad, tuhat   383.2   377.7   368.1
2                Lugejaid keskmiselt raamatukogu kohta   686.0   679.0   670.0
3  Laenutusi keskmiselt lugeja kohta, arvestusuksust    30.5    29.6    29.1
4                                          Tootajad  1554.0  1544.0  1528.0

     2015    2016
0   540.0   536.0
1   363.4   351.3
2   673.0   655.0
3    28.6    28.5
4  1471.0  1442.0
```

Tutvumine andmetega

Püüame täpsemalt tutvuda meie poolt sisse loetud tabeliga. Kuna sisseloetud tabel on meil nüüd `DataFrame`-kujul, siis saame kasutada pakutavaid Pandase võimalusi. Püüame näiteks teada saada, mitu veergu ja rida on tabelis.

Tunnus `DataFrame.shape` annab andmefreimi mõõtmed:

```
print(andmed.shape)

(6, 8)
```

```
# Nii saab teada, mitu veergu meil tabelis on.
# Veergude arv asub tunnuse shape väljundi teisel positsioonil
print("Tabelis on ", andmed.shape[1], " veergu.")

Tabelis on  8  veergu.

----------------------------------

# Veergude pealkirjad
print(andmed.columns)

Index([' ', '2010', '2011', '2012', '2013', '2014', '2015', '2016'],
dtype='object')

----------------------------------

# Nii saab teada, mitu rida meil tabelis on:
# Ridade arv asub tunnuse shape väljundi esimesel positsioonil
print("Tabelis on ", andmed.shape[0], " rida.")

Tabelis on 6 rida.

----------------------------------
```

```
# Prindime 2 esimest rida
print(andmed.head(2))

                  2010   2011   2012   2013   2014   2015   2016
0  Teatrite arv   29.0   34.0   41.0   41.0   37.0   49.0   46.0
1    Lavastused  417.0  464.0  487.0  490.0  511.0  550.0  540.0

---------------------------------

# Prindime 2 viimast rida
print(andmed.tail(2))

                                    2010    2011    2012    2013    2014  \
4                   Vaatajad, tuhat  899.9  1008.3  1143.0  1090.7  1047.1
5  Teatrisk�igud 1000 elaniku kohta   671.5   752.5   864.1   827.5   796.6

     2015    2016
4  1146.6  1186.0
5   872.2   901.4
```

Andmete töötlemise funktsioonid

Näeme, et hetkel on eraldi veerud siltidega ja veerg, mille väärtused võiksid tegelikult siltidena kasutuses olla.

Muudame soovitud veeru siltide veeruks. Veergude pealkirjade järjendist näeme, et see veerg on pealkirjaga " ".

```
andmed = andmed.set_index(' ')

# Prindime 5 esimest tabeli rida, et kontrollida.
print(andmed.head())

                    2010    2011    2012    2013    2014    2015    2016

Teatrite arv        29.0    34.0    41.0    41.0    37.0    49.0    46.0
Lavastused         417.0   464.0   487.0   490.0   511.0   550.0   540.0
..uuslavastused    173.0   190.0   203.0   186.0   196.0   216.0   196.0
Etendused         4593.0  5012.0  5678.0  5803.0  6010.0  6434.0  6573.0
Vaatajad, tuhat    899.9  1008.3  1143.0  1090.7  1047.1  1146.6  1186.0
```

Sama tulemuse saaksime ka andmete sisselugemisel parameetrit `index_col` kasutades.

```
andmed = pd.read_csv('KU086s.csv', delimiter=';', index_col=' ')
```

Vahel on vaja lisada andmeid tabelisse juurde. Seda saab teha veergude või ridade lisamise abil. Lisame veeru, kus on arvud 1-6.

