TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Markus Leivo

Enesekontrolliks mõeldud programmeerimisülesannete loomine kursusele „Programmeerimise alused II“

Bakalaureusetöö (9 EAP)

Juhendaja: Heidi Taveter, MSc

Tartu 2025

Enesekontrolliks mõeldud programmeerimisülesannete loomine kursusele „Programmeerimise alused II“

Lühikokkuvõte: Kursus „Programmeerimise alused II“ on Tartu Ülikoolis toimuv kursus, mille eesmärk on süvendada ja täiendada üliõpilaste esmaseid teadmisi programmeerimiskeeles Python. Bakalaureusetöö eesmärk on luua kursusele automaatkontrolliga enesekontrolliülesanded, mis toetavad üliõpilaste iseseisvat õppimist ja praktiliste programmeerimisoskuste arendamist. Ülesanded põhinevad kursuse õppematerjalil ning nende loomisel on arvesse võetud õpikavandusmudeli ADDIE põhimõtteid. Ülesannete lahendamine võimaldab välja selgitada potentsiaalseid puudujääke üliõpilaste teadmistes. Enesekontrolliülesanded on mõeldud lahendamiseks peale konkreetse nädala kohta käiva teoreetilise osa läbitöötamist ja enne kodutöö juurde asumist. Ülesannete eesmärk on võimaldada õppuritel iseseisvalt oma teadmisi hinnata ja seeläbi juba omandatud materjali tõhusamalt kinnistada. Kõik see annab kodutööde lahendamiseks parema ettevalmistuse. Bakalaureusetöö käigus koostati Lahenduse keskkonnas 35 automaatkontrolliga enesekontrolliülesannet. Ülesanded on kättesaadavad 2025. aasta kevadel toimunud kursuse õppematerjalides. Kursusel osalejatel paluti anda tagasisidet esimesele kümnele ülesandele. Saadud tagasiside oli valdavalt positiivne, seejuures hinnati ülesannete sisulist ja vormilist esitust ning õppeprotsessi toetavat rolli.

Võtmesõnad: Python, programmeerimine, enesekontrolliülesanded, ADDIE mudel

CERCS: P175 Informaatika, süsteemiteooria, S270 Pedagoogika ja didaktika

Creating Self-Check Programming Exercises for the Course „Introduction to Programming II“

Abstract: The course „Introduction to Programming II“ is offered at the University of Tartu and aims to deepen and expand students' initial knowledge of the Python programming language. The goal of this bachelor’s thesis is to create a set of self-assessment programming tasks for the course that support students’ independent learning and the development of practical programming skills. The tasks are based on the course's study materials and have been designed in accordance with the principles of the ADDIE instructional design model. Solving these tasks helps identify potential gaps in students’ understanding. The self-assessment tasks are intended to be completed after working through the theoretical content of a specific week and before starting the homework assignments. Their purpose is to enable students to independently evaluate their knowledge and thereby reinforce the material already learned more effectively. All this provides better preparation for completing the homework assignments. As part of the thesis, 35 self-assessment tasks with automatic feedback were created in the Lahendus environment. The tasks were included in the course materials during the spring of 2025. Course participants were asked to provide feedback on the first ten tasks. The feedback received was predominantly positive, particularly regarding the content and presentation of the tasks as well as their supportive role in the learning process.

Keywords: Python, programming, self-assessment exercises, ADDIE-model

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

Sisukord

Sissejuhatus

„Programmeerimise alused II“ (ainekood MTAT.03.256, maht 3 EAP) on Tartu Ülikoolis kevadsemestril toimuv kursus, mille eesmärk on täiendada üliõpilaste esmast programmeerimisoskust Pythonis ning süvendada nende algteadmisi programmeerimisest [1]. Õppeaine jaguneb seitsmeks suuremaks põhiteemaks ning iga põhiteema all on mitmeid alateemasid [2]. Kursuse hinne kujuneb kodutööde, praktikumide, testide, projekti, kontrolltöö ja eksamitöö tulemuste põhjal. Kursusel on kasutusel ümberpööratud klassiruumi põhimõte, mis tähendab, et üliõpilased omandavad uue materjali iseseisvalt veebikeskkonna vahendusel. Toimuvad auditoorsed praktikumid, mis võimaldavad õppejõu juhendamisel lahendada eelnevalt iseseisvalt omandatud materjalil põhinevaid ülesandeid.

Varasemalt on loodud „Programmeerimise alused II“ kursusele põhjalik ülesannete kogu, mis sisaldab praktikumi- ja koduülesandeid. Mõlemat tüüpi ülesanded on hindelise kaaluga. Kursuse materjali alateemade lõpus asuvad ka valikvastustega enesekontrolliülesanded, mis võimaldavad kiiresti äsja õpitud teemasid kinnistada. Vähevõitu on aga praktilisi programmeerimisülesandeid, mida saab teemade harjutamiseks vabalt lahendada, ilma et lahendus hindamisele läheks.

Bakalaureusetöö eesmärk on luua kursusele „Programmeerimise alused II“ enesekontrolliks mõeldud programmeerimisülesanded, mis annaksid üliõpilastele võimaluse peale kindla teoreetilise alateema läbimist testida oma teadmisi automaatkontrolliga ülesandeid lahendades. Ülesannete eesmärk on toetada iseseisvat õppimist, võimaldades üliõpilastel välja selgitada potentsiaalseid puudujääke oma teadmistes juba õppematerjali läbides. Ülesanded asuvad kursuse materjalis iga alateema lõpus ning põhinevad sellel konkreetsel alateemal. Ülesande lahenduse koodi on võimalik koheselt alateema lõpus esitada ning arvutiteaduse instituudi Lahenduse keskkonnas loodud automaatkontrolli abil kontrollida lehelt lahkumata [3]. Ülesannete koostamisel lähtutakse ADDIE mudeli põhimõtetest, mille järgi jagatakse protsess viide etappi. Nendeks on analüüs, kavandus, koostamine, rakendus ja hindamine [4]. Töö tulemusi valideeritakse üliõpilastele suunatud tagasiside küsimustiku abil.

Bakalaureusetöö koosneb neljast peatükist. Esimene peatükk annab ülevaate kursuse „Programmeerimise alused II“ õppekorraldusest ning selgitab, miks enesekontrolliülesannete loomine vajalik on. Teine peatükk käsitleb programmeerimise õpetamist Pythonis, õppematerjalide kavandamist ning ADDIE mudelit. Kolmas peatükk kirjeldab ülesannete ja automaatkontrollide väljatöötamise protsessi ning üliõpilastelt tagasiside küsimist. Neljandas

peatükis analüüsitakse tulemusi. Seal antakse ülevaade koostatud enesekontrolliülesannetest ning analüüsitakse üliõpilaste tagasisidet ülesannetele. Töö lõpus olevatest lisadest leiab kõik koostatud ülesanded ning tagasiside küsimustiku.

1. Kursus „Programmeerimise alused II“

Selles peatükis tutvustatakse kursust „Programmeerimise alused II“. Antakse ülevaade kursuse sisust, korraldusest ja kursusel kasutusel olevatest õppemeetoditest. Viimaks kirjeldatakse, milles seisneb vajadus luua kursusele eraldiseisvaid enesekontrolliülesandeid.

1.1 Ülevaade kursusest

„Programmeerimise alused II“ on Tartu Ülikoolis kevadsemestril toimuv kursus, mis on mõeldud eelkõige neile üliõpilastele, kes ei õpi informaatika erialal [1]. 2024/2025. õppeaastal kuulub kursus õppekavadesse järgmiselt:

- valikainete moodulisse „Geoloogia“, „Geograafia“, „Religiooniuuringud ja teoloogia“, „Loodusteadused ja tehnoloogia“ ning „Geoloogia ja keskkonnatehnoloogia“ õppekavadel;
- alusmoodulisse „Füüsika, keemia ja materjaliteadus“ õppekaval;
- suunamoodulisse „Loodus- ja reaalainete õpetamine põhikoolis“ õppekaval;
- vabaainete moodulisse „Poliitika ja valitsemine digiajastul“ õppekaval.

Õppeaine on jätk kursusele „Programmeerimise alused“ ning selle eesmärk on süvendada esmaseid algteadmisi programmeerimisest. Kursus on eeldusaineks mitmetele Tartu Ülikooli kursustele, mis käsitlevad programmeerimist keerulisemal tasemel, näiteks „Objektorienteeritud programmeerimine“, „Programmeerimine II“, „Keeletehnoloogia“ ja „Eesti keele töötlus Pythonis“.

Kursuse õppematerjal, mis on kättesaadav arvutiteaduse instituudi keskkonnas Courses [2], on jagatud seitsmeks suuremaks peatükiks:

1. „Kahemõõtmeline järjend“;
2. „Kahekordne tsükkel“;
3. „Andmestruktuurid“;
4. „Viitamine ja muteerimine“;
5. „Testimine ja silumine. Rekursioon“;
6. „Rekursioon II“;
7. „Objektorienteeritud programmeerimine“.

Õppeaine on mahuga 78 tundi ning selle läbimisel on võimalik saada kolm ainepunkti. Õppeinfosüsteemis on kirjas, et kursuse läbinud üliõpilane peab teadma peamisi andmetüüpe, andmestruktuure, standardoperatsioone, oskama analüüsida programmi tööd ja teadma, kuidas

programmi vajadusel täiendada [1]. Oluline on ka oskus luua algoritme vastavalt probleemile, neid testida ja vajadusel siluda.

Kursusel on programmeerimiskeelena kasutusel Python. Keel arendati välja aastal 1991 [5]. Sellest ajast alates on see olnud üks populaarsemaid programmeerimiseks mõeldud keeli. Keele populaarsust tõestab TIOBE indeks, mille järgi on Python aastal 2025 kõige populaarsem programmeerimiskeel [6]. Indeksi arvutamisel võetakse arvesse programmeerimiskeele nime sisaldavaid päringuid populaarsematel veebisaitidel ning otsingumootoritel nagu näiteks Google, Amazon ja Bing. Teoorias annab indeks hinnangu ka sellele, kui palju on maailmas oskuslikke programmeerijaid, programmeerimisega seotud kursuseid ja töökohti.

Pythoni populaarsust arvestades võiks arvata, et see keel on algajale programmeerijale mõistlik valik. Õpetaja ja teadlase David Beazley [7] sõnul on Python hea kesktee teoreetiliste ja praktiliste programmeerimisülesannete õpetamisel. Ta toob põhjenduseks selle, et Python on interpreteeritav ning seega saab kasutaja koheselt lihtsamaid programme kirjutada, käivitada ja programmi töö tulemust kuvada. Samuti on võimalik Beazley arvates Pythoniga selgeks teha ka keerulisemaid arvutiteadusega seonduvaid kontseptsioone nagu andmestruktuurid või objektorienteeritud programmeerimine. Andmestruktuuride ja objektorienteeritud programmeerimise teemasid puudutab ka kursus „Programmeerimise alused II“. Vaadates kursuse ülesehitust, mis koosneb seitsmest põhiteemast, on näha, et liigutakse lihtsamalt käsitletavate kontseptsioonide pealt, nagu kahemõõtmeline järjend ja kahekordne tsükkel, üle raskematele teemadele, milleks on andmestruktuurid ja rekursioon [2]. Kursus lõpeb objektorienteeritud programmeerimist tutvustava teemaga. See annab üliõpilastele teatud ettevalmistuse uuele keelele üleminekuks, näiteks Javale, mis on samuti objektorienteeritud programmeerimiskeel.

1.2 Õppekorraldus ja enesekontrolliülesannete koostamise vajadus

Kursusel „Programmeerimise alused II“ on kasutusel ümberpööratud õppe meetodid (ingl *flipped classroom*) [2]. Ümberpööratud klassiruum tähendab, et üliõpilased peavad põhiosa materjalist iseseisvalt omandama enne füüsilist tundi [8]. Võrreldes traditsioonilise õppega, annab see üliõpilastele vabaduse materjal läbi töötada vastavalt nende endi võimalustele. Uuringute kohaselt on ümberpööratud klassiruumil mitmeid kasulikke eeliseid [9]. Leitud on, et ümberpööratud õppe puhul on üliõpilaste huvi ja aktiivsus tunni teema vastu suurem, eneseteostus kvaliteetsem ning õppetöö tulemused paremad.

