TARTU ÜLIKOOL
Arvutiteaduse instituut
Matemaatika- ja informaatikaõpetaja õppekava

Ruth Schihalejev

Õppeprotsessi kognitiivsetele tasemetele vastavad ülesanded gümnaasiumi kursusel „Programmeerimine“

Magistritöö (15 EAP)

Juhendaja: Tauno Palts, MA

Tartu 2021

Õppeprotsessi kognitiivsetele tasemetele vastavad ülesanded gümnaasiumi kursusel „Programmeerimine“

Lühikokkuvõte:

Programmeerimise õpetamiseks ja õppimiseks on eesti keeles loodud küll erinevaid kursusi ja materjale, kuid siiani ei ole pööratud piisavalt suurt rõhku ülesannete varieeruvusele kõigi õppeprotsessi kognitiivsete tasemete hulgas. Tuginedes varem tehtud teadustöödele ülesannete kategoriseerimise kohta ja olemasolevatele materjalidele, koondati magistritöös maatrikstaksonoomia tasemetele vastavad programmeerimise ülesannete tüübid. Igale tüübile lisati näiteülesanne(/-ded), mis on mõeldud kasutamiseks gümnaasiumi valikkursusel „Programmeerimine“. Koostatud materjali põhjal tehti hindeline töö ja seda kasutati ühes Eesti gümnaasiumis. Uuringu tulemused kinnitavad varem läbiviidud teadustööde tulemusi, et ülesande tüüpide konkreetsed raskusastmed sõltuvad palju ka sihtrühmast, kuid on siiski vajalikud, et õpilased saaksid näidata oma tugevaid ja nõrku külgi, et õpetajad oskaksid neid paremini juhendada. Lisaks viidi läbi küsitlus, et uurida, kust saavad hindelisteks töödeks inspiratsiooni ja milliseid ülesannete tüüpe kasutavad programmeerimise algteadmisi õpetavad õpetajad. Mitmed vastajad tõdesid, et kasutavad olemasolevaid materjale ja näidiseks pakutud hindelisi töid, mis kinnitab vajadust erinevatele kognitiivsetele tasemetele vastavate ülesannete varieeruvust suurendada ka pakutavates õpetajamaterjalides.

Võtmesõnad:

Informaatika didaktika, õppeprotsessi kognitiivsed tasemed, programmeerimine

CERCS: S270 Pedagoogika ja didaktika, P175 Informaatika, süsteemiteooria

Exercises for levels of cognitive domain in the high school course „Programming“

Abstract:

There are various courses and materials for teaching programming in Estonian, but there is a lack of variability of exercises in some cognitive domains. Based on previous research about task categorization and existing materials, the exercise types for the Matrix taxonomy were aggregated in this Master's thesis. For each exercise type, an exemplary exercise was added, which can be used in the high school course "Programming". Based on produced material, a test was composed and it was piloted in an Estonian high school. The results of this research confirm the results of the previous studies, the difficulty of the exercise types depends on a concrete group. But different types are still necessary for students to demonstrate their strengths and weaknesses so teachers can guide them better. In addition, a survey was carried out to find out where do programming teachers get their inspiration for tests and what kind of exercise types they are using for novice programmers. Several respondents stated that they use provided materials and test examples which confirms the need to increase the variability of exercises from the cognitive domains for sample materials as well.

Keywords:

Computer science education, Learning taxonomy, Cognitive domain, Programming

CERCS: S270 Pedagogy and didactics, P175 Informatics, systems theory

Sisukord

Sissejuhatus 6

1. Programmeerimise algkursused Eestis 8

1.1 HITSA gümnaasiumi informaatika valikkursused 8

1.2 Tartu Ülikooli programmeerimise õpetamine üldhariduskoolidele 9

1.2.1 MOOCid 9

1.2.2 Tehnoloogia tarbijast loojaks 10

1.3 Tallinna Tehnikaülikooli kursus „Rakenduste loomise ja programmeerimise alused“ 11

1.4 Kursuste näidistöödes olevad ülesannete tüübid 12

2. Õppeprotsessi kognitiivsed etapid 14

2.1 Bloomi taksonoomia programmeerimises 14

2.2 Maatrikstaksonoomia 16

3. Metoodika 19

3.1 Õppeprotsessi kognitiivsetele tasemetele vastavad ülesanded 19

3.1.1 Maatrikstaksonoomia tõlkimine 19

3.1.2 Taksonoomia tasemetele vastavad ülesannete tüübid 21

3.2 Küsimuste tüüpide uuring gümnaasiumis 23

3.2.1 Kontrolltööks valmistumine 23

3.2.2 Kontrolltöö koostamine 24

3.3 Uuring praeguse praktika kohta hindeliste tööde koostamisel 25

3.3.1 Küsitluse valim 26

3.3.2 Küsimustiku ülesehitus 27

4. Tulemused 28

4.1 Juhendmaterjal veebilehena 28

4.2 Näidistöö tulemuste analüüs 33

4.2.1 Iga kategooria tulemuste analüüs 34
4.2.2 Näidistöö arutelu 37
4.3 Küsitluse tulemused 39
4.3.1 Vastajate taust seoses õpetamise ja selle teooriaga 39
4.3.2 Kasutatavad ülesannete tüübid hindelistes töödes 40
4.3.3 Küsitluse arutelu 45
Kokkuvõte 47
Viidatud kirjandus 49
Lisad 55
I. Kordamismaterjal õpilastele 55
II. Kordamistunni konspekt 61
III. Kontrolltöö variant A 63
IV. Kontrolltöö variant B 67
V. Küsimustik 71
VI. Küsimustikus pakutud ülesandetüüpide näiteülesanded 76
VII. Litsents 84

Sissejuhatus

Aastal 1956 pakkus hariduspsühholoog Benjamin Bloomi töörühm välja õppe- ja kasvatustöö eesmärkide liigituse (Bloom et al., 1956), mille kognitiivse taksonoomia liigitust kasutatakse haridusvaldkonnas siiani (Anderson et al., 2001; Gomes & Correia, 2018). Peamiseks põhjuseks on asjaolu, et taksonoomia suunab andma õpilastele erineva tasemega ülesandeid. Aegade jooksul on aga hariduse sisu, sh õppeained, muutunud ning ka uutele ainetele oleks vaja Bloomi taksonoomiat rakendada. Näiteks on õppekavasse juurde tulnud informaatika kursused, milles on võrdselt olulisel kohal nii teoreetilised teadmised kui ka praktilised oskused.

Mitmete kriitikat avaldavate autorite kõrval on teised Bloomi taksonoomiat uuesti analüüsinud ja kohendanud (Anderson et al., 2001) ning seda ka konkreetsete õppeainete kontekstis (Köksal & Ulum, 2018; Smith et al., 2020; Wang et al., 2019). Selles töös keskendutakse Bloomi taksonoomia arvutiteadusele mõeldud edasiarendusele (Fuller et al., 2007), mis võimaldab eristada teooria- ja praktikateadmiste õppeprotsessi etapilisust. Teemavalik tuleneb asjaolust, et autori hinnangul ei suuna praegused programmeerimiskursuste juurde kuuluvad (näidis)kontrolltööd õpetajaid andma ülesandeid, mis võimaldaksid õpilastel demonstreerida oma erinevaid teadmiste tasemeid. Eri liiki teadmiste harjutamise ja kontrolli vajalikkusele lisab kinnitust Lahtineni töögrupi (2007) tehtud uuring, mille tulemusena täheldati, et algajad programmeerijad liigituvad oma õpiteede järgi kuude eri rühma.

Kuigi mitmed (Gomes & Correia, 2018; Scott, 2003; Thompson et al., 2008; Whalley et al., 2006) on analüüsinud konkreetsete ülesannete paigutust Bloomi taksonoomiasse, ei ole eesti keeles teada ühtki juhendmaterjali, kus oleks kokkuvõtlikult olemas erinevaid programmeerimisega seotud ülesannete tüüpe, mida saaks kasutada nii harjutamiseks kui ka teadmiste kontrollimiseks. Magistritöö põhieesmärk on luua eestikeelne juhendmaterjal, mis oleks abiks informaatikaõpetajale, kes soovib programmeerimisega seotud õppetöös kasutada erinevaid kognitiivseid õppeprotsessi tasemeid arendavaid ja kontrollivaid ülesandeid. Selle juures on oluline analüüsida, milliseid ülesannete tüüpe esineb programmeerimises ja kuhu nad taksonoomias asetuvad. Ülesandetüüpide kogumisel lähtutakse olemasolevatest kursustest Eestis ning rahvusvahelistest teadustöödest. Näiteülesannete loomisel keskendutakse viimati arendatud gümnaasiumi valikkursusele, mille käigus tutvutakse Pythonis programmeerimise baastõdedega. Koostatud materjali

põhjal viiakse läbi kontrolltöö ühes gümnaasiumis, et analüüsida materjali kasutatavust. Töö uurimuslikus osas otsitakse vastuseid küsimustele, kust saavad õpetajad inspiratsiooni oma hindeliste tööde ülesehituseks ja kuivõrd nad kasutavad eri tüüpi ülesandeid programmeerimise algkursustes. Selleks viiakse läbi küsitlus, mille kõrvaltulemus on tutvustada vastajale magistritöö käigus kogutud ja loodud erinevaid ülesande tüüpe.

Viimastel aastatel on arendatud erinevaid kursusi tekstilise programmeerimiskeele Pythoniga tutvumiseks. Näiteks olid varasemalt väga populaarsed Tartu Ülikooli MOOC-id „Programmeerimine maalähedaselt“, „Programmeerimise alused“ (*Kursuste võrdlus*, s.a.), viimasel aastal on arendatud välja 16−26aastatele mõeldud kursus „Tehnoloogia tarbijast loojaks“ („TTL“, s.a.). Samuti on Hariduse Infotehnoloogia Sihtasutuse eestvedamisel loodud gümnaasiumi valikkursuste komplekt, milles on ka Tartu Ülikooli eestvedamisel kursus „Programmeerimine“, mida õpetatakse Pythoni baasil (*Gümnaasiumi informaatika ainekava*, s.a.). 2020. aasta kevadel valminud bakalaureusetöös selgitas Eleriin Rein muuhulgas välja, et tema valimi põhjal õpetatakse Eestis kõige sagedamini programmeerimist keeles Python. Seetõttu keskendub ka käesolev töö Pythoni algteadmiste kursusele. Kursuste sisusid on Eesti koolides erinevaid ning kuigi valikaine staatuse tõttu on kooliti teemadevalik varieeruv, on põhiteemad tingimuslaused ja tsüklid (Rein, 2020, lk 36). Need teemad läksid kokku ka valitud kooli õppegraafikuga, kus magistritöö tulemuste analüüsimiseks viidi läbi töökavasse juba varemalt planeeritud kontrolltöö. Mõlemast lähtuvalt tuuakse siinses töös ülesannete näited peamiselt tingimuslausete ja tsüklite kohta.

Kirjalik osa on ülesehituselt jagatud nelja sisupeatükki. Esimeses osas antakse ülevaade Eestis pakutavatest programmeerimise alusteadmiste kursustest programmeerimiskeeles Python ning nende kursuste juurde kuuluvatest näidistöödest. See peatükk on aluseks töö põhiosas ülesannete genereerimisel ning toeks küsitluse vastuste analüüsimisel. Teine teoreetilise tausta ülevaade käsitleb Bloomi taksonoomiat ja selle edasiarendusi, samuti teisi töid, milles on proovitud taksonoomiat arvutiteadusele, eelkõige programmeerimise õpetamisele rakendada. Kolmas peatükk keskendub tehtava töö metoodikatele ehk kuidas toimus valitud taksonoomia tõlketöö ja ülesannete leidmine, kuidas koostati uuringut gümnaasiumis ning viidi läbi küsitlust. Viimane sisupeatükk tutvustab kõigi kolme tööetapi tulemusi. Lisadest leiab uuringus kasutatud kordamistunni materjalid, läbiviidud kontrolltöö kaks varianti ning küsitluse.

1. Programmeerimise algkursused Eestis

Programmeerimisega tehakse tänapäeval Eestis tutvust juba lasteaedades, kui mängima hakatakse erinevate veebi- ja põrandarobotitega (*Tehnoloogiaõpe lasteaias*, s.a.). Visuaalsete ehk graafiliste programmeerimiskeelte juurest tekstiliste poole jõutakse peamiselt põhikooli lõpus või gümnaasiumi alguses (*Tehnoloogiaõpe 7.-9. klassile*, s.a.). Kuigi Hariduse Infotehnoloogia Sihtasutus (edaspidi ka HITSA) soodustab ja toetab igati IT-õpet igas kooliastmes (*IT-õppe teekaart*, s.a.), jõuavad mõned õpilased teadliku algoritmilise mõtlemiseni alles kõrgkooli või kutsehariduse tasandil. See töö keskendub gümnaasiumiastme tekstilise programmeerimiskeele aluskursustele Pythonis, seega siin peatükis mainitakse ka teisi kursusi, kuid keskendutakse peamiselt Pythoni baasteadmiste omandamise võimalustele Eestis ja sellele, millised on nendes kursustes näidismaterjalid hindelisteks töödeks.

1.1 HITSA gümnaasiumi informaatika valikkursused

Aastatel 2017−2019 koostas gümnaasiumi informaatika (ehk GINFi) ainekava töörühm HITSA eestvedamisel Tallinna Ülikooli, Tartu Ülikooli ja Tallinna Tehnikaülikooli koostöös viis uut gümnaasiumi informaatika valikkursust (*Metoodiline juhend õpetajale…*, s.a.). Lisaks põhjalikele toetavatele õpikutele on olemas ka tunnikavad, õppematerjalid, kursuste läbiviimise juhendid õpetajale ning Moodle'i sisustatud näidiskursused. Materjalid toetavad nii klassikalist kontaktõpet kui ka iseseisvat materjali omandamist, samuti hübriidvorme (*Metoodiline juhend õpetajale…*, s.a., lk 5).

HITSA kodulehel (*Gümnaasiumi informaatika ainekava*, s.a.) reklaamitakse, et õpilastel on võimalus valida kursuste „Programmeerimine", „Tarkvaraarendus", „Kasutajakeskne disain ja prototüüpimine", „Tarkvara analüüs ja testimine" ning „Digiteenused" vahel. Kursuste läbimise üheks suuremaks ühiseks väljundiks on võimalus õpitud teadmiste varal koostada rühmatööna digilahenduse arendusprojekt DigiTaru (*Gümnaasiumi informaatika ainekava*, s.a.). Selle töö seisukohalt on oluline baasteadmisi tutvustav kursus „Programmeerimine".

„Programmeerimine" on mõeldud sissejuhatava kursusena, kus tehakse tutvust valdkonna põhimõistete ja Pythonis koodi kirjutamisega (*Gümnaasiumi informaatika ainekava*, s.a.). Tartu Ülikooli töörühma loodud õpik (Tõnisson et al., s.a.-b) koosneb kaheksast peatükist, kus on nii baasteadmiste omandamist kui ka valdkonnapõhise silmaringi avardamist.

Põhiteadmiste hulka on siin arvatud andmetüüpide ja muutujate tundmine, kahepoolne suhtlus kasutajaga käsurea abil, tingimuslaused, tsüklid, järjendid ja sõned, funktsiooni defineerimine ning andmevahetus veebi ja failidega. Kõrvalteemadena puudutatakse kilpkonnagraafikat Turtle, juhusliku arvu moodulit random, graafilisi liideseid Tkinter ja EasyGUI (Tõnisson et al., s.a.-b).

Õpikuga on kaasas õpetajamaterjalid (Tõnisson et al., s.a.-a), kust leiab kasutamiseks ja/või inspiratsiooniks mõeldud Tartu Tamme Gümnaasiumis kasutatavad kursuse läbiviimist toetavad materjalid. Sealne hindamisjuhend näeb ette, et õpilase lõpphinne sõltub iganädalastest programmeerimisülesannetest, arvestuslikest testidest ning kontrolltööst, kusjuures kontrolltöö on analoogiline iganädalastele töödele (Tõnisson et al., s.a.-a).

1.2 Tartu Ülikooli programmeerimise õpetamine üldhariduskoolidele

Tartu Ülikooli arvutiteaduste instituut on juba 2014. aastast alates aktiivselt pakkunud võimalust teha tutvust Pythoni abil programmeerimise alusteadmistega (*Arhiiv – IT-KURSUSED*, s.a.). Ülikooli eesmärk on olnud jõuda veebipõhiste kursuste abil inimesteni, kellel mingil põhjusel ei ole võimalik programmeerimisega tutvust teha oma kooli juures (*Arhiiv – IT-KURSUSED*, s.a.). Kursused „Programmeerimine maalähedaselt“, „Programmeerimise alused“ ja „Programmeerimise alused II“ toimusid suurte osalejate arvudega kuni aastani 2019 („MOOCid“, s.a.). Alates 2020. aasta kevadest keskendutakse 16−26aastastele noortele majandus- ja kommunikatsiooniministeeriumi tellitud kursusega „Tehnoloogia tarbijast loojaks“, mille läbi(vii)mise vorme on erinevaid („TTL“, s.a.).

1.2.1 MOOCid

Tartu Ülikooli arvutiteaduse instituudi läbi viidud kolm MOOCi (ingl *massive open online course*) on mõeldud jätkukursustena, „Programmeerimine maalähedaselt“ on teistest väiksema mahuga, selle maht on üks euroainepunkt, kuid teised on kolme euroainepunktised („MOOCid“, s.a.). Kõigi materjalid asuvad vabalt kättesaadavana courses.cs.ut.ee lehel, osalejatele mõeldud testid, ülesannete esitamine-hindamine ja suhtlus toimus Tartu Ülikooli Moodle'i kursusel („MOOCid“, s.a.).

Kursust „Programmeerimine maalähedaselt“ loetakse soovituslikult järjekorras esimeseks, seal tehakse põgusalt tutvust andmetüüpide ja muutujate, kahepoolse suhtlusega kasutajaga käsurea abil, tingimuslausete, while-tsükli, järjendite ja sõnede, regulaaravaldiste,

funktsiooni defineerimise ning andmevahetusega veebi ja failidega („MOOCid“, s.a.). Kõrvalteemadena puudutatakse kilpkonnagraafikat Turtle ning mitmeid igapäevaseid valdkondi, mis on programmeerimisega seotud (Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm, 2018b).

Kursuse „Programmeerimise alused“ valimiseks soovitatakse läbida eelnevalt kirjeldatud kursus, kuid samas julgustatakse, et seda saab ka ilma eelneva kogemuseta edukalt õppida („MOOCid“, s.a.). Üldiselt on siin kõik teemad samad, kuid nendega tutvutakse süvendatumalt, lisatud on for-tsükkel, juhuarvu moodul random, graafilised liidesed Tkinter ja EasyGUI (Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm, 2018a). „Programmeerimise alused II“ eeldab juba kindlasti baasteadmiste olemasolu, sest süvendatakse eelmiste kursuste teadmisi („MOOCid“, s.a.). Kursuse kodulehelt (Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm, s.a.) leiab, et õpitakse kahekordset järjendit, kahekordset tsüklit, uusi andmestruktuure ennik, hulk ja sõnastik, uuritakse muutujate viitamissüsteeme ja muteerumisi. Samuti puudutatakse testimist ja silumist ning rekursiivseid väljakutseid. Väikeste kõrvalteemadena pakutakse taaskord nupukesi erinevatest valdkondadest.

Nende kolme kursuse lõpetamiseks on vajalik sooritada iganädalaselt test ning lahendada kohustuslik arv ülesandeid („MOOCid“, s.a.). Eduka läbimise korral anti seda tõendav diplom (*Kasulik info,* 2015). Hilisematel aastatel on olnud võimalus sooritada kontrollitud oludes arvestustöö, mida arvestati Tartu Ülikooli informaatika bakalaureuseõppesse sisse astumisel ja/või seal osalemisel (Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm, 2018a). Kursuse lehelt (Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm, 2018a) leiab arvestustöö kirjelduse, mis selgitab, et töö koosnes kahest osast, millest esimeses oli vaja lahendada ülesanded abimaterjale kasutamata paberil ning teises vajadusel materjale kasutades arvutis.

1.2.2 Tehnoloogia tarbijast loojaks

„Tehnoloogia tarbijast loojaks“ on mõnes mõttes asenduskursus eeltutvustatud MOOCidele, sest käivitati see varasemate lõppedes ning kasutusele võeti paljud varem koostatud materjalid („TTL“, s.a.). Samuti vastab see peatükis 1.1 kirjeldatud gümnaasiumi valikkursusele „Programmeerimine“, mistõttu on selleks loodud õpik kasutusel ka siin. Kursuse kodulehelt („TTL“, s.a.) saab lugeda, et erinevalt MOOCidest on siin kursusel

uudsena kasutusel HITSA eestvedamisel loodud kursuse „Tarkvaraarendus“ materjalid ning ülesannete automaatkontrolli keskkond lahendus.ut.ee.

Kursus üritab lahendada probleemi, et koolides on puudu vastavatest õpetajatest või piisavast hulgast huvitunud õpilastest. Selle kinnituseks on asjaolu, et kursust on võimalik läbida mitmes eri vormis („TTL“, s.a.). Kodulehel („TTL“, s.a.) selgitatakse, et üks variant on see, kui kursust läbitakse tavapäraselt tunniplaanis oleva ainena. Seega on koolis olemas oma õpetaja, kes kasutab etteantud materjale või leitakse see õpetaja kursuse korraldajate kaudu. Teine võimalus on, et tunniplaani aine ei sobitu, kuid õpetaja kohtub mõned korrad õpilastega koolis ja ülejäänud aja konsulteerib neid veebi teel. Kolmas variant realiseerub eelkõige olukorras, kus ühest koolist ei ole valikkursuse jagu huvitunud õpilasi või ei ole lubatud vanuses (kuni 26aastane) isik enam ühegi kooli liige. Selle valiku puhul töötab õpilane materjalidega iseseisvalt, vajadusel saab läbi interneti kaugmentorilt tuge. Vastavad mentorid leiavad õpilastele kursuse korraldajad näiteks õpetajate, IT-ettevõtete töötajate või IT-eriala tudengite seast („TTL“, s.a.).

Selle kursuse lõpetamiseks on vaja sarnaselt eelmistele kursustele lahendada jooksvalt koodi kirjutamise ülesandeid ja nädalateste. Kuid lisaks on vaja koostada, eelistatult rühmatööna, omaloominguline digilahenduse arendusprojekt ning sooritada arvestustöö („TTL“, s.a.). Arvestustöö on analoogiline iganädalastele töödele.

1.3 Tallinna Tehnikaülikooli kursus „Rakenduste loomise ja programmeerimise alused“

Sirvides Eesti gümnaasiumide õppe- ja ainekavasid, esines võrdlemisi mitmetes koolides kursuse „Rakenduste loomise ja programmeerimise alused“ nimetust. Ka sellele kursusele on avalikult kättesaadavad õppematerjalid, näiteks on olemas sisustatud Moodle’i e-kursus, õpik ja töölehed (Vilipõld et al., 2013). Sellele kursusele loodud õpik on välja antud aastal 2013 Tallinna Tehnikaülikoolis, õppematerjalide kogumis E-koolikott on väljaandjaks märgitud Sihtasutus Eesti Teadusagentuur (Vilipõld et al., 2013).

E-koolikoti keskkonnast leitav õpik (Vilipõld et al., 2013) on üles ehitatud mitmest põhimoodulist, kusjuures komplekti peamine eesmärk on tutvustada algoritmilist mõtlemist, modelleerimist ja disainimist ning mingi konkreetse programmeerimiskeele oskus on teisejärguline. Õpiku põhiteemad on näiteks objekt-orienteeritus, rakendusega seotu ja selle

loomine ning algoritmimine, sh kordused, valikud ja funktsioonide loomine. Edasi kinnistatakse ja süvendatakse neid teadmisi erinevate programmeerimiskeelte toel. Näiteks käsitletakse Scratchi, VBA-d ning Pythonit (Vilipõld et al., 2013, lk 5–10).

Pythoni juures käsitletakse andmetüüpe ja muutujaid, kilpkonnagraafikat, kahepoolset suhtlust kasutajaga käsurea abil, funktsioonide defineerimist, erinevaid mooduleid, tingimus- ehk valikulauseid, tsükleid ehk kordusi, loendeid ehk järjendeid, sorteerimisalgoritme ning andmevahetust failidega (Vilipõld et al., 2013, lk 153–230). Selle kursuse juurde ei ole antud hindeliste tööde näidiseid.

1.4 Kursuste näidistöödes olevad ülesannete tüübid

Nagu eelneva põhjal välja tuli, siis enamik Eestis pakutavatest eestikeelsetest programmeerimise alusteadmiste kursustest gümnasistidele on välja töötanud Tartu Ülikooli vastav töörühm ning vaid üks kursus on koostatud teise töörühma poolt. Viimasele ei ole lisatud hindelise töö näidist, küll aga on sellised olemas teistel kursustel. Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm ei ole üldiselt varem tehtud tööd kõrvale jätnud, vaid loonud uued kursused eelmisi materjale kohandades. Seetõttu on ka kursuse lõpetamise tingimused ja näidistööd kõigil kursustel samasugused ning ka näidistestid on identsed. Üldjoontes tuleb lahendada arvestuse saamiseks iganädalased testid ja ülesanded ning osadel kursustel kohustuslikuna lisaks arvestustöö, mis koosneb paber- ja arvutiosast. Järgnevalt kirjeldatakse, milliseid ülesannete tüüpe erinevatest materjalidest leiti.

Kursuse „Programmeerimine“ õpetajamaterjalidest (Tõnisson et al., s.a.-a) leiab, et ülesannetes ja arvestustöö arvutiosas tuleb kirjutada etteantud juhiste järgi probleemi lahendava programmi kood. Seejuures on erijuhtude kirjeldamisega üldiselt tööjuhistes ette antud ka koodi struktuur. Nagu nimigi ütleb, tuleb seda ülesannet lahendada arvutis (Tõnisson et al., s.a.-a).

