TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Mihkel Pae

Tartu Ülikooli programmeerimise algkursuse kodutööde täiendamine

Bakalaureusetöö (9 EAP)

Juhendaja: Reimo Palm, PhD

Tartu 2021

Tartu Ülikooli programmeerimise algkursuse kodutööde täiendamine

Lühikokkuvõte:

Tartu Ülikooli aines Programmeerimine (LTAT.03.001) on üle aastate kodused ülesanded olnud korduvad. Seetõttu on kodutööde lahenduste jagamine õpilaste vahel sage juhtum. Plagieerimine oleks raskem, kui õppejõududel oleks rohkem ülesandeid, mille seast valida. Käesolevas töös uuritakse ning selgitatakse välja, millised koduülesanded on hetkel liiga rasked, liiga kerged või sootuks ebavajalikud. Vastavalt sellele luuakse suurem hulk ülesandeid iga teema kohta. Autor teeb ülevaate MOOCil põhinevatest (ingl *massive open online course*) programmeerimise algkursustest maailmas ning võrdleb neid Tartu Ülikooli programmeerimise kursusega. Ülesandekomplekti loomisel kirjeldatakse ülesannete koostamise põhimõtteid ning selgitatakse, kuidas neid ülesandeid tehniliselt kasutada.

Võtmesõnad:

MOOC, Python, programmeerimisülesanded, ülesannete kogu, programmeerimise algõpe

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Renewal of homework in the basic course of programming in the University of Tartu

Abstract:

In University of Tartu, home assignments in the course Programming (LTAT.03.001) have been unchanged from year to year. Therefore sharing homework solutions among students is a common occurrence. Plagiarism would be more difficult if lecturers had more assignments to choose from. In this thesis the current assignments are examined to determine which tasks are too demanding and which tasks are unchallenging or even redundant. Afterwards a large number of tasks will be created for each topic to improve upon these issues. The author gives an overview of other basic MOOC (Massive Open Online Course)-based programming courses in the world and compares them to the programming course of the University of Tartu. When creating a set of tasks, the principles of creating tasks are described and it is explained how to use these tasks technically.

Keywords:

MOOC, Python, programming tasks, collection of tasks, teaching programming to novices

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

Sisukord

Mõisted ja terminid

Ennik (*tuple*) - Andmetüüp, mis sarnaneb järjendile. Enniku elemendid kirjutatakse ümarsulgude vahele, kuid indekseerimisel kasutatakse nurksulge ning erinevalt järjendist on ennik muudetamatu.
Muutuja (*variable*) - Aadress mälus, mis hoiab mingit väärtust.
Operaator (*operator*) - Aritmeetilise või loogilise tehte sümbol.
Rekursioon (*recursion*) - Protsess, kus näiteks arvutiprogrammis defineeritud funktsioonis on kasutatud parasjagu defineeritavat funktsiooni.
Silumine (*debugging*) - Protsess, mille käigus tuvastatakse ja eemaldatakse programmis olevad vead.
Sprait (*sprite*) - Arvutigraafikas kasutatav kahemõõtmeline pilt.
Sõne (*string*) - Suvalise pikkusega märgijada, mida Pythonis ümbritsevad jutumärgid või ülakomad.
Tingimustoiming (*conditional operation*) - Võimaldab määrata lausete valikulist täitmist sõltuvalt tingimustest.
Tõeväärtustüüp (*boolean*) - Andmetüüp, milles on vaid kaks võimalikku väärtust, tõene ja väär.
Ujukomaarv (*floating-point number*) - Reaalarv, mille iseloomulikuks tunnuseks (erinevalt täisarvust) on ka murdosa.

1. Sissejuhatus

Programmeerimine (LTAT.03.001) [1] on Tartu Ülikooli üks populaarsemaid kursuseid. Aine kuulub 13-sse erinevasse õppekavasse TÜ-s. Kursus põhineb programmeerimiskeelel Python ning kursuse eesmärgiks on omandada algteadmised programmeerimise põhikonstruktsioonidest ning osata iseseisvalt kirjutada esmaseid programme.

Kuna käsitletavat kursust pole viimastel aastatel uuendatud, võivad mõned teemad ja ülesanded olla aegunud. Probleemi üheks lahenduseks on aine põhjalik läbivaatamine ning võrdlemine teiste programmeerimise algkursustega. Töö esimeses osas võtab autor aluseks nelja ülikooli (Georgia Tehnoloogiainstituut, Michigani Ülikool, Rice'i Ülikool, Toronto Ülikool) MOOCi (ingl *massive open online course*) kursused. MOOC on interneti vahendusel vaba juurdepääsuga e-kursus. Autor kirjeldab nelja ülikooli kursuste läbitavaid teemasid, kestust, õppematerjali, õppeprotsessi, hindamisskeemi ja toob välja iga kursuse iseärasused.

Teises osas teeb autor ülevaate Tartu Ülikooli programmeerimise algkursusest, andes ülevaate, millised on teemad ja ülesanded ning mida oleks tarvis uuendada. Kursuse korrigeerimisel lähtub autor eelkõige teiste ülikoolide programmeerimise kursuste sisust ning õpilaste antud tagasisidest.

Mõned teemad ning ülesanded Tartu Ülikooli programmeerimise algkursusel on liiga lihtsad, teised rasked või ebavajalikud. Tahes-tahtmata lähevad õpilased kergemat teed pidi, otsides mooduseid, kuidas saaks lihtsamalt ülesanded tehtud. Näiteks kirjutatakse tuttavatele, küsitakse abi internetist jm. See tähendab, et ikka ja jälle peavad ülikoolid tegelema plagiaadiküsimustega. Plagiaadiks nimetatakse teiste inimeste sõnade, ideede või loodud töö avaldamist enda omandina ilma, et oleks sellele viidatud [2].

Programmeerimise kursusel peaks kodutööde ülesandeid iga 2-3 aasta tagant uuendama. Viimane põhjalik ümbertegemine antud kursusel toimus 2018. aasta sügissemestril. Kuna vajadused on välja selgitatud esimeses kahes osas, koostab autor töö kolmandas osas ülesandekomplekti ja kirjeldab seda. Antud hetkel on kursusel iga kodutöö juures 1-4 ülesannet ja samad ülesanded korduvad aastast aastasse. Kodutöid on hetkel kokku 13 ning autor koostab iga kodutöö kohta 5-7 ülesannet. Uus ülesandekomplekt parandaks kursuse kvaliteeti ning vähendaks olulisel määral plagieerimisvõimalusi.

2. Ülevaade programmeerimise algkursustest maailmas

Käesolevas peatükis uuritakse mujal maailmas toimuvaid MOOCi programmeerimise algkursuseid. Aluseks võetud MOOCi kursused on sarnaselt Tartu Ülikooli kursusele suunatud laiale osalejaskonnale, ei nõua eelteadmisi ning on õppijate hulgas populaarsed. Samuti on nende sisu ja õppeprotsess kõigile kättesaadavad. Praeguse olukorra tõttu võib arvata, et tulevikus sisaldab Tartu Ülikooli programmeerimise algkursus järjest rohkem elemente, mis on sarnased MOOCidega. Uuritakse Georgia tehnoloogiainstituudi [3], Michigani Ülikooli [4], Rice'i Ülikooli [5] ning Toronto ülikooli [6] kursuseid. Kursuseid uuritakse nii edX-st kui ka Coursera-st. Autor toob välja iga kursuse ühised omadused ning iseärasused.

2.1 MOOCi keskkonnad

2.1.1 EdX

EdX on 2012. aastal Harvardi ja MIT-i ülikoolide koostöös loodud mittetulundusühing, mis on loonud vaba juurdepääsuga e-kursusi pakkuva keskkonna. Sellist vaba juurdepääsuga kursust nimetatakse MOOCiks (ingl *massive open online course)*. Juuli 2020 seisuga oli keskkonnas üle 3500 kursuse ning oma panuse oli andnud üle 150 õppeasutuse [7]. EdX-s on ligikaudu 33 miljonit registreeritud õpilast.

Enamikuga kursustest on keskkonnas võimalik tutvuda tasuta. See tähendab, et kursuse materjalidele pääseb ligi ning kursust on võimalik läbida, kuid sellega ei kaasne sertifikaati. Sertifikaati on siiski võimalik saada, kuid selleks tuleb maksta [8].

Kursused on jaotatud nädalasteks etappideks. Igal nädal on kursuse läbijale ette nähtud lühivideod, millele järgnevad interaktiivsed harjutused, kus õpilane saab koheselt harjutada videos õpitut. Järje peal püsimiseks tuleb õpilasel igal nädalal läbida temale määratud ülesanded. Hätta jäämisel on abiks juhendid ning arutelufoorumid (ingl *discussion forum*), kus ühel kursusel osalejad saavad hinnata üskteise töid ja olla üksteisele abiks [9].

2.1.2 Coursera

Coursera on 2012. aastal kahe Stanfordi professori poolt asutatud MOOCi keskkond. Sarnaselt edX-le võimaldab ka Coursera vaba juurdepääsuga e-kursusi. Coursera pakub üle 4000 kõrge kvaliteediga kursuse rohkem kui 150 ülikoolilt ja firmalt. Näiteks on kursusi nii

Stanfordi, Londoni ülikoolidelt kui ka tippfirmadelt Google ja IBM [10]. Sarnaselt edX-le on võimalik ka Courseras kursuste materjalidega tutvuda. Sõltuvalt kursusest ei pruugi õpilane Courseras tasuta versiooni puhul ligi pääseda kõikidele kursuse kodutöödele ja ülesannetele. Kui õpilane siiski tahab ligi pääseda kõikidele kursuse materjalidele ning töödele, pakub keskkond 7-päevast tasuta prooviversiooni [9].