Kursuse loengud on videoloengute vormis ning see võimaldab üliõpilasel endal leida sobiva aja loengute vaatamiseks. Hindamine toimub eristava hindamissüsteemi alusel, kus lõpphinne kujuneb erinevate ülesannete punktide summana [10]. Iganädalaselt toimuvad praktikumid, kus lahendatakse ülesandeid õppejõu juhendamisel. Nädala jooksul tuleb lahendada ja esitada ka kodutöö. Praktikumide ja kodutööde eest on võimalik kokku saada kuni 15 punkti. Olulise osa punktidest moodustab projekt, kus üliõpilased saavad ise püstitada programmeerimisalase ülesande, millele nad kursuse käigus sobiva lahenduse leiavad. Projekti eest saab kuni 20 punkti. Kuni 17 punkti saab Moodle'i teste lahendades ning videotes esitatud H5P küsimustele vastates. Arvestatav osa punktidest moodustub kontrolltöö ja eksami põhjal, kus mõlema puhul saab kuni 26 punkti.

Enamasti toimub õppimine kursusel iseseisvalt, mistõttu on oluline, et õppijatel oleks ka võimalus oma teadmisi iseseisvalt kontrollida. Indoneesia koolis läbiviidud uuringu põhjal aitavad enesekontrolliülesanded pöörata õpilastel rohkem tähelepanu töös tehtud vigadele ja puudustele, mis omakorda motiveerib õppijaid probleemidele lahendusi otsima [11]. Enesekontrolli kasutegurit on seostatud ühe uuringu põhjal ka näiteks enesekindluse tõusuga ning edasiarendamise võimaluste tuvastamisega [12]. Kuigi kursuse õppematerjalide sees leidub juba valikvastuste vormis enesekontrolliülesandeid, mis on osaliselt loodud varasemate bakalaureusetööde käigus, puudub hetkel võimalus lahendada programmeerimist hõlmavaid ülesandeid, mis pakuksid automaatset tagasisidet [13, 14]. Puudust leevendaksid õppematerjalil põhinevad automaatkontrolliga enesekontrolliülesanded, mis võimaldaksid üliõpilastel kohe pärast teoreetilise osa läbilugemist ning valikvastustega ülesannete lahendamist praktilisi ülesandeid lahendada. Ülesanded aitaksid teadmisi kontrollida ja süvendada ning valmistaksid üliõpilasi paremini ette praktikumideks ja kodutöödeks.

2. Programmeerimise õppematerjalide koostamine

Järgnev peatükk annab ülevaate Pythonist kui õppekeelest. Seejärel kirjeldatakse õppematerjalide kavandamise ja loomise protsessi ning peatüki lõpus tutvustatakse ADDIE mudelit.

2.1 Python programmeerimise õppekeelena

Python on kujunenud maailmas üheks populaarsemaks programmeerimiskeeleks. Paljude jaoks on Python esimene programmeerimiskeel, mida koolis õpitakse. Tartu Ülikooli informaatika erialal on Python samuti esimene keel, mida üliõpilastele õpetatakse. Ukraina ülikoolides läbiviidud uuringus tuuakse esile, et esimese keele valik mõjutab üliõpilaste motivatsiooni õppida ning sellest oleneb suuresti ka see, kui edukad sellel alal tulevikus ollakse [15]. Tuuakse välja, et Python on populaarne esimese keelena, sest seda on lihtsam õpetada kui teisi keeli, see on kättesaadav kõigile ning sellele leidub palju vabavaralist abistavat materjali. Sama kinnitab ka Sundnes [16] ning toob lisaks välja, et Python sobib tänu oma lihtsale süntaksile ideaalselt esimeseks keeleks, mille kaudu programmeerimist õppida. Sundnes toob Pythoni juures välja ka ühe olulise puuduse. Kuigi Python on kõrgetasemeline keel, siis keerukamate programmide puhul võib kood kaotada struktuuri seetõttu, et Python ei sunni kasutajat otseselt järgima kindlat koodi kirjutamise stiili. Pellet jt [17] on samuti leidnud, et Javaga võrreldes on Pythonis paljusid programmeerimise kontseptsioone võimalik esitada arusaadavamalt ja ilma liigse süntaktilise keerukuseta. Seetõttu on Python sobiv valik ka kursuse „Programmeerimise alused II“ kontekstis, sest võimalik on keskenduda programmeerimiskontseptsioonide mõistmisele ilma liigsete süntaktiliste ja tehniliste takistusteta.

2.2 Õppematerjalide kavandamine ja loomine

Õppematerjalide loomisel on oluline järgida didaktilisi printsiipe. Sellepärast kasutatakse kursuste ja õppematerjalide loomisel kindlat tüüpi õpikavanduse mudeleid. Gupta [18] toob välja, et levinumad õpikavandusmudelid on näiteks ADDIE mudel, Bloomi taksonoomia (ingl *Bloom’s Taxonomy*), Dick ja Carey mudel (ingl *Dick and Carey Model*), SAM (ingl *Successive Approximation Model*) ja Gagne mudel (ingl *Gagne’s Nine Events of Instruction*). Ta rõhutab, et nende kasutamine aitab tagada õppematerjalide sisulise kvaliteedi ja eesmärgipärasuse. Lisaks märgib ta, et juhendite ja printsiipide alusel õppematerjali loomine muudab kogu protsessi tõhusamaks, säästes aega ja ressursse, sest tulemuseks on selgelt struktureeritud ja

korduvkasutatavad materjalid. Kuigi mudeleid on olemas palju erinevaid, on nende ühine eesmärk toetada õppematerjalide süsteemset ja sihipärast loomist.

Käesolevas töös keskendutakse ADDIE mudelile[1]. Spatioti jt [19] on kirjutanud, et ADDIE mudeli struktuurne terviklikkus, paindlikkus ja lihtsus muudavad selle üheks populaarseimaks õpikavanduse mudeliks. Samuti toovad Spatioli jt uuringus välja, et ADDIE mudeli kasutamine soodustab innovaatiliste õpiruumide kujundamist veebipõhises õppes, mis omakorda mõjub positiivselt õppijate motivatsioonile ja akadeemilistele tulemustele. Drljača jt [20] sõnul on oluliseks tugevuseks ka see, et mudeli üheks osaks on hindamine ja tagasiside andmine, mis võimaldab omakorda analüüsida õppematerjalide kasutegurit. Vajadusel saab selle põhjal sisse viia parandusi. ADDIE mudel on valitud käesoleva töö keskseks õpikavandusmudeliks, sest eelnevalt välja toodud eeliseid arvesse võttes, sobib mudel hästi veebipõhise õppekeskkonna vajadustega ning toetab eesmärgipärast õppematerjalide arendamist.

2.3 ADDIE mudel

Eesti Kõrg- ja Kutsehariduse Kvaliteediagentuuri ehk EKKA poolt loodud juhendi põhjal jaguneb ADDIE mudel viieks järjestikuseks etapiks [21]. Järgnevates lõikudes esitatakse iga etapi lühikirjeldus, kus tuuakse esile olulisemad eesmärgid õppematerjali loomise kontekstis.

Analüüsi etapp hõlmab endas vajaduste, tingimuste, sihtrühma ja sisu analüüsi [22]. EKKA juhendile toetudes, kirjeldatakse erinevaid analüüsietapi osi järgmiselt [21]. Vajaduste analüüsis seatakse paika õppematerjali eesmärgid ja õpiväljundid, lähtudes õppijate olemasolevatest teadmistest. Tingimuste analüüs keskendub olemasolevaletele ressurssidele, tehnoloogilistele võimalustele ja kursuse korralduslikele aspektidele. Sihtrühma analüüsi eesmärgiks on luua ettekujutus õppijate taustast, motivatsioonist ja õppimisvõimest. Sisu analüüsi põhimõtteks on struktureerida kursuse materjal vastavalt õpiväljunditele ja õppijate tasemele.

Kavandamise etapis luuakse õppematerjali esialgne kavand, mille alusel hakatakse materjale koostama [21]. Seejuures pannakse paika struktuur, õpitegevused ja ajakava. Etapi lõplikuks eesmärgiks on luua terviklik ettekujutus õppematerjalidest ja nende toimimisest [4].

[1] Hariduse ja kasvatuse sõnaraamatu järgi (https://arhiiv.eki.ee/dict/haridus/):
ADDIE mudel (ingl *ADDIE-model*) on õpikavanduse mudel, kus kursuse loomine jagatakse viieks etapiks: analüüs (ingl *Analysis*), kavandus (ingl *Design*), koostamine (ingl *Development*), rakendus (ingl *Implementation*) ja hindamine (ingl *Evaluation*). Nimetus ADDIE tuleb etappide inglisekeelsete nimetuste esitähtedest.

Koostamise etapp on protsess, mille käigus valmib terviklik õppematerjal. Selles faasis luuakse kõik eelnevalt planeeritud õppematerjalid [4]. Materjalide loomisel järgitakse paika pandud soovitusi, näiteks juhiste ja tekstide loomisel kasutatakse kindlaid formaate [21]. Oluline on see, et tekstid oleksid haaravad ja loetavad. Loetavust mõjutab oluliselt ka teksti kujundus ning seega on oluline kinni pidada kindlatest vormistamise põhimõtetest. Hoolitsetakse selle eest, et tehniline pool oleks korras ja toimiv. Tavaliselt viiakse läbi kvaliteedikontroll, et tagada vastavus disaini etapis seatud eesmärkidele [23]. Sageli hõlmab see protsess ka materjali testimist, mille kaudu on võimalik hinnata materjalide sobivust enne lõplikku rakendamist.

Rakendamise etapis kasutatakse loodud õppematerjale reaalses õppeprotsessis. Eesmärgiks on tagada, et nii õpilased kui ka õpetajad on valmis materjali kasutama [23]. Vajadusel tutvustatakse kasutatavaid vahendeid, metoodikat ja õppematerjali struktuuri, et õppeprotsess kulgeks sujuvalt ja tõhusalt [21]. Eelnevalt tagatakse ka juurdepääs kõikidele vajaminevatele materjalidele.

Hindamise etapis selgitatakse välja, millisel määral on saavutatud õpieesmärgid ning kuidas toetab loodud materjal õpilaste arengut [21]. EKKA juhendis on ära ka märgitud see, et tavaliselt saadakse olulisem tagasiside õpilastelt, kuid hindajateks võivad olla ka juhendajad ja õpetajad, kes õppematerjali aitasid koostada. Tagasiside küsimine on oluline selleks, et tuvastada õppematerjalis puudujääke. Tagasiside põhjal on võimalik kursust või õppematerjali tulevikus edasi arendada ja täiendada [22].

3. Metoodika

Peatükk käsitleb enesekontrolliülesannete koostamise protsessi, mille käigus rakendati ADDIE mudeli põhimõtteid. Seejärel antakse ülevaade tagasiside küsimustikust.