Testides ja arvestustöö paberosas on ülesandetüüpide varieeruvus suurem. Konsulteerinuna magistritöö juhendajaga, kes on ka kursuse „Tehnoloogia tarbijast loojaks“ üks korraldajatest, sain õiguse tutvuda kursuse osalejatele mõeldud Moodle’is olevate materjalidega. Sealtoodud arvestustöö näitetestis on enamus ülesandeid sellised, kus on vaja vastata, mida kuvab etteantud programm ekraanile. Seejuures on koodid sellised, et print-käske on rohkem kui üks ja samuti antakse ette erinevaid muutujate väärtusi ning tänu sellele

näeb vastusest, kuivõrd õpilane mõistab erinevaid koodifragmente. Üks ülesanne on taanete lisamise kohta, kus on etteantud kood ja õpilane peab selle korrektselt treppima ehk taanded lisama. Kui arvestustöö on ilma valikvastusteta, siis iganädalased testid on automaatkontrolliga, seega osade ülesannete puhul saab õpilane valida vastuse etteantute hulgast ning teistel kirjutama variatsioonideta vastuse, näiteks arvu. Iganädalastes testides leidub ka ülesandeid, milles on vaja vastus anda etteantud koodi sisendi kohta antud väljundi korral, ning selliseid, milles on vaja etteantud koodis märgitud reas nimetatud tegevuse täitmiseks vajalik koodirida valida või kirjutada. Nädalatestides tuleb leida ka etteantud koodis vigaseid kohti ja neid parandada, võrrelda etteantud koodifragmentide tööd ja tulemusi ning nimetada kirjeldatud funktsioone. Mõned küsimused on õpikus käsitletud teooria ja silmaringi teemade kohta.

Kursuse „Programmeerimine“ (Tõnisson et al., s.a.-a) arvestustöö näidistest ja nädalatestid on identsed eelkirjeldatud testidega. MOOCide materjalide („MOOCid“, s.a.) hulgast leitud arvestustöö vihjete ja materjalides leiduvate vaheküsimuste põhjal võib eeldada, et ka MOOCidel kasutatavad testid olid samasugused. Küll aga võib välja tuua, et kursusel „Programmeerimine“ (Tõnisson et al., s.a.-a) tehakse koondhinnet mittemõjutavate tunnitöödena paaristöid, kus on ülesannete varieeruvus suurem. Seal tuleb lisaks eelnevale analüüsida programmi koodi nii, et selgitada iga rea või kogu koodi tööd ning kirjutada etteantud lühiülesannetele ühe realisi koodivasteid.

Kirjeldatust saab järeldada, et kuigi jooksvates töödes ja harjutamisel kasutatakse mitmesuguseid ülesandeid, on suuremates ja seega kaalukamates hindelistes töödes ülesandetüüpide varieeruvus väiksem. Kuigi raskemate ülesannete vastusteni jõudmiseks on vaja lahendada mitmeid alaülesandeid, ei ole õpilasel võimalust demonstreerida kogu oma mõttekäiku ning samuti ei ole tööde ülesehituses näha süsteemset lähenemist ülesannete raskusastmete varieerimisel, sest kasutatud on vaid kahte eri tüüpi ülesannet.

2. Õppeprotsessi kognitiivsed etapid

Psühholoogias (*Cognitive Ability – APA Dictionary of Psychology*, s.a.) arvestatakse kognitiivsete võimete hulka näiteks meelde jätmine, mõistmine, teadlikkus, arutlemisvõime ja hinnangute andmine. Aastal 1956 avaldas õppeprotsessi kognitiivsete etappide liigituse ehk taksonoomia Benjamin Bloom koos oma töörühmaga (Bloom et al., 1956). Siis üritati õppimisega seotud kognitiivsed eesmärke ka järjestada ning lõpuks pakuti välja hierarhia: teadmine, mõistmine, rakendamine, analüüsimine, sünteesimine ja hindamine (Bloom et al., 1956, p. 18). Igal kategoorial on ka alamjaotused ja kirjeldused, mis aitavad mõista vastava kategooria olemust. Juba algselt raamatut välja andes tõdeti, et etappide järjestamine on väga mitmetahuline ülesanne ning tulemus ei ole kindlasti kõiki rahuldav, kuid võib anda kätte esmase suuna (Bloom et al., 1956, p. 18).

Peale seda on mitmed teadlased pakutut toetanud ja kritiseerinud, süstemaatilisemalt võeti taksonoomia vaatluse alla sajandivahetusel, kui Lorin W. Andersoni juhtimisel vaadati algne süsteem uuenenud teadmistest lähtuvalt üle (Anderson et al., 2001). Muuhulgas kohendati sõnastustes nimisõnad tegusõnadeks ning vahetati järjestuses omavahel kaks viimast kategooriat, asendades sünteesimise loomisega, kuigi samas tõdeti, et järjekorda ei saa võtta nii, et loomiseks on ilmtingimata vajalik ka hindamise oskus (Anderson et al., 2001, pp. 266–268). Samuti muudeti taksonoomia kahemõõtmeliseks, seega raskusastmed kujunevad nii kognitiivse kui ka teadmiste taseme järgi (Anderson et al., 2001, p. 266).

Ka programmeerimise õppimine sisaldab endas nimetatud taksonoomias käsitletud komponente. Näiteks peab programmeerija teadma funktsioone ja mõistma nende toimivust, tundma erinevaid konstruktsioone ning oskama kõiki teooriateadmisi kasutada praktikas koodi kirjutamisel. Lisaks peab ta aru saama enda ja teiste kirjutatud koodist ning vajadusel suutma seda parandada. Seega võiks olla lihtne kasutada ka programmeerimise õppimise protsessis Bloomi taksonoomiat ning selle kohta on avaldatud mitmeid artikleid. Järgnevalt tutvustatakse neist valituid, mis olid käesoleva töö suunast lähtuvalt süvendatud tähelepanu all, kusjuures ükski neist ei käsitlenud konkreetselt programmeerimiskeelt Python.

2.1 Bloomi taksonoomia programmeerimises

Essi Lahtinen (2007, p. 39) leidis käesoleva sajandi alguses 254 üliõpilasega tehtud uuringust, et programmeerijaks õppijad jagunevad kuude rühma, mis iseloomustavad nende

õpiteed, ning Bloomi taksonoomia järgi kõrgema taseme „loomine“ ülesannetega hakkama saajad ei pruugi osata lahendada allpool paikneva taseme „mõistmine“ ülesandeid. Sellest lähtuvalt otsustas Ursula Fuller uurida erinevaid tehtud töid ning luua informaatikale suunatud taksonoomia, mis lõpptulemusena illustreeris ka erinevaid võimalikke õpiteid taksonoomia kõrgemate tasemeteni jõudmiseks (Fuller et al., 2007). Fulleri töörühma tööd ja selle edasiarendusi tutvustatakse täpsemalt peatükis 2.2. Siiski on ka edaspidi teised teadlased sellest tööst kõrvale vaadanud ning leidnud, et tasub edasi uurida algupärasemat Bloomi taksonoomiat ja Andersoni juhtimisel tehtud edasiarendust.

Anabela Gomes (Gomes & Mendes, 2009; Gomes & Correia, 2018) on korduvalt analüüsinud Bloomi taksonoomia rakendamist programmeerimise õppimises. Tema eesmärk oli leida lahedus murekohale, et sissejuhatavate kursuste õpitulemused on väga madalad ning seetõttu oleks vaja üle vaadata ja analüüsida õpetamismeetodid, et need oleksid kujundatud õppimist toetavalt. Oma töödes tugineb ta Raimond Listeri väljatoodule, et enne on vaja osata Bloomi taksonoomia madalamaid tasemeid ning alles seejärel võib õpetada kõrgemaid (Gomes & Mendes, 2009, p. 2549; Gomes & Correia, 2018, p. 2). Aastal 2009 avaldatud töös õpetati kursust Bloomi taksonoomia alusel nii, et igas teemas mindi madalamatelt tasemetelt kõrgematele ning ka eksamisse lisati ülesandeid kõigilt taksonoomia tasemetelt (Gomes & Mendes, 2009, p. 2550). Selle tulemusena osalesid tudengid aktiivsemalt tundides, saades aru, et õppimiseks ongi vaja läbida erinevaid tasemeid ning ka eksami sooritamise suhtes olid nad positiivsemad, sest kasvõi osade ülesannete oskamine andis edasiseks pingutuseks motivatsiooni (Gomes & Mendes, 2009, p. 2551). Hiljem tehti analoogiline uuring tsüklite teemaga ning eelnevatele tulemustele lisati asjaolu, et tänu Bloomi taksonoomia järgi tööde koostamisele on ka õppejõud teadlikumad nõrgemate õppurite tasemest ja oskavad neid paremini juhendada (Gomes & Correia, 2018, p. 4).

Bloomi taksonoomia tasemeid programmeerimise õpetamise seisukohast on analüüsinud Terry Scott, tuues samas välja ka igale tasemele vastavad ülesanded (Scott, 2003). Andersoni juhtimisel tehtud edasiarenduse põhitasemetele on ülesandeid ja täpsemaid kirjeldusi-analüüse välja pakutud Errol Thompsoni (Thompson et al., 2008) eestvedamisel. Suurema töögrupiga jõuti teha ka edasiviivamaid järeldusi kui Scott üksinda. Nimelt paigutasid iga eelnevalt valitud ülesannet taksonoomiasse kõik viis autorit ning valikuid kõrvutades leiti kohe, et paigutuseks on vaja teada, mida kursusel õpitakse ja käsitletakse,

sest on mõistetav vahe, kas õpilasele on ülesanne tuttav või on see tema jaoks täiesti uus (Thompson et al., 2008, p. 156).

Töögrupp Jacqueline L. Whalley juhtimisel üritas samuti programmeerimisega seotud testi küsimusi seostada edasiarendatud Bloomi taksonoomiaga ning peale mitmeid vaidlusi jaotati üheksa valikvastusega ja üks lühivastusega küsimus kolme eri põhikategooriasse (Whalley et al., 2006, p. 245). Hindamisel kasutati lühivastuste analüüsimiseks SOLO taksonoomiat, mis näitas, kuivõrd oskab õpilane paigutada õpitavat teadmist süsteemi teiste vajalike teadmistega (Whalley et al., 2006, p. 250).

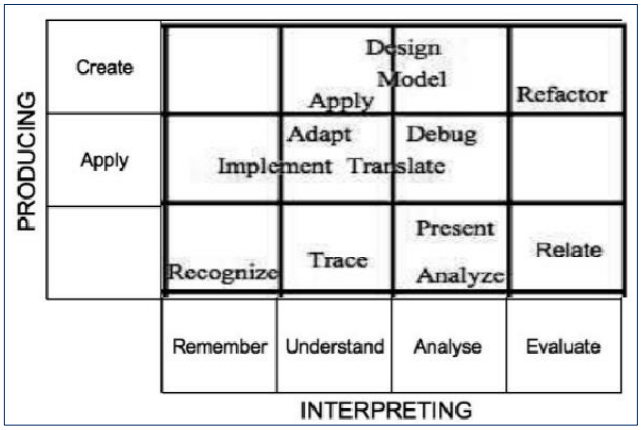
Analoogilist tööd tegid ka Shuhidan, Hamilton ja D'Souza (2009), kes analüüsisid koostatud eksamitöö 19 valikvastustega küsimuste jaotust algsesse Bloomi taksonoomiasse ning hindasid avatud vastusena koodikirjutamise ülesannet SOLO taksonoomia järgi. Ka nemad leidsid, et kategoriseerimine on keerukas protsess ning küsimused jagunesid samuti vaid kolme eri põhikategooriasse, kusjuures „mõistmise" alla liigitus 15 küsimust 19st (Shuhidan et al., 2009, pp. 149–150). Soovitusedki on sarnased Whalley jt (2006) tööle – valikvastustega küsimusi tasub klassifitseerida Bloomi taksonoomia järgi, kuid lisaks tuleb arvestada õppejõudude raskusastme hinnangut ja algajate programmeerijate tulemusi, ning koodikirjutamise ülesandeid hinnata SOLO taksonoomia järgi (Shuhidan et al., 2009, p. 152).

Aastal 2012 loodi aga veebipõhine rakendus, mis juhendab programmeerimisega seotud õpetajaid mõistma Bloomi taksonoomiat (Gluga et al., 2012). Õpetuse käigus tutvustatakse iga taset eraldi, tuues välja pikema kirjelduse, seonduvad märksõnad, ühe näiteülesande programmeerimiskeelega Java ning 1−2-lauselise üldise kokkuvõtte, mida selle taseme ülesandes lahendajalt oodatakse. Kuigi tutvustavas artiklis lubatakse, et juhend on peagi kõigile kättesaadav (Gluga et al., 2012, p. 156), ei ole selge, kust seda leida ning guugeldades magistritöö autor soovitud rakendust ei leidnud.

2.2 Maatrikstaksonoomia

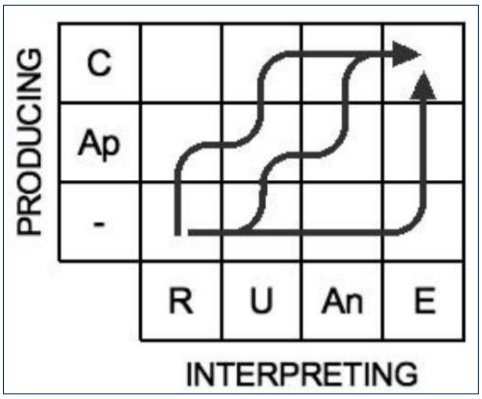
ITiCSE töögrupp koostas maatrikstaksonoomia eesmärgiga, et oleks olemas ainepõhine taksonoomia, millega saab kirjeldada igat programmeerimise õppimise õpiteed (Fuller et al., 2007, p. 153). Selleks analüüsisid nad olemasolevaid taksonoomiaid, mis on siiani loodud, ning uurisid, milleks neid õppetöös kasutatakse. Nad leidsid, et arvutiteadusele on

omapärased ka sellised aspektid, mida teistes õppeainetes niivõrd arendama ei pea. Kokkuvõttes märkisid nad olulisteks oskusteks tõhusa probleemi lahendamise, abstraktse, uudse ja loova lahenduse kujundamise, teadmiste rakendamise, liigitamise, suhtlemisoskuse tõhusaks koostööks, erialaste oskuste ja väärtuste järgimise (Fuller et al., 2007, p. 163).



Joonis 1. Maatrikstaksonoomia (Fuller et al., 2007, lk 166).

Töö inspiratsiooniks sai 2001. aastal avaldatud Bloomi taksonoomia edasiarendus, sest töörühm võttis aluseks, et etteantud programmi mõistmine-tõlgendamine, ise programmi loomine ja koodi parandamine on kolm täiesti eraldiseisvat oskust ning ei saa eeldada, et kui ühte osatakse, osatakse ka teisi (Fuller et al., 2007, pp. 163–164). Loodud maatrikstaksonoomia on näha joonisel 1 inglise keeles. Selle tõlge ei ole tähendusnüansside tõttu nii üksühene ning seetõttu leiab selgitustega eestikeelse tõlke peatükist 3.1. Kõigepealt pandi paika, kuidas asetsevad ja liiguvad tühjas maatriksis Lahtineni töögrupi leitud õpiteed, näiteks kõige kõrgemale oskuste tasemele jõudmiseks on vähemalt 3 erinevat teekonda (vt joonis 2) ning seejärel paigutati maatriksisse programmeerimisega seotud tegevused (Fuller et al., 2007, pp. 164–165).



Joonis 2. Kõige kõrgemale oskuste tasemele liikumise võimalikud teekonnad (Fuller et al., 2007, p. 165).

Maatrikstaksonoomiat on oma uuringus kasutanud näiteks Mila Kwiatkowska (2016), kes hindas kommunikatsioonipõhiselt eksamitöö ülesannete raskusastet. Selleks uuris ta valikvastustega testi küsimusi ühekaupa, võrreldes ekspertide hinnanguid raskusastmele, õigete vastuste osakaalu ja maatrikstaksonoomias asetust (Kwiatkowska, 2016, pp. 3–4).

Põhja-Carolina osariigis asuva Charlotte ülikooli õppejõud (Dorodchi et al., 2017) kasutasid maatrikstaksonoomiat, et toetada õppijaid õppeprotsessi vältel. Selleks analüüsiti õppeperioodi jooksul tehtud kolme testi vastuseid ning edaspidi pöörati rohkem tähelepanu tudengite nõrkadele kohtadele, kursuse lõpueksam koostati aga selle põhjal, kuidas olid tulemused kursusel tehtud testides (Dorodchi et al., 2017, p. 2). Maatrikstaksonoomiat kasutati peamiselt selleks, et teadvustada endale võimalikke õpiteid ja osata sellele tähelepanu pöörata ka tööde analüüsimisel ja õpingute toetamisel.

3. Metoodika

Magistritöö metoodikas on kombineeritud kvalitatiivset ja kvantitatiivset uurimismeetodit, töö koosneb kolmest osast, mis on omavahel põimitud. Kõigepealt tõlgiti peatükis 2.2 tutvustatud maatrikstaksonoomia eesti keelde ning leiti iga tegusõna juurde ülesannete tüübid ja näiteülesanded. Seejärel rakendati materjali ühes Eesti gümnaasiumis, et saada selle kasutatavusele tagasisidet. Paralleelselt katsega uuriti, milliseid ülesannete tüüpe gümnaasiumis Pythoni baaskursusi läbiviivad õpetajad üldse kasutavad.

3.1 Õppeprotsessi kognitiivsetele tasemetele vastavad ülesanded

Magistritöö algne eesmärk oli leida Bloomi taksonoomia tasemetele vastavad ülesannete tüübid ja näiteülesanded. Üsna pea jõuti aga tõdemusele, et oluline oleks eristada omavahel teooria- ja praktiliste teadmiste õppimisprotsessi etapilisust ning seda vajadust kinnitas ka Lahtineni (2007) uuring. Seetõttu võeti ülesandetüüpide kogumise ja genereerimise aluseks ITiCSE töörühma (Fuller et al., 2007) loodud maatrikstaksonoomia, mida tutvustati peatükis 2.2. Kuna autorile teadaolevalt ei olnud keegi seda veel eesti keelde tõlkinud, sai magistritöö eesmärgiks tõlkida käesolev taksonoomia eesti keelde ning seejärel leida igale kategooriale ülesande tüüp või tüübid ning tuua näiteülesanded Pythoniga programmeerimise baaskursuse teemadel.

3.1.1 Maatrikstaksonoomia tõlkimine

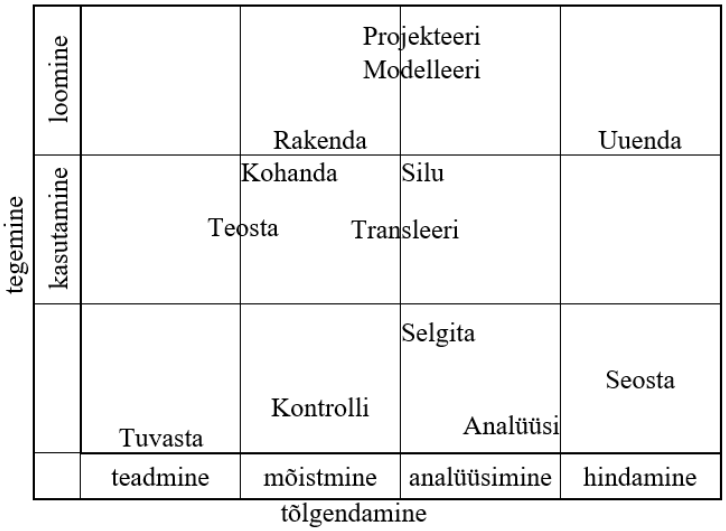
Maatrikstaksonoomia tõlkimisel lähtuti taksonoomiat tutvustavas artiklis olevatest põgusatest kirjeldustest. Kõigile maatriksis olevatele tegevustele selles artiklis siiski selgitust ei antud, näiteks ei kirjeldatud, mida täpsemalt mõeldakse „Translate“ all, samuti puudusid näiteülesanded või ülesannete tüübid. Maatriksisse paigutatud tegusõnade otsetõlget takistas avastus, et alati ei edastanud nimetatud sõna just kõige selgemalt kirjelduses avatud tegevust. Näiteks sõnade „Analyse“ ja „Present“ erinevus tuligi välja alles kirjeldustest, kus „Analyse“ selgitati kui ajalise keerukuse hindamine ning „Present“ kui lahenduse selgitamine (Fuller et al., 2007, p. 165). Samuti oleks sõnade manuaalne üksühene tõlge kaotanud ära tähendusnüansid, sest mitmele sõnale saaks anda sama eestikeelse vaste.

Maatrikstaksonoomias nimetatud tegevused on siiski programmeerimisele kohased ja erialast sõnavara kasutatakse üldiselt erinevates teadlaskondades samamoodi. Seetõttu otsiti

tõlkimiseks juhtnööre ka teistest artiklitest. Näiteks oli suureks abiks Errol Thompsoni töögrupi poolt koostatud juhis (Thompson et al., 2008), kus oli toodud programmeerimise valdkonnast lähtuvalt põhjalikumaid kirjeldusi ja ülesannete tüüpe koos näidetega. Kuigi eelnimetatud juhise koostajad ei maininud maatrikstaksonoomiat, rajasid nad oma töö Andersoni juhtimisel tehtud Bloomi taksonoomia edasiarendusele, millest lähtus ka ITiCSE töögrupp uue taksonoomia loomisel (Fuller et al., 2007, p. 164; Thompson et al., 2008, p. 156). Nii sai sealt leitud kirjelduse ühendada mõne maatrikstaksonoomia kategooriaga.

Maatriksi sees olevate verbide tõlkeprotsess nägi välja selline, et kõigepealt selgitati välja, mida iga tegusõna all mõeldakse, võrreldes ka kahes erineva töörühma artiklis (Fuller et al., 2007; Thompson et al., 2008) olevaid kirjeldusi. Tegusõnade kirjeldused on järgnevad:

- tuvasta (ingl *recognize*): tunne sõnavara, funktsioone, konstruktsioone;
- kontrolli (ingl *trace*): kontrolli programmi tulemit käsitsi;
- selgita (ingl *present*): analüüsi etteantud koodi, selgita programmi tööd;
- analüüsi (ingl *analyse*): hinda lahenduse ajalist keerukust;
- teosta (ingl *implement*): kirjuta etteantud juhiste järgi programmi kood;
- kohanda (ingl *adapt*): muuda lahendus teise struktuuri (sisaldab teadmist mida ja kuidas muuta ning loomist);
- transleeri (ingl *translate*): tõlgi ehk transleeri programm teise programmeerimiskeelde või vormingusse;
- silu (ingl *debug*): leia ja paranda vead;
- rakenda (ingl *apply*): kasuta lahendust suurema ülesande osana;
- modelleeri ja projekteeri (ingl *model*, *design*): koosta abstraktne lahendus või kavanda lahenduse struktuur;
- uuenda (ingl *refactor*): kujunda lahendus (näiteks optimeerimise eesmärgil) ümber.



Joonis 3. Maatrikstaksonoomia (Fuller et al., 2007) tõlge eesti keelde.

Kui kirjeldus oli eestikeelsena paigas, hakati erinevate sõnastike abiga sobivat eestikeelset tegusõna otsima. Sõnastikena kasutati Keeleveebi keskkonnas olevaid eriala-[1] ja tõlkesõnastikke[2], lisaks Sõnaveebi[3] ja Keelenõu[4] keskkonda, vajadusel pöörduti guugeldades ingliskeelsete sõnastike poole. Maatriksi horisontaal- ja vertikaaltelgedel asuvate teonimede tõlked lisati kõige lõpus ning need valiti selle järgi, et sõna edastaks nii algse ingliskeelse sõna mõtet kui ka võtaks kokku vastaval teljel olevad aspektid. Kõigi tõlgete puhul peeti oluliseks ka asjaolu, et see ei oleks liiga pikk, eelistatult ühesõnaline. Tulemuseks saadud maatriks on näha joonisel 3. Tõlkimine on keerukas protsess ning igal inimesel on sellest oma nägemus. Nii võib olla ka selle töö puhul, et kõik tõlgetega ei nõustu, kuid siin on tehtud tõlkeprotsessiga algust ja pakutud välja teadaolevalt esimene maatrikstaksonoomia eestikeelne versioon.

3.1.2 Taksonoomia tasemetele vastavad ülesannete tüübid

Vestlustest kolleegidega on välja tulnud, et informaatikaõpetajad tunnevad puudust õppe- ja juhendmaterjalidest. Magistritöö koostamise käigus leiti mitmeid töid, kus oli toodud

[1] Kasutatud erialasõnastikud: Keeleveebi keskkonnas olevad „Arvutisõnastik“ https://www.keeleveeb.ee/dict/speciality/computer/ ja „IT terministandardi projekti (1998-2001) sõnastik“ https://www.keeleveeb.ee/dict/speciality/itstandard/.

[2] Kasutatud tõlkesõnastikud: Keeleveebi keskkonnas olevad „Inglise-eesti-inglise sõnaraamat Aare“ https://aare.edu.ee/dictionary.html ja „Inglise-eesti / eesti-inglise veebisõnastik Nastik“ https://nastik.palat.ee/.

[3] Sõnaveeb on leitav aadressilt https://sonaveeb.ee/.

[4] Keelenõu keskkond on leitav aadressilt http://kn.eki.ee/.