Kursused kestavad üldjuhul neli kuni kaksteist nädalat. Igal nädalal on ette nähtud üks kuni kaks tundi videomaterjali, lugemismaterjalid, ülesanded ja projektid. Tasulise versiooni puhul kontrollivad automaatkontrolli puudumisel õpilaste lahendusi õppejõud. Samuti saavad õpilased anda tagasisidet üksteise töödele. Alates 2017. aastast pakub Coursera täismahus magistrikraadi omandamist [10].

2.2 Kursuste ühised omadused

Tartu Ülikooli kursus Programmeerimine keskendub ainult programmeerimiskeelele Python, mistõttu võttis autor võrdluseks samuti Pythoni algkursused. Kursused on e-õppe vormis ning põhinevad nelja – Georgia, Michigani, Rice'i ning Toronto Ülikoolide materjalidel. Kursused algavad iga nädala või kahe nädala tagant. Ühele kursusele registreerunud õpilastel on ühine arutelufoorum, kus saab üksteist aidata, küsida nõu ning anda tagasisidet üksteise töödele.

Kursused on üles ehitatud nädalaste etappidena. Selle ajaga tuleb läbi töötada, sõltuvalt kursusest, videomaterjalid, lugemismaterjalid ning lahendada ülesanded. Iganädalasel koduülesandel on võimalik anda kaasõpilaste tehtud kodutöödele tagasisidet. Coursera-s näiteks on üheks ülesandeks iga nädal üle vaadata ja hinnata vähemalt viie kaasõpilase tööd. Selline õpetamismetoodika on kasulik, sest õpilased näevad, et nende kaasõpilased näevad vaeva ja on pühendunud, mis omakorda motiveerib ka neid pingutama.

Kursustel osalemise jaoks pole tarvis eelteadmisi, s.t pole tarvilik tunda mõnda teist programmeerimiskeelt. Seetõttu tutvustatakse alguses programmeerimist üldiselt, leitakse vastus küsimustele, miks me programmeerime, ning mida on võimalik programmeerimisega saavutada. Nii saavad õpilased edaspidises faasis paremini aru, miks ja kuidas arvuti mõnda koodijuppi täidab.

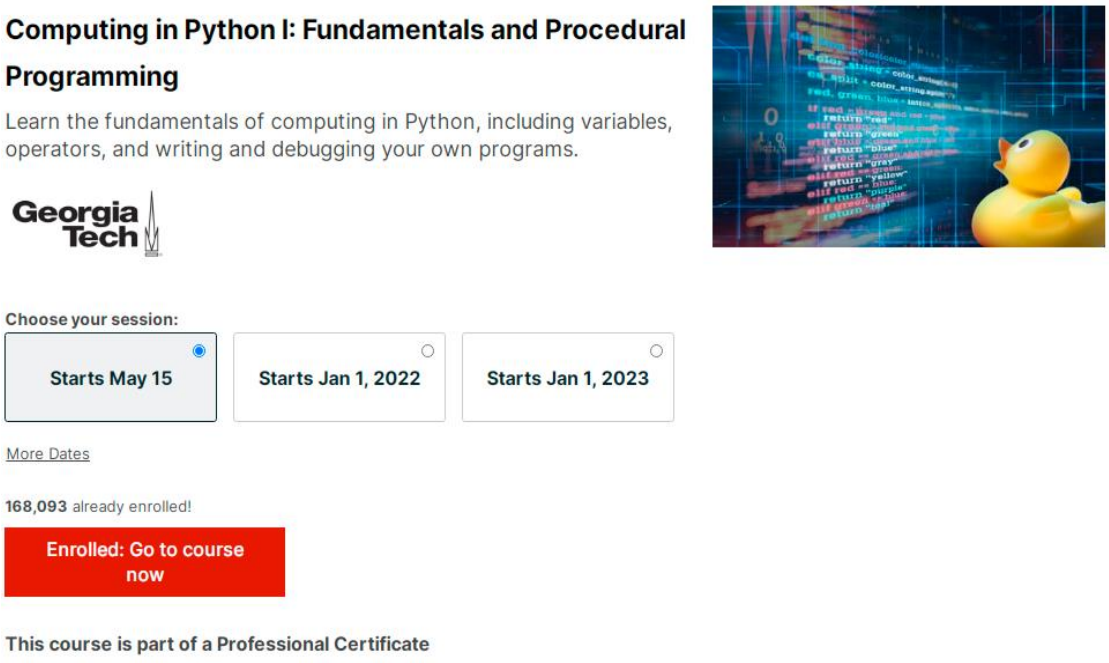
Kursuste eduka läbimise puhul väljastatakse õpilastele sertifikaat, kuid seda vaid tasulise versiooni korral. Enamus MOOCe on lõpetanud tasuta sertifikaatide jagamise, kuna selline süsteem ei olnud jätkusuutlik ning liiga paljud õpilased alustasid kursust, kuid ei lõpetanud.

2.3 Kursuste iseärasused

Antud peatükis tehakse ülevaade neljast kursusest eraldi. Tuuakse välja, mille poolest üks või teine kursus eristub. Võrreldakse läbitavaid teemasid, õppematerjale, kursuse kestust. Lisaks uuritakse kursuse läbijate tagasisidet jm.

2.3.1 Arvutamine Pythonis I-IV (*Computing in Python I-IV*). Georgia Ülikool

Antud kursust uurib autor keskkonnast edX. Varem oli Computing in Python edX-s ühtse tervikuna ning kursuse pikkuseks oli 16 nädalat. Igal nädalal oli ette nähtud 10 tundi õppemahtu. Kursuse haldajad leidsid, et kuna enamik kursusi edX-s kestavad 4-5 nädalat, on vajalik see väiksemateks osadeks jagada. Nüüd jaguneb Computing in Python neljaks eraldi kursuseks, kus iga kursuse kestus on 4-5 nädalat. Nelja kursuse edukal läbimisel väljastatakse õpilasele kutsetunnistus, mis on kasulik näiteks tööle kandideerimisel. Kursuse materjalidele on võimalik ligi pääseda ka tasuta, kuid sel juhul ei kaasne kutsetunnistust [13].



Joonis 1. Pilt EdX-st kursuse esimese osa avalehelt [3].

Kursused koosnevad õppevideotest, lugemisülesannetest, programmeerimisülesannete lahendamisest ning projektidest. Iga õppevideo all on video kohta käiv küsimus või ülesanne, millele peab õpilane õigesti vastama. Vale vastuse korral saab õpilane uuesti proovida. Samuti on kursusel testid, mis pole küll kohustuslikud, aga siiski kujundavad lõpphinde. Tasuta versiooni puhul ei saa õpilane osaleda hindelistel töödel, aga edasi liikumiseks peavad olema ülesanded läbitud ja seega läbib õpilane hindelised tööd automaatselt. Õpilane läbib kursuse

juhul, kui ta on saanud testidest, projektidest, eksamist ja aktiivsuse eest kokku üle 60% punktidest. See tähendab, et mõni ülesanne või projekt võib olla tegemata, aga kui kogu punktisumma on vähemalt 60%, saab õpilane sertifikaadi ja on kursuse läbinud (vt tabel 1). Kõigi nelja kursuse läbimisel saab õpilane kutsetunnistuse. Kutsetunnistust aktsepteerivad paljud ülikoolid ning ka näiteks tööle kandideerimisel tuleb kasuks kutsetunnistuse ettenäitamine.

Tabel 1. Kursuse hindamissüsteem.

	Aeg	Alampiir	Max punktid
Ülesandekomplektid	1. nädal - 4. või 5. nädal, sõltuvalt kursusest	60% punktidest	70% kursuse kogupunktidest
Eksam	Viimane nädal	60% punktidest	30% kursuse kogupunktidest

Kursusel kasutatakse uut mugavat nutiraamatu (Smartbook) tehnoloogiat (vt joonis 2). Nutiraamat kasutab masinõppega välja töötatud mudelit, mis ühildab õppimise ja hindamise ning isikupärastab õppeprotsessi. Nutiraamatu kasutamine antud kursusel pole kohustuslik, kuid annab õppimisele enesekindlust ning loob hea õppekeskkonna.

Joonis 2. Smartbook võimaldab korraga õppida ning harjutada [3].

Esimeses osas (Fundamentals and Procedural Programming) räägitakse programmeerimisest üldiselt - mis see on, kus seda kasutatakse ja miks seda kasutatakse. Käiakse läbi sellised teemad nagu silumine, kompileerimine, muutujad, kommenteerimine, loogikaoperaatorid ning matemaatilised operaatorid (vt tabel 2).

Teises osas (Control Structures) tutvustatakse tingimuslauseid, tsükleid, funktsioone ja vigadega toimetulekut. Need on esmatähtsad teadmised algaja programmeerija jaoks.

Kolmandas osas (Data Structures) räägitakse, nagu nimigi ütleb, andmestruktuuridest. Õpetatakse käsitlema sõnesid, järjendeid, faili sisendit ja väljundit ning sõnastikke. Õpilased on väitnud, et see osa on nende jaoks kõige raskem.

Neljandas osas (Objects & Algorithms) õpitakse objekte ja algoritme: objektorienteeritud programmeerimine, ajalise keerukuse hindamine, rekursioon, sorteerimisalgoritmid, otsingualgoritmid jm. Need on järgmised suured teemad programmeerimises. Selle kursuse lõppedes ootab õpilast probleemide komplekt (ingl *Problem set*), mis on nii-öelda ettevalmistus eksamiks. Probleemide komplekt valmistab õpilase hästi ette, kuna need ülesanded on raskemad, võrreldes eksami ülesannetega. Komplekt katab kõiki neljal kursusel õpitud teemasid. Probleemid võivad olla raskemad kui need, mida õpilane siiani kohanud on, mistõttu lahendusteni jõudmine võib aega võtta tunde. Mõni nendest probleemidest on näiteks algelise plagiarismidetektori kirjutamine, *blackjacki bot*-i tegemine jm.