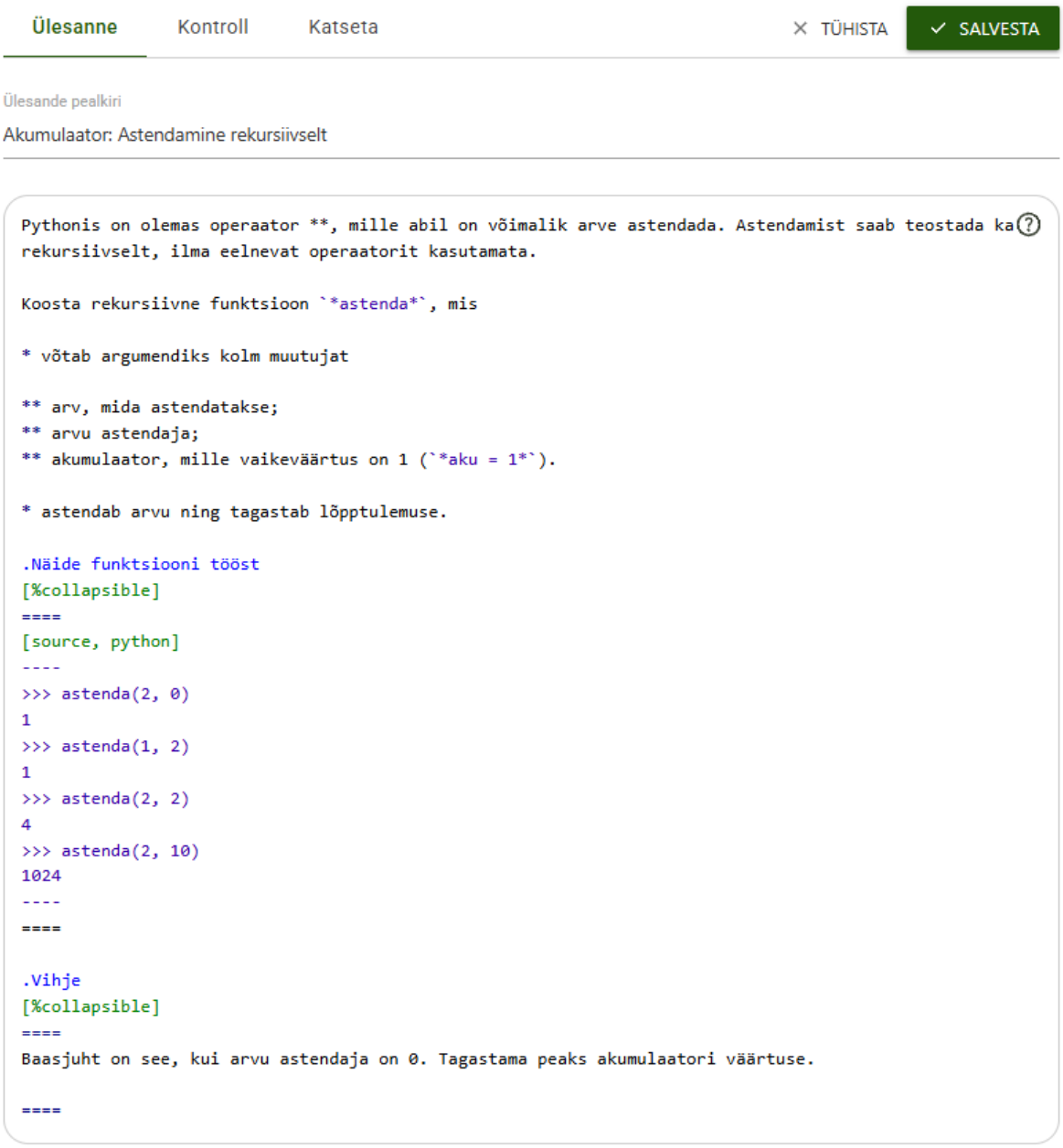
3.1 Ülesannete koostamise protsess

Ülesannete koostamise protsess põhines ADDIE mudeli viiel etapil – analüüs (ingl *Analysis*), kavandus (ingl *Design*), koostamine (ingl *Development*), rakendamine (ingl *Implementation*) ja hindamine (ingl *Evaluation*).

Protsess algas analüüsi etapist, mille käigus töötati läbi olemasolev kursuse materjal. Kuna loodavad ülesanded pidid põhinema konkreetse kursuse materjalil ning olema sama raskustasemega, tutvuti esmalt põhjalikult Coursese lehel oleva lugemismaterjaliga [2]. Lisaks loeti ja osaliselt ka lahendati läbi kursuse Moodle'i keskkonnas olevad kodutööde juhendid [24]. Seda tehti selleks, et aru saada, millist tüüpi enesekontrolliülesanded toetaksid kõige paremini kodutöödeks valmistumist. Kodutööde analüüsi tulemusel otsustati, et loodavad ülesanded peaksid olema lihtsama raskusastmega ning valdavalt elulise sisuga. Seejuures püüti vältida temaatikat, mis kattuks olemasolevate koduülesannete sisuga, et õppematerjal oleks mitmekesisem.

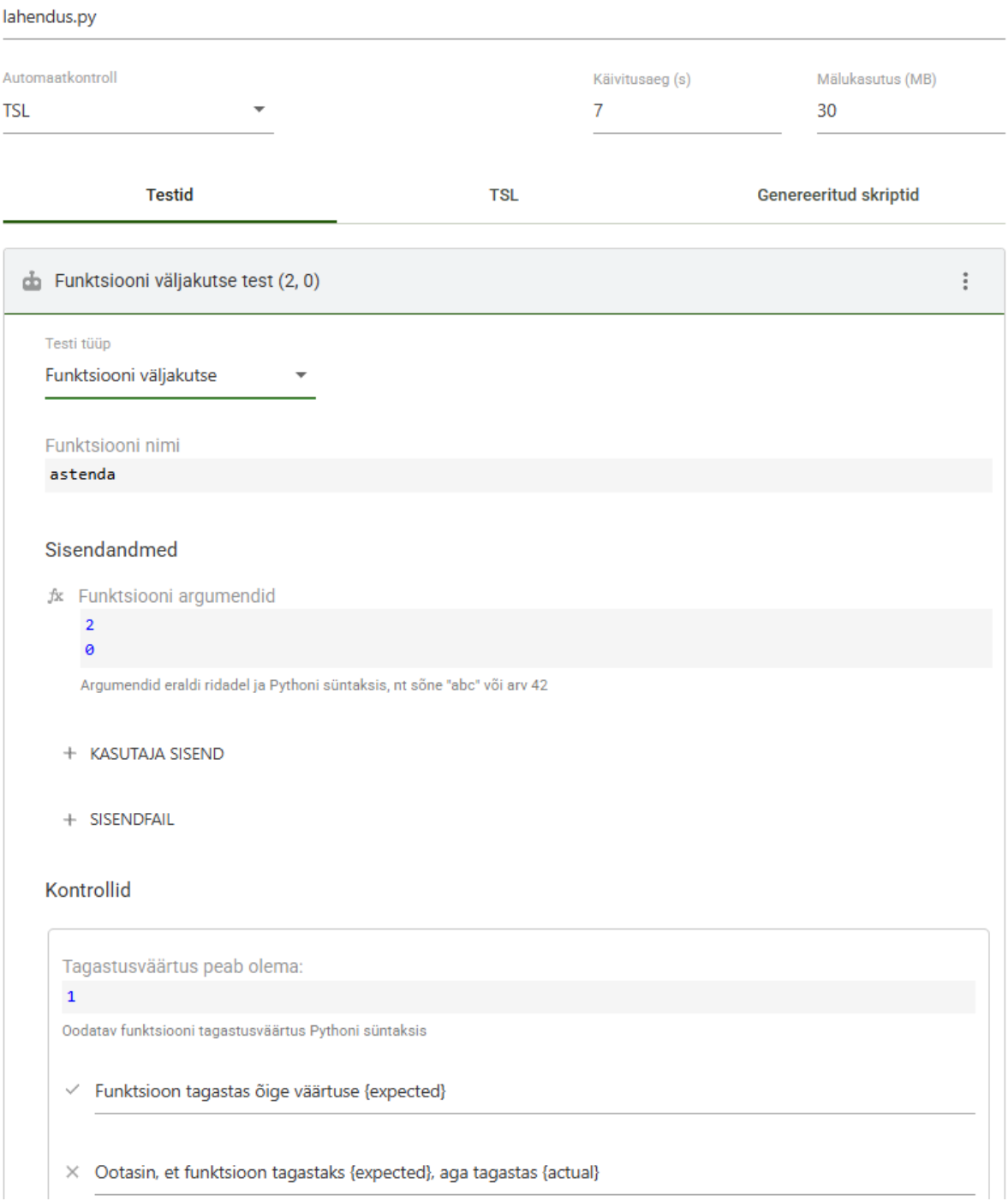
Kavandamise etapis mõeldi läbi ülesannete üldine struktuur ja lähenemisviis. Otsustati, et ülesandeid hakatakse looma alateemade kaupa, andes võimaluse koheselt kontrollida alateemas omandatut. Määratleti ülesannete eesmärk ja sobiv raskusaste. Eesmärgiks sai, et ülesanded toetaksid äsja läbitud materjali kinnistamist. Vaadati lähemalt ka kursuse ülesannete keskkonda. Kodu- ja praktikumiülesannete kontrollimiseks on kasutusel Lahenduse keskkonna automaatkontrollide süsteem [3]. Uuriti olemasolevate ülesannete näitel, kuidas ülesannete juhendeid kirjutada ning automaatkontrolle seadistada. Analüüsiti, milliseid ülesandetüüpe on võimalik kontrollida ning millised kontrollfunktsioonid on Lahenduse süsteemis kasutusel.

Koostamise etapis alustati konkreetsete ülesannete väljamõtlemist. Ülesandeid loodi alateemade kaupa ja enamasti kindlas järjestuses. Esmalt mõeldi välja ülesande teema ja juhend, lähtudes alateema sisust ja keerukusest. Juhendeid koostati AsciiDoc formaadis, mis on näha joonisel 1.



Joonis 1. Ülesande juhendi kirjutamine AsciiDoc formaadis.

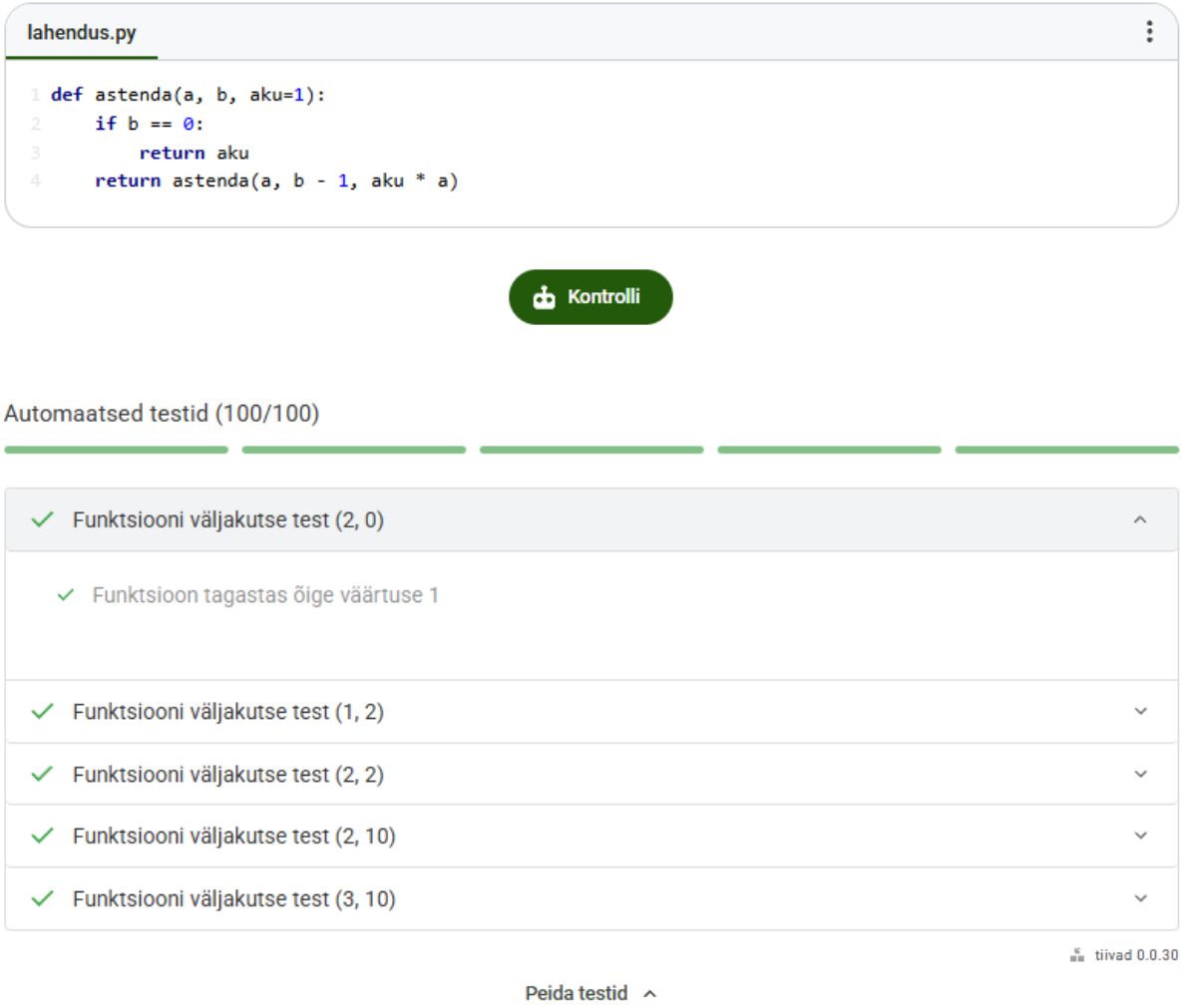
Kuna eelnevalt polnud töö autor AsciiDoc formaati kasutanud, tuli kasuks juhendi järgimine. Lahenduse lehel pääses juhendi kirjutamise aknas küsimärgi ikooni alt ligi põhjalikule juhendile [25]. Peale juhendi valmimist, loodi ülesandele ka näidislahendus, mida esialgu testiti arenduskeskkonnas Thonny. Töö autor kasutas Thonny keskkonda sellepärast, et see on kasutusel ka kursusel. Pärast juhendi ja näidislahenduse valmimist, seadistati automaatkontrollid. Automaatkontrollide võimalused on esitatud joonisel 2.