üksikuid näiteülesandeid, kuid tõlgitud maatrikstaksonoomias nimetatud tegevustele ei ole autori teada veel keegi ülesannetega juhendmaterjali koostanud, sest peatükis 2.2 tutvustatud töödest esimeses (Kwiatkowska, 2016) kasutati testis valikvastustega küsimusi ning teises (Dorodchi et al., 2017) ei koostatud ülesandeid otseselt maatrikstaksonoomia järgi. Ülesandetüüpide valimisel ja näiteülesannete koostamisel peeti silmas, et need oleksid kergelt kohandatavad peatükis 1.1 tutvustatud kursusele „Programmeerimine“, sest selle baasteadmiste õpetamisest lähtuva sisu ja terviklike õppematerjalide kättesaadavuse tõttu on see eeldatavasti kõige sagedamini õpetatav-õpitav kursus.

Õppimisel saadud teadmised salvestatakse erinevatesse aju piirkondadesse (Tokuhama-Espinosa, 2019, p. 1) ning seetõttu on oluline, et teadmise kinnistamiseks harjutataks seda erinevates situatsioonides. Nii areneb abstraktne mõtlemine ja võimekus kasutada sama teadmist uutes olukordades (Vakil & Heled, 2016, p. 212–213). Sellest kaasaegse psühholoogia teadmisest ajendatuna soovis magistritöö autor pakkuda välja erinevaid ülesannete tüüpe, seejuures ka ühele maatriksis nimetatud tegevusele leida võimalikult palju variatiivsust.

Ülesande tüüpide ideid saadi tutvudes erinevate artiklitega ja arutledes juhendajaga, kusjuures tavaliselt oli vaja saadud idee paigutada õige tegusõna juurde, sest konkreetsele maatriksis nimetatud tegevusele ülesande mõtlemine osutus vähem inspireerivaks variandiks. Kõige suuremaks abiks olid õpiku „Programmeerimine“ juurde kuuluvad õpetajamaterjalid (Tõnisson et al., s.a.-a), kus oli harjutamiseks kasutatud erinevat sorti ülesandeid, ning Errol Thompsoni jt (2008) artikkel, kus oli üldsõnaliselt lahti selgitatud, milliseid ülesandeid võiks erinevate Bloomi taksonoomia tasemete juures lahendada. Samuti leiti mõtteid Terry Scotti (Scott, 2003) artiklist, kus oli toodud igale algse Bloomi taksonoomia tasemele näiteülesanded programmeerimiskeeles C++. Leitud ülesanded aitasid genereerida ka uusi tüüpe. Ülesannete tüübid koondati lühikirjeldustena, mitte märksõnadena.

Igale leitud ülesande tüübile lisati ka konkreetne näiteülesanne peatükis 1.1 tutvustatud kursusel „Programmeerimine“ kasutamiseks. Näiteülesannete sõnastamine oli tänu kirjeldatud tüüpidele lihtsam ning sisu poolest ammutati inspiratsiooni õpetajatöös nähtud veaohtlikemast kohtadest ja suuremate ülesannete puhul temaatiliselt elulistest valdkondadest.

3.2 Küsimuste tüüpide uuring gümnaasiumis

Koostatud juhendmaterjalile tagasiside saamiseks viidi läbi uuring gümnaasiumis, kus kahe kümnenda klassi (IT- ja reaalsuuna) õpilased olid lõpetamas kursust „Programmeerimise alused I“, mille sisu ühildub suures osas gümnaasiumi valikkursusega „Programmeerimine“. Selle konkreetse kooli valimise põhjuseks oli asjaolu, et kuigi kursust viis läbi teine õpetaja, oli töö autor seal kasutatavad materjalid ise kogunud ja (osaliselt) koostanud ning hästi kursis olemine lihtsustas uuringu ettevalmistust ja läbiviimist. Kahe erineva klassi valimisse kaasamine oli vajalik, et valim oleks tausta poolest mitmekesisem ja sama teadmiste taseme erinevaid ülesandeid lahendaksid erineva taustaga inimesed.

Uuringu käigus sooviti teada saada õpilaste vastukaja materjalile ning analüüsida kontrolltöö tulemuste abil kategooriate raskusastmete asetust taksonoomias. Vormiks valiti kontrolltöö, sest see ei too tänu kohustuslikkusele lisakoormust õpilastele, seda nii valmistumise kui ka tundide sisu arvelt. Samuti vähendas see õpetaja tööd, sest põhitöö variandid anti talle magistritöö koostajalt. Samas võimaldas see näidete varal õpetajal materjali katsetada järeltöö koostamisel.

Arutelust õpetajaga selgus, et üldiselt lahendavad nad õppetöö käigus ülesandeid, kus on vaja etteantud kirjelduse järgi kirjutada programmi kood, palju harvem kirjutatakse koodi plokkskeemi järgi. Teooria õppimise käigus tutvutakse koodinäidetega, mille puhul tuleb leida programmi väljund ning ühise aruteluna selgitatakse ka õige vastuse leidmiseks tehtud lahenduskäiku. Teisi ülesannete tüüpe nende õppetöös ei kasutata. Seetõttu viidi läbi kordamistunni raames uute ülesannete tutvustuseks mõeldud tund ning hiljem eraldi kontrolltöö.

3.2.1 Kontrolltööks valmistumine

Kontrolltööks valmistumise oluline osa on kordamine, selleks töötati välja kordamismaterjalid ning magistritöö autor viis läbi igale klassile ühe kordamistunni. Uuringuks valiti välja kõigist maatrikstaksonoomia kategooriatest ülesannete tüübid, mis esindaksid kategooriat kõige paremini ning oleksid sisuliselt võimalikult erinevad. Samuti arvestati, et neid oleks võimalik kohaldada kontrolltöösse tulevatele teemadele, mis olid järjendid, for- ja while-kordused. Õpilastele valiti harjutamiseks iga tüübi kohta 1–3 näiteülesannet.

Õpilastele anti kordamismaterjalidena lisas 1 olev tabel ning harjutuste tööleht, kuhu olid tõstetud lisas 1 oleva tabeli viimases veerus olevad ülesanded vormistatult paberil ja arvutis lahendamiseks eraldi. Mõlema klassi kordamistund viidi läbi kaks nädalat (1 õppenädal ja vaheaeg) enne kontrolltööd, pandeemia tõttu distantsõppel Google Meeti vahendusel. Tunnikonspekti leiab lisast 2, kus on toodud õpilastega jagatud kontrolltöös käsitletavate teemade nimekiri ning ülesanded koos nendele mõeldud kommentaaridega. Seda tundi ei andnud nende enda õpetaja, vaid töö autor.

Osaliselt ka kordamistunni veebivormi tõttu õpilastelt väga palju tagasisidet ega küsimusi ei saanud. Üksikud küsimused hõlmasid näiteks märkusi ettenäidatud lahenduse lõpetamatuse või eelmise kontrolltöö teemade kohta, nt „Mida teeb hüüumärk Pythonis?“. Üks õpilane avaldas muret, et uusi ülesandeid on nii palju, et ei oska kusagilt alustada, kuid jäi rahule vastusega, et tuleb etteantud teemad erineval moel läbi töötada, näiteks teooria ja harjutusülesannete abil.

3.2.2 Kontrolltöö koostamine

Kontrolltöösse valiti igast kategooriast üks ülesande tüüp, mis seda kategooriat kõige paremini esindab. Erandiks oli juhtum, kus mitu kategooriat pandi kokku ühte ülesandesse, „Kontrolli“ (ehk kontrolli programmi tulemit käsitsi) ning „Selgita“ (ehk selgita programmi tööd) ühildati ühte ülesandesse, hindamisel arvestati neid eraldi. Tööst jäeti välja analüüsimine, mis algses maatriksis oli kirjeldatud kui ajalise keerukuse hindamine. Põhjuseks oli asjaolu, et ka selles gümnaasiumis ei õpetata ajalise keerukuse hindamist. Ülesannete valimisel töösse oli eesmärk kasutada võimalikult erinevaid ülesandeid. Kontrolltööst valmis kaks varianti, sest sooritusaegasid oli kaks. Erinevates variantides olid üldiselt igas kategoorias samad küsimuste tüübid, kuid ülesande sisu oli erinev. See pidi ennetama olukorda, et absoluutselt kõigil läheb mingi ülesande lahendus valesti, sest käsitletavat nüanssi ei olnud õpilased piisavalt harjutada saanud. Kahes variandis erines kategooria „Seosta“ küsimuse tüüp, ühes variandis oli see paberosa ja teises arvutiosa ülesanne. Variandis A oli vaja lahendada paberosas 4 ja arvutiosas 3 ülesannet, B variandis paberosas 4 ja arvutiosas 2 ülesannet. Kontrolltöö variandid A ja B on leitavad vastavalt töö lisas 3 ja 4. Kontrolltöö soorituse kasutamiseks magistritöös andis nõusoleku variandi A 32 lahendajast 27 ja variandi B 29 lahendajast 25 õpilast.

Teise variandi (lisas 4) ülesannete komplekti lahendas läbi kõrvaline isik, kes on lõpetanud peatükis 1.2.1 tutvustatud MOOC kursused „Programmeerimise alused“ ja

„Programmeerimise alused II“. Katselahendamise võib lugeda õnnestunuks, sest koguajast (70 min) kulus tal lahendamiseks 60 minutit ning selle käigus tuli välja mitmeid kohti, kus parandada sõnastust või teha vormistus lahendaja seisukohalt mugavamaks. Näiteks sai parandatud sõnastus „Selgita, mis kuvatakse programmi tulemusel käsureale“ algusega „Mis ja miks kuvatakse…“. Ühes ülesandes oli kasutatud näitena funktsiooni *sqrt* ning see asendati operaatoriga **, et koodi pikkust optimeerida. Veel oli kasutatud plokkskeemi esimeses rööpkülikus mõistet „sõnade jada“, mis kirjeldati teiste sõnadega lahti. Samuti anti selle ülesande juurde näidissisend, et õpilasel ei kuluks oma lahenduse kontrollimisel aega sisendi genereerimiseks. Paberosa ülesanne 6 oli algselt arvutiosas, kuid kordamisharjutuste eeskujul tõsteti paberosasse.

Katselahendamine oli toeks ka tööde modifitseerimisel. Nimelt selgus, et kontrolltöö tegemiseks ettenähtud aeg on lühem, 60 minutit, ja lisaks anti vahetunnist 3 minutit, et esitada töö. Seetõttu otsustati töös kasutatavate kategooriate koguarvu mitte vähendada, vaid vormistada töö Moodle'i testina, kus teatud kategooriate vahel loositakse igale õpilasele vaid üks küsimus. Loosimisse võetavad kategooriad valiti paberosas selle järgi, mille sisu on üksteisele kõige sarnasem. Näiteks „Kontrolli ja selgita“ ning „Silu“ vajavad kõik koodist arusaamist ja selle järgi tegutsemist. Teises variandis lisati nende hulka ka „Seosta“. Arvutiosas oli kõigile kohustuslik ülesanne „Teosta“, sest oleks ebaõiglane osadele anda ülesande tüüp, mida nad alati koos on harjutanud ning teistele küsimus, mida nad on saanud vähem harjutada. Seetõttu ülejäänud arvutiosa kategooriad „Kohanda“ ja „Transleeri“ paigutati loositavate hulka. Ühes variandis jäi kohuslikuks ka arvutiosasse paigutatud „Seosta“, milles taanete lisamine ja arvutiga lahenduse kontrollimine ei tohiks palju aega võtta.

3.3 Uuring praeguse praktika kohta hindeliste tööde koostamisel

Magistritöö teine eesmärk oli uurida, millised on õpetajate praegused praktikad hindeliste tööde koostamisel ehk milliste ülesannete tüüpide abil hinnatakse õpilaste teadmisi ja oskusi. Küsimustele vastuste saamiseks viidi Pythoniga programmeerimise algteadmisi õpetavate õpetajate hulgas läbi küsitlus, kus uuriti nende tausta seoses õpetamise ja selle teooriateadmistega ning paluti avada oma hindeliste tööde sisu. Küsimustikuga sooviti pakkuda midagi ka õpetajatele: esiteks said nad teadmise, et ülesannete tüüpe programmeerimisega seotult uuritakse ning küsitluse täitjad said loodetavasti inspiratsiooni

juba vastuseid andes. Küsimustikus (vt lisa 5) olid nii avatud kui ka valikvastusega küsimused, seega saab teha nii kvalitatiivset kui ka kvantitatiivset analüüsi.

3.3.1 Küsitluse valim

Küsitluses osalejate leidmiseks kasutati Eesti Hariduse Infosüsteemi (lühendatult EHIS) õppeasutuste otsingut[5], kus määrati tegutsemise vormiks „üldhariduse alla kuuluv gümnaasium“. Nii saadi kätte kõigi Eesti gümnaasiumide kodulehtede aadressid, üksikutel juhtudel tuli kooli nime järgi guugeldades uus veebiaadress leida. Kodulehelt otsiti üles õppekava, ainekavad ja –passid ning õppesuundade ja valikkursuste kirjeldused, et teha kindlaks, kas koolis pakutakse kursust, milles õpetatakse programmeerimist. Sellekohane viide leiti 65 gümnaasiumi kohta. Nende koolide veebilehelt võeti õpetajate kontaktide hulgast informaatika, arvutiõpetuse ja programmeerimise õpetaja kontakt(id), üksikutel juhtudel ka IT-juhi ja/või haridustehnoloogi meiliaadress. Konkreetse isikuga seotud meiliaadresse koguti 89, nelja kooli puhul ei olnud õpetajate nimekirjas märgitud eelnimetatud ametinimetusi või meiliaadresse ja seetõttu võeti nende koolide puhul üldine meiliaadress. Selline valimisse võtmise viis oli kiirem ja efektiivsem, kui näiteks EHISe päring, sest kursuste nimed ja sisud võivad olla väga erinevad. Samas on arusaadav, et tausta uurimata kõikidele koolidele pole mõtet infot saata, eriti kui sooviti, et vastaksid just programmeerimist õpetavad inimesed.

Küsitlus viidi läbi Tartu Ülikooli LimeSurvey keskkonnas. Seal oli võimalik lisada uuringu valimis osalevate inimeste nimed ja meiliaadressid ning tänu sellele oli mugav saata osalejatele nimelised kutsed, mis personaliseerituse tõttu võiks innustada vastama. Samuti sai korduskutse saata vaid neile, kes polnud veel küsitlusele vastanud või polnud keeldunud osalemast. Korduskutse juurde märgiti ka palve, et kui inimene tegelikult ei kuulu uuringu sihtrühma ehk ei õpeta programmeerimise aluseid Pythonis, siis anda sellest teada kutsest keeldumise lingile vajutamisega. Kuigi vastamine oli anonüümne, on keskkonnas konkreetsete vastustega seostamatult näha, kes on ankeedi lõpuni täitnud või keeldunud, seega saab teha järeldusi ka selle kohta, millistest koolidest vastajad olid.

Küsimustikule reageeriti nii saadetud linkide kaudu kui ka meili teel. Meili teel anti näiteks teada, kes nende koolis tegelikult uuritavat kursust õpetab või toodi välja, et õpilaste vähese

[5] Eesti Hariduse Infosüsteemi koolide otsingu veebileht: https://enda.ehis.ee/avalik/avalik/oppeasutus/OppeasutusOtsi.faces.

huvi tõttu on kursust õpetatud vaid üks kord. Vastukaja oli üldiselt positiivne, tehtavast magistritööst olid huvitunud ka need, kellel ise pole olnud veel võimalust seda kursust õpetada. Kokku keeldus uuringus osalemisest 21 õpetajat, kusjuures mõni oli enda koolis märgitud ainsaks õpetajaks, kes seda kursust välja toodud ametinimetuste poolest võiks õpetada. Kirja teel tulnud täpsustuste põhjal võib keeldumise põhjusi olla kolm: õpetaja ei soovi uuringus osaleda, õpilaste vähese huvi tõttu ei ole olnud võimalust väljapakutud kursust läbi viia või õpetati täpselt peatükis 1.1 tutvustatud näidiskursuse materjalidega ja seetõttu ei pidanud õpetaja omapoolseid andmeid väärtuslikuks. Alustatud küsimustikke oli 45, neist 29 ankeeti oli täidetud poolikult, lõpuni täidetud ankeete laekus 16, vastajad olid 15 eri koolist.

3.3.2 Küsimustiku ülesehitus

Küsitlus koosnes kahest plokist. Esimeses küsiti, kust leitakse inspiratsiooni hindeliste tööde ülesehituseks, milline on kokkupuude õppe- ja kasvatustöö eesmärkide liigitustega ning kuivõrd teadlikult jälgitakse, et hindelistes töödes oleksid küsimused erinevatelt õppeprotsessi kognitiivsetelt tasemetelt. Need aspektid on olulised, sest ajakirjanduses on läbi käinud, et informaatikaõpetajatest on suur puudus ning seetõttu on koolides õpetamas inimesi, kellel puudub vastav kvalifikatsioon („Lõunaeestlane“, 2020). See omakorda võib mõjutada õppeprotsessi vastavust õpetamise teoreetilistele suundadele.

Teises osas paluti vastata, milliseid ülesannete tüüpe hindelistes töödes kasutatakse ja kas need on avatud või valikvastustega küsimused. Samuti paluti kommenteerida etteantud ülesannete tüüpe ning pakuti võimalust tutvustada enda koostatud hindelisi töid. Vastamiseks antud ülesannete tüübid valiti peatükis 2.2 tutvustatud maatrikstaksonoomia järgi koostatud ülesannete tüüpide hulgast. Juhendmaterjalis (tutvustatud peatükis 3.1) pakuti välja 38 erinevat ülesannete tüüpi, mis koormavad rohkuse tõttu ebavajalikult küsimustiku täitjat, lisaks on nende hulgas omavahel väga analoogilisi tüüpe. Küsimustikku valiti 28 ülesannete tüüpi, mõnel juhul algseid teis(t)ega ühendamisel ümber sõnastades. Iga tüübi juurde oli võimalik lisalingilt vaadata 1−3 näiteülesannet. Küsimustiku täitjale lingiga antud näiteülesanded leiab töö lisast 6. Täitmise mugavust silmas pidades jaotati ülesannete tüübid nelja rühma, millest esimese tüübid võiks kokku võtta märksõnadega „teadmine ja primitiivsem mõistmine“, teises rühmas „rakendamine ja analüüsimine“, kolmandas „sünteesimine ja loomingulisus“ ning neljandas rühmas olevad tüübid koonduvad märksõnade „hindamine ja parandamine“ alla.

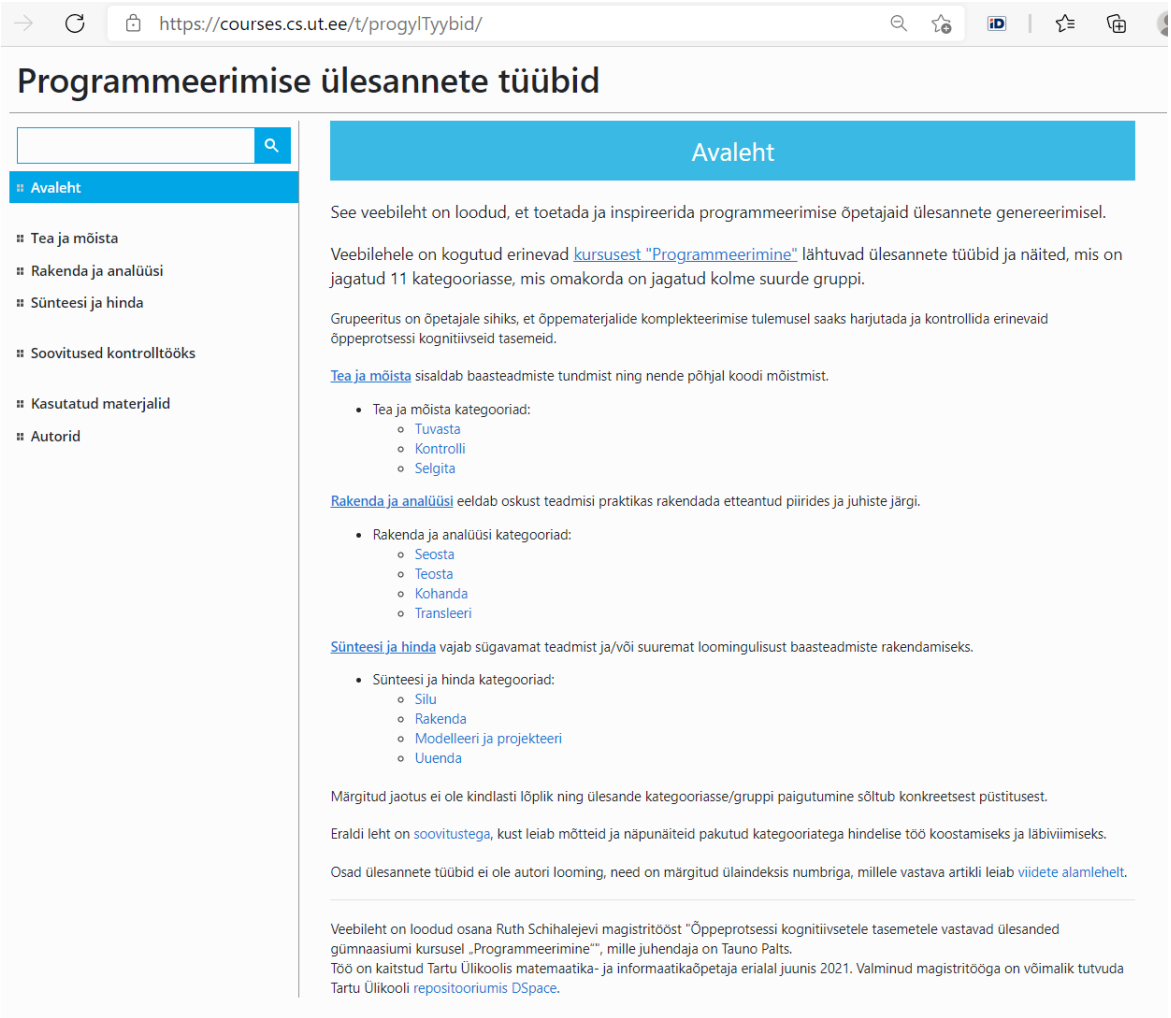
4. Tulemused

Metoodikast lähtuvalt jagunevad ka tulemused kolme põhiossa. Maatrikstaksonoomia tõlkest, kogutud ülesannete tüüpidest ja näiteülesannetest koostati kasutajasõbralikkuse eesmärgil veebileht. Gümnaasiumis tehtud uuringu tulemuste põhjal saab anda mitmeid soovitusi järgmiste tööde koostamiseks ja mõtteid edasiseks arenduseks-uurimiseks. Õpetajate hulgas läbiviidud uuringu tulemusi analüüsides keskendutakse eelkõige püstitatud uurimisküsimustele vastamisele.

4.1 Juhendmaterjal veebilehena

Peatükis 3.1 kirjeldatud maatrikstaksonoomia tõlge ja täiendatud materjal õpetajale on leitav veebiaadressilt https://courses.cs.ut.ee/t/progylTyybid/. Materjal vormistati just veebilehena, et see oleks kättesaadav ja sisu mugavalt jälgitav. Liigendamiseks kasutati courses.cs.ut.ee lehe funktsionaalsusi, et jagada infot alamlehekülgedele ja ühel lehel kõrvalist infot soovi järgi peita ja nähtavaks teha. Veebilehel liikumiseks on vasakul servas menüü ning ka alamlehtedel on võimalik vajutada tegevuste nimedele, millelt viib link vastavale lehele.

Avalehele (vt joonis 4) lisati üldine teave materjali kohta ning viide magistritööle, et oleks võimalus uurida avalikustatud töö tagamaid. Liigendamiseks jaotati maatrikstaksonoomias olevad 13 tegevust võimalikult võrdselt kolme gruppi. Neid gruppe eristab omavahel see, mida ülesannete lahendamiseks peab oskama – vastav kirjeldus on iga grupi nime juures. Gruppide nimedeks võeti inspiratsiooni Bloomi taksonoomia tasemetest, kuid ei olegi mõeldud, et kõikidesse gruppidesse valitud tegevused ka algses Bloomi taksonoomias seal paikneksid. Seega jaotamisel lähtuti kirjeldusest, mitte nimest, nimed on vajalikud, et vasakul asuvas menüüs gruppe eristada.

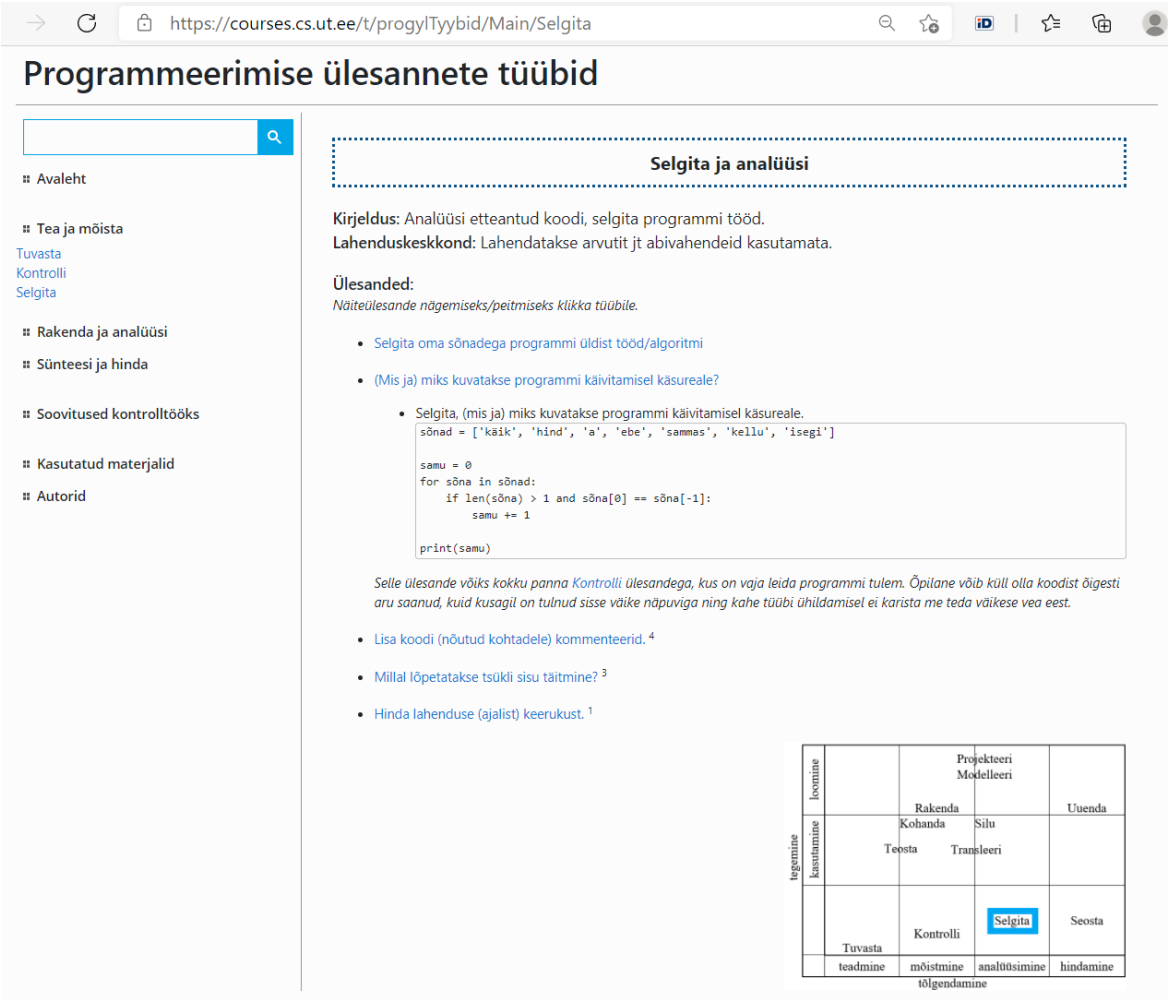


Joonis 4. Kuvatõmmis juhendmaterjaliga veebilehe avalehest.