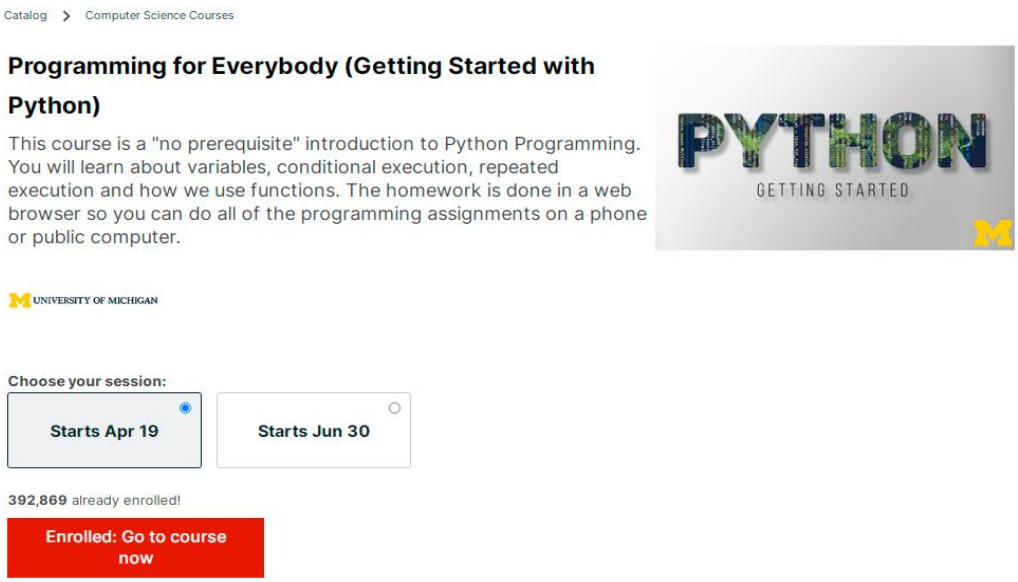
Tabel 2. Nelja kursuse ajakava nädalate kaupa.

I osa	
1. nädal	Sissejuhatus, arvutamine Pythoniga, silumine
2. nädal	Protseduuriline programmeerimine, kommenteerimine
3. nädal	Muutujad
4. nädal	Loogilised operaatorid, matemaatilised operaatorid
II osa	
5. nädal	Juhtimisstruktuurid
6. nädal	Tingimuslaused
7. nädal	Tsüklid
8. nädal	Funktsioonid
9. nädal	Vigade käsitlemine
10. nädal	Andmestruktuurid

III osa	
11. nädal	Sõned, ennik
12. nädal	Järjendid
13. nädal	Faili sisend- ja väljundfunktsioonid
14. nädal	Sõnastikud
IV osa	
15. nädal	Objektorienteeritud programmeerimine, rekursioon
16. nädal	Ajaline keerukus, erinevad sorteerimis- ja otsimisalgoritmid

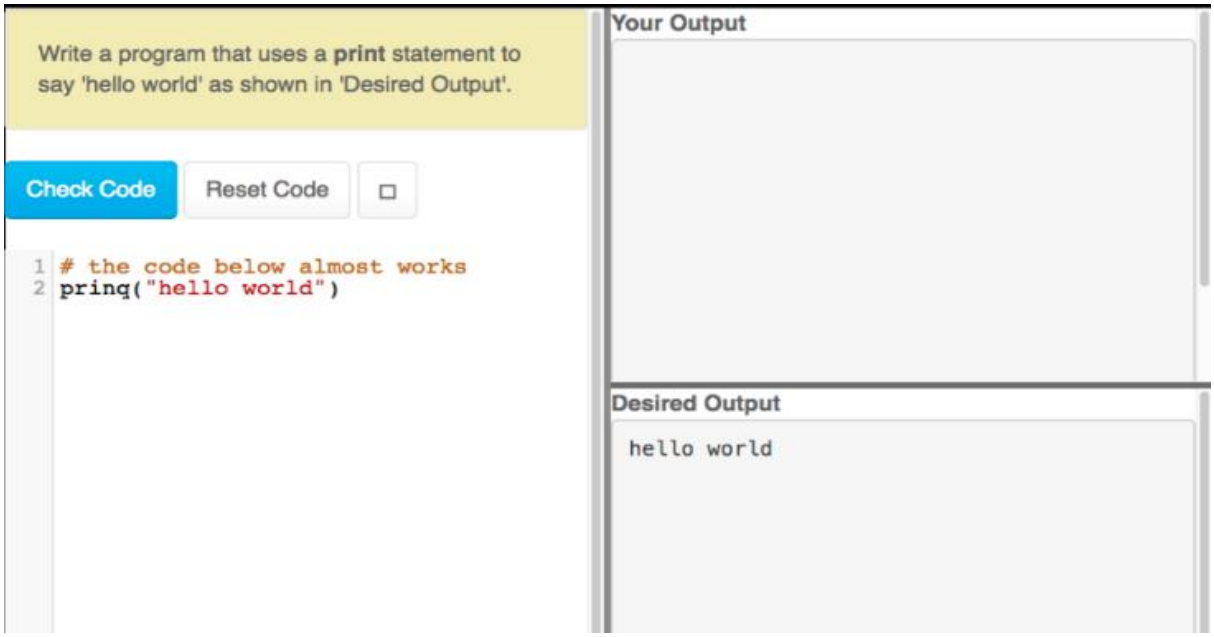
2.3.2 Programmeerimine kõigile (*Programming for Everybody*). Michigani Ülikool

Kursus, mida autor uurib, pärineb taaskord keskkonnast edX (vt joonis 3). Populaarsel lehel Class Central, kus õpilased saavad jagada enda kogemusi antud kursusega ning hinnata kursust viie palli süsteemis, on kursuse keskmiseks hindeks antud 4,9. Suur enamus kursuse läbijatest on kursust kõvasti kiitnud. Tutvustavas osas on öeldud, et kursuse läbimiseks pole vaja eelteadmisi programmeerimisest. Samuti mainitakse, et õpilased, kes on tuttavad mõne teise programmeerimiskeelega, peaks läbima kursuse 2-3 nädalaga, ilma eelneva kogemuseta 5-6 nädalaga. Hetkel on käsitletaval kursusel ainult üks osa. Varem viidi kursust läbi Python 2 versioonis, kuid Python 3 versiooni saabumisel viidi ka see kursus üle uuele Pythoni versioonile.



Joonis 3. Pilt võetud edX kursuse avalehelt [4].

Sarnaselt Tartu Ülikooli kursusele, kus enamikel kodutöödel on automaatkontrollid, on ka siin sarnane tööriist – automaathindaja. Automaathindaja (vt joonis 4) antud kursusel on Michigani Ülikooli poolt välja töötatud ning lingitud otse edX-le. Kui õpilane on automaathindajast mingi ülesandega ühe korra juba läbi saanud, võib õpilane koodi muuta, kustutada, kuid läbinud sooritus jääb püsima.



Joonis 4. Automaathindaja edX kursusel [4].

Antud kursusel omandatakse programmeerimise algteadmised ning kursuse töömaht ei ole sama suur kui näiteks Tartu Ülikooli programmeerimise kursusel. Kursus on jagatud viieks peatükiks. Lugemist on õpilasel vähe, mistõttu on kursusel soovitatud endale ise lisalugemist juurde võtta. Näiteks veebipoest Amazon on võimalik osta sellesama kursuse teemaline raamat, mis süvendab õpitut.

Esimesel nädalal räägitakse programmeerimisest üldiselt. Miks me programmeerime, tutvustatakse arvuti riistvara komponente ja tehakse tutvust Pythoni kui programmeerimiskeelega. Teises ja kolmandas peatükis tehakse tutvust avaldiste, muutujate ning tingimuslausetega. Neljandas peatükis õpitakse käsitlema funktsioone. Viimases peatükis vaadatakse üle tsüklid ja iteratsioonid (vt tabel 3). Peatükid koosnevad suuresti videomaterjalidest ning ülesannetest. Iga peatüki lõpus on inspireeriv intervjuu programmeerimise alal tuntud inimesega, kes räägib, kuidas tema teekond programmeerimises alguse sai.