Joonis 2. Automaatkontrolli seadistamine.

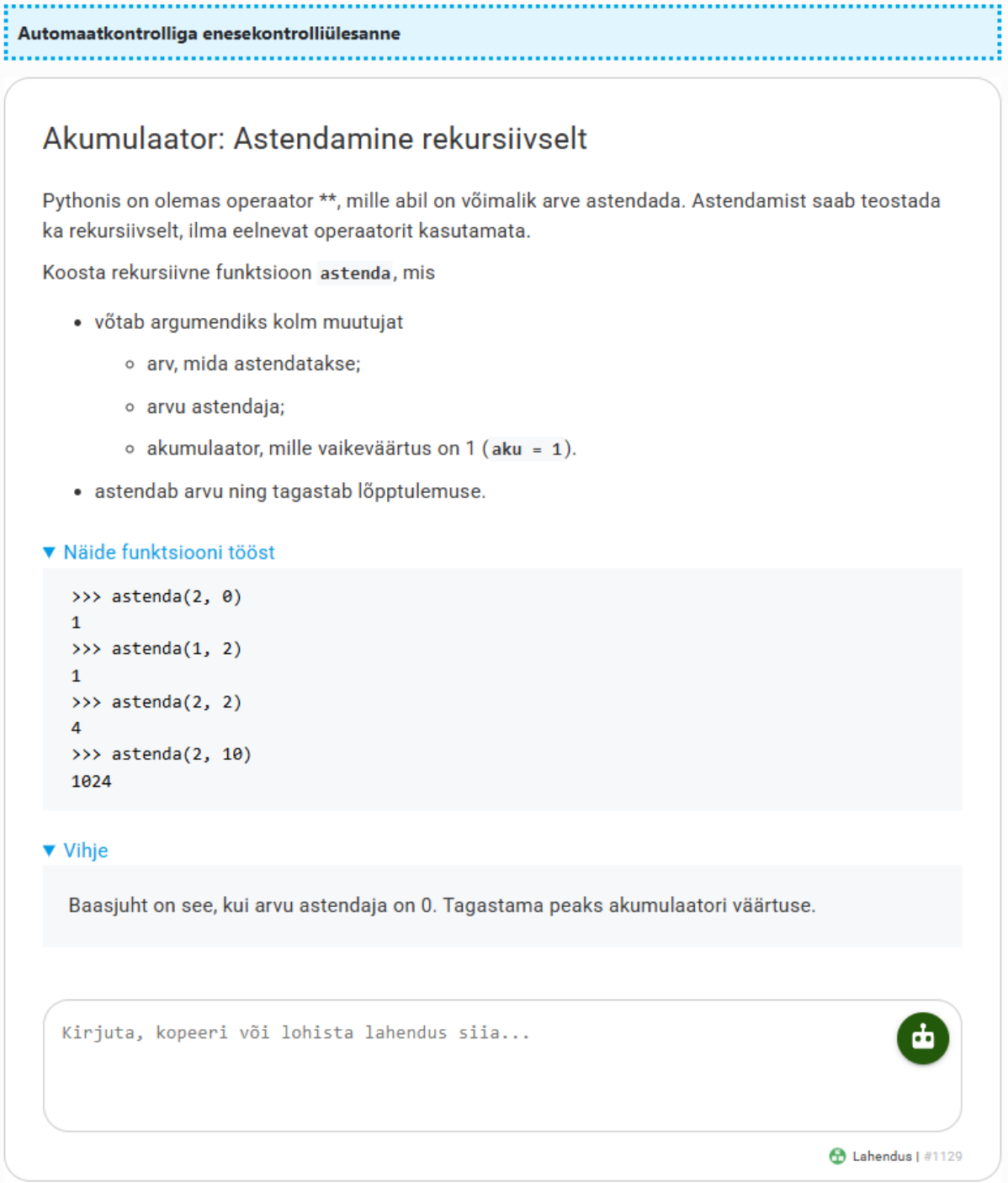
Automaatkontrolli seadistamisel tuli esmalt valida automaatkontrolli tüüp. Kuna varasemalt on kursusel kasutatud TSL'i (ingl *Test Specific Language*) tüüpi kontrolle ning see oli ka juhendaja soovitus, seadistati kõik autori poolt loodud ülesanded selliselt. Seejärel sai valida testi tüübi, milleks oli kas funktsiooni väljakutse või programmi käivitus. Vastavalt ülesande temaatikale ja sisule, kasutas autor töö vältel mõlemaid testi tüüpe. Seadistamisel sai veel valida funktsioonile etteantavaid argumente, kasutaja sisendit, sisendfaili ning määrata õige

tagastusväärtuse. Peale automaatkontrollide seadistamist oli võimalik näidislahendust juba kontrollida. Kontrollimine on välja toodud joonisel 3.



Joonis 3. Ülesande lahenduse kontrollimine.

Rakendamise etapis lisati valminud ülesanded lugemismaterjalide hulka. Kursuse Coursese lehel asuvad ülesanded alateemade lõpus ning iga ülesanne on tähistatud pealkirjaga „Automaatkontrolliga enesekontrolliülesanne“. Kõik ülesanded olid kättesaadavad 14. aprillil 2025 alanud „Programmeerimise alused II“ kursusele. Ühe konkreetse ülesande välimus Coursese lehel on nähtaval joonisel 4.



Joonis 4. Automaatkontrolliga enesekontrolliülesanne kursuse Coursese veebilehel.

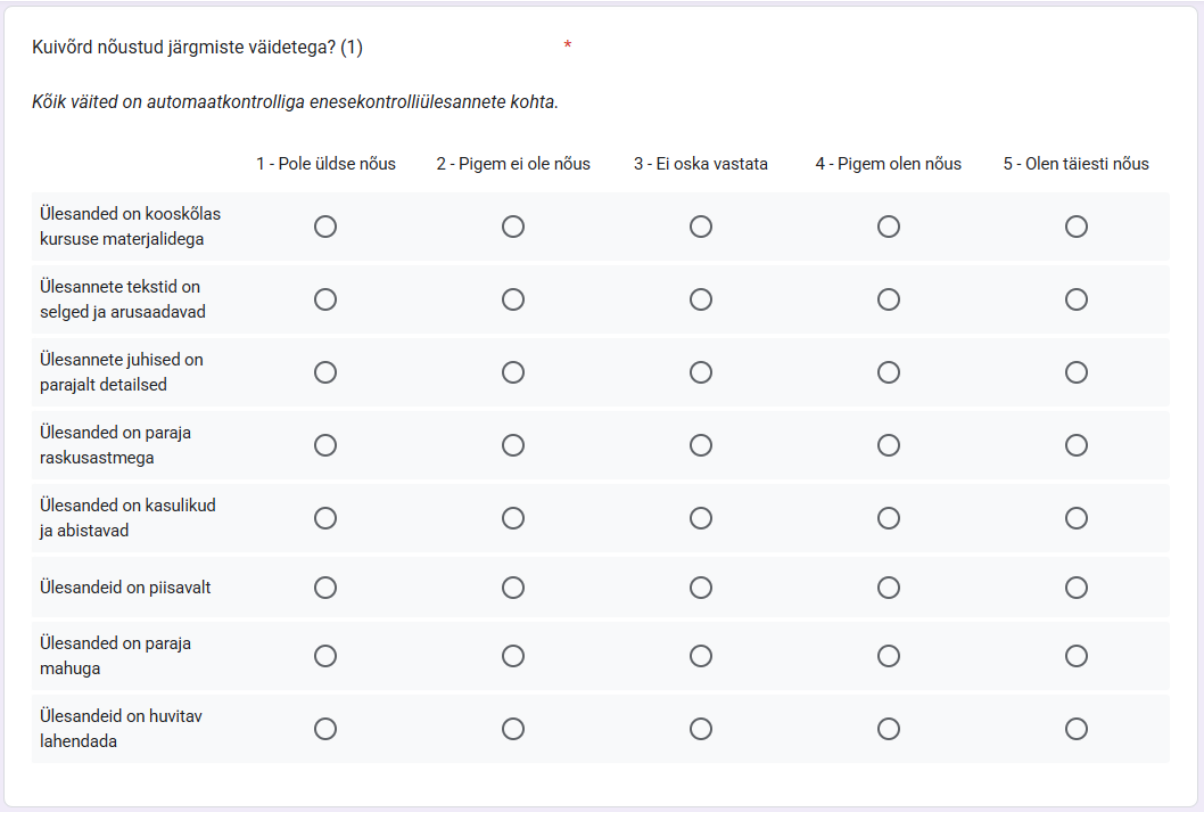
Hindamise etapis viidi läbi kursusel osalevatele üliõpilastele suunatud küsitlus, mille abil koguti tagasisidet ülesannete sobivuse, kasulikkuse ja raskusastme kohta. Tagasiside võimaldas hinnata ülesannete tõhusust ning viia sisse muudatusi.

3.2 Tagasiside kogumine

Tagasiside kogumiseks koostati veebipõhine küsimustik Google Formsi keskkonnas, mille eesmärk oli hinnata koostatud ülesannete kasulikkust, kvaliteeti ja sobivust (lisa I). Küsimustik sisaldas nii valikvastustega kui ka tekstilisi küsimusi. Tagasisidet küsiti esimese kahe teema „Kahemõõtmeline järjend“ ja „Kahekordne tsükkel“ kohta. Kokku oli nende teemade kohta kümme automaatkontrolliga ülesannet. Küsimustik oli üliõpilastele avatud nädal aega.

Küsimustiku alguses uuriti, kas vastajad olid üldse automaatkontrolliga enesekontrolliülesandeid lahendanud. Selle küsimuse vastusest lähtuvalt jagunes küsimustik kaheks. Üliõpilastel, kes vastasid eitavalt, paluti oma vastust lühidalt põhjendada, pärast mida nad said vastused ära saata. Need kes, vastasid jaatavalt, suunati edasi küsimustiku põhiosse.

Esimeses osas paluti vastajatel märkida, mitu ülesannet nad esimese kahe teema kohta kokku lahendasid. Seejärel esitati väiteid enesekontrolliülesannete sisulise ja vormilise kvaliteedi kohta. Väited on näha joonisel 5.



Kuivõrd nõustud järgmiste väidetega? (1) *

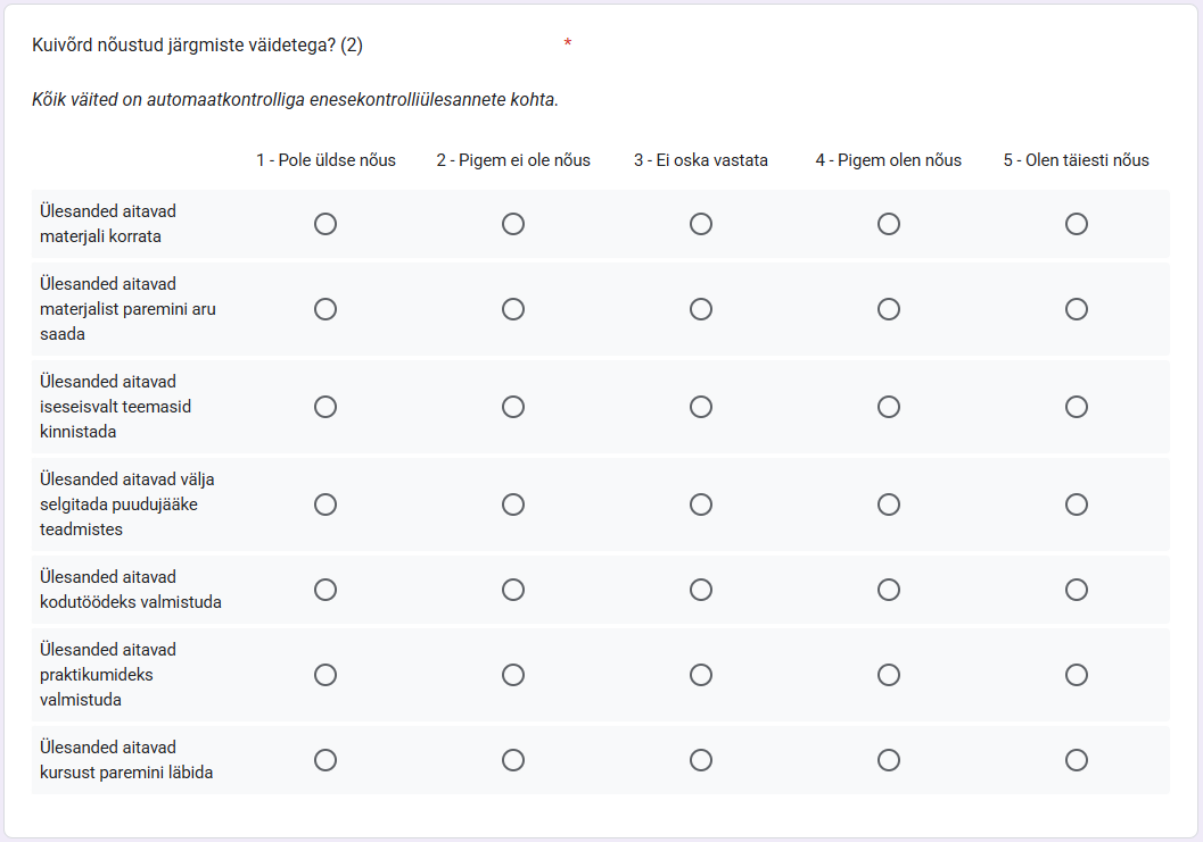
Kõik väited on automaatkontrolliga enesekontrolliülesannete kohta.