Veebilehel eristatakse sõnu „grupp“ ja „kategooria“ – grupi tähendust selgitati eelmises lõigus, kategooriate all (siin ja edaspidi) mõeldakse maatrikstaksonoomias nimetatud tegevusi. Iga grupi nimele vasakul menüüs vajutades avaneb väike selgitus (vt näidet joonisel 5), mis võtab kokku kategooria tegevused. Siit viivad lingid iga selles grupis oleva kategooria alamlehele.

Joonis 5. Kuvatõmmis konkreetset gruppi tutvustavast alamlehest.

Iga kategooria jaoks on tehtud eraldi alamlehekülg (vt näidet joonisel 6), millel on toodud kategooria nimi, kirjeldus, soovituslik lahenduskeskkond, ülesannete tüübid, asetus maatriksis ning vahel ka lisamärkused või –soovitused. Nimi ja kirjeldus on saadud peatükis 3.1 tutvustatud meetodil. Lahenduskeskkonna all on kaks valikut: lahendatakse abivahendeid kasutamata või arenduskeskkonda kasutades. Loomulikult jääb lõplik otsus abimaterjalide kasutamise kohta materjali kasutajale ehk õpetajale, kuid oleme andnud omapoolse soovituse, vajadusel seda ka konkreetse ülesande juures põhjendades. Ülesande tüüpide peale klõpsates avanevad ja peituvad näiteülesanded ja üksikutel juhtudel ka konkreetselt selle tüübi juurde kuuluvad soovitused, mis lähtuvad peatükis 3.2 tutvustatavast uuringust. Lisatud joonis maatrikstaksonoomias paiknemise kohta annab kasutajale aimduse, kuidas võiks kategooria suhestuda teistega.



Joonis 6. Kuvatõmmis konkreetset kategooriat tutvustavast alamlehest, avatud on üks näiteülesanne.

Veebis olev juhendmaterjal on kohandatud gümnaasiumi kursusele „Programmeerimine“. Selle käigus pandi kokku algses maatriksis eraldi olevad kategooriad „Analüüsi“ ja „Selgita“, sest esimene tähendab programmi ajalise keerukuse hindamist ja teine programmi töö kirjeldamist. Kuna ajalise keerukuse hindamine ei ole kursuse gümnaasiumiastmes ette nähtud oskus, siis ei olnud otstarbekas paigutada seda eraldi, kaalumisel oli ka „Analüüsi“ välja jätmine. „Analüüsi“ ja „Selgita“ on algses maatriksis samas kastis ning tutvustavas artiklis (Fuller et al., 2007, p. 165) toodud mõte, et nimetused võivad liikuda olenevalt konkreetsele situatsioonile nii vertikaal- kui ka horisontaalsuunaliselt, andis julguse kaks ühendada. Teine kohendus, mis veebilehel võrreldes algse maatriksiga tehti, oli see, et ühendati „Modelleeri“ ja „Projekteeri“. Selle põhjuseks oli, et kursusel „Programmeerimine“ pannakse suuremat rõhku programmeerimiskeele omandamisele, mitte niivõrd ise programmi kujundamisele. Kuna see on aga arendatav oskus gümnaasiumis

õpetamiseks mõeldud jätkukursustel, jäeti mõlemate tegevuste nimed ka kategooria nimetusse alles.

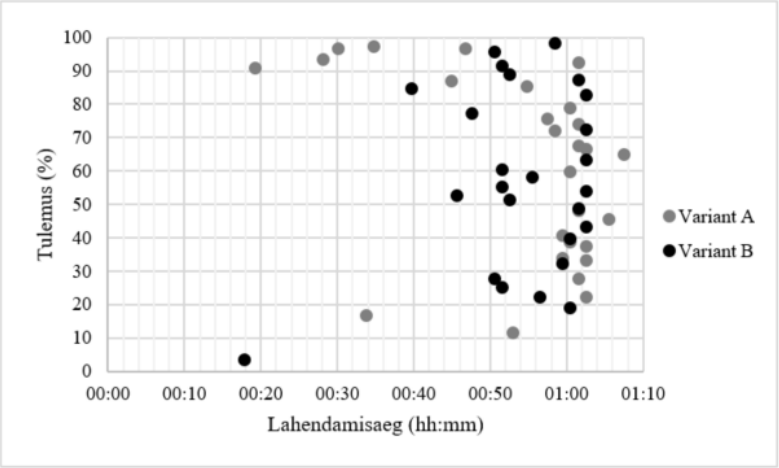
Lisaks on veebilehel kolm alamlehte: „Soovitused kontrolltööks“, „Kasutatud materjalid“ ja „Autorid“. Soovituste all (vt joonis 7) tuuakse välja siin töös tutvustatud gümnaasiumis tehtud uuringu analüüsidest ja tähelepanekutest tehtud järeldused, mida peaks arvestama pakutud kategooriatega hindelise töö koostamisel ja läbiviimisel. Soovituste kohta saab lugeda ka peatükist 4.2.2. Kategooriate juurde, mida ei ole autor enne magistritöö tegemist välja mõelnud ja õpetajatöös kasutanud, on lisatud ülaindeksiga viitenumbrid, mille täiskirjed leiab alamlehelt „Kasutatud materjalid“. Alamleht „Autorid“ on oluline eraldi välja tuua selleks, kui keegi soovib täpsemalt uurida materjali tausta või anda sellele tagasisidet.

Joonis 7. Kuvatõmmis soovitustega alamlehest.

See materjal on loodud kursusele „Programmeerimine“, kuid on autori hinnangul kasutatav ka teistel programmeerimist õpetavatel kursustel. Näiteks peatükis 1.1 mainitud kursusel „Tarkvaraarendus“ eeldatakse samuti veel mõningate koodi kirjutamise algtõdede õpetamist. Kursustel, kus tegeletakse tarkvaraprojekti koostamisega, sh justmainitud „Tarkvaraarendus“, saaks rakendada pea kõiki ülesannete tüüpe, mis on veebilehel välja toodud. Erinevus kursusega „Programmeerimine“ seisneks siis aga selles, et õpilane peab ühe projekti koostamisel läbi tegema kõik ülesannete tüübid ning erinevate kategooriate tulemused ei ole üksteisest lahutatavad vaid mõjutavad järgmisi. Teiste õppeainete kursustel ei pruugi väljapakutud ülesandetüüpide kasutamine suure tõenäosusega õnnestuda, sest mõeldud on väga konkreetselt programmeerimisele.

4.2 Näidistöö tulemuste analüüs

Kontrollimaks eelmises peatükis kirjeldatud ülesannete tüüpide ja näiteülesannete vastavust algse maatriksi asetusele, viidi ühes Eesti gümnaasiumis kontrolltöö vormis läbi uuring. Koondtulemused on üsna kehvad, variandi A keskmine resultaat on 61% ja variandi B puhul 57%. Tulemused ja lahendamiseks kulunud ajad on näha joonisel 8.



Joonis 8. Kontrolltöö tulemused sõltuvalt lahendamisajast.

Kuna variandi A sooritamisel esitati enamik töid etteantud aja viimastel sekunditel, otsustati teises soorituskatses võtta üks ülesanne vähemaks – kategooria „Seosta“ paigutus nüüd esimeste loositavate ülesannete hulka. Kuigi viimastel minutitel esitatud töid oli vähem, ei andnud ülesannete koguarvu vähendamine sooritustulemustele oodatud positiivset mõju.

Analüüsides olukorda ja kogudes õpilaste arvamusi, jäi kõlama mitu töö sooritust mõjutavat tegurit. Kõik tegurid olid põhjustatud pandeemiast tingitud ebakindlast olukorrast nii pikas kui ka lühikeses perspektiivis.

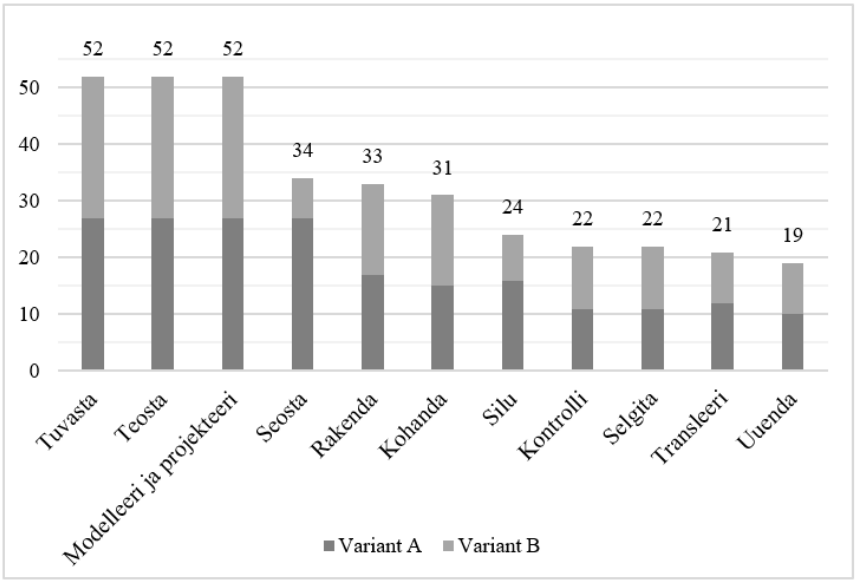
- Distantsõpe – õpilased ei olnud veel ühekski arvutiõpetuse kontrolltööks pidanud nii suures mahus õppima ja kordama veebitundides ning ka kontrolltöö sooritamine veebi teel oli neile informaatika tunnis võõras, sest erinevalt teistest ainetest pidi käima panema nii kaamera kui ka ekraanisalvestuse.
- Tehnilised probleemid – kuigi süsteemi katsetati ka viimases kordamistunnis, kimbutasid kontrolltöö ajal ikkagi mitmed tehnilised probleemid, millega erinevalt klassiruumist pidi seekord ilma rahustava õpetaja toeta üksi jagu saama.
- Valmistumise ajagraafik – mõjutada võis ka asjaolu, et kuigi suuliselt oli öeldud kontrolltöö sooritamise kuupäevadeks 3. ja 5. märts, oli veel vastavalt 2 või 4 päeva enne töö sooritamist e-päevikusse Stuudium märgitud eksitavalt 9. ja 10. märts.

- Kordamisülesannete varieeruvus – varem kordamist alustanud õpilased tundsid puudust rohkematest ülesannetest, just paberosas.

Hindamisel paistis välja ka see, et õpilased ei olnud korralikult lugenud ülesannetes küsitavat. Näiteks variandi B ülesandes „Rakenda“ oli vaja lünka kirjutada täpselt tööjuhises kirjeldatud tingimus, kuid esitatud vastus seda ei edasta. Ekraanivideoid vaadates on aga väga selgelt näha, kuidas õpilane kerib ülesande tööjuhisest üsna kiirelt edasi ning mõtlemisajal on ees ainult etteantud kood. „Kontrolli ja selgita“ ülesannete juures olid mitmed jätnud vastamata küsimusele „Miks“ ja seetõttu kaotasid väga palju punkte. Koodi parandamise puhul („Silu“) oli jäetud märkimata tööjuhises nõutud parandatava rea number, mis on oluline mainida, sest pakutud koodilõigu järgi ei pruugi olla aru saada, mida täpselt parandati.

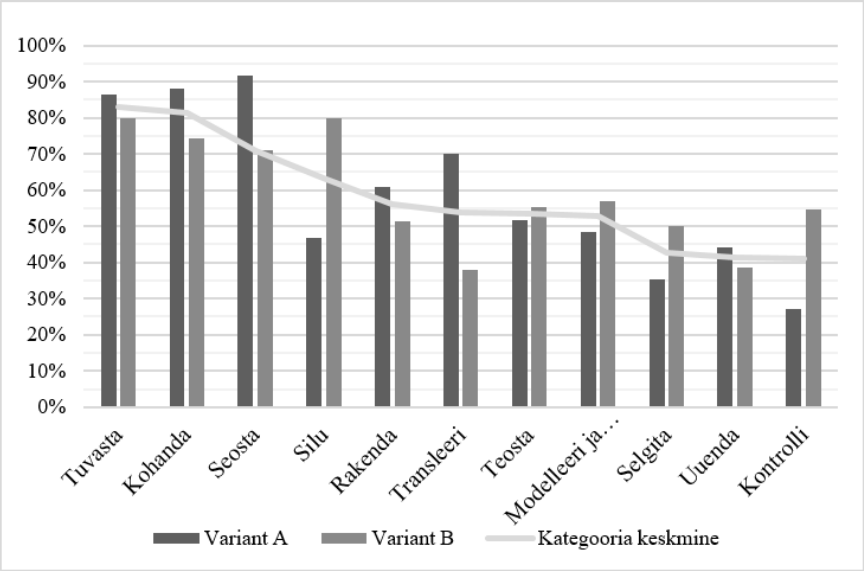
4.2.1 Iga kategooria tulemuste analüüs

Nagu juba eelnevalt mainitud, siis igale õpilasele osad ülesanded loositi teatud valikute hulgast (vt kontrolltöid lisas 3 ja 4) juhuslikult, seega ei lahendatud iga ülesannet võrdne arv kordi. Lahendajate arvud on toodud joonisel 9. Kategooriad „Tuvasta“, „Teosta“ ja „Modelleeri ja projekteeri“ olid kohustuslikud mõlema variandi kõigile õpilastele, seega lahendas neid 52 õpilast. „Seosta“ oli kohustuslik ülesanne variandi A lahendajatele ning ülesannete arvestuses lahendas kõige vähem (7 õpilast) variandi B „Seosta“ ülesannet. Mõlema variandi peale kokku lahendati kõige harvem kategooriat „Uuenda“, 19 korda.



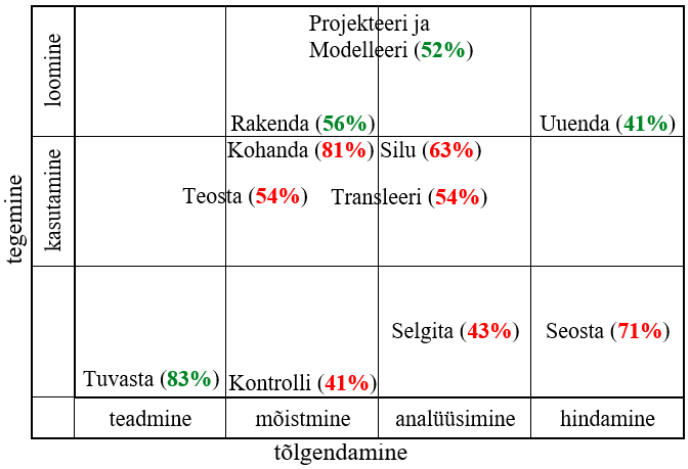
Joonis 9. Iga ülesande lahendajate arv.

Vaatamata kehvadele koondtulemustele joonistuvad välja kategooriad, mille ülesanded olid õpilastele kergemad ja raskemad (vt joonis 10). Kategooriate raskusastmete võrdlemiseks anti mõlemas variandis lahendamiseks üks ülesanne, kusjuures ühe kategooria kahe ülesande kontrollitavad teadmised olid erinevad. Kuna maatriksis ei paigutu ülesanded nii üheselt järjekorda, võrreldakse edaspidi iga kategooriat eraldi teda ümbritsevatega.



Joonis 10. Kategooriate keskmised tulemused protsentides.

Maatriksis oleva paigutuse järgi (vt joonis 11) on õige raskusastmega ülesanded tehtud kategooriatele „Tuvasta“, „Rakenda“, „Modelleeri ja projekteeri“ ja „Uuenda“. Nimetatute puhul peab paika, et nendest paremal ja üleval olevate kategooriate tulemused on kehvemad ning vasakul ja allpool olevate ülesannete lahendused edukamad. Teiste kategooriate puhul need seaduspärad ei kehti ning seetõttu analüüsitakse neid igaüht põhjalikumalt.



Joonis 11. Kategooriad koos keskmiste tulemustega (protsentides) asetatuna algsesse maatriksisse.

„Kontrolli“ peaks olema maatriksis asetuse järgi lihtsam ülesanne kui „Transleeri“ ja „Selgita“, kuid on selgitamisest veidi ja tõlkimisest tunduvalt kehvema tulemusega. Kontrolltöö variantides oli kokku pandud „Kontrolli“ ja „Selgita“ ülesanded, kus ette oli antud programmi kood, mille puhul oli vaja nimetada programmi tulem ja selgitada selle põhjust. Seetõttu on mõistetav, et nende kategooriate tulemused on üsna sarnased ning lahendustes esines olukordi, kus õpilane oli selgitanud korrektselt, kuid programmi tulemi leidmisel oli jäänud mõni tingimustele vastav element loendamata. Tõlkimise ülesandeks oli kirjutada plokkskeemi järgi kood. Siinkohal oli vaja õpilasel aru saada kirjeldusest ning kirjutada iga kujundi järgi üks koodirida. Analoogiliselt „Kontrolli“ ja „Selgita“ erinevuste põhjendusele, on ka siin mõjutajaks asjaolu, et „Transleeri“ abil saab õpilane oma teadmisi laiemalt avada ja hindaja ka osaliselt õige vastuse eest punkte anda, kuid „Kontrolli“ puhul on tegemist põhimõtteliselt õige-vale küsimusega, kus osaliselt õiget vastust ei esine.

„Selgita“ juures on samuti problemaatiline olukord kahe kategooriaga, „Kontrolli“ ja „**Seosta**“. Viimase puhul tuleb võtta arvesse, et variandis A oli kõigile kohustuslikus ülesandes vaja lisada koodi taanded, kusjuures vastust oli võimalik kontrollida arvuti abil. Variandis B lahendas sama kategooria ülesannet tehete järjekorda paberile märkides 7 õpilast. Seega on kõrge tulemus suure tõenäosusega tingitud asjaolust, et enamusel oli ülesannet võimalik tänu arvuti abile kerge vaevaga lahendada katseeksituse meetodil. „Selgita“ ja „Kontrolli“ vaheline ebakõla kirjeldati lahti eelmises lõigus.

Kategooria „**Teosta**“ keskpärane tulemus oli üllatuslik, sest just sellist tüüpi ülesandeid olid õpilased varasemalt kõige rohkem lahendanud. Maatriksis asetuse suhtes on problemaatiline varasemalt mainitud võrdlus kategooriaga „Kontrolli“ ning asjaolu, et „Kohanda“ tulemus on madalamal olevast palju parem. Viimase puhul võib mõjutav asjaolu olla see, et „**Kohanda**“ ülesandes oli vaja sisuliselt teha palju vähem, kui „Teosta“ puhul. „Teosta“ juures oli vaja kirjutada terve programm, üle kümne koodirea, kuid „Kohanda“ eeldas vaid teatud aspekti muutmist. Kui võtta ka arvesse eespool kirjeldatud olukord, et õpilased ei lugenud korralikult töö käske, siis ise programmi kirjutamiseks oli etteantud üsna pikk tööjuhend tingimustega, mida tuleb järgida. Samas on etteantud koodi muutmiseks vaja tööjuhises tähele panna vaid märksõna, mis peab lahenduses esinema. „Teosta“ ja „Kontrolli“ ebakõla on selgitatud eespool „Kontrolli“ lõigus.

„**Transleeri**“ kohta on juba eelnevalt selgitatud vastuolusid kategooriatega „Kontrolli“ ja „Selgita“. Kategooriaga „Teosta“ on ümardades saadud keskmine tulemus võrdväärne, kuid tagaplaanil on „Transleeri“ tulemus 0,4% „Teosta“ omast kõrgem, seega on maatriksis

selles osas kõik paigas. Külla aga on problemaatiline, et „**Silu**“ tulemus on temast madalamal olevast 9% kõrgem. Nagu eelmises lõigus kirjeldatud, võib ka siin olla mõjutajaks asjaolu, et „Silu“ eeldas koodist aru saamist ning ühe koodirea muutmist, kuid „Transleeri“ taaskord täpse juhise järgi pikema koodi kirjutamist.

4.2.2 Näidistöö arutelu

Tööde läbiviimise, tagasisidestamise ja analüüsimise käigus tulid välja mõningad aspektid, mida arvestada kontrolltöö koostamise ja läbiviimise protsessis. Läbiviidud tööde põhjal järelduste üldistamist segavad aga mõned piirangud.

Soovitused kordamiseks, töö korralduseks ja küsimuste sõnastamiseks

Peatükis 4.1 tutvustati, et juhendmaterjalist koostatud veebilehele lisati ka mõned soovitused kontrolltöö koostamiseks ja läbiviimiseks. Need soovitused lähtuvad näidistöö korraldamisest ja tulemustest. Kuna veebilehel ei ole varem mainitud, et töö võiks olla kaheosaline, paber- ja arvutiosast koosnev, siis see ongi toodud koos lühikese põhjendusega esimeseks soovituseks.

Kuigi töö koosneb kahest osast ja osade sooritamine on ühesuunaline, on oluline, et õpilane saaks teadlikult planeerida kahele osale jagunevat aega. Näidiskontrolltöö läbiviimisel tunnistasid mitmed sooritajad, et kiirustasid paberosa ülesannete esitamisega, sest ei teadnud, mis neid arvutiosas ootab.

Hindelistes töödes ei piirduta tavaliselt ühe teemaga, mistõttu kõigis teemades kõigi taksonoomia tasemete kontrollimine teeb töö mahu väga suureks. Seega peab kindlasti tegema valikuid, kuid võimalusel võiks siiski silmas pidada, et ülesanded oleksid erinevate raskusastmetega. Tehtud töö puhul oli iga ülesanne erinevalt raskustasemelt ning pea igas ülesandes kontrolliti erinevat teadmist. Siiski võib argumenteerida, kas ülesannete koguarv liiga suur ei olnud. Terry Scott (2003, p. 271) on oma uurimuses välja toonud, et ühes ülesandes mitme kategooria kontrollimine võimaldab ka väiksema ülesannete koguarvuga rohkemaid tasemeid kontrollida. Ühes ülesandes mitme kategooria kontrollimisel oleks aga oluline jälgida, et kaks kategooriat ei sõltuks üksteisest. See aitab vältida olukorda, et kus õpilane lahendab ühe kategooria valesti ning seetõttu karistatakse teda automaatselt ka teises kategoorias eksimise eest.

Hindamise juures on veel kaks olulist aspekti. Esiteks peab hindamissüsteem olema ülesannete lõikes võrdne. Punktisüsteemi koostamisel ei tohi unustada, et sageli peab

vastuse andmiseks tegema ka taustal analüüse. Näiteks kategooria „Kontrolli“ jaoks programmi tulemi leidmiseks peab õpilane enne aru saama, kuidas programm üldse toimib, ehk tegema läbi ka kategooria „Selgita“. Lisaks on oluline, et õpilased oleksid teadlikud, kui põhjalikult on vaja küsimustele vastata, sest õpilasel peab ülesannete iseseisvaks edukaks lahendamiseks olema varasem kogemus (Selby, 2015, p. 85). Näiteks küsimus „Kuidas toimib funktsioon …?“ ei pruugi õpilasele ilma varasemat tähelepanu pööramata viidata, et tuleb vastata ka programmeerimise seisukohalt olulistele kõrvalaspektidele, näiteks järjendi funktsioonide puhul ka sellele, kas funktsioon tagastab vastuse või muudab algset järjendit.

Piirangud ja edasiarendusvõimalused

Gümnaasiumis tehtud uuringu puhul on mitmed piirangud, mis mõjutavad saadud tulemusi. Kõige suuremaks teguriks on uuringu ajaline kestus ja seega õpilaste ettevalmistus. Käesolev uuring vältas koolis 3 nädalat, mille jooksul tutvustati ja lahendati uusi ülesannete tüüpe ning sooritati kontrolltöö. Kuigi õpilased olid mitmete ülesandetüüpidega tutvunud varem, näiteks Exceli funktsioonide õppimisel, ei olnud nad Pythoni teadmisi enamikes sellistes ülesannetes enne kordamistundi üldiselt kasutanud. Tulemusi mõjutas ka asjaolu, et õpilased ei pidanud oma iseseisvat valmistumist piisavaks, kuigi seda eeldasid uuringu läbiviijad töö hindelist vormi arvestades.