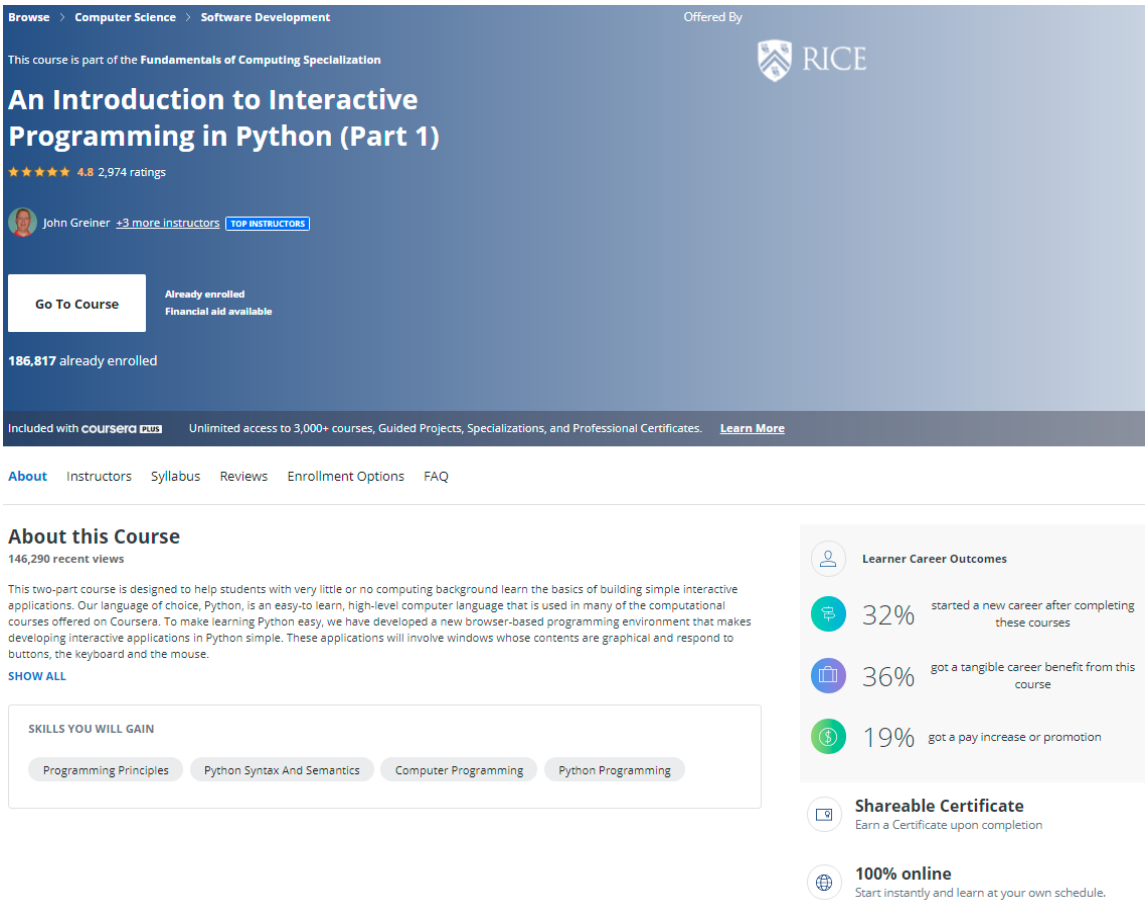
Tabel 3. Kursuse ajakava nädala kaupa.

1. nädal	Sissejuhatus
2. nädal	Avaldised ja muutujad
3. nädal	Tingimuslaused
4. nädal	Funktsioonid
5. nädal	Tsüklid ja iteratsioonid

Sõltuvalt peatükist on iga peatüki lõpus kodutöö või test. Kodutöö koosneb ühest raskemat sorti probleemist või ülesandest. Autori arvates on kursus natukene aegunud, kuna pole interaktiivseid harjutusi ja pole vaheldust õpilastele. Kodutöödel hätta jäämisel on abiks foorumid, kus ühes klassis osalevad õpilased saavad aidata üksteist. Kursuselt läbi saamiseks peab kodutöödest ning testidest koguma vähemalt 60% punktidest. Kursuse tasuta versiooni puhul ei saa õpilane sooritada hindelisi kodutöid ega pälvi kursuse lõpus sertifikaati. Kursusel on iga nädala lõpus ülesanne, mis võtab tavaliselt 1-2 tundi lahendamiseks. Et kursuselt läbi saada, on vaja kõik 5 ülesannet saada arvestatud.

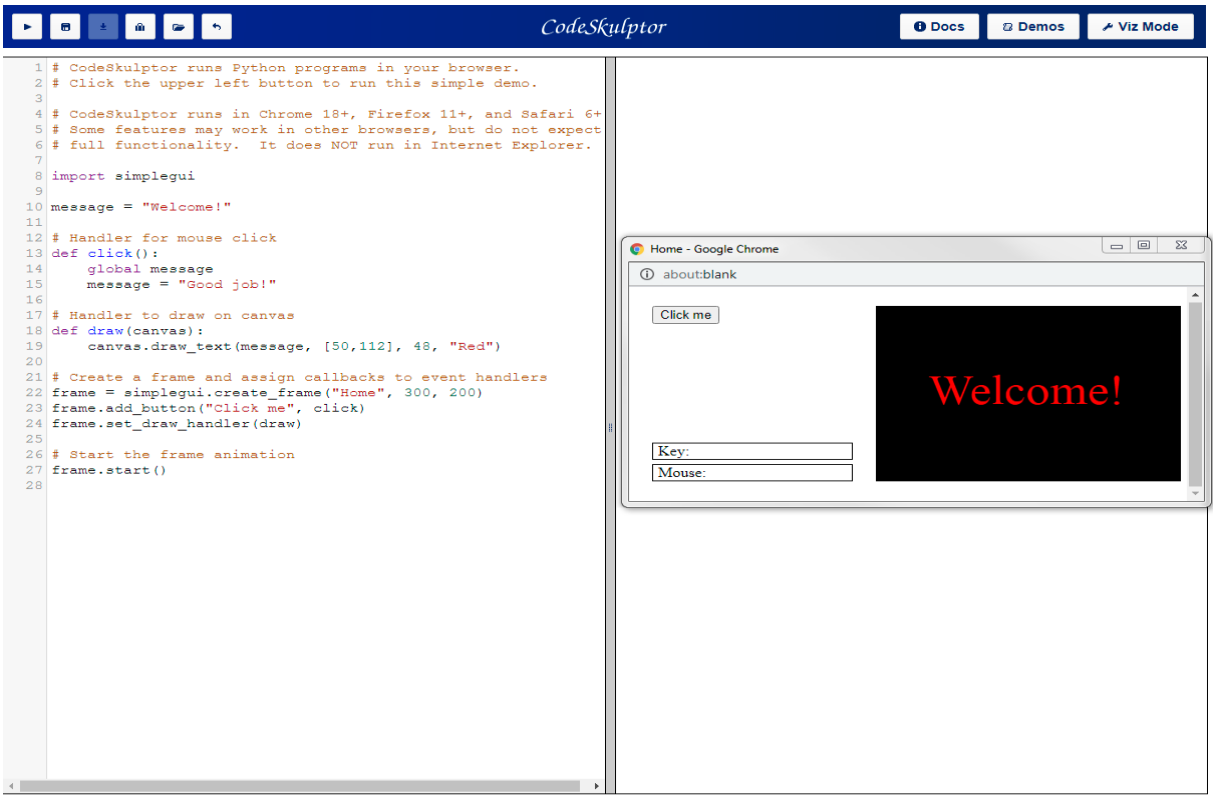
2.3.3 Sissejuhatus interaktiivsesse programmeerimisse Pythonis (*An Introduction to Interactive Programming in Python*). Rice'i Ülikool

Antud kursust uurib autor Coursera-st (vt joonis 5). Kursusel on kaks osa. Esimese osa kestuseks on 5 nädalat ning teise osa kestuseks 4 nädalat. Leheküljel Class Central on üle 3200 inimese andnud tagasisidet kursusele ning keskmiseks hindeks viie palli süsteemis on antud 4,9. Sarnaselt eelnevatele kursustele ei pea antud kursusele registreerides olema programmeerimise eelteadmisi ning kursust on võimalik läbida tasuta. Sellisel juhul ei kaasne kursuse läbimisel sertifikaati.



Joonis 5. Pilt võetud Coursera-st kursuse esimese osa avalehelt [5].

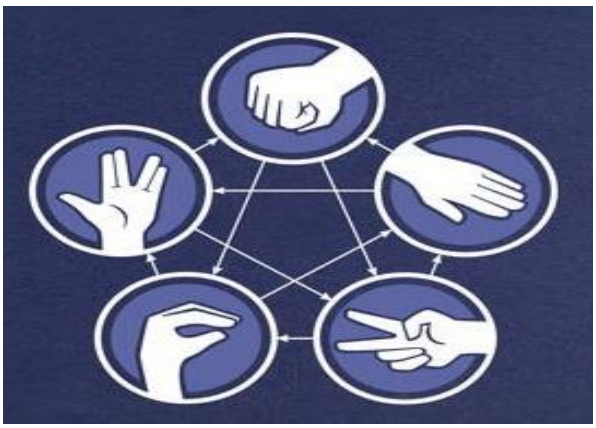
Kursuse esimesel nädalal tutvustatakse CodeSkulptorit (vt joonis 6), mis on veebipõhine arenduskeskkond. Eeliseks on see, et arvutisse ei pea eraldi programmi installeerima. Samas saab Codeskulptoris importida kõiki vajalikke mooduleid ning see on vägagi kasutajasõbralik [15]. Huvitav lähenemine kursusel on see, et kursusel ei tutvustata programmeerimist üldiselt, vaid hakatakse kohe koodi kirjutama. Alustatakse aritmeetilistest avaldistest ning muutujatest. Esimesel nädalal tuleb osalejal läbida test ning osalejatel on ka valikuline miniprojekt. Testist peab osaleja saama vähemalt 70% punktidest.



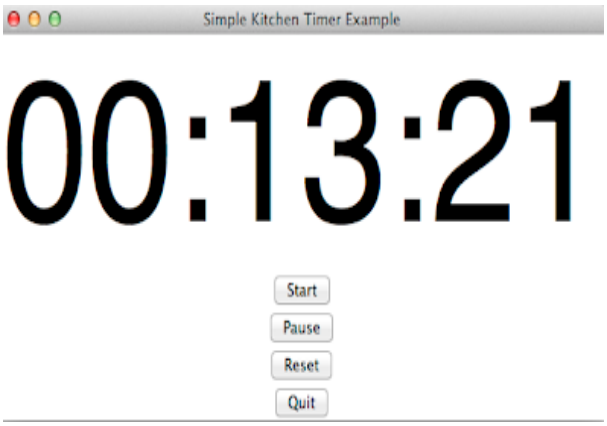
Joonis 6. Pilt CodeSkulptori kasutamisest [5].