	1 - Pole üldse nõus	2 - Pigem ei ole nõus	3 - Ei oska vastata	4 - Pigem olen nõus	5 - Olen täiesti nõus
Ülesanded on kooskõlas kursuse materjalidega	○	○	○	○	○
Ülesannete tekstid on selged ja arusaadavad	○	○	○	○	○
Ülesannete juhised on parajalt detailsed	○	○	○	○	○
Ülesanded on paraja raskusastmega	○	○	○	○	○
Ülesanded on kasulikud ja abistavad	○	○	○	○	○
Ülesandeid on piisavalt	○	○	○	○	○
Ülesanded on paraja mahuga	○	○	○	○	○
Ülesandeid on huvitav lahendada	○	○	○	○	○

Joonis 5. Tagasiside küsimustiku esimese osa väited.

Väiteid hinnati Likerti skaalal[2] ühest kuni viieni. Hinnang üks tähistas, et vastaja pole väitega üldse nõus, hinnang kaks, et pigem ei ole nõus. Hinnang kolm väljendas neutraalsust ehk ei oska vastata. Hinnang neli viitas sellele, et vastaja on väitega pigem nõus ning viis seda, et on täiesti nõus. Likerti skaala põhjal hinnati väiteid ka küsimustiku teistes osades. Esimese osa väidete hinnatavad aspektid hõlmasid ülesannete selgust, kooskõla kursuse materjalidega, juhiste detailsust, raskusastet, kasulikkust, huvitavust ning mahu sobivust.

Teises osas keskenduti ülesannete õppeprotsessi toetavale rollile. Teise osa väited on näha joonisel 6. Väidete kaudu hinnati, kuivõrd aitavad ülesanded materjali korrata, mõista ja kinnistada, puudujääke tuvastada ning kodutöödeks ja praktikumideks valmistuda.


Kuivõrd nõustud järgmiste väidetega? (2) *

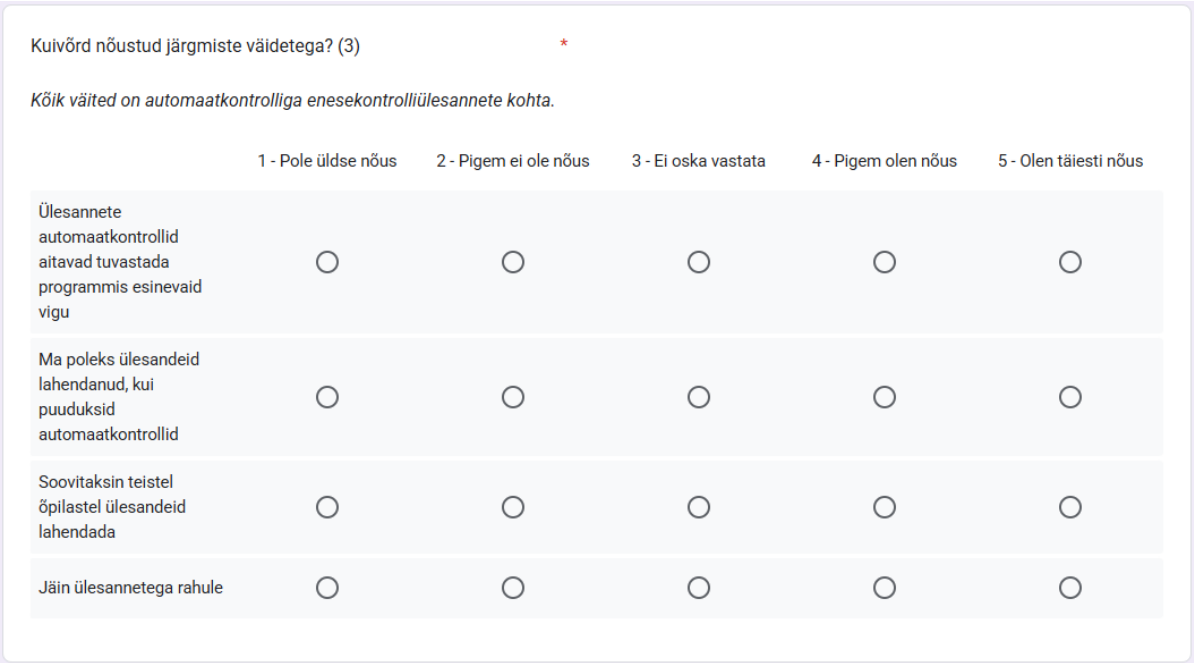
Kõik väited on automaatkontrolliga enesekontrolliülesannete kohta.

	1 - Pole üldse nõus	2 - Pigem ei ole nõus	3 - Ei oska vastata	4 - Pigem olen nõus	5 - Olen täiesti nõus
Ülesanded aitavad materjali korrata	○	○	○	○	○
Ülesanded aitavad materjalist paremini aru saada	○	○	○	○	○
Ülesanded aitavad iseseisvalt teemasid kinnistada	○	○	○	○	○
Ülesanded aitavad välja selgitada puudujääke teadmistes	○	○	○	○	○
Ülesanded aitavad kodutöödeks valmistuda	○	○	○	○	○
Ülesanded aitavad praktikumideks valmistuda	○	○	○	○	○
Ülesanded aitavad kursust paremini läbida	○	○	○	○	○

Joonis 6. Tagasiside küsimustiku teise osa väited.

[2] Hariduse ja kasvatuse sõnaraamatu järgi (https://arhiiv.eki.ee/dict/haridus/):
Likerti skaala (ingl *Likert scale*) on suhtumisi ja arvamusi uuriva küsimustiku skaala, mis määrab ära teatud väitega nõustumise astme.

Kolmandas osas hinnati automaatkontrollide tööd. Väited on näha joonisel 7. Vastajad hindasid, kas automaatkontrollid aitasid vigu leida, kas mõjutasid ülesandeid lahendama, ning kas nad soovitaksid automaatkontrolliga ülesannete lahendamist teistele üliõpilastele. Viimasena hinnati üldist rahulolu ülesannetega.


Kuivõrd nõustud järgmiste väidetega? (3) *

Kõik väited on automaatkontrolliga enesekontrolliülesannete kohta.

	1 - Pole üldse nõus	2 - Pigem ei ole nõus	3 - Ei oska vastata	4 - Pigem olen nõus	5 - Olen täiesti nõus
Ülesannete automaatkontrollid aitavad tuvastada programmis esinevaid vigu	○	○	○	○	○
Ma poleks ülesandeid lahendanud, kui puuduksid automaatkontrollid	○	○	○	○	○
Soovitaksin teistel õpilastel ülesandeid lahendada	○	○	○	○	○
Jäin ülesannetega rahule	○	○	○	○	○

Joonis 7. Tagasiside küsimustiku kolmanda osa väited.

Küsimustiku lõppu olid lisatud mitmed avatud küsimused, mille kaudu oli võimalik jagada vabas vormis kommentaare, ettepanekuid parandusteks või tuua esile muid kogemusi seoses ülesannetega. Neist oli kohustuslik vabas vormis mõtete jagamine ülesannete kohta. Motiveerimaks üliõpilasi küsimustele vastama, sai vormi täitmise eest kursusel ühe boonuspunkti. Selleks pidi vastaja sisestama tagasiside lõpus oma ees- ja perenime. Küsimustiku vastuseid kasutati analüüsi käigus vaid üldistatud ja anonüümsel kujul ning ei seostatud kuidagi vastaja isikuga.

4. Tulemused

Peatükk käsitleb bakalaureusetöö tulemusi. Antakse ülevaate valminud automaatkontrolliga enesekontrolliülesannetest. Seejärel analüüsitakse üliõpilastelt kogutud tagasisidet.

4.1 Ülevaade koostatud ülesannetest

Bakalaureusetöö raames koostati 35 automaatkontrolliga enesekontrolliülesannet (lisa II). Ülesanded loodi kursuse kõigi seitsme teema kohta. Ülesanded jaotusid teemade lõikes järgmiselt:

- „Kahemõõtmeline järjend“ – 3 ülesannet;
- „Kahekordne tsükkel“ – 7 ülesannet;
- „Andmestruktuurid“ – 5 ülesannet;
- „Viitamine ja muteerimine“ – 4 ülesannet;
- „Testimine ja silumine. Rekursioon“ – 4 ülesannet;
- „Rekursioon II“ – 10 ülesannet;
- „Objektorienteeritud programmeerimine“ – 2 ülesannet.

Kõik koostatud ülesanded on omakorda seotud kindlate alateemadega, mis on välja toodud tabelis 1. Alateemade sisu mõjutas otseselt ülesannete teema, sisu ja raskusastme valikut.

Tabel 1. Automaatkontrolliga enesekontrolliülesannete ülevaade teemade lõikes.

Teema	**Alateema**	**Ülesanne**
1. Kahemõõtmeline järjend	1.1 Sissejuhatus	Sissejuhatus: Istekohtade info väljastamine
	1.2 Järjendite järjend	Järjendite järjend: Kursused kahemõõtmelisse järjendisse
	1.3 Maatriks	Maatriks: Ruutmaatriksist elemendi leidmine
2. Kahekordne tsükkel	2.1 Järjend ja tsükkel	Järjend tsüklis: Restorani tellimused
		Tsükkel ja indeksid: Ainepunktide summa
		Kahekordne tsükkel ja indeksid: Maatriksi transponeerimine
	2.2 Järjend ja funktsioon	Järjend funktsiooni argumendina: Juku hinded

		Järjend funktsiooni väärtusena: Ruutmaatriks
	2.3 Reaalsete andmete kasutamine	Andmed failist: Punktide kokkulugemine
		Reaalsete andmete kasutamine: Tartu elanike arvu suurimad muutused
3. Andmestruktuurid	3.1 Järjend ja ennik	Järjend ja ennik: Temperatuuride analüüs
	3.2 Hulk	Hulk: Ostunimekirja uuendaja
		Hulk: Ostunimekirja kontrollija
	3.3 Sõnastik	Sõnastik: Autode hooldusajalugu
	3.4 Andmestruktuuride võrdlus	-
	3.5 Sõne	Sõne: Isikukoodist sünnikuupäeva eraldamine
4. Viitamine ja muteerimine	4.1 Viitamine	-
	4.2 Muteerimine	-
	4.3 Järjendi muteerimine	Järjendite "korrutamine": Toolide paigutus
	4.4 Funktsioon ja muteerimine	Funktsioon ja muteerimine: Ostukorvi soodustus
	4.5 Kordamine	Töövoost, mida ikka kasutame: Maakondande kilometraaž
		Sama programm hoopis funktsiooniga: Ülepinge kontroll
5. Testimine ja silumine. Rekursioon	5.1 Testimine ja silumine	-
	5.2 Rekursioon: sissejuhatus	Rekursiivne väljakutse: Kahe naturaalarvu summa
		Rekursiivne väljakutse: Kahe naturaalarvu vahe
	5.3 Rekursioonist detailsemalt	Rekursiivne väljakutse: Kolmnurga joonistamine
		Rekursioon sõnede ja järjenditega: Järjendi arvude summa leidmine