Piiranguna võib välja tuua ka aspekti, et puudus kontrollrühm. Sellel on aga eetiline põhjendus, sest ühe klassi eri rühmadel ja paralleelklassidel peaksid olema võimalikult võrdsed tingimused samaks tööks valmistumiseks ja selle sooritamiseks. Edaspidi võiks sellise ülesehitusega hindelist tööd läbi viia ja uurida ka teistes koolides, toetades sellega muuhulgas õpetaja üleminekut kohandatud kontrolltöö vormi juhendamisele.

Uuringust selgub, et kategooriate asetus maatriksis sõltub konkreetsest ülesande püstitusest ja sellest, kuidas on see seotud varasemalt harjutatud ülesannetega. Seda tõid välja ka varem tehtud uuringud (Thompson et al., 2008, p. 156) ja maatrikstaksonoomia loojad (Fuller et al., 2007, p. 165), seega ei ole tulemuste mittevastavus algsele asetusele probleem, vaid kinnitus ja julgustus teistele kasutajatele. Tulemused näitavad siiski selgelt ära õpilaste probleemsed kohad ja teadmised, millele on vaja edasistes õpingutes rohkem tähelepanu pöörata, mis ongi eri tasemega ülesannete andmise üks eesmärke.

Kindlasti mõjutas tulemusi ka hindamissüsteem, just võrdluses erinevatele kategooriatele omistatud kaaludega. Edaspidi võiks uurida, kuidas kategooriaid ühte töösse võtta, näiteks,

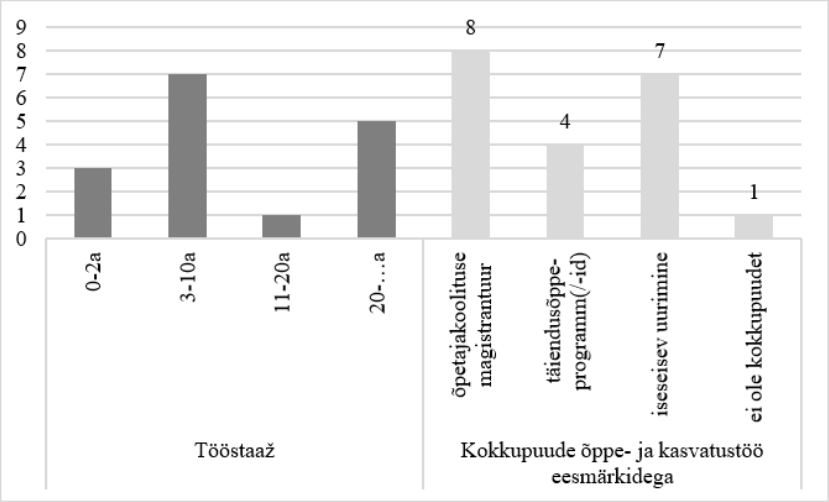
kas oleks vajalik tasakaalustada ühte „Uuenda“ kategooria ülesannet mitme „Tuvasta“ ülesandega. Samuti, kuidas kujundada selliseks tööks hindamissüsteem nii, et eri õpistiiliga õppijatel oleks võrdne võimalus töö edukalt sooritada ja kuivõrd on see üldse oluline. Kõrvuti hindamissüsteemiga oleks oluline pöörata tähelepanu ka ühe ülesande tüübi erinevatele näiteülesannetele – kui palju mõjutab ülesande raskusastet näiteks etteantava koodi pikkus, vihjete hulk ja küsimuse sõnastus. Kõiki eelmainitud aspekte arvestades võiks välja töötada näidiseks arvestus- ja kontrolltööd, samas pöörates tähelepanu ka Eesti gümnaasiumides sagedamini kasutatavatele sooritusaegadele, näiteks kontrolltöö 45- ja 75-minutilise tunni jaoks ning pikema sooritusajaga arvestustöö.

4.3 Küsitluse tulemused

Õpetajate hulgas läbiviidud küsitluse eesmärk oli teada saada, kust saadakse hindelisteks töödeks inspiratsiooni ja milliseid ülesannete tüüpe kasutatakse. Seejuures ei ole väheoluline nende varasem kokkupuude õppe- ja kasvatustöö eesmärkide liigitamisega.

4.3.1 Vastajate taust seoses õpetamise ja selle teooriaga

Joonisel 12 näeb vastajate tööstaaže ja kokkupuudet õppe- ja kasvatustöö eesmärkidega. Vastajate hulgas oli tööstaaži poolest nii programmeerimise õpetamist alles alustanud kui ka juba pika õpetamiskogemusega inimesi, rohkem oli siiski kogenud õpetajaid. Kuueteistkümnest kaheksa õpetajat oli läbinud või läbis õpetajakoolituse magistrantuuri, kokku vähemalt 60 EAPd, ning neli oli läbinud õppe- ja kasvatustöö eesmärkide kohta täiendusprogrammi(/-e). Vaid üks vastaja ei olnud eesmärkide liigitustega varem kokku puutunud, iseseisvalt oli teemat uurinud seitse vastajat, kellest kolm ei olnud sellega mujal kokku puutunud.



Joonis 12. Vastajate jaotus tööstaaži ja teoreetilise tausta poolest.

Avatud vastusega küsimusele, kuivõrd jälgitakse hindeliste tööde koostamisel erinevaid õppeprotsessi kognitiivsete tasemete esindavatust, jagunesid vastused üldiselt kolmeks. Neli vastajat tunnistas, et ei jälgi ning kolm arvas, et jälgib alateadlikult. Kaheksa vastas, et jälgib ning kaks neist tõid konkreetselt ka välja, et alati on esindatud ülesanne mõistete tundmise, teadmiste rakendamise ning selgitamise kohta. Üks õpetaja märkis, et on kognitiivsete tasemete varieerimise suhtes paindlik ja lähtub konkreetsest rühmast.

4.3.2 Kasutatavad ülesannete tüübid hindelistes töödes

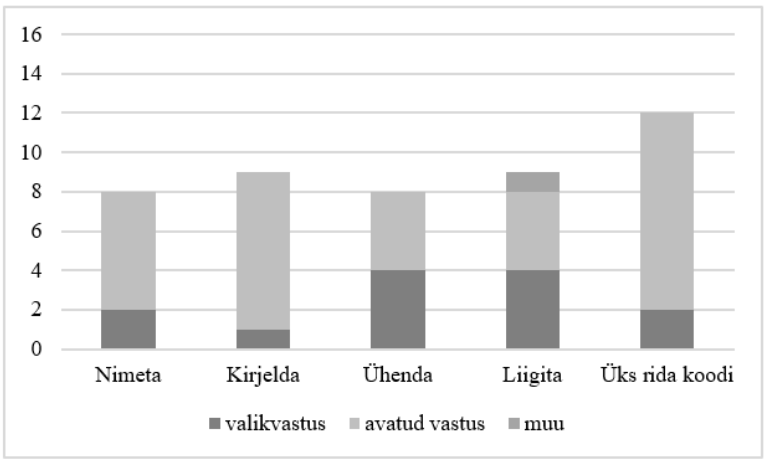
Oma inspiratsiooni allikateks tööde ülesehituse ja/või ülesannete tüüpide jaoks nimetasid vastajad ülikoolide pakutud kursusi, näiteks Tartu Ülikooli „Programmeerimine“ koos tudengitele mõeldud õpikuga[6] ja peatükis 1.2.1 tutvustatud MOOCid, mis olid algselt täiskasvanutele mõeldud. Välja toodi ka gümnaasiumi valikkursuseks mõeldud õpiku õpetajamaterjalid, mida tutvustati peatükis 1.1 ja kursuse juurde loodud ülesannete automaatkontrolli keskkond lahendus.ut.ee. Mitmed on otsinud mõtteid ka internetist, lisaks mainiti ühte raamatut ja Tallinna Tehnikaülikooli kursusi. Kusjuures ülikooli kursuste puhul ei ole vahet, mis programmeerimiskeelele algselt ülesanne mõeldud oli, vajadusel kohendatakse sõnastust Pythonile. Kuigi küsimus oli töö ülesehituse ja ülesannete tüüpide kohta, soovisid vastajad avaldada oma inspiratsiooniallikaid ka teemade suhtes.

[6] Tartu Ülikooli arvutiteaduse instituudis õpetatakse kursust „Programmeerimine“ samanimelise õpikuga, mis on leitav aadressilt https://progeopik.cs.ut.ee/.

Ülesanneteks sisu on leitud näiteks igapäevaelust ja kogemustest, õppeainetest nimetati matemaatikat ja statistikat.

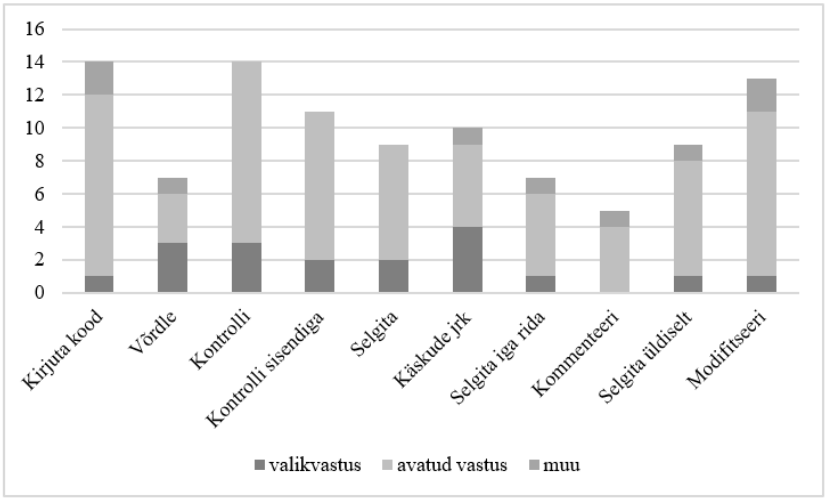
Järgmisena oli vaja vastata, milliseid pakutud ülesannete tüüpe hindelistes töödes kasutatakse ja kas pigem valik- või avatud vastusega küsimusena. Lisaks sai valida varianti „muu“ ja „ei ole kasutanud“. Võrreldes küsitluses osalejate märgitud valikvastuseid ja avatud vastuseid, näib, et variant „muu“ tekitas segadust ning seda vastati ka siis, kui tegelikult oleks oodatav valik olnud „avatud vastusega küsimus“. Näiteks, kui õpilane peab terve programmi koodi või suurema osa sellest ise kirjutama, on see avatud vastusega küsimus. Vastusevariandi „muu“ lisamisega loodeti aga tegelikult, et tuleksid välja uued küsimuste tüübid. Kahjuks see ei õnnestunud, sest valikut „muu“ kommenteeriti väga üksikutel juhtudel ning uut infot see ei andnud. Seda asjaolu tasub silmas pidada ka tulemusi vaadates.

Kõige rohkem (14 vastajat) kasutati ülesannete tüüpe, kus oli vaja kirjutada tekstilise juhendi järgi programmi kood, leida programmi väljund või etteantud koodi parandada, populaarsed (13 vastajat) olid ka plokkskeemi järgi koodi kirjutamine ja etteantud koodi modifitseerimine. Need viis tüüpi on sagedased ka Tartu Ülikooli koostatud ja korraldatud kursustel (vt ptk 1.1−1.2). Enamik vastajatest (12) märkisid veel üherealise koodi kirjutamise ja lünga täitmise kasutatavust hindelistes töödes. Ülejäänud ülesannete tüüpe analüüsitakse peatüki 3.3.2 lõpus tutvustatud rühmade kaupa eraldi ning nende kohta on ka joonised, kus on muuhulgas näha, kui palju on iga tüüpi kasutatud avatud ja valikvastusega küsimusena ning muus vormis. Joonistele on lisatud ülesande tüüpi tähistavad märksõnad ning vasakult paremale lugedes on need samas järjekorras lisas 5 toodud küsitlusega ja lisas 6 olevate näiteülesannetega.



Joonis 13. Märksõnade „teadmine ja primitiivsem mõistmine“ alla kuuluvate tüüpide kasutamine hindelistes töödes.

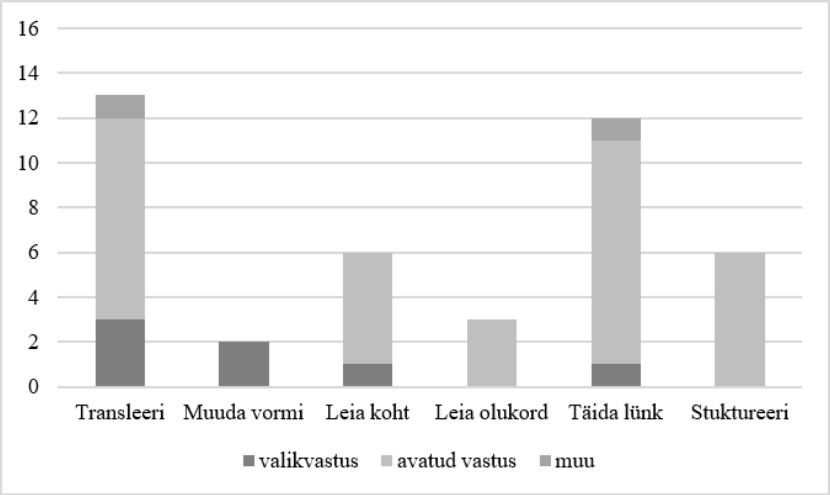
Joonisel 13 on toodud esimese rühma, mis võeti kokku märksõnadega „teadmine ja primitiivsem mõistmine“, ülesannete tüüpide kasutamine. Kõige rohkem kasutati neid nii, et õpilane sai vastata oma sõnadega. Kõiki ülesandeid kasutasid oma hindelistes töödes vähemalt pooled vastajatest. Kuna oodatult kasutati väga palju ka juhiste järgi koodi kirjutamise ülesannet (vt joonis 14, märksõna „Kirjuta kood“), oli autori jaoks üllatuslik, et nii mitmed paluvad õpilastel kirjutada ka ainult ühe rea koodi. Vastajad tõid välja, et selle rühma ülesannete tüüpe kasutatakse arvestuslikes tunnikontrollides või suulistel vastamistel. Üks õpetaja tõdes, et ei ole neid kasutanud, aga võiks.



Joonis 14. Märksõnade „rakendamine ja analüüsimine“ alla kuuluvate tüüpide kasutamine hindelistes töödes.

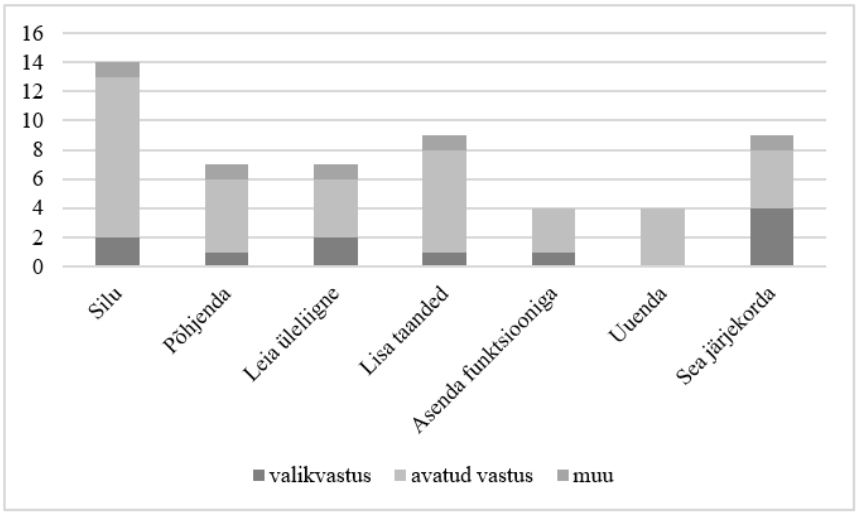
Teises rühmas, märksõnade „rakendamine ja analüüsimine“ juures, oli kõige rohkem erinevaid ülesannete tüüpe, nende tulemusi näeb joonisel 14. Siingi oli kõige populaarsem,

et ülesanne oli avatud vastusega. Oodatult oli populaarne kasutada tüüpi, kus õpilane peab leidma programmi tulemi (märksõnaga „Kontrolli“). Muret tekitab seejuures asjaolu, et mõned õpetajad ei palu õpilasel oma mõttekäiku selgitada ning seetõttu võib tekkida olukord, kus õpilane saab programmi tööst aru, aga läbimängul jääb mõni aspekt märkamata. Ootamatu oli, et juhiste järgi koodi kirjutamist ei kasuta kõik vastajad, kuigi just see on kõige tavapärasem ülesande tüüp, mida programmeerimise kursusel harjutatakse, neid leidub kõikides õpikutes, ülesannete kogudes, kursustel, näidiskontrolltöödes ja võitlusülesannetes. Kasutamisrohkuse (13 vastajat) poolest üllatas aga tüüp, mille puhul on vaja etteantud koodi uutele tingimustele vastavaks modifitseerida.



Joonis 15. Märksõnade „sünteesimine ja loomingulisus“ alla kuuluvate tüüpide kasutamine hindelistes töödes.

Kolmanda rühma, joonisel 15 toodud ja märksõnade „sünteesimine ja loomingulisus“ juures olevate tüüpide kasutajaid ja mittekasutajaid oli peaaegu võrdselt, kuid kui kasutati, siis taas avatud vastusena. Arvestades ka õpikutes olevaid näidisharjutusi, siis oligi oodata, et kõige rohkem tehakse siit hulgast plokkskeemi järgi koodi kirjutamist või selle vastupidist ülesannet. Samuti on õpikutest läbi käiv tüüp lünga täitmise ülesanne. Teised ei ole niivõrd levinud ka mujal, näiteks ei toodud selliseid näiteülesandeid üheski peatükis 2 tutvustatud uuringus. Nende ülesannete juurde kirjutas üks vastaja, et ei kasuta neid, sest ei soovi piirata õpilaste loovust. Selle rühma ülesanded on tõesti enamus ühe õige lahendusega, kuid ka siin on, eelkõige vähemkasutatute puhul, võimalik ülesanne püstitada avatumalt, mis võimaldaks õpilasel teha lahendus vastavalt oma võimetele ja loovusele.



Joonis 16. Märksõnade „hindamine ja parandamine“ alla kuuluvate tüüpide kasutamine hindelistes töödes.

Joonisel 16 on toodud viimase rühma, märksõnade „hindamine ja parandamine“ juurde paigutatud ülesannete tüüpide kasutused. Väga populaarne oli siin avatud vastusena koodi silumine ehk parandamine ning see on mõistetav, sest tegevust tuleb teha ka ise koodi kirjutades ning vastavad näiteharjutused on õpiku materjalides. Peatükis 1.1 tutvustatud õpetajamaterjalide näidiskontrolltöös on toodud üksikud ülesanded taanete lisamise kohta, seega on mõistetav ka selle tüübi sage märkimine. Siinsetegi ülesannete puhul leidis üks vastaja, et need tekitavad stereotüüpe ja piiravad lahendajate loovust. Teine vastaja leidis aga, et kui sellised ülesanded oleksid lisatud näidistestidesse, kasutaks ta neid kindlasti õppetöös.

Ükski vastaja ei osanud täiendavalt ühtki uut ülesande tüüpi lisada, toodi välja, et pakutud tüübid olid aga inspireerivad. Kui paluti anda tagasisidet pakutud ülesannete tüüpidele, tõid väga mitmed välja, et kasutavadki ainult peatükis 1.1 tutvustatud „Programmeerimise“ õpiku juurde kuuluvaid õpetajamaterjale ning lasevad õpilaste kirjutatud koode kontrollida lahendus.ut.ee keskkonna automaatkontrollil. Sedasi on õpetajal vähem tööd, pakutud tüübid vajaksid aga õpetaja suuremat süvenemist.

Etteantud ülesande tüüpide kommentaariks tõid vastajad välja, et need aitavad kontrollida, kuivõrd õpilased on teemadest aru saanud ja oskavad neid erinevates kontekstides kasutada, samal ajal ei pea selle kontrolliks lahendama suurt ülesannet. Märgiti juurde ka asjaolu, et õpilane võib kirjutada näidete varal koodi kokku, sisuliselt aspektidest aru saamata, kuid sellised ülesanded aitaksid õpilastel paremini mõista ja õpetajal mõistmist jälgida. Samuti

nimetati, et eri tüüpi ülesanded tooksid vaheldust ka õpilastele, huvitavad oleksid näiteks modifitseerimise ja parandamise ülesanded.

Pakutud ülesandeid analüüsisid vastajad ka sellest küljest, kuidas neid lahendada või küsida. Näiteks mõistete ja konstruktsioonide tundmise ülesandeid võiks lahendada üle klassi, sest neil on sisuliselt üks vastus. Lühemad ja pikemad koodi kirjutamise ülesanded on head klassis arutelude tegemiseks, sest lahendusi on erinevaid. Samuti on oluline klassis arutleda, kuidas programm töötab. Kuigi küsimustikus pakuti, et iga ülesande tüüpi võiks küsida nii, et antud on ka vastuste valikud, leidsid vastajad, et kõiki ei saagi valikvastustega küsida. Vastamiste vormid sõltuvad ka situatsioonist – näiteks harjutamiseks mõeldud automaatkontrolliga testid tehakse valikvastustega, aga kontrolltöös lastakse õpilasel oma sõnadega vastata.

Pakutud võimalusest jagada oma tööd paljud keeldusid ning mõned põhjendasid seda sellega, et kasutavad täpselt kursuse „Programmeerimine“ juurde kuuluvates õpetajamaterjalides olevat kontrolltööd. Esitatud vastustest sai aga kinnitust näiteks see, et mitmed õpetajad loovad ise teooriamaterjale, milles on rõhutatud sihtrühmale huvipakkuvaid ja vajalikke aspekte ning kõrvalised teemad hoopis välja jäetud. Lisaks tuli välja distantsõppele kohandatud kontrolltöö, kus õpilastel paluti kasutada sooritamise ajal vahendeid, mis salvestavad ekraanil toimuvat.

4.3.3 Küsitluse arutelu

Läbiviidud küsitluse juures on piiranguid, mis ei võimalda tulemusi piisavalt üldistada. Esiteks vastajate vähesus, sest lõpuni täidetud ankeete laekus 16 inimeselt 15 koolist, kuigi küsitlus saadeti laiali 65 Eesti gümnaasiumi õpetajale. Seejuures võib arvestada, et kõigis neis koolides ei pruugigi uurimisalust kursust toimuda, sest kooli kodulehelt leitud info põhjal võeti ka ainult viite „õpetatakse kursust Programmeerimine“ korral see kool valimisse. Mitmed kirja teel reageerijad tõid välja ka asjaolu, et kursust pakutakse, kuid siiani ei ole toimunud või kursust hakatakse õpetama esimest korda alles eesoleval perioodil.

Piiranguna võib välja tuua asjaolu, et viimastel aastatel on aktiivselt propageeritud ja pakutud võimalust katsetada peatükis 1.1 tutvustatud kursust, millel on olemas kõik nädalaülesanded ja näidiskontrolltööd. Paljud õpetajad kasutavadki just neid materjale, äärmisel juhul üksikuid ülesandeid ära jättes või juurde lisades ning ise nad tööde koostamisega suurt loomingulisust ei rakenda. Sellest johtuvalt võiks mitmekesistada ka

neid õpetajamaterjale selles töös välja pakutud ülesannete tüüpe kasutades, seda tõid välja ka küsitlusele vastajad.

Praegu saadud vastustest tuli välja, et teadvustatakse, et näiteks ülesandes, kus on vaja kirjutada etteantud probleemi lahendav kood, on tegelikult õpilasel vaja lahendada mitmeid eri tüüpi (ala)ülesandeid, mida oli ka siin küsimustikus välja toodud. Teine küsimus on aga selles, kas neid etappe või tasemeid siis ka hindamisel eristatakse. Üks vastaja tõi välja, et süüvib hindamisel koodi sisusse alles siis, kui programm ei anna oodatavat vastust. Vahel teevad aga õpilased nii-öelda käsitööd ja kirjutavad koodi, mis kuvabki ainult oodatud vastused ning muude juhtumite korral programm ei tööta.

Püstitatud uurimisküsimused said tulemusi analüüsides vastused. Õpetajate praegused praktikad hindeliste tööde koostamisel on mitmekesised, kuid üldiselt on siiski levinumad viis ülesannete tüüpi: plokkskeemi või tekstilise juhendi järgi koodi kirjutamine, etteantud programmi tulemi leidmine ja koodi parandamine või muutmine. Vastustest tuli välja, et töödes on kasutatakse analoogilisi ülesannete tüüpe, mida tutvustati peatükis 1.4. Seega võib järeldada, et õpetajad kasutavad hindeliste tööde koostamisel neid näiteid, millega nad tutvusid enda õpingute kestel, sealhulgas gümnaasiumi kursuste ja õpikutega iseseisva või täienduskoolitusel tutvumise käigus. Mitmed õpetajad ei pea vajalikuks koostada uusi ülesandeid, vaid kasutavadki õpiku juurde kuuluvates õpetajamaterjalides näidiseks pakutud kontrolltöid.

Kokkuvõte

Mitmed õppejõud ja teadlased on analüüsinud testides olevaid ülesandeid, et need oleksid kooskõlas erinevate õppeprotsessi kognitiivsete tasemetega. Kuna on leitud, et olemasolevad taksonoomiad ei toeta täielikult kõigi programmeerimise õppimise õpiteede esindajaid, on maatrikstaksonoomia näol arendatud arvutiteaduse-spetsiifiline taksonoomia. Selle magistritöö eesmärk oli lähtudes nimetatud maatrikstaksonoomiast koondada kokku õppeprotsessi kognitiivsetele tasemetele vastavad ülesannete tüübid, mis oleksid kohandatavad ka Pythoniga programmeerimise alusteadmiste kursusele, ning uurida, millised on tegevõpetajate harjumused diferentseerida ülesandeid hindelistes töödes.