Teisel ja kolmandal nädalal õpitakse funktsioone, tingimuslauseid, lokaalseid ja globaalseid muutujaid, graafilisi liideseid, sisendvälju jm. Teisest nädalast on osavõtjatel vaja teha iganädalane projekt. Projektid on huvitavad ning kinnistavad nädala jooksul õpitut. Teisel nädalal tuleb näiteks teha projekt nimega „Kivi-paber-käärid-sisalik-Spock“ mäng (vt joonis 7), mis on modifitseeritud mäng teada-tuntud kivi-paber-käärid mängust. Selline projekt juba teisel nädalal näitab, et tempo kursusel on üsnagi kiire ja õpilastel tuleb vastavalt kohaneda. Hätta jäämisel on abiks foorumid ning abi saab ka internetist. Samuti on vaja õpilastel teha iganädalane test, mis on kohustuslik kursuse läbimiseks.

Neljandal ja viiendal nädalal uuritakse sõnesid, järjendeid, klaviatuurisisendeid ning jätkatakse graafiliste liidestega - õpitakse kanvaad (ingl *canvas*), interaktiivset joonistamist, visualiseerimist, kokkupõrget (ingl *collision*) ja peegeldust (ingl *reflection*). Neljanda nädala projektis tuleb teha interaktiivne köögitaimer (vt joonis 8).



Joonis 7. Kivi-paber-käärid-sisalik-Spock mängu formaat.



Joonis 8. Neljanda nädala projekti näidistöö.

Teises osas jätkatakse sealt, kus esimeses osas pooleli jäädi. Leheküljel Class Central kursuse hindeks antud keskmiselt 4,8 (viie palli süsteemis). Esimesel ja teisel nädalal vaadatakse üle hiiresisend, järjendifunktsioonid, sõnastikud, klassid ning objektorienteeritud programmeerimine. Sarnaselt murelahendajale Tartu Ülikooli kursusel on siin koodikliinik (ingl *Code Clinic*) ehk kui projektitöös jäädakse hätta, siis saab koodikliinikust kasulikke nõuandeid. Viimasel kahel nädalal õpitakse algelist mängufüüsikat, spraitide kokkupõrget ning animatsioone. Teise osa projektitööd on lihtsamate veebimängude tegemine. Näiteks tuleb teha kasutajaliidesega *Blackjack* ning mäng, kus peab kosmoselaevaga meteoriitide eest põgenema. Tabelis 4 on toodud kursuse ajakava nädalate kaupa.

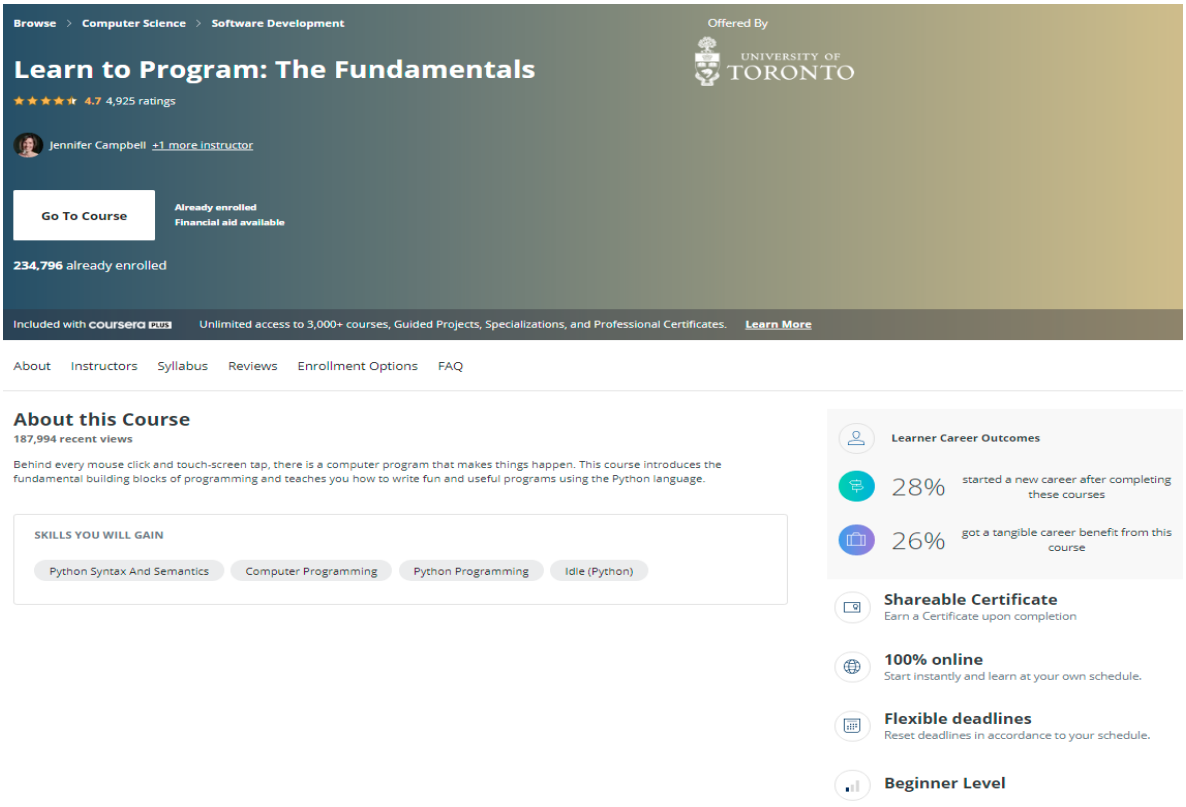
Tabel 4. Kursuse ajakava nädala kaupa.

I osa	
1. nädal	Avaldised, muutujad, lihtlaused
2. nädal	Funktsioonid, loogika, tingimuslaused
3. nädal	Sündmusjuhitud programmeerimine, lokaalsed/globaalsed muutujad
4. nädal	Kanvaa, joonistamine, taimer
II osa	
5. nädal	Järjendid, klaviatuurisisend, liikumise modelleerimine
6. nädal	Hiiresisend, järjendifunktsioonid, sõnastikud
7. nädal	Klassid ja objektorienteeritud programmeerimine
8. nädal	Mängufüüsika, spraidid
9. nädal	Animatsioonid

Kursusel on iganädalased testid, kus tuleb läbimiseks saada vähemalt 70% punktidest. Esimesel kahel nädalal on üks test nädala kohta ning seejärel edaspidi kaks testi nädala kohta kuni teise osa lõpuni välja. Testi lahendamiseks on 30 minutit ja test koosneb tavaliselt kümnest küsimusest. Alatest teisest nädalast on vaja teha iga nädal ka üks nn miniprojekt. Kõikides miniprojektides on vaja saada arvestus, et kursust läbida. Miniprojekte hindavad nii kaasõpilased kui ka kursuse läbiviijad.

2.2.4 Õpime programmeerima: Põhialused (*Learn to Program: The Fundamentals*). Toronto Ülikool

Käesolevat kursust uurib autor Coursera-st. Kursuse tagasiside keskmine hinne leheküljel Class Central on 4,8. Kursuse kestuseks on 7 nädalat, õppimine on intensiivne ning õpitakse tundma enamikku Pythoni põhitõdedest. Kui varasemate kursuste juures on mõne teema kohta videoloeng või lugemismaterjal, siis sellel kursustel on kõikidel teemadel nii video kui ka lugemine.



Joonis 9. Pilt võetud kursuse avalehelt [6].

Kursuse esimesel ning teisel nädalal tehakse ülevaade kursuse hindamiskriteeriumidest ning kursuse logistikast. Räägitakse Pythonist üldiselt ning näidatakse, milliseid programme on võimalik Pythoniga teha. Samuti räägitakse arvuti mälust,

mis on programmeerimisel tähtis. Õpitakse tundma Pythonis muutujaid, matemaatilisi tehteid, sisendit-väljundit, sõnesid, sisseehitatud funktsioone ja funktsioonide defineerimist. Teise nädala lõpus on kodutöö, kus pannakse proovile osalejate õpitu kahe nädala jooksul. Kodutöö teemaks on ajavööndite teisendamine. Kodutööl oodatakse õpilaselt funktsioonide ja matemaatiliste tehete oskuseid.

Kolmandal ja neljandal nädalal korratakse kõigepealt esimesel kahel nädalal õpitut. Lisaks õpitakse konverteerimist – näiteks täisarvutüüpe sõnedeks. Tehakse ka tutvust tingimuslausetega, *for*-tsüklitega ning silumisega.

Kui kursuse keskel tundub, et tempo ei ole liiga kiire, siis viimased kolm nädalat on õppijatele vägagi suure koormusega. Vaadatakse üle erinevad tsüklid, järjendimeetodid, tüübiloendid, kuidas asjakohaselt kommenteerida, faili lugemine ja kirjutamine, korteežid, sõnastikud. Teemasid ei pruugi olla palju, aga iga teema vaadatakse põhjalikult üle. Kursuse lõppedes eksam, mis kestab 30 minutit. Eksam koosneb 16 küsimusest, küsimusi on enamiku teemade kohta, mida kursusel õpiti, ning läbimiseks tuleb igaühest saada vähemalt 80% punktidest. Tabelis 5 on toodud täpne kursuse ajakava nädalate kaupa.

Tabel 5. Kursuse ajakava nädala kaupa.

1. nädal	Muutujad, funktsioonid
2. nädal	Sõned, funktsioonid
3. nädal	Tõeväärtustüüp, importimine, tingimuslaused
4. nädal	Tsüklid, sõneoperatsioonid
5. nädal	Tsüklid, järjendid
6. nädal	Failide lugemine/kirjutamine, indeksid, järjendite järjendid
7. nädal	Ennikud, sõnastikud

Kursusel on kokku 7 testi, mille läbimiseks tuleb saada vähemalt 80% punktidest. Testi saab sooritada 3 korda iga 8 tunni tagant. Teisel, neljandal ja kuuendal nädalal on programmeerimisülesanded, mis on üpriski mahukad. Keskmiselt võtab ülesande lahendamine aega 3 tundi. Kõigil kolmel ülesandel tuleb saada maksimaalsed punktid, et läbida. Kursuse tegijad on öelnud, et eeldatakse õpilaselt mitu korda kodutöö läbi lugemist, enne kui õpilane ülesandest täielikult aru saab. Kursuse viimasel nädalal on eksam, mis moodustab 25% lõpphindest.