	5.4 Andmetöötlus mooduliga Pandas	-
	5.5 Teatrikülastuste andmed	-
	5.6 Silmaring: Keeletöötlus	-
6. Rekursioon II	6.1 Tsükkel ja rekursioon	Millal lõpetada?: Tordi lõikamine
		Akumulaator: Astendamine rekursiivselt
		Sabarekursioon: Arvude 1 kuni n summa
		Järjendi summa näide: Kontojäägi arvutamine
	6.2 Rekursiooni tüüpe, hargnev rekursioon	Kahendotsing: Lennupiletite otsimine
		Hargnev rekursioon e puurekursioon: Mündivisete kombinatsioonid
		Fibonacci arvud: Trepiastmed
		Vastastikune rekursioon: Palgatõus
	6.3 Rekursiivsed andmestruktuurid	Järjendid järjendite sees: Paarisarvude korrutis
		Failide ja kaustade kuvamine: Järjend failisüsteemina
7. Objektorienteeritud programmeerimine	-	Meetodid: Raamatute haldamine
		Lõpetuseks: Pangakonto haldamine

Kõikide teemade ja alateemade kohta ülesandeid ei koostatud. Põhjuseid oli mitmeid. Näiteks alateemad „3.4 Andmestruktuuride võrdlus“, „4.1 Viitamine“, „4.2 Muteerimine“ ja „5.1 Testimine ja silumine“ olid sisult lühemad ja lihtsamad. Samuti käsitlesid need osaliselt selliseid Pythoni aspekte, mida oleks automaatkontrolliga väga keeruline või isegi võimatu hinnata. Alateemad „5.4 Andmetöötlus mooduliga Pandas“, „5.5 Teatrikülastuste andmed“ ja „5.6 Silmaring: Keeletöötlus“ käsitlesid Pythoni lisafunktsioone ja -pakette, mis vajasid eraldi paigaldamist või mille kasutamist ei toeta Lahenduse keskkonna automaatkontrolli TSL-i

süsteem. Seetõttu ei olnud nende teemade põhjal ülesannete koostamine tehniliselt teostatav. Seitsmenda teema „Objektorienteeritud programmeerimine“ kohta koostati küll kaks ülesannet, kuid Lahenduse keskkonna automaatkontrolli süsteemil puudus võimalus kontrollida objektorienteeritud programme. See tähendas, et ei olnud võimalik automaatselt kontrollida programmi sees olevate klasside tööd. Seega otsustati nende ülesannete puhul kontrollida vaid programmi poolt ekraanile väljastatud teksti.

4.2 Üliõpilaste tagasiside analüüs

Tagasisidet küsiti 2025. aasta kevadel toimunud „Programmeerimise alused II“ kursuse üliõpilastelt. Kuna kursus jätkus ka pärast käesoleva töö esitamist, küsiti tagasisidet ajalistel põhjustel vaid esimese kümne automaatkontrolliga enesekontrolliülesande kohta. Kursusel osales õppeinfosüsteemi põhjal kokku 80 üliõpilast [1]. Tagasiside küsimustikule vastas 11 üliõpilast. Vastajate tagasihoidlikku arvu võis mõjutada asjaolu, et kursuse „Programmeerimise alused“ lõpus, mis eelnes kursusele „Programmeerimise alused II“, küsiti üliõpilastelt teise lõputöö raames samuti tagasisidet sealsete enesekontrolliülesannete kohta. Lisaks oli küsimustik ajaliste piirangute tõttu avatud vaid ühe nädala, mis võis ka mõjutada vastajate arvu. Vastanutest üheksa inimest lahendas enesekontrolliülesandeid. Kaks inimest ülesandeid ei lahendanud.

Ülesannete mittelahendamise selgituseks toodi välja kaks peamist põhjust: teemad olid juba selged ning seetõttu tundus lahendamine mittevajalik või ei leitud aega, et ülesandeid lahendada. Küsimusele, kas plaanitakse järgmiste teemade juures ülesandeid lahendada, vastati, et seda tehakse sel juhul, kui teemad on keerulisemad. Üks üliõpilane tõi siinkohal välja, et talle oleks tulevikus ülesannete lahendamine kindlasti kasulik, kuid ta ei leia selle jaoks piisavalt aega. Pakuti välja ka mõte, et ülesandeid motiveeriks rohkem lahendama boonuspunktide süsteem.

Õppijatelt, kes ülesandeid lahendasid, küsiti esimeses osas, mitu ülesannet nad kümnest võimalikust kokku lahendasid. Neli ülesannet lahendas üks üliõpilane, viis ülesannet kaks, kuus ülesannet kaks, seitse ülesannet kaks ja kõik kümme ülesannet kaks üliõpilast. Seejärel hinnati viie palli skaalal ülesannete sisulist ja vormilist kvaliteeti kaheksa väite põhjal. Hinnangud on näha tabelis 2.

Tabel 2. Tagasiside küsimustiku esimese osa väidete hinnangud.

Väide	**Hinnangute jaotus Likerti skaalal**				
	Pole üldse nõus	**Pigem ei ole nõus**	**Ei oska vastata**	**Pigem olen nõus**	**Olen täiesti nõus**
	1	**2**	**3**	**4**	**5**
Ülesanded on kooskõlas kursuse materjalidega	-	-	-	2	7
Ülesannete tekstid on selged ja arusaadavad	-	-	-	5	4
Ülesannete juhised on parajalt detailsed	-	-	-	5	4
Ülesanded on paraja raskusastmega	-	1	-	5	3
Ülesanded on kasulikud ja abistavad	-	-	1	3	5
Ülesandeid on piisavalt	-	1	-	6	2
Ülesanded on paraja mahuga	-	-	-	6	3
Ülesandeid on huvitav lahendada	-	1	-	5	3

Kõige madalam hinnang, mis esimeses sektsioonis väidetele anti, oli kaks. Seda hinnangut anti kolmel korral ja erinevatele väidetele. Nendeks olid väited „Ülesanded on paraja raskusastmega“, „Ülesandeid on piisavalt“ ja „Ülesandeid on huvitav lahendada“. Ühel korral anti hinnanguks kolm ehk neutraalne hinnang. See oli seotud väitega „Ülesanded on kasulikud ja abistavad“. Enamasti olid väidete hinnanguteks neli ja viis. Hinnangut neli jäeti 37 korda ja hinnangut viis 31 korda. Esimese osa väidete hinnangute põhjal võib öelda, et valdavalt olid üliõpilaste arvamused positiivsed ning ülesannete sisulise ja vormilise poolega oldi rahul.

Teises osas hinnati seda, kuidas ülesanded toetavad kursuse materjali omandamist. Väiteid, mida hinnata, oli seitse. Hinnangud on nähtavad tabelis 3. Teise osa väidete hinnangud olid samuti positiivsed. Madalaim hinnang oli kolm ning seda esines ainult ühel korral väite „Ülesanded aitavad materjalist paremini aru saada“ juures. Ülejäänud väiteid hinnati hinnetega

neli ja viis. Hinnangut neli jäeti 25 korda ning hinnangut viis 37 korda. Seega saab öelda, et üldiselt olid ülesanded abistavad ning aitasid üliõpilasi materjali läbitöötamisel.

Tabel 3. Tagasiside küsimustiku teise osa väidete hinnangud.

Väide	**Hinnangute jaotus Likerti skaalal**				
	Pole üldse nõus	**Pigem ei ole nõus**	**Ei oska vastata**	**Pigem olen nõus**	**Olen täiesti nõus**
	1	**2**	**3**	**4**	**5**
Ülesanded aitavad materjali korrata	-	-	-	3	6
Ülesanded aitavad materjalist paremini aru saada	-	-	1	2	6
Ülesanded aitavad iseseisvalt teemasid kinnistada	-	-	-	2	7
Ülesanded aitavad välja selgitada puudujääke teadmistes	-	-	-	4	5
Ülesanded aitavad kodutöödeks valmistuda	-	-	-	4	5
Ülesanded aitavad praktikumideks valmistuda	-	-	-	4	5
Ülesanded aitavad kursust paremini läbida	-	-	-	6	3

Kolmandas osas hinnati automaatkontrollide kvaliteeti ning üldist rahulolu. Seda hinnati nelja väite põhjal. Hinnangud on näha tabelis 4. Kolmanda osa hinnangud olid valdavalt positiivsed. Hinnangut kaks anti kolmel korral. Ühel korral väitele „Ülesannete automaatkontrollid aitavad tuvastada programmis esinevaid vigu“. Kahel korral anti hinnang kaks väitele „Ma poleks ülesandeid lahendanud, kui puuduksid automaatkontrollid“. Küll aga saab seda hinnangut vaadata positiivsena, sest vastanud üliõpilased oleksid ülesandeid lahendanud ka automaatkontrollide puudumisel. Hinnangut kolm anti kahel korral ja erinevatele väidetele. Hinnangut neli jäeti 17 korda ja hinnangut viis 14 korda. Hinnangute põhjal saab teha positiivseid järeldusi. Automaatkontrollidest oli järelikult ülesannete juures pigem kasu kui kahju, üliõpilased jäid ülesannetega üldiselt rahule ning soovitaksid ülesandeid ka teistele üliõpilastele.

Tabel 4. Tagasiside küsimustiku kolmanda osa väidete hinnangud.

Väide	**Hinnangute jaotus Likerti skaalal**				
	Pole üldse nõus	**Pigem ei ole nõus**	**Ei oska vastata**	**Pigem olen nõus**	**Olen täiesti nõus**
	1	**2**	**3**	**4**	**5**
Ülesannete automaatkontrollid aitavad tuvastada programmis esinevaid vigu	-	1	1	5	2
Ma poleks ülesandeid lahendanud, kui puuduksid automaatkontrollid	-	2	1	3	3
Soovitaksin teistel õpilastel ülesandeid lahendada	-	-	-	5	4
Jäin ülesannetega rahule	-	-	-	4	5

Tagasiside küsimustiku viimases osas oli vastajatel võimalik anda vabas vormis tagasisidet. Samuti küsiti üliõpilastelt seda, kas nad plaanivad kontrolltööks kordamisel automaatkontrolliga enesenkontrolliülesandeid lahendada. Eitavalt ei vastatud küsimusele mitte ühtegi korda. Viis üliõpilast vastasid jaatavalt. Üks üliõpilane lisas, et lahendab ülesandeid kindlasti uuesti, sest see on kiire viis harjutamiseks. Ülejäänud neli olid kahevahel ning seejuures kommenteeris üks üliõpilane, et lahendaks kontrolltööks kordamisel enesekontrolliülesandeid, kui kontrolltöö jaoks tehakse uued eraldiseisvad automaatkontrolliga kordamisülesanded. Vabas vormis saadud tagasiside oli konstruktiivne ning valdavalt positiivselt meelestatud. Näiteks kommenteeriti ülesandeid järgmiselt: „Meeldib, et ülesanded ongi täpselt nende teemade peale, mida just lugesin, nii et saab kohe teooriat ka praktikas kinnitada.“, „Ülesanded aitavad teemasid kinnistada ning näitavad, millele peaks neid programmeerimise võtteid kasutades tähelepanu juhtima.“, „Aitavad iseseisvalt õppida - saab ülesandeid lahendada siis, kui tahan, seal kus tahan ja ma saan kohe tagasisidet.“, „Uue osa läbitöötamisel väga kasulikud.“ ja „Automaatkontroll on elupäästja, ilma ei saaks ma hakkama.“. Kaks üliõpilast leidsid, et ülesanded on küll kasulikud, kuid kohati liiga ajamahukad ja keerulised. Samuti pakkusid kaks üliõpilast välja ideid automaatkontrollide edasiarendamiseks. Mõlemad ideed olid seotud vihjetega ning välja pakuti, et automaatkontrollid võiksid anda õigete lahenduste kohta vihjeid, kui ülesandeid on proovitud

juba pikalt lahendada. Ühel juhul leiti ka, et automaatkontrolli süsteem ei anna piisavalt tagasisidet, kui lahendus on vale ning seega on raske aru saada, mis esitatud lahenduses valesti võiks olla. Siinkohal nõustub töö autor, et Lahenduse keskkonna automaatkontrolli süsteemi saaks tulevikus edasi arendada nii kontrollimise võimaluste kui tagasiside andmise poole pealt.

Kokkuvõttes näitas üliõpilaste tagasiside, et esimesed kümme koostatud ülesannet olid kursuse sisu ja eesmärkidega valdavalt kooskõlas, selged ning abistavad. Enamik vastanutest hindas ülesandeid positiivselt ning seda kinnitasid ka vabas vormis saadud vastused. Seejuures toodi välja ka, et kohati on ülesanded ajamahukad ning automaatkontrollide tagasisidet oleks võimalik paremaks muuta. Üldiselt täitsid seega ülesanded bakalaureusetöö eesmärki – toetada iseseisvat õppimist ja aidata välja selgitada puudusi teadmistes.