Töös analüüsiti, millised on viimasel kümnel aastal Eestis levinud Pythoni alusteadmiste kursused ja milliseid ülesannete tüüpe leidub nende juurde kuuluvates hindeliste tööde näidistes. Leiti, et enamus toimunud ja toimuvaid kursusi on loodud Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühma poolt ning nende kursuste näidistööd on samasugused. Kursuse lõpus toimuv hindeline töö koosneb paberosas ilma kirjalikku selgitust andmata koodi tulemi leidmisest ja arvutiosas juhendi järgi koodi kirjutamisest.

Kõrvuti sellega toodi välja, milliste tulemusteni on jõudnud erinevad uuringud, mis käsitlevad Bloomi taksonoomia õppeprotsessi kognitiivsete tasemete rakendamist programmeerimisega seotud testides. Enamikes töödes leiti, et tasemete varieeruvuse saavutamine on keerukas ja konkreetsete ülesannete kategoriseerimist mõjutavad mitmed tegurid. Samas toetab erinevate raskusastmetega testi koostamine õpetajal õppija toetamist, sest selle abil on selgemalt näha, kuhu on õppur kinni jäänud.

Kuna kõigis leitud töödes analüüsiti vaid üksikuid ülesandeid, siis magistritöö käigus koondati kokku erinevad ülesannete tüübid koos näiteülesannetega gümnaasiumi valikkursusele „Programmeerimine“, kus õpitakse programmeerimise alusteadmisi tekstilise programmeerimiskeelega Python. Ülesandetüüpidest ja näiteülesannetest tehti juhendmaterjal, mis vormistati courses.cs.ut.ee keskkonnas veebilehena. Ülesanded koguti maatrikstaksonoomia kategooriate järgi, mis veebilehel omakorda rühmitati, et õpetajal oleks selgem ülevaade, millised võiksid olla raskemad ja millised lihtsamad ülesanded.

Lisaks koostati ja viidi läbi näidiskontrolltöö ühes Eesti gümnaasiumis. Uuringu eesmärk oli saada koostatud ülesannete tüüpidele tagasisidet, sh kas tulemused ühtivad algse maatriksi kategooriate paigutusega. Töö tulemusel selgus, et kategooriaid esindavate

ülesannete tulemused ei suhestunud omavahel algse maatriksi järgi. Analüüside tulemusel leiti, et selle põhjuseks võib olla näiteks ülesande pikkus ja alaülesannete arv. Seetõttu võiks töö edasiarendamisel uurida just ülesannete keerukust ja mahtu, et paika panna hindelise töö ülesehitus, sh ligikaudne ülesannete arv üldlevinumateks tunnipikkusteks.

Magistritöö teine eesmärk oli välja selgitada, millised on praegused õpetajate praktikad hindeliste tööde koostamisel ning kust saadakse nendeks inspiratsiooni. Küsimustikule vastanud 16 õpetajat tõi välja, et üldiselt kasutavad nad materjale, mis on mõeldudki kursuste läbiviimiseks ja on vabalt kättesaadavad, näiteks Tartu Ülikooli arvutiteaduse instituudi courses lehel olevate kursuste materjalid ja valikkursuse õpiku juurde kuuluvad õpetajamaterjalid. Seetõttu ühtisid ka märgitud ülesannete tüübid teoreetilises osas tutvustatud töödega. Kuigi õpetajad märkisid kasutamisalusteks kõiki pakutud 28 ülesandetüüpi, olid kõige levinumad juhise järgi koodi kirjutamine, programmi tulemi kontrollimine ja etteantud koodi parandamine. Populaarsed olid veel koodi uutele tingimustele vastavaks tegemine ja ülesanne, kus on vaja lahendus teisele kujule viia, näiteks plokkskeemi tõlkimine koodiks või vastupidi. Tihti märgiti ka ühe realise koodi kirjutamise ja lünga täitmise tüüpi. Mitmete pakutud ülesandetüüpide kohta öeldi, et kui see oleks materjalides olemas, siis kasutaks seda kindlasti. See kinnitab selle magistritöö vajalikkust ja annab indu teostada pakutud edasiarendust, et välja töötada kogutud ülesandetüüpidega hindeliste tööde näidised.

Viidatud kirjandus

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., & Wittrock, M. C. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman,. https://eduq.info/xmlui/handle/11515/18345

Arhiiv – IT-KURSUSED. (s.a.). Tartu Ülikooli Arvutiteaduse Instituudi IT-Kursused. Salvestatud 8. veebruar 2021, https://programmeerimine.ut.ee/arhiiv/

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain*. Longrnans, Green and Company. https://www.uky.edu/~rsand1/china2018/texts/Bloom%20et%20al%20-Taxonomy%20of%20Educational%20Objectives.pdf

Cognitive ability – APA Dictionary of Psychology. (s.a.). Salvestatud 23. märts 2021, https://dictionary.apa.org/cognitive-ability

Dorodchi, M., Dehbozorgi, N., & Frevert, T. K. (2017). "I wish I could rank my exam's challenge level!": An algorithm of Bloom's taxonomy in teaching CS1. *2017 IEEE Frontiers in Education Conference (FIE)*, 1–5. https://doi.org/10.1109/FIE.2017.8190523

Fuller, U., Johnson, C. G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I., Jackova, J., Lahtinen, E., Lewis, T. L., Thompson, D. M., Riedesel, C., & Thompson, E. (2007). Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin*, *39*, 152–170. https://doi.org/10.1145/1345375.1345438

Gluga, R., Kay, J., Lister, R., Kleitman, S., & Lever, T. (2012). Coming to terms with Bloom: An online tutorial for teachers of programming fundamentals. *Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123*, 147–156. https://dl.acm.org/doi/10.5555/2483716.2483734

Gomes, A., & Correia, F. B. (2018). Bloom's Taxonomy Based Approach to Learn Basic Programming Loops. *2018 IEEE Frontiers in Education Conference (FIE)*, 1–5. https://doi.org/10.1109/FIE.2018.8658947

Gomes, A., & Mendes, A. J. (2009). Bloom's taxonomy based approach to learn basic programming. *Proceedings of ED-MEDIA 2009--World Conference on Educational Multimedia, Hypermedia & Telecommunications*, 2547–2554. https://www.researchgate.net/publication/243962268_Bloom's_taxonomy_based_approach_to_learn_basic_programming

Gümnaasiumi informaatika ainekava. (s.a.). #HITSA. Salvestatud 8. veebruar 2021, https://www.hitsa.ee/ikt-haridus/progetiiger/gumnaasiumi-informaatika-ainekava

IT-õppe teekaart. (s.a.). #HITSA. Salvestatud 8. veebruar 2021, https://www.hitsa.ee/teekaart

Kasulik info. (2015, veebruar 20). Tartu Ülikool. E-õpe, MOOCid. https://www.ut.ee/et/oppimine/kasulik-info

Kursuste võrdlus – ATI programmeerimise õpetamise töörühm. (s.a.). Salvestatud 8. veebruar 2021, https://progmooc.cs.ut.ee/moocid/kursuste-vordlus/

Kwiatkowska, M. (2016). Measuring the Difficulty of Test Items in Computing Science Education. *Proceedings of the 21st Western Canadian Conference on Computing Education*, 1–6. https://doi.org/10.1145/2910925.2910950

Köksal, D., & Ulum, Ö. G. (2018). Language assessment through Bloom's Taxonomy. *Journal of Language and Linguistic Studies, 14*, 76–88. https://dergipark.org.tr/en/pub/jlls/527924

Lahtinen, E. (2007). A Categorization of Novice Programmers: A Cluster Analysis Study. *PPIG, 16*, 32–41. https://www.ppig.org/files/2007-PPIG-19th-Lahtinen.pdf

Metoodiline juhend õpetajale gümnaasiumi informaatika uute valikkursuste õpetamiseks. (s.a.). HITSA. Salvestatud 8. veebruar 2021, https://media.voog.com/0000/0034/3577/files/Metoodiline%20juhend%20%C3%B5petajale%20g%C3%BCmnaasiumi%20informaatika%20uute%20valikkursuste%20%C3%B5petamiseks.docx.pdf

MOOCid. (s.a.). *Tartu Ülikooli Informaatika didaktika töörühm*. Salvestatud 8. veebruar 2021, https://didaktika.cs.ut.ee/moocid/

Programmeerimiskursus "Tehnoloogia tarbijast loojaks". (s.a.). *Tartu Ülikooli Informaatika didaktika töörühm*. Salvestatud 8. veebruar 2021, https://didaktika.cs.ut.ee/progttl/

Rein, E. (2020). *Eesti gümnaasiumites õpetatavad programmeerimise kursused* [Bakalaureusetöö, Tartu Ülikool]. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69746&year=0

Scott, T. (2003). Bloom's taxonomy applied to testing in computer science classes. *Journal of Computing Sciences in Colleges*, *19*, 267–274. https://dl.acm.org/doi/abs/10.5555/948737.948775

Selby, C. C. (2015). Relationships: Computational thinking, pedagogy of programming, and Bloom's Taxonomy. *Proceedings of the Workshop in Primary and Secondary Computing Education*, 80–87. https://doi.org/10.1145/2818314.2818315

Shuhidan, S., Hamilton, M., & D'Souza, D. (2009). A taxonomic study of novice programming summative assessment. *Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95*, 147–156. https://dl.acm.org/doi/abs/10.5555/1862712.1862734

Smith, E. B., Gellatly, M., Schwartz, C. J., & Jordan, S. (2020, september 10). Training Radiology Residents, Bloom Style. *Academic Radiology*. https://doi.org/10.1016/j.acra.2020.08.013

Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm. (s.a.). *MOOC Programmeerimise alused II*. Kursused - Arvutiteaduse instituut. Salvestatud 8. veebruar 2021, https://courses.cs.ut.ee/2019/eprogalused2/spring/Main/HomePage

Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm. (2018a). *MOOC Programmeerimise alused*. Kursused - Arvutiteaduse instituut. https://courses.cs.ut.ee/2018/eprogalused/fall/

Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühm. (2018b). *Programmeerimisest maalähedaselt*. Kursused - Arvutiteaduse instituut. https://courses.cs.ut.ee/2018/progmaa/fall/Main/HomePage

Tehnoloogiaõpe 7.-9. Klassile. (s.a.). #HITSA. Salvestatud 8. veebruar 2021, https://www.hitsa.ee/teekaart/7-9-klassile

Tehnoloogiaõpe lasteaias. (s.a.). #HITSA. Salvestatud 8. veebruar 2021, https://www.hitsa.ee/teekaart/lasteaiale

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). Bloom's taxonomy for CS assessment. *Proceedings of the tenth conference on Australasian computing education - Volume 78*, 155–161. https://dl.acm.org/doi/abs/10.5555/1379249.1379265

Tobe lugu: Eesti digiriiki ähvardab informaatika õpetajate puudus. (2020, juuni 10). *Lõunaeestlane*. https://lounaeestlane.ee/tobe-lugu-eesti-digiriiki-ahvardab-informaatika-opetajate-puudus/

Tokuhama-Espinosa, T. (2019). *Five Pillars of the Mind: Redesigning Education to Suit the Brain* (1st edition). W. W. Norton & Company. https://www.amazon.com/Five-Pillars-Mind-Redesigning-Education/dp/0393713210

Tõnisson, E., Palts, T., Säde, M., Tõnisson, K., & jt. (s.a.-a). Kaitstud: Õpetajamaterjal. *Programmeerimine*. Tartu Ülikool. Salvestatud 8. veebruar 2021, https://web.htk.tlu.ee/digitaru/programmeerimine/chapter/opetajamaterjal/

Tõnisson, E., Palts, T., Säde, M., Tõnisson, K., & jt. (s.a.-b). *Programmeerimine*. Tartu Ülikool. Salvestatud 8. veebruar 2021, https://web.htk.tlu.ee/digitaru/programmeerimine/

Vakil, E., & Heled, E. (2016). The effect of constant versus varied training on transfer in a cognitive skill learning task: The case of the Tower of Hanoi Puzzle. *Learning and Individual Differences, 47*, 207–214. https://doi.org/10.1016/j.lindif.2016.02.009

Vilipõld, J., Antoi, K., & Amitan, I. (2013). *Rakenduste loomise ja programmeerimise alused*. E-koolikott. https://e-koolikott.ee/oppematerjal/7364-Rakenduste-loomise-ja-programmeerimise-alused

Wang, Y., Chen, A., Schweighardt, R., Zhang, T., Wells, S., & Ennis, C. (2019). The nature of learning tasks and knowledge acquisition: The role of cognitive engagement in physical education. *European Physical Education Review, 25*, 293–310. https://doi.org/10.1177/1356336X17724173

Whalley, J. L., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P. K. A., & Prasad, C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using the bloom and SOLO taxonomies. *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, 243–252. https://dl.acm.org/doi/abs/10.5555/1151869.1151901

Lisad

I. Kordamismaterjal õpilastele

Gümnaasiumis läbiviidud kordamistunni materjal sisaldas kahte faili. Esimeses oli siintoodud tabel ning teise oli kopeeritud kontrolltöö vormistusega (vt lisa 3 ja 4) siinse tabeli viimases veerus olevad ülesanded. Tabelis on lisatud iga ülesande ette sulgudes indeks, mille esimene täht tähistab (P) paber- või (A) arvutiosa ning number viitab järjenumbrile. Samadele indeksitele viidatakse lisas 2 olevas tunnikonspektis.

Kirjeldus	Tüübid	Keskkond	Näited
Tunne sõnavara, funktsioone, konstruktsioone jm.	○ Nimeta funktsioon, mis… ○ Kirjelda, mida teeb funktsioon… ○ Selgita, kuidas toimib funktsioon… ○ Liigita konkreetne osa koodist (nt märgi tingimuslaused või nimeta koodis märgitud konstruktsioon).	Lahendatakse arvutit jt abivahendeid kasutamata.	○ *(P1)* Nimeta funktsioon, mille abil saab kontrollida, kas järjend või sõne sisaldab otsitavat üksust. ○ *(P2)* Kirjelda, mis andmestruktuuri või -tüübiga kasutatakse ning mida teeb funktsioon `append`. ○ *(P3)* Kuidas toimib sorteerimisfunktsioon `sort` ehk millised on tulemuse omadused? ○ *(P4)* Märgi joonisel koodiosa, mida sooritatakse ühe käivitamise jooksul korduvalt. `a = 1` `s = 0` `while a < 15:` `  if a % 4 == 0:` `  s += a` `  a += 1` `print(s)`
Kontrolli programmi tulemit käsitsi.	○ Mis väljastatakse programmi tulemusel käsureale? • Mida tagastab funktsioon antud argumentide korral? • Mitu korda täidetakse tsüklit?	Lahendatakse arvutit jt abivahendeid kasutamata.	`for i in range(6, 25, 3):` `  if i % 2 == 0:` `  a = i` `print(a)` ○ *(P5)* Mis väljastatakse programmi tulemusel käsureale? ○ *(P6)* Mis väljastatakse programmi tulemusel käsureale, kui sisestatakse 3 ja 1? `x = int(input("Sisesta x-i väärtus: "))` `y = int(input("Sisesta y-i väärtus: "))` `while x > 0:` `  x *= y` `  x -= 1` `print(x)` ○ *(P7)* Mitu korda täidetakse tsüklit? `arv = 24` `while int(arv) != 1:` `  if arv % 2 == 0:` `  print(arv)` `  arv = arv // 2` `  else:` `  arv = arv % 2`

<table>
<tr><th>Kirjeldus</th><th>Tüübid</th><th>Keskkond</th><th>Näited</th></tr>
<tr><td>Kirjuta etteantud juhiste järgi kood.</td><td>○ Kirjuta kirjelduse järgi tingimustele vastav kood (väike kood või väga täpsed juhised).</td><td>Lahendatakse kasutades arenduskesk-konda.</td><td>○ (A1) Vaata näiteid tunnitöödest.</td></tr>
<tr><td>Muuda lahendus teise struktuuri (sisaldab teadmist, mida ja kuidas muuta ning loomist).</td><td>○ Muuda etteantud kood uutele tingimustele vastavaks.</td><td>Olenevalt ülesande keerukusest lahendatakse kasutades arenduskesk-konda või arvutit jt abivahendeid kasutamata.</td><td>○ (A2) Muuda koodi nii, et arvu pannakse astmesse 2 kuni see ei jagu täpselt kahega. Kui arv jagub kahega, siis lahutatakse 2. Lõpuks väljastatakse tulemus.
<pre>arv = int(input("Sisesta arv: "))

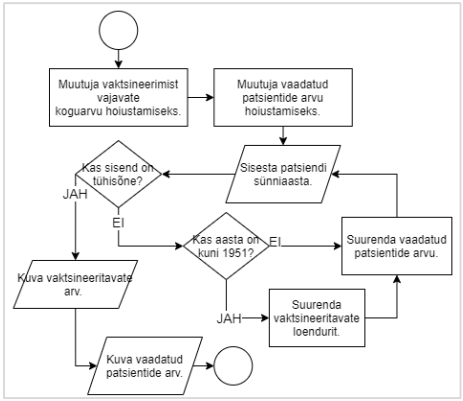
if arv % 2 != 0:
    arv += arv
    print(arv)
else:
    arv -= 2
    print(arv)</pre></td></tr>
<tr><td>Selgita programmi tööd.</td><td>○ Selgita oma sõna sõnadega programmi tööd/algoritmi.</td><td>Lahendatakse arvutit jt abivahendeid kasutamata.</td><td>○ (P8) Selgita oma sõnadega programmi üldist tööd (Märkus: REL on rahvastiku ja eluruumide loendus).
<pre>rel2011 = ["luterlane", "õigeusklik", "baptist", "maausuline",
           "jehoovatunnistaja", "vanausuline", "katoliiklane",
           "taarausuline"]
rel2000 = ["luterlane", "õigeusklik", "baptist", "vanausuline",
         "jehoovatunnistaja", "nelipühilane", "katoliiklane", "adventist"]

erinevused = []

for u1 in rel2011:
    if u1 not in rel2000:
        erinevused += [u1]

for u0 in rel2000:
    if u0 not in rel2011:
        erinevused += [u0]

print(erinevused)</pre></td></tr>
</table>

Kirjeldus	Tüübid	Keskkond	Näited
Tõlgi programm teise vormingusse.	○ Algoritmi tõlkimine ühelt esitusvormilt teisele (nt plokkskeem -> kood või vastupidi).	Lahendatakse kasutades arenduskesk-konda.	○ *(A3−A4)* Kirjuta plokkskeemi järgi programmi kood. 
Kasuta lahendust suurema ülesande osana.	○ Täida koodis olev lünk (rida/argument/funktsioon), et kood vastaks tingimus(t)ele.	Olenevalt ülesande keerukusest lahendatakse kasutades arenduskesk-konda või arvutit jt abivahendeid kasutamata.	○ *(P10)* Tööjuhis palub leida, mitmendal positsioonil on sisendarvus esimene 2ga jaguv number (kui seda ei leidu, siis väljastatakse -1). Täida koodis olev lünk, et kood vastaks tingimus(t)ele. `arv = input("Sisesta arv: ")` `leiti = False` `for i in range(_____):` `  nr = int(arv[i])` `  if nr%2 == 0:` `  print(i)` `  leiti = True` `  break` `if leiti == False:` `  print(-1)`
Leia ja paranda vead.	○ Leia koodis viga ja paranda see. ○ Selgita, miks programm ei anna oodatavat vastust.	Olenevalt ülesande keerukusest lahendatakse kasutades arenduskesk-	○ *(P11)* Programmi oodatavad väljundid algse järjendi modifitseerimisel on `Algne ['1', '2', '3', '4', '5', '6', '7', '8']` `12` `34` `56` `78` ja `Algne ['1', '2', '3', '4', '5']` `12` `34` .

<table>
<tr><th>Kirjeldus</th><th>Tüübid</th><th>Keskkond</th><th>Näited</th></tr>
<tr><td></td><td></td><td>konda või arvutit jt abivahendeid kasutamata.</td><td>Selgita, miks järgmine kood annab veateate ning mida tuleks muuta, et programm annaks oodatud tulemuse.
<pre>
j = ['1', '2', '3', '4', '5', '7', '8']
print("Algne", j)
i = 1
while i < len(j):
    print(j[i-1]+j[i])
    i += 2
</pre></td></tr>
<tr><td>Lähtuvalt ümbritsevast, mõista osa lahendusest.</td><td>○ Lähtuvalt etteantud koodist selgita, miks just sellist konstruktsiooni/funktsiooni on kasutatud.
○ Tuvasta käskude täitmise järjekord.
○ Lisa koodi vajalikud taanded.</td><td>Olenevalt ülesande keerukusest lahendatakse kasutades arenduskeskkonda või arvutit jt abivahendeid kasutamata.</td><td>○ (A5) Lisa koodi vajalikud taanded nii, et programm kuvaks käsureale arvu 88.
<pre>
m = 10
k = 5
p = 3

while m <= 20:
k -= p
if k % 2 == 0:
m += p
p *= -5

print(m)
</pre>
○ (P12) Mis järjekorras täidetakse käske?
<pre>"Teet sööb magusat tortillapitsat".lower().strip("t")[-5:]</pre>
○ (P13) Põhjenda, miks on siin ülesandes mõistlik kasutada while-True tsüklit.
<pre>
nimed = []

while True:
    print("Sisesta võistleja nimi.")
    print("Lõpetamiseks sisesta tühik.")
    nimi = input("Sisend: ")
    if nimi == " ":
        print("Lõpetasid sisestamise")
        break
    else:
        nimed.append(nimi.strip())

print("Võistlete nimekiri", nimed)
</pre></td></tr>
</table>

Kirjeldus	Tüübid	Keskkond	Näited
Kavanda abstraktne lahendus või selle struktuur.	○ Kirjuta lahendus pseudokoodina või plokkskeemina. ○ Jaga suurem ülesanne osadeks. ○ Kavanda programm, mis lahendab etteantud olukorra ehk kirjuta ise ülesande juhis.	Lahendatakse arvutit jt abivahendeid kasutamata.	○ *(P14)* Kavanda programm, mis leiab etteantud järjendist ebasobiva elemendi (ebasobilik algab a-tähega) ehk kirjuta ise ülesande juhis. Ülesande lahendusena esita plokkskeem, pseudokood ja/või lahenduskäigu kirjeldus.
Kujunda lahendus (näiteks optimeerimise) eesmärgil ümber.	○ Leia koodis ebaoptimaalne osa ning paranda see. ○ Mõista, mida teeb etteantud kood ning kirjuta see olemasolevate funktsioonide abil lühemalt. ○ Tõsta koodifragmendid õigesse järjekorda.	Olenevalt ülesande keerukusest lahendatakse kasutades arenduskeskkonda või arvutit jt abivahendeid kasutamata.	○ *(P9)* Mõista, mida teeb etteantud kood ning kirjuta see olemasoleva funktsiooni abil ümber (tulemuses kuni 3 koodirida). `rahvuspargid = [2018, 1993, 1971, 2004, 1993, 1993]` `vanim = 2021` `r = 0` `while r < len(rahvuspargid):` `  park = rahvuspargid[r]` `  if park < vanim:` `  vanim = park` `  r += 1` `if vanim != 2021:` `  print(vanim)` ○ *(A6)* Allolev kood kuvab paarisarvud -10st kuni 10ni. Kirjuta kood ümber pikkuse suhtes optimaalsemalt (kuni 2 rida). `i = -10` `while i <= 9:` `  print(i)` `  i += 2` `print(i)` ○ *(A7)* Optimeeri koodi ning ühtlasi paranda muutujate nimed matemaatikamõistetele vastavaks.

Kirjeldus	Tüübid	Keskkond	Näited
			`` arvud = [5,6,7,8,9,10,11,12,13,14] #järjendi elementide arv elemente = 0 for i in arvud: elemente += 1 #keskmise leidmine keskmineIndx = elemente // 2 elemente = 0 for i in arvud: elemente += 1 if elemente == keskmineIndx: keskmineEl = i print(keskmineEl) ``` ○ *(A8)* Tõsta koodiread õigesse järjekorda ``` #kas kõik elemendid algavad sama sümboliga else: for i in j[1:]: eelmine = str(i) print(samad) samad = samad and False j = [392421, 3570, 38093, 30202, 3378, 366] samad = samad and True if str(i)[0] == eelmine[0]: eelmine = str(j[0]) samad = True ```

II. Kordamistunni konspekt

Õppeaine ja klass: Informaatika, 10. klass
Tunni kestus: 60 min
Teemad:

- Järjend
  - (negatiivsed) indeksid;
  - elemendi saamine ([]), järjendi viilutamine ([]);
  - elemendi kontrollimine (in);
  - append, remove, reverse, sort, len, sum, max, min.
- Sõne kui järjend
  - erinevad järjendi operatsioonid sõnedel (nt elemendi kontrollimine ja viilutamine);
  - meetod split;
  - reavahetus (\n), tabulaator (\t).
- While-kordus
  - tsükkel ehk kordus, while-kordus, while-tsükli osad;
  - tsüklimuutuja;
  - muutuja väärtuse muutmine (lühivariandid += jt);
  - tsüklivälise muutuja muutmine;
  - lõpmatu tsükkel while True;
  - break.
- For-kordus
  - tsükkel ehk kordus, for-kordus, for-tsükli osad;
  - (abi)funktsioon range;
  - for- vs while-kordus.

Tunni ülesehitus:

Aeg	Sisu	Tegevus
3 min	Sissejuhatus	Tervitus. Miks siin: magistritöös uurin erinevaid ülesannete tüüpe ning palun teie abi nende hindamisel. Plaan on selline, et täna tutvume nende tüüpidega kordamisülesannete näol ning peale vaheaega kontrolltöös kasutatakse erinevat tüüpi ülesandeid. Tänase kordamistunni ja kontrolltöö peavad kõik sooritama, kuid teil on võimalus anda nõusolek, et tohin teie kontrolltöö sooritust arvestata anonüümselt oma magistritööks. Ootan väga teie küsimusi ja märkusi, näiteks kui ei saa küsimusest või ülesandest täpselt aru, milline on teie meelest eriti tore ülesanne või milline mitte nii väga ja kõige muu kohta ka, mis mõtted tekivad.