3. Ülevaade Tartu Ülikooli programmeerimise kursusest

Antud peatükis uurib autor, mis olukord on Tartu Ülikooli kursusel. Autor teeb ülevaate kursusel käsiteldavatest teemadest, ülesannetest, materjalidest ja hindamisskeemist. Kursust võrreldakse eelnevas peatükis käsitletud kursustega ning tuuakse välja kasulikud asjad, mida saaks Tartu Ülikooli kursusele üle võtta ja tehakse ka lühike ülevaade õpilaste tagasisidest antud kursusel.

3.1 Olukord kursusel

Antud kursust uurib autor kursuse kodulehelt courses.cs.ut.ee ning Moodle'ist. Kursuse kestuseks on 16 nädalat. Programmeerimise algkursuse eesmärgiks on õpilastele edasi anda teadmised programmeerimise põhikonstruktsioonidest ning oskused koostada esmaseid algoritme ja programme. Kursuse mahuks on 6 EAP, kusjuures ühe EAP maht on 26 tundi. Seega keskmiselt peab õpilane panustama 156 tundi oma ajast kursuse läbimiseks. Kursus koosneb 32 arvutipraktikumist, 92 tunnist iseseisvast tööst ning erinevatest loenguvideodest. Kursus toimus 2020/2021. aasta sügissemestril esimesed 12 nädalat tavapäraselt kontaktõppes, ülejäänud semester toimus kursus pandeemia tõttu e-õppe vormis. Kursuse tagasisides on mainitud, et kursus on vähemalt sama mahukas, kui mitte isegi mahukam kui 6 EAP-d. Sellel kursusel maha jäädes on raske järje peale jõuda. Kursuse läbinud tudengid on tagasisides öelnud, et teha tuleks kõik harjutused ja ülesanded, mis antakse. Samuti tuleks lahendada ka mittehindelised harjutused, et valmistuda paremini kontrolltöödeks ja eksamiks. Kursuse õppejõud on olnud läbi aastate väga abivalmid tudengeid aitama.

Kursus on üles ehitatud nädalaste etappidena. Igal nädalal võetakse uus teema või korratakse eelmisel nädalal õpitut (vt tabel 6). Kokku on kursusel 13 kodutööd ja 13 praktikumitööd. Ühe kodu- ja praktikumitöö eest on võimalik saada kuni 0,5 punkti. Kodutöödes on 1-4 ülesannet ja praktikumitöös kuni 5 ülesannet. Alampiir koduülesannete ja praktikumide ülesannete eest on 7 punkti. Esitama peab üsna palju ülesandeid ning õpilase jaoks on tempo suhteliselt kiire. Loengute eest on kokku võimalik saada 7 punkti, 0,5 punkti iga loengu eest. Usinamatel õpilastel on võimalik lisaülesandeid lahendada ja lisapunkte teenida. Kokku on 10 lisaülesannet, kus iga ülesande eest on võimalik maksimaalselt saada 1 punkt. Kursuse keskosas, kui õpilastel on baasteadmised omandatud, tuleb leida omale kaaslane, idee ning hakata projekti tegema. Projekti idee antud kursusel on autori arvates kasulik, kuna paneb õpilaste fantaasia tööle ning õpetab õpilasi iseseisvaks programmide

kirjutamiseks. Projekti eest saab maksimaalselt 10 punkti ning alampiir on 5 punkti. Ilma alampiiri täitmata pole võimalik kursuselt läbi saada. Kursusel on kolm suuremat hindelist tööd, millest kaks on kontrolltööd ning üks on eksam. Kontrolltööd on vastavalt 6. ning 12. nädalal. Kontrolltööl on kaks osa – Moodle'i test ning programmeerimine. Moodle'i test on arvestuslik ning programmeerimise osa annab maksimaalselt 20 punkti. Kontrolltöös küsitakse kõike seni õpitut. Eksam toimub tavaliselt jaanuarikuus ning koosneb arvestuslikust ja punktilisest osast. Kokku on võimalik eksamilt saada 30 punkti. Nii kontrolltööde kui eksami sooritamiseks on tarvis saada arvestus arvestuslikus osas.

Tabel 6. Tartu Ülikooli programmeerimise algkursuse hindamisskeem.

	Aeg	**Alampiir**	**Max**	**Kommentaar**
Loengud	1.-16. nädal	-	7	0,5 punkti loengust
Kodutöö+praktikum	1.-16. nädal	7	13	0,5+0,5 kodutööst ja praktikumist
Projekt	7.-16. nädal	5	10	Kahes osas (5+5)
1. kontrolltöö	6. nädal	Arvestusliku osa arvestus	20	Kahes osas: arvestuslik ja punktiline osa
2. kontrolltöö	12. nädal	Arvestusliku osa arvestus	20	Kahes osas: arvestuslik ja punktiline osa
Eksam	jaanuaris	Arvestusliku osa arvestus	30	Kahes osas: arvestuslik ja punktiline osa
Lisaülesanded		-	10	Kokku 10 lisaülesannet kursuse peale ära jaotatud

Tabel 7. Kursuse ajakava nädala kaupa.

Nädal	Teema
1. nädal	Sissejuhatus
2. nädal	Avaldised ja lihtlaused
3. nädal	Tingimus- ja korduslaused
4. nädal	Funktsioonid
5. nädal	Algoritmid
6. nädal	Kordamine. Järjendid

7. nädal	Järjendid ja for-tsükkel
8. nädal	Järjendite kasutamise skeemid
9. nädal	Järjendite muutmine
10. nädal	Andmestruktuurid
11. nädal	Andmestruktuurid II
12. nädal	Kordamine
13. nädal	Rekursioon
14. nädal	Rekursioon II
15. nädal	Mitmesuguseid algoritme
16. nädal	Kordamine ja projekt

3.2 Tartu Ülikooli kursus võrreldes välismaa ülikoolide kursustega

Suurimaks erinevuseks Tartu Ülikooli kursusel võrreldes teiste välismaiste kursustega on tempo. Antud Tartu Ülikooli kursusel ei ole lugemis- ja videomaterjali oluliselt rohkem kui teistel kursustel, küll aga on palju kodutöid ja ülesandeid. Tartu Ülikooli kursus on üles ehitatud nõnda, et kiirematel õppijatel oleks ka „huvitav“. On antud palju harjutusülesandeid ja lugemismaterjale, mida kiiremad õppijad saavad uurida ning seeläbi lihvida oma programmeerimise oskust veelgi. Aeglasemad õppijad ei pruugi kõike informatsiooni ja ülesandeid hoomata, sest ühe nädalaga kaetakse ära suur hulk teemasid. Samuti on Tartu Ülikooli programmeerimise algkursuses enamus ülesandeid hindelised ning mõjutavad suurel määral lõpphinnet. Erinevalt teistest kursustest on Tartu Ülikooli kursus hindeliselt eristav. Autori arvates on Tartu Ülikooli kursuse metoodika tõhusam, kuna õpilased tahavad head hinnet saada ja see paneb tihtipeale rohkem pingutama.

Struktuuri poolest sarnaneb Tartu Ülikooli algkursus kõige rohkem neljaosalisele Georgia Ülikooli kursusele. Teemad on suhteliselt sarnased, kuigi Georgia Ülikoolis ei uurita selliseid teemasid nagu näiteks rekursioon ja ei pöörata nii palju rõhku failide lugemisele/kirjutamisele. Samas pööratakse palju rõhku objektorienteeritusele, mida Tartu Ülikoolis õpitakse jällegi kursus hiljem, vähemasti informaatika erialal.

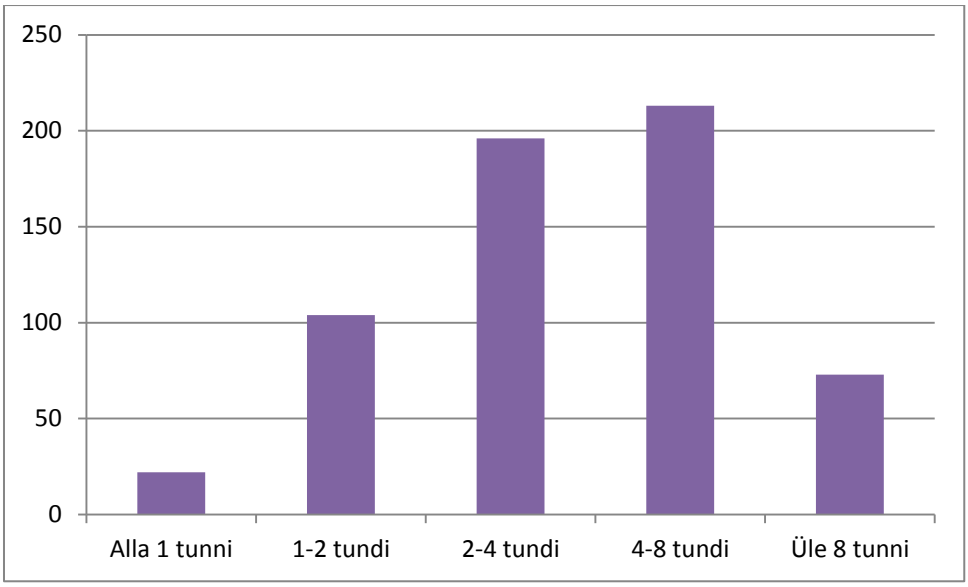
Antud Tartu Ülikooli kursusel proovitakse hoolitseda selle eest, et ükski õpilane maha ei jääks. Selleks on kursusel olemas konsultatsioonid, abi saab küsida õppejõududelt, oma praktikumijuhendajalt ning erinevalt rahvusvahelisest kursusest on lihtsam küsida abi

kaasõpilastelt. Õpilastele on tehtud mittehindelised „Kontrolli ennast“ testid, kus saab katsetada, kas üks või teine teema on kinnistunud.