Oluline on märkida seda, et ülejäänud 25 ülesande puhul ei saa täielikult kindel olla, kas need on sisu poolest sobilikud ja kursuse materjali omandamist abistavad ülesanded. Seega peaks tulevikus läbi viima veel vähemalt ühe küsitluse, mille abil saaks sarnasel viisil hinnata ülejäänud 25 ülesande sobivust ja kasutegurit.

Kokkuvõte

Bakalaureusetöö eesmärk oli luua kursusele „Programmeerimise alused II“ automaatkontrolliga enesekontrolliülesanded, mille lahendamine annaks üliõpilastele võimaluse peale kindla teoreetilise alateema läbimist iseseisvalt testida oma teadmisi ja saada tagasisidet. Töö teoreetilises osas kirjeldati kursuse õppekorraldust ning selgitati enesekontrolliülesannete vajadust kursuse kontekstis. Analüüsiti Pythonit programmeerimise õppekeelena, tutvustati õppematerjalide kavandamise võimalusi ning analüüsiti ADDIE õpikavanduse mudelit. Metoodika peatükis kirjeldati ülesannete koostamise protsessi, mis põhines ADDIE mudeli etappidel. Kirjeldati ka tagasiside kogumise protsessi ja küsimustiku sisu. Tulemuste peatükis esitati ülevaade koostatud ülesannetest ning analüüsiti üliõpilaste tagasisidet koostatud ülesannetele.

Töö käigus koostati 35 automaatkontrolliga enesekontrolliülesannet kõigile seitsmele kursuse teemale. Ülesanded jaotusid kursuse Coursese veebilehe materjalidesse alateemade kaupa, kusjuures kõikide alateemade kohta ülesandeid ei koostatud. Peamisteks põhjusteks olid teatud alateemade lihtsus ja väike maht ning Lahenduse keskkonna TSL-i põhise automaatkontrolli süsteemi piiratud funktsionaalsus.

Kõik koostatud ülesanded olid kättesaadavad 2025. aasta kevadsemestril toimunud kursusele. Ülesannete hindamiseks koostati tagasiside küsimustik, kus üliõpilastelt küsiti hinnanguid esimese kümne enesekontrolliülesande kohta. Hinnati ülesannete sisu, vormi, õppimist toetavaid aspekte ning automaatkontrollide kvaliteeti. Võimalik oli anda vabas vormis tagasisidet. Küsimustikule vastas 11 üliõpilast. Neist kaks ülesandeid ei lahendanud. Ülejäänud üheksa üliõpilase üldine tagasiside oli positiivne. Hinnangute põhjal selgus näiteks, et ülesanded olid paraja mahuga, ülesannete juhendid selged ja detailsed ning materjalidega kooskõlas. Positiivselt hinnati ka ülesannete rolli õppeprotsessi toetamisel. Kõige kõrgemalt hinnati seda, et ülesanded aitasid iseseisvalt teemasid korrata ja kinnistada. Ülesannete sisu kohta ei pakutud välja konkreetseid muudatusi, kuid lisati ideid automaatkontrollide edasiarenduse osas. Paari õppija tagasiside põhjal olid ülesanded kohati liiga keerulised ning ajaliselt mahukad. Vabas vormis tagasisidest selgus, et ülesanded olid kasulikud, abistavad ning iseseisvat õppimist toetavad. Tulevikus oleks soovitatav ka ülejäänud 25 ülesande puhul koguda tagasisidet, et tuvastada võimalikke puudujääke ning luua terviklik arvamus loodud automaatkontrolliga enesekontrolliülesannete kasulikkusest.

Viidatud kirjandus

[1] Tartu Ülikooli õppeinfosüsteem. MTAT.03.256 „Programmeerimise alused II“ õppeaastal 2024/2025. https://ois2.ut.ee/#/courses/MTAT.03.256/version/67a8d9d1-c977-7eed-d9c9-c73bc8c61e10/details (06.12.2024).

[2] Tartu Ülikool. Arvutiteaduse instituut. „Programmeerimise alused II“ õppeaastal 2024/25. https://courses.cs.ut.ee/2025/progalused2/spring (06.12.2024).

[3] Arvutiteaduse instituudi programmeerimise õpetamise keskkond Lahendus. https://lahendus.ut.ee/ (23.04.2025).

[4] Downes A. ADDIE: 5 Steps for Effective Training & Learning Evaluation. 2019. https://www.watershedlrs.com/blog/learning-evaluation/addie-instructional-design-model/ (07.05.2025).

[5] van Rossum G. Python 0.9.1 part 01/21. https://www.tuhs.org/Usenet/alt.sources/1991-February/001749.html (06.12.2024).

[6] Jansen P. TIOBE Index. https://www.tiobe.com/tiobe-index/ (09.05.2025).

[7] Downey A. How to Think Like a Computer Scientist : Learning with Python. Minneapolis, MN: Open Textbook Library. 2008.

[8] Lage M. J., Platt G. J., Treglia M. Inverting the Classroom: A Gateway to Creating an Inclusive Learning Environment. *The Journal of Economic Education*, 2000, 31, 1, 30–43. https://doi.org/10.1080/00220480009596759 (07.12.2024).

[9] Senali M. G., Iranmanesh M., Ghobakhloo M., Gengatharen D., Tseng M.-L., Nilsashi M. Flipped classroom in business and entrepreneurship education: A systematic review and future research agenda. *The International Journal of Management Education*, 2022, 20, 1. https://doi.org/10.1016/j.ijme.2022.100614 (07.12.2024).

[10] Tartu Ülikool. Arvutiteaduse instituut. „Programmeerimise alused II“. Hindamine. https://courses.cs.ut.ee/2025/progalused2/spring/Main/Hindamine (17.04.2025).

[11] Dewi N. P. A. K., Nitiasih P. K., Santosa M. H. The Self- and Peer-Assessments in Creative Writing: Students’ Benefits and Reactions. *SOSHUM : Jurnal Sosial dan Humaniora*, 2024, 14, 3, 241–248. https://doi.org/10.31940/soshum.v14i3.241-248 (24.04.2025).

[12] Ezeugo N. C., Eleje L. I., Obiasor G. E., Mbelede N. G., Osonwa K. E., Metu I. C., Eleje G. U. Self-Assessment Techniques in Clinical Studies in Public Universities in Anambra State: Benefits and Alignment With Supervisors Evaluation as Perceived by Medical Students. *Journal of Medical Education and Curricular Development*, 2024, 11. https://doi.org/10.1177/23821205241308787 (24.04.2025).

[13] Tagam K. E-kursuse „Programmeerimise alused II“ rekursiooni temaatika küsimuste ja ülesannete loomine. Tartu Ülikooli arvutiteaduse instituut. 2017. https://thesis.cs.ut.ee/2d9d095c-19f7-41ce-bfd0-b4150e8a2128 (24.04.2025).

[14] Rõmmel K. E-kursuse „Programmeerimise alused II“ kahemõõtmelise järjendi esialgsete materjalide koostamine ning analüüs. Tartu Ülikooli arvutiteaduse instituut. 2017. https://thesis.cs.ut.ee/c7c8594b-73eb-4d15-b566-f4db35cba45b (17.04.2025).

[15] Prokop Y., Trofimenko E., Loginova N., Zadereyko A., Gerganov M. Multivariate Analysis when Choosing the First Programming Language Studied in Universities. *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, 2019, 1224–1228. https://doi.org/10.1109/ukrcon.2019.8879810 (07.12.2024).

[16] Sundnes J. Introduction to Scientific Programming with Python. Cham: Springer International Publishing. 2020.

[17] Pellet J.-P., Dame A., Parriaux G. How beginner-friendly is a programming language? A short analysis based on Java and Python examples. 2019. https://www.researchgate.net/publication/337388460_How_beginner-friendly_is_a_programming_language_A_short_analysis_based_on_Java_and_Python_examples (24.04.2025).

[18] Gupta D. 8 Effective Instructional Design Models in 2025. *The Whatfix Blog*, 2023. https://whatfix.com/blog/instructional-design-models/ (24.04.2025).

[19] Spatioti A. G., Kazanidis I., Pange J. A Comparative Study of the ADDIE Instructional Design Model in Distance Education. *Information*, 2022, 13, 9, 402. https://doi.org/10.3390/info13090402 (24.04.2025).

[20] Drljača D. P., Tomić S., Ris K. BENCHMARKING INSTRUCTIONAL DESIGN MODELS - ADDIE WINS. *Economy & Market Communication Review / Casopis za Ekonomiju i Trzisne Komunikacije*, 2024, 14, 2, 688–698. https://doi.org/10.7251/emc2402688d (16.04.2025).

[21] Pilt L., Kusmin M., Plank T., Villems A., Varendi M., Rogalevitš V., Rosenberg A., Kirikal M., Požogina K., Dremljuga-Telk M. Juhend kvaliteetse e-kursuse loomiseks. 2021. https://oppevara.edu.ee/ekursus/ (24.04.2025).

[22] Villems A., Aluoja L., Pilt L., Naulainen M.-M., Kusmin M., Rogalevitš V., Tokko U. Digitaalse õppematerjali loomise soovitused – Juhend digitaalse õppematerjali autorile. https://oppevara.edu.ee/kvaliteet/ (07.05.2025).

[23] Allen W. C. Overview and Evolution of the ADDIE Training System. *Advances in Developing Human Resources*, 2006, 8, 4, 430–441. https://doi.org/10.1177/1523422306292942 (07.05.2025).

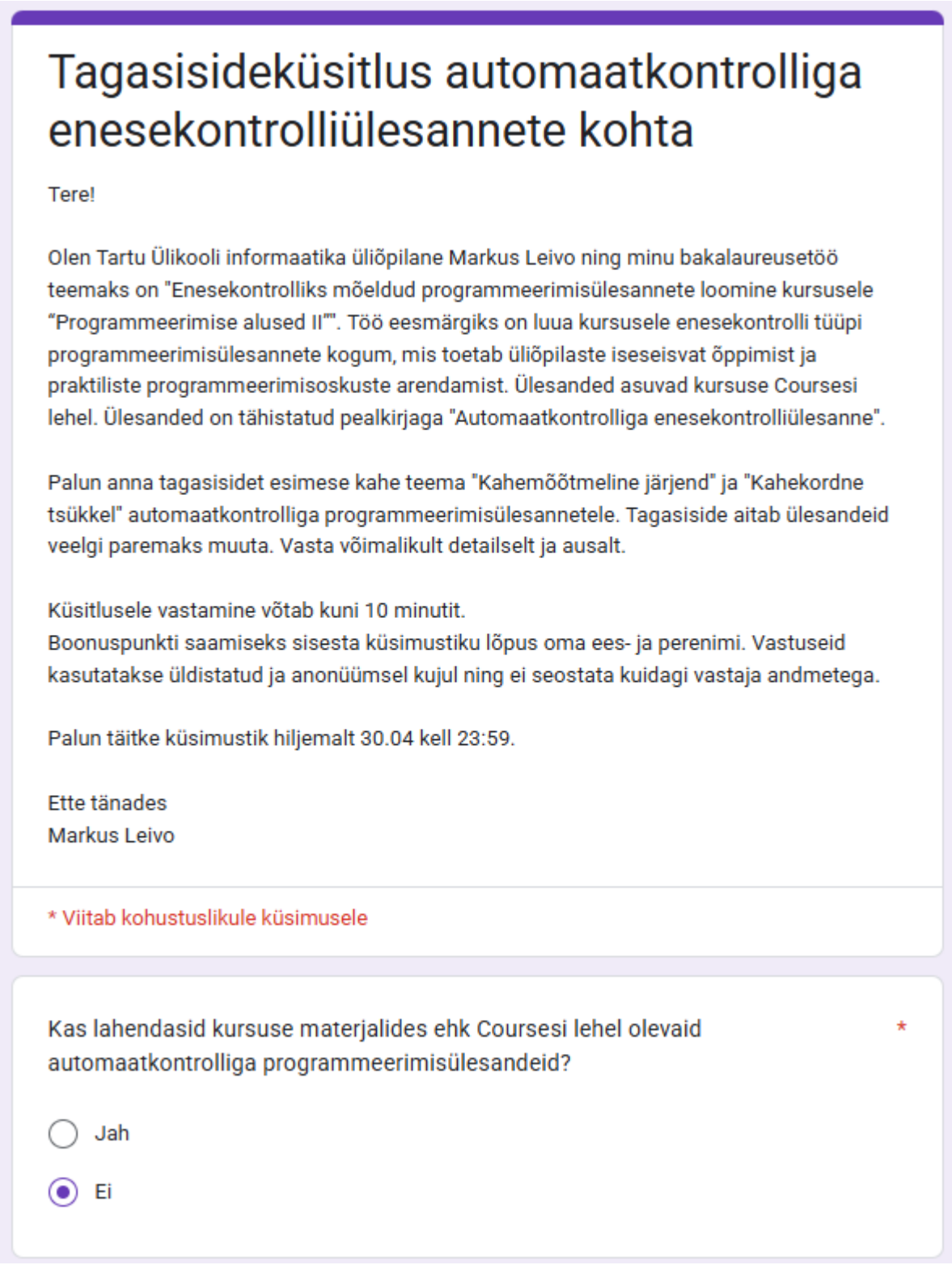
[24] Tartu Ülikooli Moodle. Programmeerimise alused II (MTAT.03.256). https://moodle.ut.ee/course/view.php?id=677 (24.04.2025).