Aeg	Sisu	Tegevus
5 min	Failide ülesehitus, kontrolltöö olemus	Tutvusta failide jaotust (ülesannete tüübid-keskkonnad, ülesanded lahendamiseks teises failis). Osa ülesandeid on vaja lahendada paberil ilma materjalideta, kt teine pool on arvutis Thonnys.
52 min	Ülesanded ja nende märkused (iga ülesannet ei lahenda läbi, vaid annan kaasa märkmed)	Hakkame otsast vaatama, kõigepealt õpilased lahendavad, saavad küsida. Vahepeal võtan sõna ning kommenteerin ülesandeid/plokki. PABEROSA: • **Ül 1-3**: Vaata üle funktsioonid ja kirjeldused, et oskaksid selgitada/ära tunda. • **Ül 5-7**: Tuleta meelde range 3 võimalust. Vajadusel võid mõtlemiseks kirjutada vahetulemused leheservale, lisalehele. • **Ül 8**: Jälgi, mida küsitakse – siin on vaja üldist tööd, mitte täpset, teinekord on vaja täpset (konkreetne rida vms). • **Ül 9**: Õpitud funktsioone ei ole palju, mõtle, millist see meenutab. • **Ül 11**: Selgita! Muuda! – pane tähele, mida on vaja. Selgita mõjutatavaid ridu, mitte kogu koodi. • **Ül 14**: Mis on pseudokood. Oodatakse algoritmi, üldist lahenduskäiku. ARVUTIOSA: • **Ül 4**: Lahendame koos põhjalikult. (Siin on while-True tsükkel) • **Ül 7**: Ürita aru saada, millist õpitud funktsiooni mingi osa koodist meenutab. (pigem lisaülesandena boonuspunktideks) • **Ül 8**: Taanded on jäetud paika, üksikud read on jäetud samale kohale. Küsimuste korral alati võimalus kirjutada meilile või Stuudiumi Suhtlusesse.

III. Kontrolltöö variant A

Kontrolltöö sooritamise ajaks tuleb **liituda Meeti veebitunniga** ning **sisse lülitada oma kaamera**.

Kogu kontrolltöö sooritamisest (ülesannete avamine kuni esitamine) tuleb **ekraani(de)l toimuv salvestada ja esitada.** Video tuleb esitada (suuremat mahtu ja salvestusaega arvestades) kuni 3 tunni jooksul peale ülesannete esitamist, **saates fail ruth.schihalejev@*xxx*.ee**.

Enne kontrolltöö alustamist **sulge kõik arvutis avatud** rakendused ja veebilehed, avatuks jäävad Google Meet ning Moodle. Thonny on (kiusatuste vältimiseks) soovitatav avada peale paberosa lahenduste esitamist.

Kontrolltöö ajal on suhtlemine (v.a õpetajaga) ja ülesannete (sh vastuste) levitamine keelatud.

Test koosneb kahest osast:
1. paberosa (lahenda ilma abimaterjalideta, soovituslikult 15−20 min)
2. arvutiosa (lahenda Thonnys, soovituslikult 40−45 min)

PABEROSA (8,75−11p)

Siin näed korraga kõiki paberosa ülesandeid.

Kirjuta lahendused otse ülesande juures olevasse tekstivälja. Vajadusel kirjuta vahetehted jm enda jaoks mõeldud märkmed Wordi, Notepadi vms.

Muid abivahendeid ega -materjale ei ole lubatud kasutada.

1. Kirjelda, mis andmestruktuuri või –tüübiga kasutatakse ning mida teeb funktsioon `remove`.
(3p)

2. JUHUSLIK valikust: „Kontrolli ja selgita“ vs „Silu“

2.1) Mis ja miks kuvatakse programmi käivitamisel käsureale. (4p)

```
sõnad = ['käik', 'hind', 'a', 'ebe', 'sammas', 'kellu', 'isegi']

samu = 0
for sõna in sõnad:
    if len(sõna) > 1 and sõna[0] == sõna[-1]:
        samu += 1

print(samu)
```

2.2) Leia koodis viga ja paranda see, kirjutades vigane rida uuesti (märgi nr). (2p)

```
kraad = float(input("Sisesta kraadimise tulemus: "))

while kraad > 37.5 and < 39:
    print("Palavik on mõõdukas")
    if kraad >= 38:
        print("Alanda palavikku")
    kraad = float(input("Sisesta kraadimise tulemus: "))
```

3. JUHUSLIK valikust: „Rakenda“ vs „Uuenda“

3.1) Täida koodis olev lünk. (1,25p)

```
#finiši protokoll
lõpetajad = ["Mai", "Koit", "Tuule", "Kirsi", "Lume", "Tormi", "Mari"]

koht = input("Sisesta, mitmendat lõpetajat soovid näha: ")

soovitudLõpetaja = _____________________________
print(koht, "lõpetaja oli", soovitudLõpetaja)
```

3.2) Mõista, mida teeb etteantud kood ning kirjuta see olemasoleva funktsiooni abil ümber (tulemuses kuni 3 koodirida). (1,25p)

```
arvud = [-100, 1, 278, 5, 91, -4, 0]
arvud2 = []

while len(arvud) > 0:
    võetav = min(arvud)
    arvud2.append(võetav)
    arvud.remove(võetav)

print(arvud2)
```

4. Kujunda programm (pseudokoodi või täpse kirjeldusena), mis leiab otsitava elemendi asukohad etteantud järjendis. Kasuta lahenduses tsüklit. (2,5p)

Näide programmi tööst:

```
Algne järjend: ['o', 'm', 'p', 'm', 'm', 'n', 'p', 'm', 'o', 'p', 'n']
Otsitav: m
Asukohad: [1, 3, 4, 7]
```

Paberosa vastuste esitamiseks ja arvutiosa ülesannete nägemiseks vajuta nuppu "Järgmine leht".

NB! Paberosa juurde tagasi tulla siis enam ei saa!

ARVUTIOSA (15,25−19,75p)

Järgnevalt näed kõiki <u>arvutiosa</u> ülesandeid. Nende lahendamiseks kasuta arenduskeskkonda Thonny ja selle võimalusi. Lubatud on kasutada ka Moodle'is olevaid materjale.

Lahenda kõik ülesanded ühte Thonny faili (nimi kujul *EesnimiPerenimi_Proge1_KT2.py*).
Esita kõigi ülesannete lahendused ja Thonny logid (nimi kujul *EesnimiPerenimi_Proge1_KT2.zip*) **viimase ülesande lahenduseks**.

1. Ette on antud kolm järjendit, esimeses on hommikusöögi valikud, teises lõuna ja kolmandas õhtusöögi võimalused. Kirjuta programmi kood, mis kuvab kasutajale need söögid, mis sobivad kaheks või kõigiks kordadeks. (Kokku 10,75p)

Selleks vaata läbi esimene järjend ja kui vaadeldav element on ka teises ja/või kolmandas järjendis, kuva vastav teade (5p). Seejärel vaata teise järjendi elemente ning kui vaadeldav sisaldub ka kolmandas järjendis, kuva vastav teade (3p). Iga valikut tohib kuvada vaid üks kord, selleks kogu vaadeldud elemendid eraldi muutujasse (2,75p).

Näide programmi tulemusest:

```
salat sobib igaks toidukorraks
võileib sobib kaheks toidukorraks
praad sobib kaheks toidukorraks
pitsa sobib kaheks toidukorraks
```

, kui algjärjendid on

```
hommik = ["puder", "salat", "jogurt", "praemuna", "helbed", "võileib"]
lõuna = ["praad", "supp", "salat", "pitsa", "ahjuvorm"]
õhtu = ["pirukas", "võileib", "panniroog", "pitsa", "praad", "salat"]
```

2. JUHUSLIK valikust: „Kohanda“ vs „Transleeri“

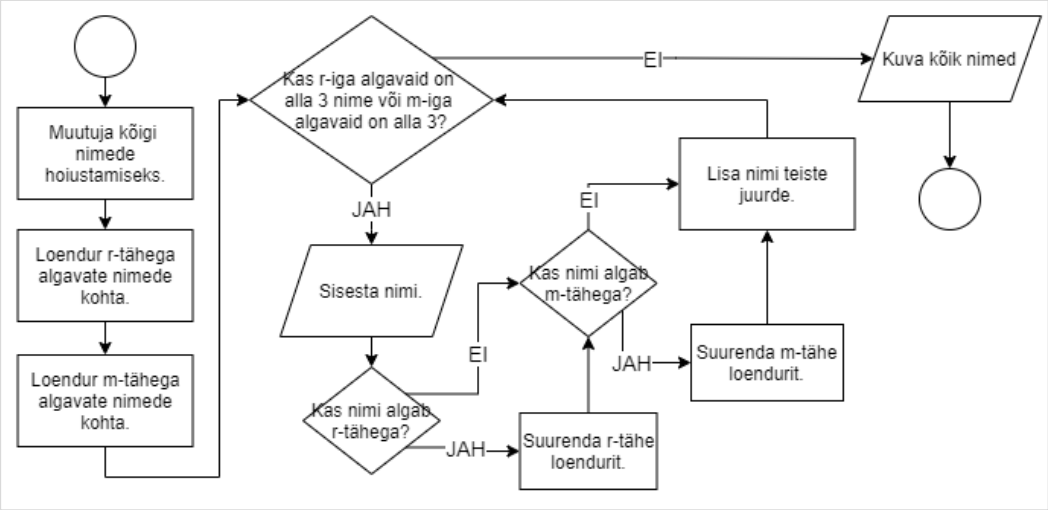
2.1) Kirjuta etteantud programm, kasutades while-tsükli asemel for-tsüklit. (3,5p)

```
aeg = int(input("Sisestage minutite arv: "))
laike = 0
loendur = 0

while loendur < aeg:
    loendur += 1
    if loendur % 2 == 1:
        laike += loendur

print("Laikide koguarv on " + str(laike) + ".")
```

2.2) 2020. sündinud laste 10 populaarseima nime hulgas algavad poiste nimed enamasti r-tähega ja tüdrukute nimed m-tähega. Kirjuta plokkskeemi järgi programmi kood. (8p)



3. Lisa koodi vajalikud taanded, et programm kuvaks ekraanile arvu 20. (1p)

```
x = 5
y = 3
z = 1

while x > 0:
if x==y:
break
z *= x
x -= 1

print(z)
```

Lae siia arvutiosa lahendused (lahendusfail + logifailid) siia ülesande alla, jälgi ka failinimede õigsust!

Lahendusfaili nimi on kujul *EesnimiPerenimi_Proge1_KT2.py* ning logifailide nimi kujul *Eesnimi-Perenimi_Proge1_KT2.zip*.

Faili laadimise kast.

Olen nõus, et minu kontrolltöö sooituskatse tulemusi kasutatakse anonüümselt Ruth Schihalejevi (TÜ) magistritöös.

O Tõene

O Väär

IV. Kontrolltöö variant B

Kontrolltöö sooritamise ajaks tuleb **liituda Meeti veebitunniga** ning **sisse lülitada oma kaamera**.

Kogu kontrolltöö sooritamisest (ülesannete avamine kuni esitamine) tuleb **ekraani(de)l toimuv salvestada ja esitada.** Video tuleb esitada (suuremat mahtu ja salvestusaega arvestades) kuni 3 tunni jooksul peale ülesannete esitamist, **saates fail ruth.schihalejev@*xxx*.ee**.

Enne kontrolltöö alustamist **sulge kõik arvutis avatud** rakendused ja veebilehed, avatuks jäävad Google Meet ning Moodle. Thonny on (kiusatuste vältimiseks) soovitatav avada peale paberosa lahenduste esitamist.

Kontrolltöö ajal on suhtlemine (v.a õpetajaga) ja ülesannete (sh vastuste) levitamine keelatud.

Test koosneb kahest osast:

3. paberosa (lahenda ilma abimaterjalideta, soovituslikult 15−20 min)
4. arvutiosa (lahenda Thonnys, soovituslikult 40−45 min)

PABEROSA (9−11,75p)

Siin näed korraga kõiki paberosa ülesandeid.

Kirjuta lahendused otse ülesande juures olevasse tekstivälja. Vajadusel kirjuta vahetehted jm enda jaoks mõeldud märkmed Wordi, Notepadi vms.

Muid abivahendeid ega -materjale ei ole lubatud kasutada.

1. Kirjelda, mis andmestruktuuri või -tüübiga kasutatakse ning mida teeb funktsioon `reverse`. (2p)

2. JUHUSLIK valikust: „Kontrolli ja selgita“ vs „Silu“ vs „Seosta“

2.1) Mis ja miks kuvatakse programmi käivitamisel käsureale. (4p)

```
sõnad = ['maias', 'luba', 'äi', 'veepudel', 'kaal', 'päike']

samu = 0
for sõna in sõnad:
    if sõna[1] == sõna[-2] and len(sõna) > 3:
        samu += 1

print(samu)
```

2.2) Leia koodis viga ja paranda see, kirjutades vigane rida uuesti (märgi nr). (2p)

```
#kuva temperatuurid, mis on alla 1 kraadi
temperatuurid = [0, 1, 3, 1, 0, -1, 0, 2, 3]

for temperatuurid in el:
    if el <= 0:
        print(el)
```

2.3) Kirjuta üksteise alla teises reas olevate käskude täitmise järjekord. (2p)

```
aastapäev = "24.02.1918 Eesti Vabariik"
len(aastapäev.split()[0].split("."))
```

3. JUHUSLIK valikust: „Rakenda“ vs „Uuenda“

3.1) Ema saab õunu jagada, kuni iga laps on (1−2) õuna saanud või õunu veel jätkub. Täida koodis olev lünk. (2p)

```
from random import randint
õunu = 14

lapsi = int(input("Õuna soovivate laste arv: "))

i = 1
while _____________________________:
    lapsele = randint(1,2)
    print(lapsele)
    õunu -= lapsele
    i += 1

print("Emale jäi", õunu)
```

3.2) Mõista, mida teeb etteantud kood ning kirjuta see olemasoleva funktsiooni abil ümber (tulemuses kuni 3 koodirida). (1,25p)

```
arvud = [55, -84, 70, 377, -215, 13, 9]
vastus = 0

while len(arvud) > 0:
    vastus += arvud[0]
    arvud.remove(arvud[0])

print(vastus)
```

4. Kujunda programm (pseudokoodi või täpse kirjeldusena), mis kogub algsest järjendist meeste ja naiste palgad eraldi järjenditesse. (3,75p)

Näide programmi tööst:

```
Algsed andmed: ['n1095', 'n1398', 'm2010', 'n1950', 'm1834', 'm993',
'n1520', 'm1490', 'm1330', 'n2000', 'm1700', 'n1690', 'm1840']

Naiste palgad: [1095, 1398, 1950, 1520, 2000, 1690]
Meeste palgad: [2010, 1834, 993, 1490, 1330, 1700, 1840]
```

Paberosa vastuste esitamiseks ja arvutiosa ülesannete nägemiseks vajuta nuppu "Järgmine leht".
NB! Paberosa juurde tagasi tulla siis enam ei saa!

ARVUTIOSA (14,75−19,75p)

Järgnevalt näed kõiki arvutiosa ülesandeid. Nende lahendamiseks kasuta arenduskeskkonda Thonny ja selle võimalusi. Lubatud on kasutada ka Moodle'is olevaid materjale.

Lahenda kõik ülesanded ühte Thonny faili (nimi kujul *EesnimiPerenimi_Proge1_KT2.py*). **Esita kõigi ülesannete lahendused ja Thonny logid** (nimi kujul *EesnimiPerenimi_Proge1_KT2.zip*) **viimase ülesande lahenduseks**.

1. Linnal on tehtud leping lume väljaveo mahule (igal aastal uus). Kui lund veetakse välja vähem, kui kokku lepitud, tagastab teenusepakkuja linnale vastava summa, kui aga lund on vaja rohkem välja vedada, maksab linn vajaliku osa lisaks. (Kokku 11,5p)

Kirjuta programmi kood, mis küsib kasutajalt kõigepealt algse lumeveo mahu ning seejärel tsükli abil (st ükshaaval) iga nädala väljaveomahu. (4p) Lepingumahu ületamise kohta tuleb üks kord anda teade (teavituse edastamise kohta hoia infot eraldi muutujas). (3,5p) Programmi lõppedes tuleb kuvada, kes ja kui palju peab teisele maksma (näiteks vastavalt sellele, kas lumeveo kogumaht on negatiivne, positiivne või null). (4p)

Näide programmi tööst:

```
Sisesta kokkulepitud kogus: 15000
        Sisesta nädala kogus: 100
        Sisesta nädala kogus: 392
        Sisesta nädala kogus: 1405
        Sisesta nädala kogus: 5931
        Sisesta nädala kogus: 750
        Sisesta nädala kogus: 2472
        Sisesta nädala kogus: 1850
        Sisesta nädala kogus: 9380
Sel aastal on lund rohkem, kui viimasel 20 aastal.
        Sisesta nädala kogus: 3523
        Sisesta nädala kogus: 180
        Sisesta nädala kogus:

Linn peab teenuse eest maksma lisaks: 10983.0
```

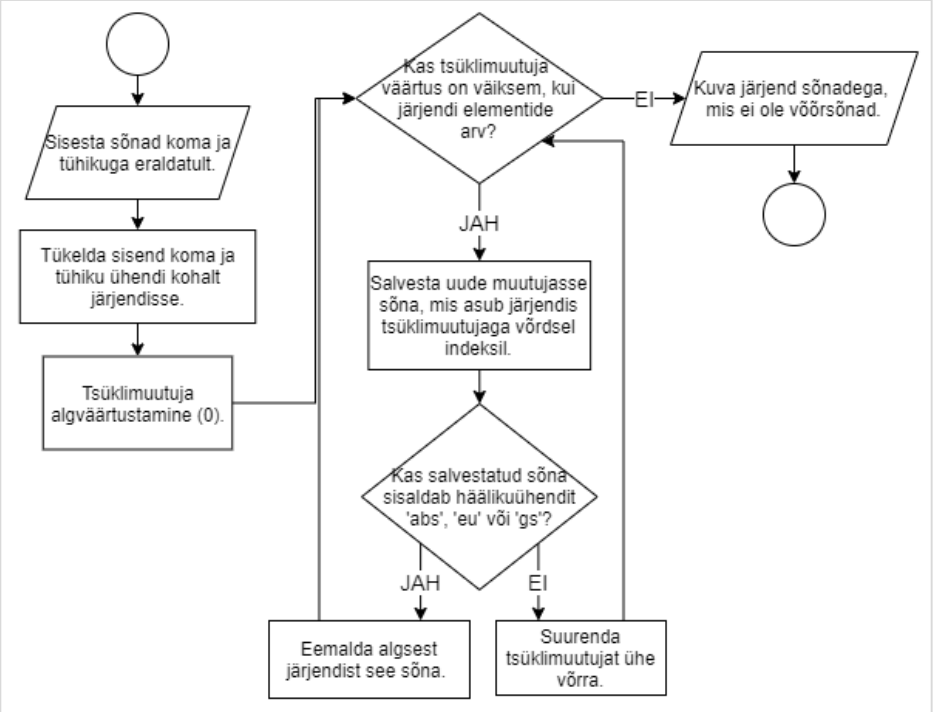
2. JUHUSLIK valikust: „Kohanda“ vs „Transleeri“

2.1) Muuda while-tsüklit nii, et sisendit küsitakse lõpmatult, kuid tsükkel lõpetatakse siis, kui sisend on suurem nullist. (3,25p)

```
while sisend < 0:
    print("Ruutjuurt ei saa võtta negatiivsest arvust.")
    sisend = input("Sisesta arv, millest soovid ruutjuurt: ")
```

2.2) Võõrsõnade üheks tunnuseks on tavatu häälikuühend, näiteks „gs“, „abs“, „eu“. Kirjuta plokkskeemi järgi programmi kood. (8,25p)

Programmi sisend võib olla näiteks „gangster, absoluutne, ujula, bakalaureus, tõrv, piisk, heuristika, lütseum, kalm“.



Lae siia arvutiosa lahendused (lahendusfail + logifailid) siia ülesande alla, jälgi ka failinimede õigsust!

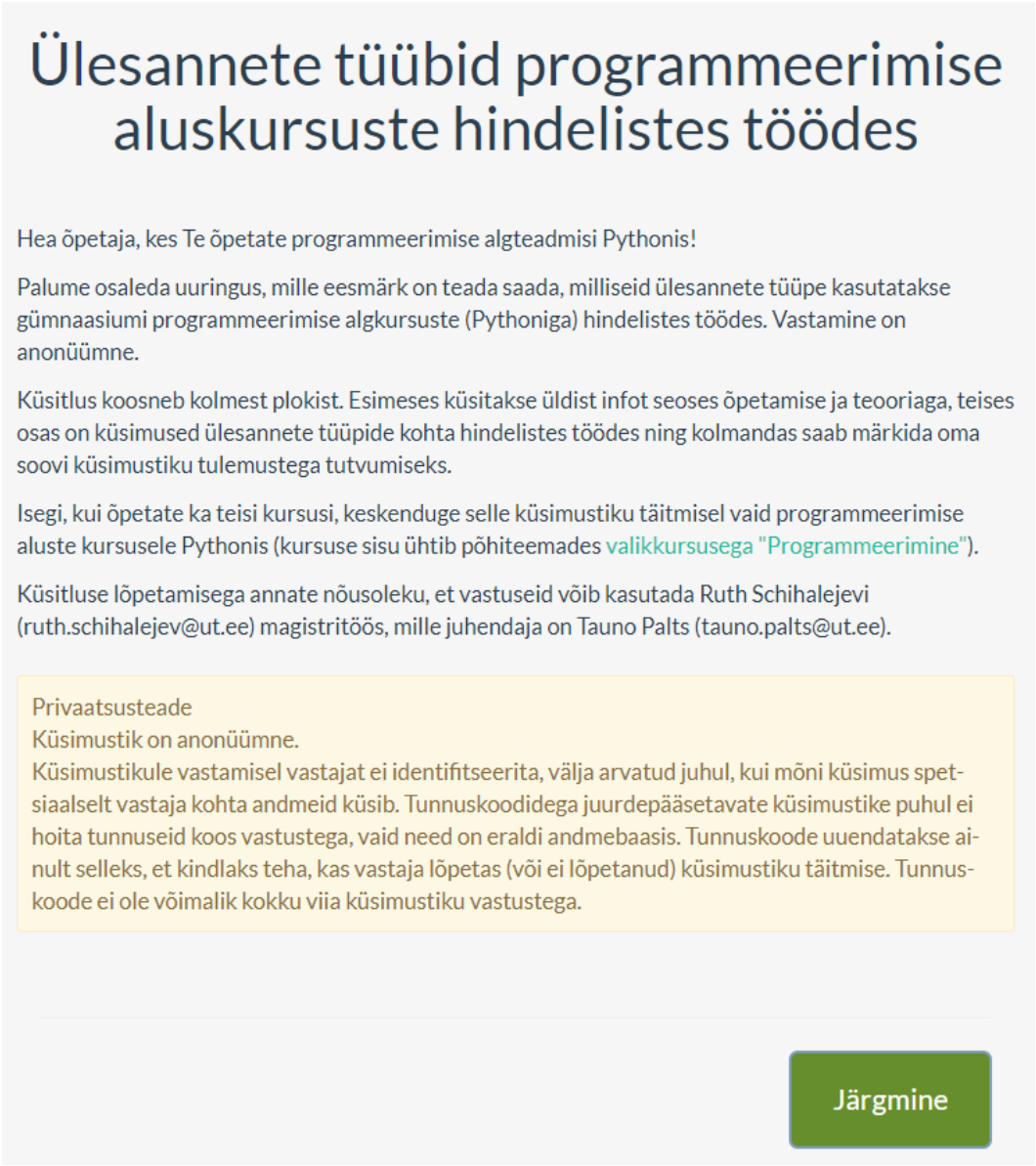
Lahendusfaili nimi on kujul *EesnimiPerenimi_Proge1_KT2.py* ning logifailide nimi kujul *Eesnimi-Perenimi_Proge1_KT2.zip*.

Faili laadimise kast.

Olen nõus, et minu kontrolltöö sooituskatse tulemusi kasutatakse anonüümselt Ruth Schihalejevi (TÜ) magistritöös.

- O Tõene
- O Väär

V. Küsimustik



Ülesannete tüübid programmeerimise aluskursuste hindelistes töödes

Hea õpetaja, kes Te õpetate programmeerimise algteadmisi Pythonis!

Palume osaleda uuringus, mille eesmärk on teada saada, milliseid ülesannete tüüpe kasutatakse gümnaasiumi programmeerimise algkursuste (Pythoniga) hindelistes töödes. Vastamine on anonüümne.

Küsitlus koosneb kolmest plokist. Esimeses küsitakse üldist infot seoses õpetamise ja teooriaga, teises osas on küsimused ülesannete tüüpide kohta hindelistes töödes ning kolmandas saab märkida oma soovi küsimustiku tulemustega tutvumiseks.

Isegi, kui õpetate ka teisi kursusi, keskenduge selle küsimustiku täitmisel vaid programmeerimise aluste kursusele Pythonis (kursuse sisu ühtib põhiteemades valikkursusega "Programmeerimine").

Küsitluse lõpetamisega annate nõusoleku, et vastuseid võib kasutada Ruth Schihalejevi (ruth.schihalejev@ut.ee) magistritöös, mille juhendaja on Tauno Palts (tauno.palts@ut.ee).