```
andmed['Uus veerg'] = [1, 2, 3, 4, 5, 6]
print(andmed)
```

```
                                  2010    2011    2012    2013    2014  \

Teatrite arv                      29.0    34.0    41.0    41.0    37.0
Lavastused                       417.0   464.0   487.0   490.0   511.0
..uuslavastused                  173.0   190.0   203.0   186.0   196.0
Etendused                       4593.0  5012.0  5678.0  5803.0  6010.0
Vaatajad, tuhat                  899.9  1008.3  1143.0  1090.7  1047.1
Teatrisk�igud 1000 elaniku kohta   671.5   752.5   864.1   827.5   796.6

                                  2015    2016  Uus veerg

Teatrite arv                      49.0    46.0          1
Lavastused                       550.0   540.0          2
..uuslavastused                  216.0   196.0          3
Etendused                       6434.0  6573.0          4
Vaatajad, tuhat                 1146.6  1186.0          5
Teatrisk�igud 1000 elaniku kohta   872.2   901.4          6
```

Kui on vaja lisada väärtused vastavatesse kindla sildiga ridadesse, saab seda teha uue seeria (Series) loomise ning selle tabelisse lisamise abil.

```
andmed['Uus veerg'] = pd.Series([1,2,3,4,5,6], index =
['Lavastused', 'Teatrite arv', '..uuslavastused', 'Etendused', 'Vaatajad,
tuhat', 'Teatrisk�igud 1000 elaniku kohta'])
print(andmed)
```

```
                                  2010    2011    2012    2013    2014  \

Teatrite arv                      29.0    34.0    41.0    41.0    37.0
Lavastused                       417.0   464.0   487.0   490.0   511.0
..uuslavastused                  173.0   190.0   203.0   186.0   196.0
Etendused                       4593.0  5012.0  5678.0  5803.0  6010.0
Vaatajad, tuhat                  899.9  1008.3  1143.0  1090.7  1047.1
Teatrisk�igud 1000 elaniku kohta   671.5   752.5   864.1   827.5   796.6

                                  2015    2016  Uus veerg

Teatrite arv                      49.0    46.0          2
Lavastused                       550.0   540.0          1
..uuslavastused                  216.0   196.0          3
Etendused                       6434.0  6573.0          4
Vaatajad, tuhat                 1146.6  1186.0          5
Teatrisk�igud 1000 elaniku kohta   872.2   901.4          6
```

Nüüd aga kustutame loodud veeru. Kasutame selleks 3 erinevat funktsiooni: drop(), pop() ja del.

pop()

```
# pop() funktsioon tagastab eemaldatud veeru, eemaldab veeru
tabelist, ei vaja uuesti omistamist

eemaldatud = andmed.pop('Uus tulp')
print(eemaldatud)
```

```
Teatrite arv                          2
Lavastused                            1
..uuslavastused                       3
Etendused                             4
Vaatajad, tuhat                       5
Teatrisk�igud 1000 elaniku kohta      6
Name: Uus tulp, dtype: int64
```

drop()

```
# drop() funktsioon tagastab uuendatud tabeli, milles on märgitud veerg
eemaldatud, vajab uuesti omistamist, sest protseduuri tehakse tabeli koopia
peal, veeru eemaldamiseks tuleb kasutada parameetrit axis=1

andmed = andmed.drop(['Uus tulp'], axis=1)
print(andmed)
```

```
                                  2010    2011    2012    2013    2014  \

Teatrite arv                      29.0    34.0    41.0    41.0    37.0
Lavastused                       417.0   464.0   487.0   490.0   511.0
..uuslavastused                  173.0   190.0   203.0   186.0   196.0
Etendused                       4593.0  5012.0  5678.0  5803.0  6010.0
Vaatajad, tuhat                  899.9  1008.3  1143.0  1090.7  1047.1
Teatrisk�igud 1000 elaniku kohta  671.5   752.5   864.1   827.5   796.6

                                  2015    2016
Teatrite arv                      49.0    46.0
Lavastused                       550.0   540.0
..uuslavastused                  216.0   196.0
Etendused                       6434.0  6573.0
Vaatajad, tuhat                 1146.6  1186.0
Teatrisk�igud 1000 elaniku kohta  872.2   901.4
```

Proovime tabelist ka rea eemaldada. Teeme seda algul indeksipõhiselt ja seejärel eemaldame rea, mis vastab mingile määratud kriteeriumile.

drop()

```
andmed = andmed.drop(['Teatrite arv'], axis=0)
print(andmed)