3.3 Programmeerimise kursuse õpilaste tagasiside

Käesoleva bakalaureusetöö peamiseks eesmärgiks on teha uus ülesandekomplekt Tartu Ülikooli programmeerimise kursusele. Kursuse korraldajad on koostanud küsimustiku, kus õpilased saavad anda tagasisidet eesmärgiga parandada kursuse kvaliteeti. Samuti teeb see järgmisel kursuse toimumisel õpilastel õppimise ning õppejõududel õpetamise sujuvamaks.

Kursuse jooksul oli igal nädalal õpilastel võimalus vastata küsimustikule, kus muuhulgas küsiti eelneva nädala kodutöö keerulisust. Küsiti näiteks „Kui palju aega kulutasid kodus selle nädala teemadele (sh materjali lugemine ja koduülesannete lahendamine)?“ ning „Kui keeruline oli sinu jaoks selle nädala kodutöö?“. Paljud õpilased väitsid tagasisides, et kodutööd on tihti liiga rasked, võtavad liiga kaua aega ning loengumaterjalid pole kooskõlas koduste ülesannetega. Joonis 10 näitab, kui palju aega kulutasid õpilased nädalas teemade läbivaatamiseks, sealhulgas materjalide lugemisele ning koduülesannete lahendamisele.



Joonis 10. Kui palju aega kulus õpilastel nädalas materjalide läbi töötamiseks ja koduülesannete lahendamiseks nädalatal 2-15. Y-teljel on märgitud vastanud õpilaste koguarv.

Mõningad õpilased väitsid, et kodutööde püstitus oli tihtipeale segadust tekitav ja kursuse kodulehe õpikus ning loenguslaididel polnud piisavalt infot, et koduülesandega hakkama saada. Kindlasti tuleks õpilastele jagada vihjeid, kui alguses mõni ülesanne raske

tundub. Algaja programmeerija jaoks võib olla kahe osa kokkupanek keeruline ning ei pruugi osata internetist abi otsida. Samas on programmeerimisoskuse arendamine üsna ajarohke ning kõik ei pruugigi algul vajalikke oskuseid kiirelt selgeks saada.

Tagasisidest selgus, et osad teemad olid üleliigsed või mõnda teemat oli liiga palju. Näiteks kolmandal nädalal õpiti Pykkarit (Pykkar on moodul, mis tekitab endale mängumaailma ning kus saab robotit mängumaailmas liigutada käskude abil) ja mõnegi õpilase arvates oli antud teema segane ning ebavajalik. Õpilased väitsid mitme nädala jooksul, et failist lugemine ja faili kirjutamine oli nende jaoks kõige raskem teema ning seda teemat oleks võinud rohkem praktikumides korrata.

Programmeerimise algkursuse ainet Tartu Ülikoolis võtavad mitmete õppekavade õpilased, mistõttu ei tohiks aine läbimine eeldada mingeid teadmisi muudest valdkondadest. Näiteks 10. nädala kodutöös oli juttu maatriksitest, mida mõnes õppekavas ei õpita ning see ajas õpilasi segadusse.

Õpilased on tagasisides öelnud, et kursusel tekkis kodutööde ja harjutustega lumepalliefekt. Kui liiga palju kuhjub, siis jääbki kuhjuma ja raske oli järele jõuda. Autor on sellega nõus. Õnneks olid kursusel olemas konsultatsioonid, mida õpilased kiitsid ja pakkusid välja, et neid võiks isegi rohkem olla.

4. Ülesandekomplekti kirjeldus

Antud peatükis teeb autor ülevaate ülesandekomplektist. Kirjeldatakse, mis on ülesannete koostamise head tavad ning samas ka vead, mida tihtipeale tehakse.

4.1 Eelprotsess

Lister [11] on öelnud, et iga õppejõu ülesanne peaks olema realiseerida õpilase täispotentsiaal. Selleks, et õpilastelt kätte saada nende potentsiaal, tuleb materjalide koostamisel vaadata materjali läbi õpilase silmade. Kui õpilane saab materjalist aru nii, nagu materjali autor seda mõelnud on, omandab õpilane teadmised kiiremini.

Igal kursusel on tugevamad ja nõrgemad õpilased. Antud kursus on mõeldud eelkõige algteadmiste omandamiseks ning õpilaste programmeerimise oskuse ühtlustamiseks. Tihtipeale leiab programmeerimise kursustest ülesandeid, mille püstitus on segane isegi kogenud programmeerijale. Programmeerimisülesanded on enamasti tekstülesanded, mistõttu leiavad õpilased tihti end olukorrast, kus tekstist arusaamine valmistab rohkem raskusi kui programmeerimine ise. Programmeerimise ülesannete koostamise üks põhitavasid on, et ülesande sõnastus oleks selge, arusaadav ning täpne. Samuti peaks olema tekst kaasahaarav ning ülesanne võiks seostuda õpilase endaga. Seda on autor üritanud jälgida ülesandekomplekti loomisel.

Antud kursus on kolmeteistkümnes erinevas õppekavas, mistõttu ülesannete lahendamine ei tohiks eeldada teadmisi mõnest muust valdkonnast, näiteks kõrgemast matemaatikast. Matemaatiliste ülesannete lahendamine võib tekitada õpilastele segadust ning arusaamatust, kuna õpilased eeldavad, et on vaja matemaatilisi oskusi. Seetõttu võib õpilastel kaduda huvi ning ei realiseerita enda potentsiaali.

4.2 Ülesannete koostamine

Iganädalasele kodutööle koostas autor 5-7 ülesannet. Keskmiselt on kodutöös vaja lahendada 2-3 koduülesannet, seega saavad õppejõud ka järgnevatel semestritel valida autori koostatud komplektist erinevaid ülesandeid. See parandab kindlasti kursuse kvaliteeti ning vähendab plagiarismi. Küll aga ei olnud autori töö peamine eesmärk plagiarismi vähendamine. Autor loodab, et uue ülesandekomplektiga on õpilastel koduülesannete lahendamine huvitavam ja motiveerivam. Samuti tahab autor näidata algajatele programmeerijatele, kuidas on võimalik päriselu seostada programmeerimisega.

Ülesannete koostamisel jälgis autor, et ülesanded oleksid kooskõlas antud kursuse materjalidega ning ülesanded oleksid lahendatavad kõigile. Ülesandekomplekti koostamine koosnes pikast protsessist. Kõigepealt tutvuti teemaga ning tehti endale selgeks, milliseid funktsioone, elemente antud peatükis käsitletakse. Seejärel vaadati ülesannete raskuseid nii Tartu Ülikooli kursusel kui ka Georgia Tehnoloogiainstituudi, Michigani Ülikooli, Rice'i Ülikooli ning Toronto Ülikooli kursustel. Peale seda mõeldi välja ülesannete taust. Mida elulisem taust, seda huvitavam õpilasel on. Kirjutati valmis ülesande programm ja püstitus. Vajadusel lisati ka ülesandesse vihjeid ja näpunäiteid. Autor pani kirja, millised on peamised reeglid, mida võiks programmeerimise kursuse kodutööde koostamisel järgida:

- Ülesande püstitus peab olema selge ja täpne.
- Mida rohkem kasutatakse ülesande püstitusel elulist konteksti, seda paremini hoomab õpilane ülesannet.
- Ülesandel ei tohiks olla vaid üks lahendus. Õpilased peaksid taipama, et ülesandele saab mitut erinevat moodi läheneda.
- Keerulised matemaatilised ülesanded ei too alati välja õpilaste potentsiaali.

Koostatud ülesandekomplekti on autor läbi lahendanud, et näha, kui raske on ühte või teist ülesannet lahendada. Samuti proovis autor lahendada ülesandeid mitmel viisil, kuna igal ülesandel võiks olla mitmeid mooduseid lahenduseni jõudmiseks.

4.3 Ülesandekomplekt

Kokku loodi ülesandeid kolmeteistkümne nädala kodutöödele. Autor on koostanud 74 uut ülesannet, mistõttu kõiki ülesandeid eraldi tutvustada ei ole otstarbekas. Autor toob välja mõned koostatud ülesannete püstitused ning moodused, kuidas õpilane ülesannetele lähenema peaks. Ülesannete püstitused on üpriski sarnased. Kõigepealt on ülesannetes räägitud kontekstist, seejärel kirjeldatud nõudeid programmile ning lõpetuseks esitatud näiteväljund(id) ning vihje(d). Näiteväljundid annavad paremini mõista, mida ülesannetes täpsemalt küsitud on, ning vihjed aitavad õpilasi ideede puudumisel.

Pitsa

Pitseerias on müügil väike, keskmine ning suur pitsa.
Väikese pitsa läbimõõt on 25 cm ning see maksab 5.50 €.
Keskmise pitsa läbimõõt on 33 cm ning see maksab 10 €.
Suure pitsa läbimõõt on 40 cm ning see maksab 16 €.

Arvuta, millise pitsa ruutsentimeetri hind on kõige väiksem. Kasutades tingimuslauset, prindi tulemused välja kujul:
„Väikese pitsa ruutsentimeetri hind on kõige väiksem“
„Ruutsentimeetri hind antud pitsal on ...“

Joonis 11. Teise nädala kodutöö üks ülesannetest.