Privaatsusteade
Küsimustik on anonüümne.
Küsimustikule vastamisel vastajat ei identifitseerita, välja arvatud juhul, kui mõni küsimus spetsiaalselt vastaja kohta andmeid küsib. Tunnuskoodidega juurdepääsetavate küsimustike puhul ei hoita tunnuseid koos vastustega, vaid need on eraldi andmebaasis. Tunnuskoode uuendatakse ainult selleks, et kindlaks teha, kas vastaja lõpetas (või ei lõpetanud) küsimustiku täitmise. Tunnuskoode ei ole võimalik kokku viia küsimustiku vastustega.

Järgmine

Eelküsimused

*** Kui pikk on Teie tööstaaž (aastates) programmeerimise aluste õpetajana?**

Valige üks järgnevatest vastustest

- ○ kuni 2 aastat
- ○ 3–10 aastat
- ○ 11–20 aastat
- ○ üle 20 aasta

Ümardage aastad ülespoole (näiteks, kui te õpetate kolmandat õppeaastat, valige 3-10).

*** Kust olete saanud inspiratsiooni hindelise töö ülesehituseks ja/või ülesannete tüüpideks?**

*** Milline on Teie kokkupuude õppe- ja kasvatustöö eesmärkide liigitustega?**

Märkige palun kõik, mis sobivad

- ☐ Olen lõpetanud/läbimas õpetajakoolituse magistrantuuri (vähemalt 60EAP).
- ☐ Olen läbinud õppe- ja kasvatustöö eesmärkide kohta täiendusõppeprogrammi(/-e).
- ☐ Olen iseseisvalt uurinud.
- ☐ Pole õppe- ja kasvatustöö eesmärkidega varem kokku puutunud.
- ☐ Muu:

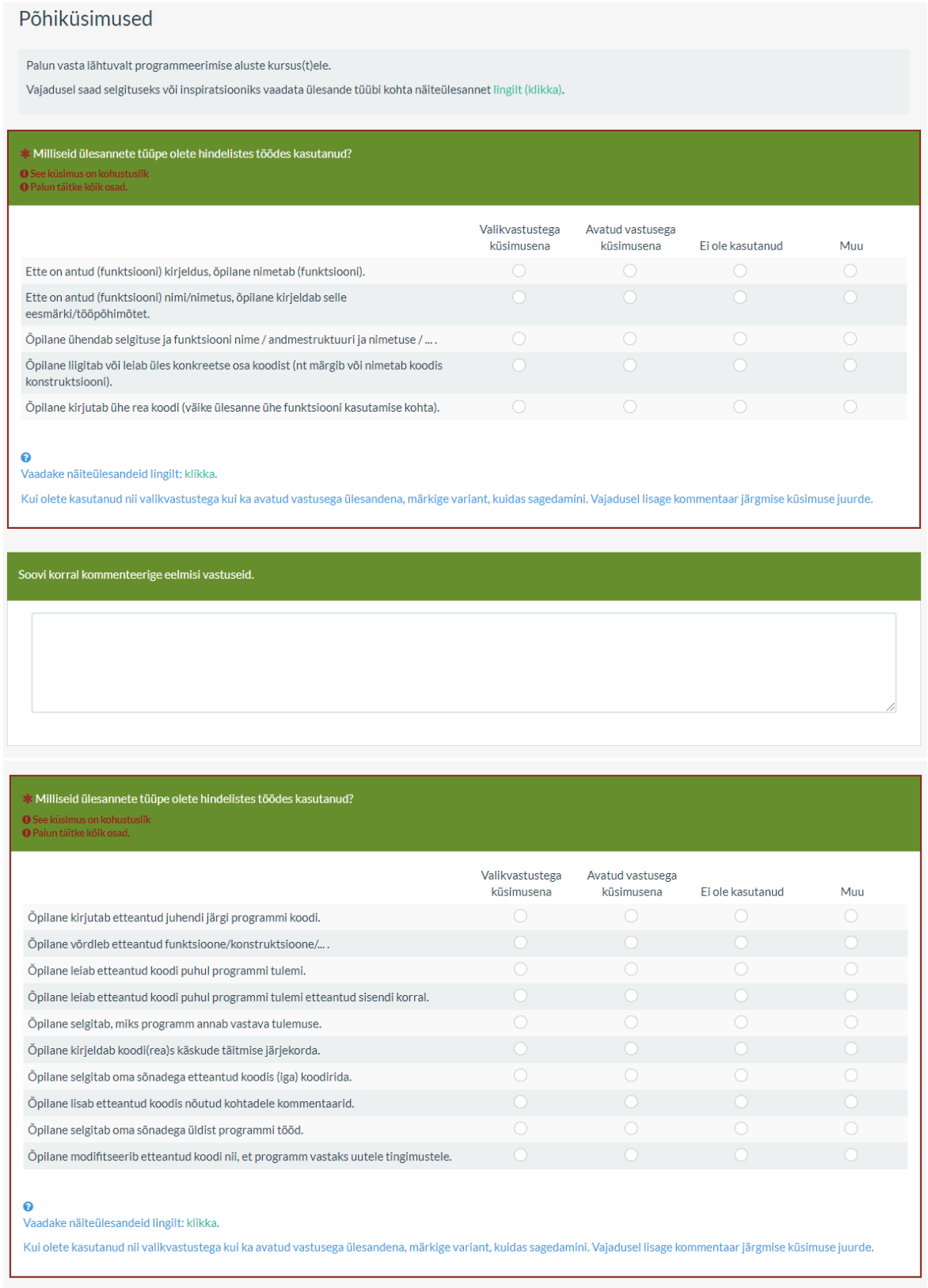
*****

Kommenteerige väidet:

Hindeliste tööde koostamisel jälgin teadlikult, et selles kontrollitaks erinevaid õppeprotsessi kognitiivseid tasemeid.

Näiteks on õppeprotsessi kognitiivsete tasemete liigitus üks Bloomi taksonoomia osadest (teadmine, mõistmine, rakendamine, analüüs, süntees, hindamine).

Põhiküsimuste juures on lingid, millelt leiab iga ülesande tüübi kohta ühe näiteülesande. Kõik küsimuste tüübid oli koondatud ka ühte faili, selle leiab lisast 6.



Põhiküsimused

Palun vasta lähtuvalt programmeerimise aluste kursus(t)ele.

Vajadusel saad selgituseks või inspiratsiooniks vaadata ülesande tüübi kohta näiteülesannet lingilt (klikka).

✱ Milliseid ülesannete tüüpe olete hindelistes töödes kasutanud?

See küsimus on kohustuslik
Palun täitke kõik osad.

	Valikvastustega küsimusena	Avatud vastusega küsimusena	Ei ole kasutanud	Muu
Ette on antud (funktsiooni) kirjeldus, õpilane nimetab (funktsiooni).	○	○	○	○
Ette on antud (funktsiooni) nimi/nimetus, õpilane kirjeldab selle eesmärki/tööpõhimõtet.	○	○	○	○
Õpilane ühendab selgituse ja funktsiooni nime / andmestruktuuri ja nimetuse /	○	○	○	○
Õpilane liigitab või leiab üles konkreetse osa koodist (nt märgib või nimetab koodis konstruktsiooni).	○	○	○	○
Õpilane kirjutab ühe rea koodi (väike ülesanne ühe funktsiooni kasutamise kohta).	○	○	○	○

Vaadake näiteülesandeid lingilt: klikka.

Kui olete kasutanud nii valikvastustega kui ka avatud vastusega ülesandena, märkige variant, kuidas sagedamini. Vajadusel lisage kommentaar järgmise küsimuse juurde.

Soovi korral kommenteerige eelmisi vastuseid.

✱ Milliseid ülesannete tüüpe olete hindelistes töödes kasutanud?

See küsimus on kohustuslik
Palun täitke kõik osad.

	Valikvastustega küsimusena	Avatud vastusega küsimusena	Ei ole kasutanud	Muu
Õpilane kirjutab etteantud juhendi järgi programmi koodi.	○	○	○	○
Õpilane võrdleb etteantud funktsioone/konstruktsioone/... .	○	○	○	○
Õpilane leiab etteantud koodi puhul programmi tulemi.	○	○	○	○
Õpilane leiab etteantud koodi puhul programmi tulemi etteantud sisendi korral.	○	○	○	○
Õpilane selgitab, miks programm annab vastava tulemuse.	○	○	○	○
Õpilane kirjeldab koodi(rea)s käskude täitmise järjekorda.	○	○	○	○
Õpilane selgitab oma sõnadega etteantud koodis (iga) koodirida.	○	○	○	○
Õpilane lisab etteantud koodis nõutud kohtadele kommentaarid.	○	○	○	○
Õpilane selgitab oma sõnadega üldist programmi tööd.	○	○	○	○
Õpilane modifitseerib etteantud koodi nii, et programm vastaks uutele tingimustele.	○	○	○	○

Vaadake näiteülesandeid lingilt: klikka.

Kui olete kasutanud nii valikvastustega kui ka avatud vastusega ülesandena, märkige variant, kuidas sagedamini. Vajadusel lisage kommentaar järgmise küsimuse juurde.

Soovi korral kommenteerige eelmisi vastuseid.

Milliseid ülesannete tüüpe olete hindelistes töödes kasutanud?

See küsimus on kohustuslik
Palun täitke kõik osad.

	Valikvastustega küsimusena	Avatud vastusega küsimusena	Ei ole kasutanud	Muu
Õpilane leiab etteantud koodis vea ja parandab selle.	○	○	○	○
Õpilane selgitab, miks just sellist konstruktsiooni/funktsiooni on kasutatud.	○	○	○	○
Õpilane leiab koodist ebavajalikud osad (nt programmi tööd mitte mõjutavad osad).	○	○	○	○
Õpilane lisab koodi taanded nii, et programm annaks oodatava vastuse.	○	○	○	○
Õpilane optimeerib koodi, asendades koodifragmendid olemasolevate funktsioonidega.	○	○	○	○
Õpilane optimeerib etteantud koodi, leides ebaoptimaalsed kohad (vt nt harjutust "Miks on ronk nagu kirjutuslaud" - link)	○	○	○	○
Õpilane tõstab koodifragmendid järjekorda nii, et programm annaks oodatava vastuse.	○	○	○	○

Vaadake näiteülesandeid lingilt: klikka.

Kui olete kasutanud nii valikvastustega kui ka avatud vastusega ülesandena, märkige variant, kuidas sagedamini. Vajadusel lisage kommentaar järgmise küsimuse juurde.

Soovi korral kommenteerige eelmisi vastuseid.

Milliseid ülesannete tüüpe olete veel hindelistes töödes kasutanud?

Võib tuua nii kirjelduse kui ka näite.

*
Palun kommenteerige eeltoodud ülesannete tüüpe ja nende näiteid.
(Kõik kirjeldused ja näited leiab lingilt: klikka)

See küsimus on kohustuslik

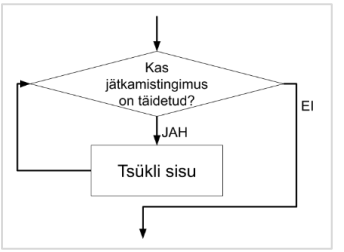
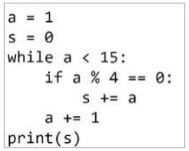
* Kas olete nõus jagama mõnda oma hindelist tööd?

Valige üks järgnevatest vastustest
See küsimus on kohustuslik

- Jah, soovin jagada faili oma seadmest.
- Jah, soovin jagada tekstina (näiteks veebiaadressina või tekstina).
- Ei soovi jagada.

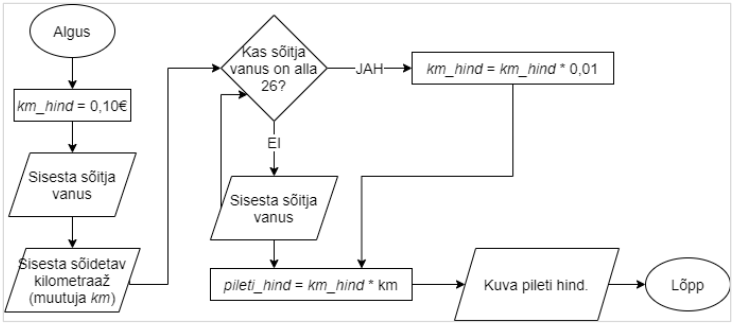
VI. Küsimustikus pakutud ülesandetüüpide näiteülesanded

Lisas 5 olevas küsimustikus olid toodud iga ülesande tüübi kohta ka näiteülesanded, et vastajad mõistaksid paremini küsitavat.

<table>
<tr><th></th><th>Ülesande tüüp</th><th>Näiteülesanne</th></tr>
<tr><td>1</td><td>Ette on antud (funktsiooni) kirjeldus, õpilane nimetab (funktsiooni).</td><td>Nimeta funktsioon, mille abil saab kontrollida, kas järjend või sõne sisaldab otsitavat üksust.
Mis tsüklit on skeemil kujutatud?
 
</td></tr>
<tr><td>2</td><td>Ette on antud (funktsiooni) nimi/nimetus, õpilane kirjeldab selle eesmärki/tööpõhimõtet.</td><td>Kirjelda, mis andmestruktuuri või -tüübiga kasutatakse ning mida teeb funktsioon <code>append</code>.
Kuidas toimib sorteerimisfunktsioon <code>sort</code> ehk millised on tulemuse omadused?</td></tr>
<tr><td>3</td><td>Õpilane ühendab selgituse ja funktsiooni nime / andmestruktuuri ja nimetuse / ... .</td><td><table><tr><td>Jagamine (vastuseks ujukomaarv)</td><td>//</td></tr><tr><td>Täisosa leidmine</td><td>%</td></tr><tr><td>Jäägi leidmine</td><td>/</td></tr></table></td></tr>
<tr><td>4</td><td>Õpilane liigitab või leiab üles konkreetse osa koodist (nt märgib või nimetab koodis konstruktsiooni).</td><td>Märgi joonisel koodiosa, mis kuulub korduse alla.
 
</td></tr>
<tr><td>5</td><td>Õpilane kirjutab ühe rea koodi (väike ülesanne ühe funktsiooni kasutamise kohta).</td><td>Tükelda muutuja <code>lõik</code> väärtus tabulaatorite (taanete) kohalt järjendisse.</td></tr>
<tr><td>6</td><td>Õpilane kirjutab etteantud juhendi järgi programmi koodi.</td><td>Kirjuta programmikood, mis teisendab eurod kroonideks:
* kui sisend lõppeb EUR, siis korruta 15,5455,
* kui sisend lõppeb EEK, siis jaga 15,5455,
* muul juhul küsi uuesti sisendit.</td></tr>
</table>

	Ülesande tüüp	Näiteülesanne
7	Õpilane võrdleb etteantud funktsioone/konstruktsioone/... .	Võrdle for- ja for-range tsükleid.
8	Õpilane leiab etteantud koodi puhul programmi tulemi.	Mis kuvatakse pildil oleva programmi tulemusel käsureale? ```for i in range(3, -3, -1): print(i)``` Mitu korda täidetakse tsüklit? ```arv = 24 while int(arv) != 1: if arv % 2 == 0: print(arv) arv = arv // 2 else: arv = arv % 2```
9	Õpilane leiab etteantud koodi puhul programmi tulemi etteantud sisendi korral.	Mis kuvatakse programmi tulemusel käsureale, kui sisestatakse 3 ja 1? ```x = int(input("Sisesta x-i väärtus: ")) y = int(input("Sisesta y-i väärtus: ")) while x > 0: x *= y x -= 1 print(x)```
10	Õpilane selgitab, miks programm annab vastava tulemuse.	Selgita, miks kood ei anna vastuseks `[1, 2, 3]`. ```jarjend = [1, 2, 3, 4, 5] jarjend2 = jarjend[-2:-5] print(jarjend2)```
11	Õpilane kirjeldab koodi(rea)s käskude täitmise järjekorda.	Mis järjekorras täidetakse teisel real olevaid käske? ```aastapäev = "24.02.1918 Eesti Vabariik" len(aastapäev.split()[0].split("."))```
12	Õpilane selgitab oma sõnadega etteantud koodis iga koodirida.	Selgita oma sõnadega iga koodirea tööd. ```failid = ["arvutus.xlsx", "w.eeskiri.docx", "programm.py", "esitlus.pptx"] lõpud = [] for andmekogum in failid: laiend = andmekogum.split(".")[-1] lõpud.append(laiend) print("Tulemusjärjend on\n\t" + str(lõpud))```

	Ülesande tüüp	Näiteülesanne
13	Õpilane lisab etteantud koodis nõutud kohtadele kommentaarid.	Kommenteeri koodi (ette on antud nõutud kohad). `täish = ['a', 'e', 'i', 'o', 'u', 'õ', 'ä', 'ö', 'ü']` `kaash = ""` `kaash_max = ""` `nimi = input("Sisesta nimi: ")` `#` `for täht in nimi.lower():` `  if täht in täish:` `  #` `  if len(kaash)>len(kaash_max):` `  kaash_max=kaash` `  kaash = ""` `  else:` `  #` `  kaash += täht` `#` `if len(kaash)>len(kaash_max):` `  kaash_max=kaash` `print("Nimi oli "+nimi+".")` `print("Selles on kõige rohkem kõrvuti",len(kaash_max), "kaashäälikut.")` `print("Need kaashäälikud on järjekorras järgmiselt:",kaash_max+".")`
14	Õpilane selgitab oma sõnadega üldist programmi tööd.	Selgita oma sõnadega programmi tööd. `hinded = [84,32,65,41]` `print("Algsed tulemused:\n",hinded)` `uued_hinded = []` `for i in hinded:` `  if i < 50:` `  uus = int(input("Millega asendad tulemuse "+str(i)+" : "))` `  uued_hinded.append(uus)` `  else:` `  uued_hinded.append(i)` `print("Lõplikud tulemused:\n", uued_hinded, "\n")` `print("Keskmine tulemus on: "+str(sum(uued_hinded)/len(uued_hinded)))` `print("Madalaim tulemus on: "+str(min(uued_hinded)))` `print("Kõrgeim tulemus on: "+str(max(uued_hinded)))`

	Ülesande tüüp	Näiteülesanne
15	Õpilane modifitseerib etteantud koodi nii, et programm vastaks uutele tingimustele.	Muuda while-tsüklit nii, et sisendit küsitakse lõpmatult, kuid tsükkel lõpetatakse siis, kui sisend on suurem nullist. `while sisend < 0:` `  print("Ruutjuurt ei saa võtta negatiivsest arvust.")` `  sisend = input("Sisesta arv, millest soovid ruutjuurt: ")` `print("Ruutjuur arvust", sisend, "on", sqrt(sisend))`
16	Õpilane tõlgib ühelt esitlusvormilt teisele (nt plokkskeem → kood).	Kirjuta plokkskeemi järgi programmi kood. 
17	Õpilane muudab ühelt vormilt teisele (järjendid → sõnastik, while → for, ...).	Kirjuta etteantud programm, kasutades while-tsükli asemel for-tsüklit. `aeg = int(input("Sisestage minutite arv: "))` `laike = 0` `loendur = 0` `while loendur < aeg:` `  loendur += 1` `  if loendur % 2 == 1:` `  laike += loendur` `print("Laikide koguarv on " + str(laike) + ".")`
18	Õpilane paigutab etteantud koodijupi suurema koodi sobivasse kohta.	Kirjuta sobivasse kohta koodifragment *`uus.reverse()`*

	Ülesande tüüp	Näiteülesanne
		`#Ülesanne:` `#"sulav - alla sadas sammal suli" -> "ilus lammas sadas alla - valus"` `tekst = "sulav - alla sadas sammal suli".split()` `tekst.reverse()` `tulemus = ""` `for yksus in tekst:` `  uus = list(yksus)` `  uus="".join(uus)` `  tulemus += " "+uus` `print(tulemus)`
19	Õpilane leiab (pseudokoodina/kirjeldusena/...) olukorra, kus kasutada etteantud koodijuppi.	Leia olukord, kus oleks võimalik kasutada koodijuppi *`sõne[3:5]`* (kirjuta pseudokoodi abil lahendus, kasutades antud osa).
20	Õpilane täidab etteantud koodis oleva lünga (rida/funktsioon/lõik/...).	Tööjuhis palub leida, mitmendal positsioonil on sisendarvus esimene 2ga jaguv number (kui seda ei leidu, siis kuvatakse -1). Täida koodis olev lünk, et kood vastaks tingimus(t)ele. Täida koodis olev lünk. `arv = input("Sisesta arv: ")` `for i in range(__________):` `  nr = int(arv[i])` `  if nr%2 == 0:` `  print(i)` `print(-1)` `#finiši protokoll` `lõpetajad = ["Mai", "Koit", "Tuule", "Kirsi", "Lume", "Tormi", "Mari"]` `koht = input("Sisesta, mitmendat lõpetajat soovid näha: ")` `soovitudLõpetaja = _____________` `print(koht, "lõpetaja oli", soovitudLõpetaja)`
21	Õpilane jagab suurema ülesande osadeks (kirjeldusena/plokkskeemina/pseudokoodina/...) ehk kirjutab ise koodi kirjutamise juhendi.	Koosta plokkskeem või tööjuhis ülesandele, mis leiab etteantud järjendist ebasobiva elemendi (ebasobilik algab a-tähega).

<table>
<tr><th></th><th>Ülesande tüüp</th><th>Näiteülesanne</th></tr>
<tr><td>22</td><td>Õpilane leiab etteantud koodis vea ja parandab selle.</td><td>Leia koodis viga ja paranda see.

```
#Kuva temperatuurid, mis on all 1 kraadi
temperatuurid = [0, 1, 3, 1, 0, -1, 0, 2, -3]
for temperatuurid in el:
    if el <= 0:
        print(el)
```

Korrasta kood.

```
while True:
    1nimi = input("Sisesta nimi: ")
    Vanus = input("Sisesta vanus: ")
    if Vanus.isnumeric() == True: break
        Vanus = int(Vanus)
    sünnikuupäev = input("Sisesta sünnikuupäev: ")
    print("Sisesta vanus täisarvuna!")

print(1nimi, 'tohib vaadata filmi "A Quiet Place":', Vanus>12)
```

Paranda koodis olevaid numbreid nii, et programm annaks vastuseks `[1, 2, 3]`.

```
jarjend = [1, 2, 3, 4, 5]
jarjend2 = jarjend[-2:-5]
print(jarjend2)
```
</td></tr>
<tr><td>23</td><td>Õpilane selgitab, miks just sellist konstruktsiooni/funktsiooni on kasutatud.</td><td>Põhjenda, miks on siin ülesandes mõistlik kasutada while-True tsüklit.

```
nimed = []

while True:
    print("Sisesta võistleja nimi.")
    print("Lõpetamiseks sisesta tühik.")
    nimi = input("Sisend: ")
    if nimi == " ":
        print("Lõpetasid sisestamise")
        break
    else:
        nimed.append(nimi.strip())

print("Võistlete nimekiri", nimed)
```
</td></tr>
<tr><td>24</td><td>Õpilane leiab koodist ebavajalikud osad (nt programmi tööd mitte mõjutavad osad).</td><td>Korrasta kood.

```
while True:
    1nimi = input("Sisesta nimi: ")
    Vanus = input("Sisesta vanus: ")
    if Vanus.isnumeric() == True: break
        Vanus = int(Vanus)
    sünnikuupäev = input("Sisesta sünnikuupäev: ")
    print("Sisesta vanus täisarvuna!")

print(1nimi, 'tohib vaadata filmi "A Quiet Place":', Vanus>12)
```
</td></tr>
</table>

	Ülesande tüüp	Näiteülesanne
25	Õpilane lisab koodi taanded nii, et programm annaks oodatava vastuse.	Lisa koodi vajalikud taanded, et programm kuvaks ekraanile arvu 20. `x = 5` `y = 3` `z = 1` `while x > 0:` `if x==y:` `break` `z *= x` `x -= 1` `print(z)`
26	Õpilane optimeerib koodi, asendades koodifragmendid olemasolevate funktsioonidega.	Mõista, mida teeb etteantud kood ning kirjuta see kood olemasoleva funktsiooni abil ümber (tulemuses on 3 koodirida). `arvud = [23, 58, 13, 794]` `print("Algne järjend",arvud)` `summa = 0` `for arv in arvud:` `summa += arv` `print("Summa on", summa)`
27	Õpilane optimeerib etteantud koodi, leides ebaoptimaalsed kohad (vt nt harjutust "Miks on ronk nagu kirjutuslaud" - link)	Antud kood kuvab paarisarvud -10st kuni 10ni. Kirjuta kood ümber pikkuse suhtes optimaalsemalt (kuni 2 rida). `i = -10` `while i <= 9:` `print(i)` `i += 2` `print(i)`

	Ülesande tüüp	Näiteülesanne
		Optimeeri koodi ning lõpeta programm. `#krossijooksu finiš` `esimene = input("1. lõpetaja: ")` `teine = input("2. lõpetaja: ")` `kolmas = input("3. lõpetaja: ")` `neljas = input("4. lõpetaja: ")` `viies = input("5. lõpetaja: ")` `kuues = input("6. lõpetaja: ")` `seitsmes = input("7. lõpetaja: ")` `kaheksas = input("8. lõpetaja: ")` `üheksas = input("9. lõpetaja: ")` `kümnes = input("10. lõpetaja: ")` `koht = input("Sisesta, mitmendat lõpetajat soovid näha: ")`
28	Õpilane tõstab koodifragmendid järjekorda nii, et programm annaks oodatava vastuse.	Tõsta koodiread õigesse järjekorda. `while salasona != "Salas6na" and kordi <= 3:` `else:` `    print("Kasutaja on lukustatud")` `    print("Sisselogimine õnnestus")` `kordi = 0` `if kordi >= 3:` `    salasona = input("Sisesta parool: ")` `salasona = input("Sisesta parool: ")` `    kordi += 1`

VII. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Ruth Schihalejev**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Õppeprotsessi kognitiivsetele tasemetele vastavad ülesanded kursusel „Programmeerimine“**, mille juhendaja on Tauno Palts, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Ruth Schihalejev

25.05.2021