2010    2011    2012    2013    2014  \

Lavastused                          417.0   464.0   487.0   490.0   511.0
..uuslavastused                     173.0   190.0   203.0   186.0   196.0
Etendused                          4593.0  5012.0  5678.0  5803.0  6010.0
Vaatajad, tuhat                     899.9  1008.3  1143.0  1090.7  1047.1
Teatrisk?igud 1000 elaniku kohta    671.5   752.5   864.1   827.5   796.6

                                    2015    2016

Lavastused                          550.0   540.0
..uuslavastused                     216.0   196.0
Etendused                          6434.0  6573.0
Vaatajad, tuhat                    1146.6  1186.0
Teatrisk?igud 1000 elaniku kohta    872.2   901.4
```

ix()

Eemaldame read, kus on 2012. aasta väärtused väiksemad või võrdsed arvust 1000.

```
andmed = andmed.ix[df['2012'] >= 1000]
print(andmed)

                   2010    2011    2012    2013    2014    2015    2016

Etendused        4593.0  5012.0  5678.0  5803.0  6010.0  6434.0  6573.0
Vaatajad, tuhat   899.9  1008.3  1143.0  1090.7  1047.1  1146.6  1186.0
```

Muudame indeksi `Teatrisk?igud 1000 elaniku kohta` nimetust, et imeliku ä-tähe asendaja asemel päris ä-täht saada.

```
andmed = andmed.rename(index={'Teatrisk?igud 1000 elaniku kohta' : 'Teatriskäigud
1000 elaniku kohta'})
print(andmed.tail(1))
                                2010   2011   2012   2013   2014   2015   2016

Teatriskäigud 1000 elaniku kohta  671.5  752.5  864.1  827.5  796.6  872.2  901.4
```

Andmete kajastamine graafikul

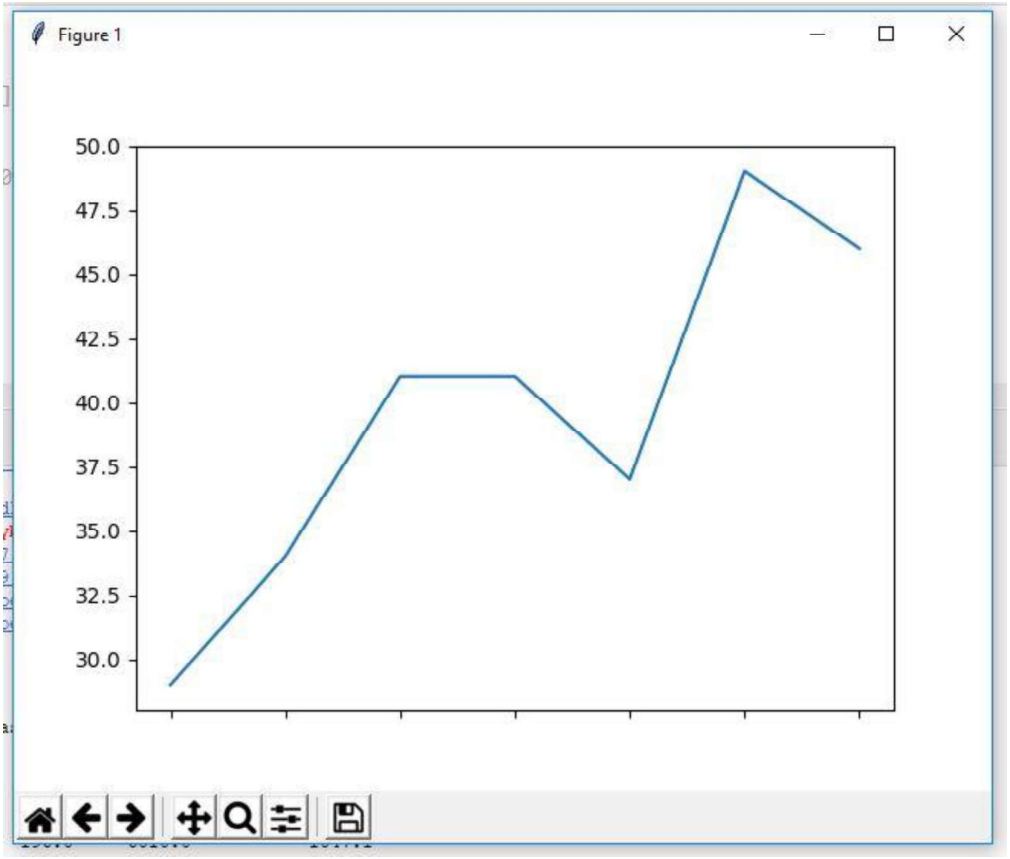
Andmete põhjal saab graafiku teha mooduli matplotlib abil. Moodul tuleb installide enne importimist. Sellega tegelesime eelmises peatükis.