Joonisel 11 toodud ülesanne on eluline ning seega on tudengitel ülesannet huvitav lahendada, kuna ülesannet saab seostada endaga. Ülesanne kontrollib õpilaste loogikat ning kasutab tingimuslauset, mida õpitakse teisel nädalal. Samuti tuleb ülesande lahendamiseks kasutada matemaatilist konstanti π, mida uuritakse esimeses ning teises praktikumis. Nädala kodutööd koosnevad tavaliselt 1-2 lihtsamast ning 1-2 keerukamast ülesandest. Joonisel toodud ülesanne kuulub teise nädala ülesannete seas lihtsamat sorti ülesannete sekka.

Jõusaal

Tauri tahtis hakata käima jõusaalis. Ta tegi uuringu, kui palju maksavad jõusaalides piletid ning kirjutas tulemused faili *piletid.txt*, kus igas reas on kirjas jõusaali nimi, pileti hind ning kuude arv, mitmeks kuuks antud pilet kehtib. Tauri soovis teada, millises jõusaalis ning millisel piletil on kõige väiksem kuumaks. Kirjuta programm, mis arvutab, milline pilet on kõige soodsam ning prindib välja vastava jõusaali ning kuumaksu antud piletil.

Faili *piletid.txt* sisu:
Lõvikonservi jõusaal, 96.0,6.0
Jõusaal kanajalad, 22.0,1.0
Rind trimmi, 47.0,3.0

Joonis 12. Seitsmenda nädala kodutöö üks ülesannetest.

Joonisel 12 toodud ülesanne „Jõusaal“ on juba keerulisem. Seitsmendal nädalal õpitakse järjendeid ning *for*-tsüklit. Antud ülesandes on vaja osata mõlemat. Kõigepealt tuleks faili lugeda *for*-tsükliga ning seejärel kasutada *split*-funktsiooni. Ülesanne on huvitav, kuna lahendamiseks on mitmeid viise. Ülesande lahendamiseks võib kuluda mõnda aega, sest õpilastel on selleks perioodiks osade kokkupanemine veel raske. Samas on ülesanne arendav, eluline ning paneb kõik senimaani õpitu proovile.

4.4 Ülesannete kasutamine

Gomes [12] on öelnud, et tavaliselt on õpilaste suurim takistus programmeerimise ülesannete lahendamisel ülesannete tekstist arusaamine. Selleks võib olla kaks põhjust. Esiteks, ülesande koostaja on tihtipeale juba kogenud programmeerija, ning ei pruugi taibata, kui vahepeal mõni tema jaoks lihtne ülesanne võib õpilastele osutuda keeruliseks. Seega peab ülesande koostaja olema hoolas, et ka mittekogenud programmeerijad saaksid ülesande püstitusest üheselt aru. Antud töö autor on üritanud panna rõhku ülesannete püstituste lihtsusele ja arusaadavusele. Teiseks põhjuseks on see, et õpilased ei ole läbi töötanud kursuse materjale. Tartu Ülikooli kursusel on materjalideks programmeerimise õpik [14], loenguvideod ning praktikumid ja konsultatsioonid. Koostatud ülesandekomplekti lahendamine eeldab, et õpilased on läbi töötanud kursuse materjalid. Ülesannete lahendamisel ei tohiks vaja minna interneti abi, kuigi vahel enda ideede puudumisel on seda soovituslik kasutada. Ideede puudumisel saab ka abi kas praktikumijuhendajatelt või konsultatsioonidest.

Ülesannete lahendamisel peaks õpilased kõigepealt lugema läbi kogu ülesande teksti. Kui ülesande tekst jääb esimesel korral arusaamatuks, tuleks lugeda ka teist korda ning panna kokku kõigepealt suur pilt - mida tahetakse, millised nõuded peavad olema täidetud. Tavaliselt on ülesandes olemas näidisväljund(id) või näidis(ed) programmi tööst, mis aitavad õpilastel aru saada, mida ülesannetes täpsemalt küsitud on. Kõigepealt tuleks õpilastel peas läbi mõelda või panna paberile kirja funktsioonid ja toimingud, mida ülesannete lahendamiseks tarvis on. Kui kohe alguses kogu pilti kokku ei pane, tuleks ülesanne juppideks teha. Seejärel, kui mõningad ülesande osad on valmis, tuleks proovida neid omavahel ühildada. Alati tasub ka koodiridadega proovida ja mängida. Näiteks tasub katsetada, et mis juhtub, kui koodiridu natukene muuta, vahetada või asendada. Kuna ülesannetel on lahenduskäike mitmeid, on hea kui õpilased oskavad mitut eri moodi ülesannet lahendada.

5. Kokkuvõte

Lõputöö eesmärgiks oli luua uus ülesannete kogu Tartu Ülikooli kursuse Programmeerimine (LTAT.03.001) jaoks. Antud töös loodi kokku 74 ülesannet 13 nädala kodutöödele. Kahjuks ei jõutud koguda tagasisidet õpilaste käest, mis oleks autorile andnud parema ülevaate ülesannete sobilikkusest. Ülesannete koostamisel jälgiti eelkõige seda, et ülesannete püstitused oleks üheti mõistetavad, ülesanded oleksid huvitavad ja elulised, kataksid peatükkides õpitut ning lisaks kordaks ka varasemaid peatükke.

Enne ülesannete koostamist tutvuti nelja välismaa ülikooli MOOCi kursustega, uuriti materjale ning tehti tutvust koduste ülesannetega. Seejärel tehti põhjalik uuring Tartu Ülikooli programmeerimise algkursusele, vaadati üle kursuse õpik [14], loengud, kodutööd ning õpilaste tagasiside. Tehti selgeks, millist laadi ülesanded on vajalikumad ning sellest tingitult mõeldi kõigepealt välja ülesannete kontekst, pandi kirja ülesannete püstitus ning tehti ka valmis lahenduskoodid. Ülesannetele lisati näidislahendused ja -väljundid, vajadusel ka vihjed. Ülesannete lahenduste saamiseks võib pöörduda kursust Programmeerimine (LTAT.03.001) läbiviivate õppejõudude poole.

Edasiarendamise võimalusteks on koguda õpilastelt tagasisidet loodud ülesandekomplekti kohta ja vastavalt vajadusele viia sisse muudatused. Lisaks saaks teha antud komplektile ka automaatkontrollid, mis lihtsustaks õppejõudude tööd ja aitaks õpilastel aru saada enda tehtud vigadest. Samuti saab ka ülesannetele koostada murelahendajad, mis hätta jäämisel aitaksid õpilasi.

6. Viidatud kirjandus

[1] Tartu Ülikooli kursus „Programmeerimine“ (LTAT.03.001). https://ois2.ut.ee/#/courses/LTAT.03.001 (05.05.2021)

[2] Maurer, H. A., Kappe, F., & Zaka, B. (2006). Plagiarism-A survey. Journal of Universal Computer Science, *12*(8), 1050-1084. (20.03.2021)

[3] Arvutamine Pythonis I-IV. https://www.edx.org/course/computing-in-python-i-fundamentals-and-procedural (25.03.2021), https://www.edx.org/course/computing-in-python-ii-control-structures (27.03.2021), https://www.edx.org/course/computing-in-python-iii-data-structures (27.03.2021), https://www.edx.org/course/computing-in-python-iv-objects-algorithms (28.03.2021)

[4] Programmeerimine kõigile. https://www.edx.org/course/programming-for-everybody-getting-started-with-pyt (30.03.2021)

[5] Sissejuhatus interaktiivsesse programmeerimisse Pythonis I-II. https://www.coursera.org/learn/interactive-python-1 (30.03.2021), https://www.coursera.org/learn/interactive-python-2 (30.03.2021)

[6] Õpime programmeerima: Põhialused. https://www.coursera.org/learn/learn-to-program (31.03.2021)

[7] EdX Learner Help Center. What is edX? https://support.edx.org/hc/en-us/articles/115011202727-What-is-edX- (25.03.2021)

[8] Edukatico. Review of Coursera, edX, and Udacity: What's Good and What's Bad https://www.edukatico.org/en/report/review-of-coursera-edx-and-udacity-what-s-good-and-what-s-bad (31.03.2021)

[9] Oktavia, T., Prabowo, H., & Supangkat, S. H. (2018, September). The comparison of MOOC (massive open online course) platforms of edX and Coursera (Study case: Student of programming courses). In *2018 International Conference on Information Management and Technology (ICIMTech)* (pp. 339-344). IEEE.

[10] What is Coursera? https://www.businessinsider.com/what-is-coursera (26.03.2021)

[11] Lister, R., & Leaney, J. (2003, January). First year programming: Let all the flowers bloom. In *Proceedings of the fifth Australasian conference on Computing education-Volume 20* (pp. 221-230).

[12] Gomes, A., Carmo, L., Bigotte, E., & Mendes, A. (2006, September). Mathematics and programming problem solving. In *3rd E-learning Conference–Computer Science Education* (pp. 1-5).

[13] Georgia Tech, McGraw-Hill and edX Take Undergraduate Computer Science Online. https://www.mheducation.com/news-media/press-releases/georgia-tech-mooc-undergraduate-computer-science.html (01.04.2021)

[14] Programmeerimise õpik. https://progeopik.cs.ut.ee (10.04.2021)

[15] Ajayi, R. O. (2014). Interactive game programming with Python (CodeSkulptor). https://www.theseus.fi/bitstream/handle/10024/81687/Ajayi_Richard.pdf?sequence=1 (30.03.2021)

Lisad

I Lisafailid

Lisadesse on kaasa pandud kokkupakitud fail (kodutööd.zip), mis sisaldab töö jooksul koostatud 13 nädala kodutöid. Ülesannete nägemiseks tuleb fail lahti pakkida.

Lahenduskoodid on saadetud kursust Programmeerimine (LTAT.03.001) läbiviivatele õppejõududele.

II Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Mihkel Pae,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Tartu Ülikooli programmeerimise algkursuse kodutööde täiendamine**, mille juhendajaks on Reimo Palm, 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.

Tartus, **07.05.2021**