```
# Impordime mooduli ja tekitame oma andmete kohta graafiku.
import matplotlib.pyplot as plot

# Transponeerime andmed, muudame veergude pealkirjad indeksiteks ja
indeksid veergude pealkirjadeks.
# Näitame graafikul, kuidas muutus aastate jooksul teatrite arv
transponeeritud_andmed = andmed.T
transponeeritud_andmed['Teatrite arv'].plot()
plot.show()
```

Transponeeritud andmed:

```
                0           1                2           3                4  \
     Teatrite arv  Lavastused  ..uuslavastused  Etendused  Vaatajad, tuhat
2010            29         417              173        4593            899.9
2011            34         464              190        5012           1008.3
2012            41         487              203        5678             1143
2013            41         490              186        5803           1090.7
2014            37         511              196        6010           1047.1
2015            49         550              216        6434           1146.6
2016            46         540              196        6573             1186

                                    5
     Teatrisk?igud 1000 elaniku kohta
2010                            671.5
2011                            752.5
2012                            864.1
2013                            827.5
2014                            796.6
2015                            872.2
2016                            901.4
```

Graafik:



Andmete kirjutamine faili

Kui andmed on töödeldud, võib tekkida vajadus need uude faili kirjutada. Seda saab teha funktsiooni `.to_csv` abil. Kirjutame muudetud andmed faili nimega uued_andmed.csv, kasutame eraldajana semikoolonit.

```
andmed.to_csv('uued_andmed.csv', sep=';', encoding='utf-8')
```

II. Loenguslaidid

Andmetöötlus Pythonis

Moodul Pandas

Andmetöötlus

- Digitaalsete andmete maht on aastatega hüppeliselt kasvanud
  - kuidas neid andmeid hästi töödelda
- Näiteks masinõppeks
  - algoritmide testimiseks ja väljatöötamiseks ning hiljem nende põhjal ennustamiseks
- Andmetöötluseks saab samuti kasutada
  - programmeerimiskeeli R, MATLAB, Java, Julia ja Scala
  - https://medium.freecodecamp.org/which-languages-should-you-learn-for-data-science-e806ba55a81f
- Meid huvitab Python

Moodul Pandas

- Moodul Pandas on üles ehitatud põhiliselt 2 teisele moodulile: NumPy ja SciPy.
- NumPy on moodul andmemassiivide haldamiseks ja töötlemiseks.
- SciPy on moodul teaduslike ja tehniliste arvutuste tegemiseks.
- Andmete esitamiseks visuaalselt võib kasutada Matplotlib moodulit.

Veel mooduleid
https://medium.com/activewizards-machine-learning-company/top-15-python-libraries-for-data-science-in-in-2017-ab61b4f9b4a7

Moodulite importimine

`import numpy` - mooduli funktsioonide rakendamiseks tuleb kasutada kuju `numpy.*`, nt `numpy.array([1,2,3])`
`import numpy as np` - funktsioonide rakendamiseks tuleb kasutada kuju `np.*`, nt `np.array([1,2,3])`
`from numpy import *` - kõiki funktsioone saab vabalt kasutada, nt `array([1,2,3])`
`from numpy import array, vectorize, dot` - võib funktsioone ka eraldi importida

Analoogiliselt SciPy jt moodilid

Põhilised andmestruktuurid

Series

- Ühemõõtmeline, sarnastest andmetest koosnev, muutmatu suurusega seeria.
- Võib mõelda sellest ka kui kindla suurusega sõnastikust, milles võtmele vastab väärtus.


```
import pandas as pd
järjend = ['a','b','c','d']
print(järjend)

['a','b','c','d']

seeria = pd.Series(järjend, index=[2,1,3,0])
print(seeria)

2    a
1    b
3    c
0    d
dtype: object
```



```
print('Element indeksiga 0 listis on:',
 list[0])
```

Väljund: `Element indeksiga 0 listis on: a`

```
print('Element indeksiga 0 seerias on:',
 jada[0])
```

Väljund: `Element indeksiga 0 seerias on: d`

Indeksid saavad olla ükskõik, millised:

```
s = pd.Series([7, 'Tartu Ülikool', 3.14,
 -1789710578, 'Informaatika!'],index=['A', '2',
 'C', '4', 'E'])
```

Väljund:

```
A                  7
2     Tartu Ülikool
C              3.14
4       -1789710578
E     Informaatika!
dtype: object
```

Tüübiks object, sest nii string, integer kui ka float kuuluvad kõrgema klassi object alla.

Näide

```
import numpy as np
import pandas as pd
andmed = np.array([1,2,3,4,5])
s = pd.Series(andmed,index=['a','b','c','d','e'])
```

Prindime välja: `print(s)`

```
a    1
b    2
c    3
d    4
e    5
dtype: int32
```

Võib mõelda kui võtmetest sõnastikus, kus igale võtmele vastab mingi väärtus.

Andmete tüüp, mida hoitakse seerias.

Mõningad tunnused

```
import numpy as np
import pandas as pd
andmed = np.array([1,2,3,4,5])
s = pd.Series(andmed,index=['a','b','c','d','e'])
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int32
```

Andmete saamine asukoha järgi:

`print(s[0])` → `1`

`print(s[2:4])` →

```
c    3
d    4
dtype: int32
```

`print(s['b'])` → `2`

`print(s[['b','d','c']])` →

```
b    2
d    4
c    3
dtype: int32
```



```
import numpy as np
import pandas as pd
andmed = np.array([1,2,3,4,5])
s = pd.Series(andmed,index=['a','b','c','d','e'])
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int32
```

Indeksite/võtmete saamine: `print(s.axes)`

```
[Index(['a','b','c','d','e'],dtype='object')]
```

Kas on tühi?

`print(s.empty)` False

Suurus:

`print(s.size)` 5

Väärtused:

`print(s.values)` `[1 2 3 4 5]`

Veel tunnuseid ja funktsioone...

- shape
- iloc
- loc
- data
- T
- head()
- tail()
- abs()
- add()
- all()
- any() jne

Funktsioonid on kujul: funktsiooninimi()

- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html

DataFrame

- Kahemõõtmeline, erinevat tüüpi andmeid endas hoidev, muudetav massiiv.
- Põhimõtteliselt tabel, milles on read ja veerud.

```
import pandas as pd
isikud = [['Juss',10],['Illar',12],['Mari-Liis',13]]
andmestruktuur =
  pd.DataFrame(isikud,columns=['Nimi','Vanus'],dtype=int)
print(andmestruktuur)
```

Väljund:

	Nimi	Vanus
0	Juss	10
1	Illar	12
2	Mari-Liis	13

DataFrame'i funktsioonid

	Nimi	Vanus
0	Juss	10
1	Illar	12
2	Mari-Liis	13

Elukoha veeru lisamine:

```
isikud =
  [['Juss',10],['Illar',12],['Mari-Liis',13]]
andmed =
  pd.DataFrame(isikud,columns=['Nimi','Vanus'],dty
  pe=int)

andmed['Elukoht']=pd.Series(['Tartu','Elva','Võru
  ']
,index=[0,1,2])

print(andmed)
```

	Nimi	Vanus	Elukoht
0	Juss	10	Tartu
1	Illar	12	Elva
2	Mari-Liis	13	Võru

Ridade lisamine:

```
uued_read = pd.DataFrame([['Kalmer', 9,
  Tallinn'],['Jaanika', 11, 'Haapsalu']], columns
  = ['Nimi','Vanus','Elukoht'], index=[3,4])

andmed = andmed.append(uued_read)
```

	Nimi	Vanus	Elukoht
0	Juss	10	Tartu
1	Illar	12	Elva
2	Mari-Liis	13	Võru
3	Kalmer	9	Tallinn
4	Jaanika	11	Haapsalu

Veeru kustutamine:

```
del andmed['Vanus'] või
andmed.pop('Vanus')
```

	Nimi	Elukoht
0	Juss	Tartu
1	Illar	Elva
2	Mari-Liis	Võru
3	Kalmer	Tallinn
4	Jaanika	Haapsalu

Siin on väga oluline omistada muudetud andmestruktuur.

Rea kustutamine:

```
andmed = andmed.drop(2)
print(andmed)
```

	Nimi	Elukoht
0	Juss	Tartu
1	Illar	Elva
3	Kalmer	Tallinn
4	Jaanika	Haapsalu

Veel funktsioone...

Tunnused on väga sarnased Series omadega ja osa funktsioone samuti.

- duplicated()
- equals()
- divide()
- get_values()
- groupby()
- max()
- mean()
- mod()
- pow() jne

- https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html

Kuidas andmeid programmi saada?

CSV failist - meetodid read_csv, read_table

```
andmed = pd.read_csv("minuAndmed.csv")
```

Struktureeritud andmed failist JSON - read_json
Excel tüüpi failid - read_excel
Andmeid saab avada kätte ka andmebaasist - read_sql_table, read_sql_query

Kust andmeid saada?

Internetis on erineval kujul andmeid:

- https://www.stat.ee/
- HaridusSILM - https://goo.gl/qTJpbr
- http://ec.europa.eu/eurostat/data/database
- https://www.kaggle.com/
- http://datasci.ee/ressursid/#andmestikud

Isiklikud andmed nutirakendustest ja ka pulsikellast:

- https://support.polar.com/en/support/how_do_i_export_individual_training_sessions_from_polar_flow_web_service

Kuidas kasutada Thonnys?

Select Tools -> Manage Packages
Otsida moodulit nime järgi: Pandas.
Installib pandase ja sellega seotud moodulid nagu nt NumPy.

Demo

Kodutöö

Suurem ülesanne - Projekt

Ülesande täpse püstituse teeb üliõpilane ise. Temaatika peab teile endale huvi pakkuma.

Programm peab vastama järgmistele nõuetele.

- Peab olema enda tehtud.
- Orienteeruv töömaht lahendamisel 8 tundi. (Ajakulu esitada eraldi ligikaudse aruandena.)
- Programm peab töötlema andmeid
  - Andmed võivad olla pärit veebist (stat.ee vm) või enda omad
  - Andmed tuleb lugeda failist (nt csv)
- Programm peab midagi küsima kasutajalt (kasvõi failinime)
  - Võib eeldada, et kasutaja sisestab sobivate andmetega faili nime
- Programm peab andmete põhjal midagi mõistlikku arvutama ja tulemuse ekraanil esitama.
- Järgmistest nõuetest võib jätta kaks täitmata
  - graafiline väljund (diagramm vms)
  - filtreeritud andmete teise faili kirjutamine
  - kasutaja valiku põhjal arvutuste tegemine
  - iseseisva (ilma Pythonita käivituva) programmi tegemine

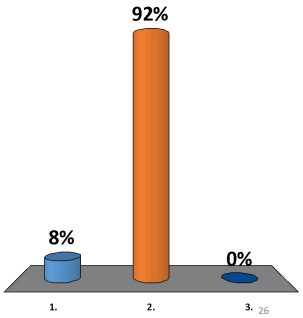
Suurem ülesanne - Projekt

Idee: hiljemalt 5. detsember Moodle'is

Esitlus: esitlus 21. detsembril, 11. ja 24. jaanuaril?? (peab olema tehtud enne eksamit)

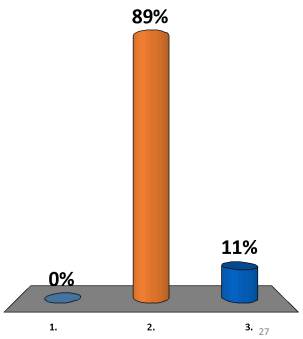
Loengu tempo oli

1. liiga kiire
2. paras
3. liiga aeglane



Materjal tundus

1. liiga lihtne
2. parajalt jõukohane
3. liiga keeruline



Suur tänu osalemast!
Kohtumiseni!

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Inga Konovalova**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Andmetöötluse mooduli pandas õppematerjalide koostamine programmeerimise kursustele

   mille juhendaja on Eno Tõnisson

   1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 14.05.2018