[25] Asciidoctor Docs. AsciiDoc Syntax Quick Reference. https://docs.asciidoctor.org/asciidoc/latest/syntax-quick-reference/ (25.04.2025).

Lisad

I. Koostatud tagasiside küsimustik

Küsimustiku tutvustus



Tagasisideküsitlus automaatkontrolliga enesekontrolliülesannete kohta

Tere!

Olen Tartu Ülikooli informaatika üliõpilane Markus Leivo ning minu bakalaureusetöö teemaks on "Enesekontrolliks mõeldud programmeerimisülesannete loomine kursusele "Programmeerimise alused II"". Töö eesmärgiks on luua kursusele enesekontrolli tüüpi programmeerimisülesannete kogum, mis toetab üliõpilaste iseseisvat õppimist ja praktiliste programmeerimisoskuste arendamist. Ülesanded asuvad kursuse Coursesi lehel. Ülesanded on tähistatud pealkirjaga "Automaatkontrolliga enesekontrolliülesanne".

Palun anna tagasisidet esimese kahe teema "Kahemõõtmeline järjend" ja "Kahekordne tsükkel" automaatkontrolliga programmeerimisülesannetele. Tagasiside aitab ülesandeid veelgi paremaks muuta. Vasta võimalikult detailselt ja ausalt.

Küsitlusele vastamine võtab kuni 10 minutit.
Boonuspunkti saamiseks sisesta küsimustiku lõpus oma ees- ja perenimi. Vastuseid kasutatakse üldistatud ja anonüümsel kujul ning ei seostata kuidagi vastaja andmetega.

Palun täitke küsimustik hiljemalt 30.04 kell 23:59.

Ette tänades
Markus Leivo

* Viitab kohustuslikule küsimusele

Kas lahendasid kursuse materjalides ehk Coursesi lehel olevaid automaatkontrolliga programmeerimisülesandeid? *

- ( ) Jah
- (•) Ei

Küsimused vastajatele, kes ülesandeid ei lahendanud

Automaatkontrolliga enesekontrolliülesanded

Küsimused on järgnevate ülesannete kohta:

- "Sissejuhatus: Istekohtade info väljastamine";
- "Järjendite järjend: Kursused kahemõõtmelisse järjendisse";
- "Maatriks: Ruutmaatriksist elemendi leidmine";
- "Järjend tsüklis: Restorani tellimused";
- "Tsükkel ja indeksid: Ainepunktide summa";
- "Kahekordne tsükkel ja indeksid: Maatriksi transponeerimine";
- "Järjend funktsiooni argumendina: Juku hinded";
- "Järjend funktsiooni väärtusena: Ruutmaatriks";
- "Andmed failist: Punktide kokkulugemine";
- "Reaalsete andmete kasutamine: Tartu elanike arvu suurimad muutused".

Palun selgita lühidalt, miks Sa automaatkontrolliga enesekontrolliülesandeid ei lahendanud? *

Teie vastus

Kas plaanid järgnevate nädalate teemade juures automaatkontrolliga enesekontrolliülesandeid lahendada? *

- ◯ Jah
- ◯ Võib-olla
- ◯ Ei

Soovi korral kommenteeri eelnevat vastust.

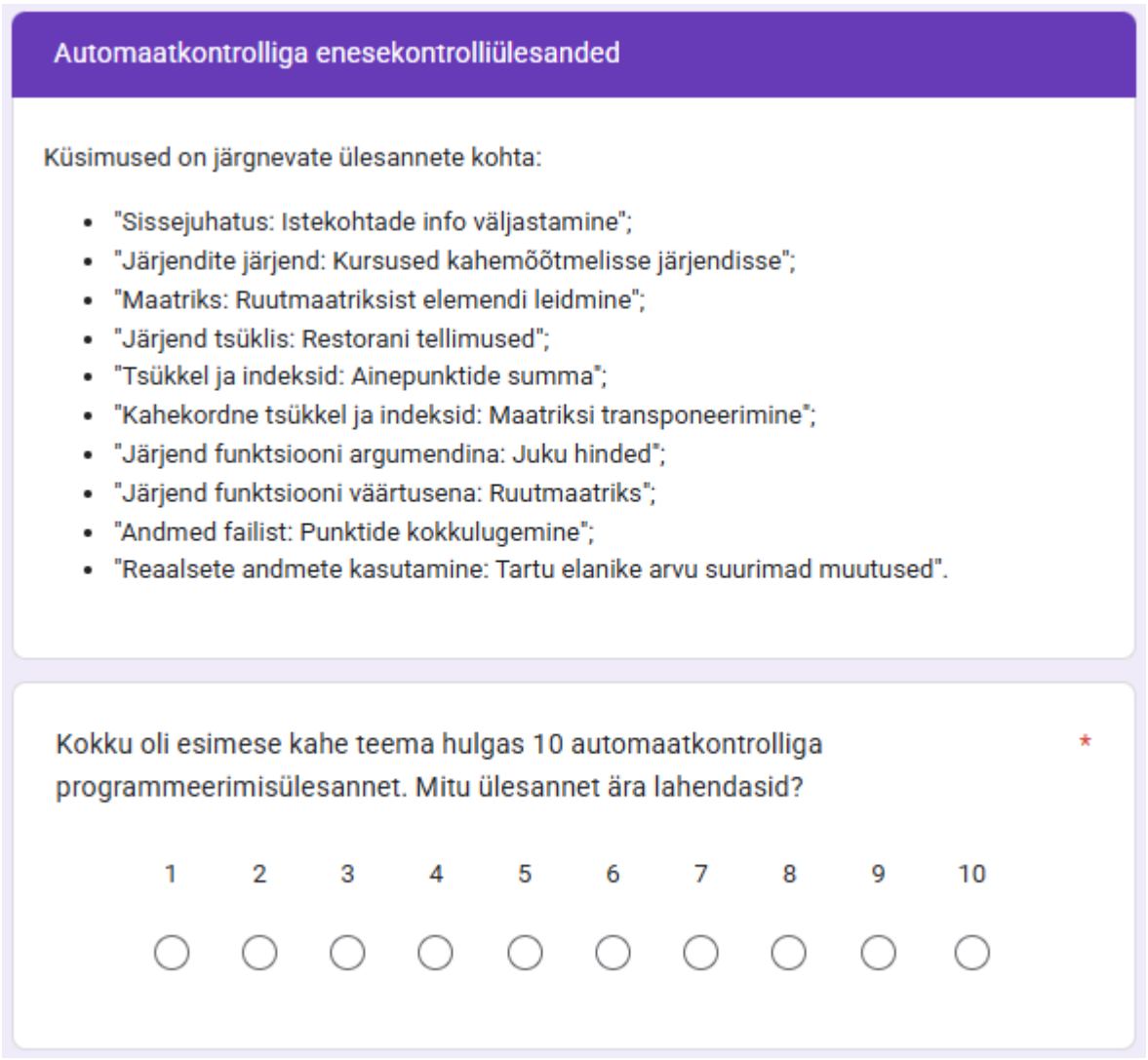
Teie vastus

Boonuspunkti saamiseks sisesta oma ees- ja perenimi.

Teie vastus

Tänan, et leidsid aega küsimustikule vastata!
Ära unusta vajutada nuppu "Saada ära"!

Küsimused vastajatele, kes ülesandeid lahendasid



Automaatkontrolliga enesekontrolliülesanded

Küsimused on järgnevate ülesannete kohta:

- "Sissejuhatus: Istekohtade info väljastamine";
- "Järjendite järjend: Kursused kahemõõtmelisse järjendisse";
- "Maatriks: Ruutmaatriksist elemendi leidmine";
- "Järjend tsüklis: Restorani tellimused";
- "Tsükkel ja indeksid: Ainepunktide summa";
- "Kahekordne tsükkel ja indeksid: Maatriksi transponeerimine";
- "Järjend funktsiooni argumendina: Juku hinded";
- "Järjend funktsiooni väärtusena: Ruutmaatriks";
- "Andmed failist: Punktide kokkulugemine";
- "Reaalsete andmete kasutamine: Tartu elanike arvu suurimad muutused".

Kokku oli esimese kahe teema hulgas 10 automaatkontrolliga programmeerimisülesannet. Mitu ülesannet ära lahendasid? *

1 2 3 4 5 6 7 8 9 10

I väidete sektsioon

Kuivõrd nõustud järgmiste väidetega? (1) *

Kõik väited on automaatkontrolliga enesekontrolliülesannete kohta.

	1 - Pole üldse nõus	2 - Pigem ei ole nõus	3 - Ei oska vastata	4 - Pigem olen nõus	5 - Olen täiesti nõus
Ülesanded on kooskõlas kursuse materjalidega	○	○	○	○	○
Ülesannete tekstid on selged ja arusaadavad	○	○	○	○	○
Ülesannete juhised on parajalt detailsed	○	○	○	○	○
Ülesanded on paraja raskusastmega	○	○	○	○	○
Ülesanded on kasulikud ja abistavad	○	○	○	○	○
Ülesandeid on piisavalt	○	○	○	○	○
Ülesanded on paraja mahuga	○	○	○	○	○
Ülesandeid on huvitav lahendada	○	○	○	○	○

Soovi korral kommenteeri eelnevaid vastuseid.

Teie vastus

II väidete sektsioon

Kuivõrd nõustud järgmiste väidetega? (2) *

Kõik väited on automaatkontrolliga enesekontrolliülesannete kohta.

	1 - Pole üldse nõus	2 - Pigem ei ole nõus	3 - Ei oska vastata	4 - Pigem olen nõus	5 - Olen täiesti nõus
Ülesanded aitavad materjali korrata	◯	◯	◯	◯	◯
Ülesanded aitavad materjalist paremini aru saada	◯	◯	◯	◯	◯
Ülesanded aitavad iseseisvalt teemasid kinnistada	◯	◯	◯	◯	◯
Ülesanded aitavad välja selgitada puudujääke teadmistes	◯	◯	◯	◯	◯
Ülesanded aitavad kodutöödeks valmistuda	◯	◯	◯	◯	◯
Ülesanded aitavad praktikumideks valmistuda	◯	◯	◯	◯	◯
Ülesanded aitavad kursust paremini läbida	◯	◯	◯	◯	◯

Soovi korral kommenteeri eelnevaid vastuseid.

Teie vastus

III väidete sektsioon

Kuivõrd nõustud järgmiste väidetega? (3) *

Kõik väited on automaatkontrolliga enesekontrolliülesannete kohta.

	1 - Pole üldse nõus	2 - Pigem ei ole nõus	3 - Ei oska vastata	4 - Pigem olen nõus	5 - Olen täiesti nõus
Ülesannete automaatkontrollid aitavad tuvastada programmis esinevaid vigu	◯	◯	◯	◯	◯
Ma poleks ülesandeid lahendanud, kui puuduksid automaatkontrollid	◯	◯	◯	◯	◯
Soovitaksin teistel õpilastel ülesandeid lahendada	◯	◯	◯	◯	◯
Jäin ülesannetega rahule	◯	◯	◯	◯	◯

Soovi korral kommenteeri eelnevaid vastuseid.

Teie vastus

Vabas vormis tagasiside

Kas plaanid lahendada kontrolltööks kordamisel automaatkontrolliga enesekontrolliülesandeid? *

- ◯ Jah
- ◯ Võib-olla
- ◯ Ei

Soovi korral kommenteeri eelnevat vastust.

Teie vastus

Soovi korral lisa ideid, kuidas muuta automaatkontrolliga enesekontrolliülesandeid veelgi paremaks.

Teie vastus

Jaga vabas vormis oma mõtteid automaatkontrolliga enesekontrolliülesannete kohta. Oodatud on igasugune konstruktiivne tagasiside. *

Teie vastus

Boonuspunkti saamiseks sisesta oma ees- ja perenimi.

Teie vastus

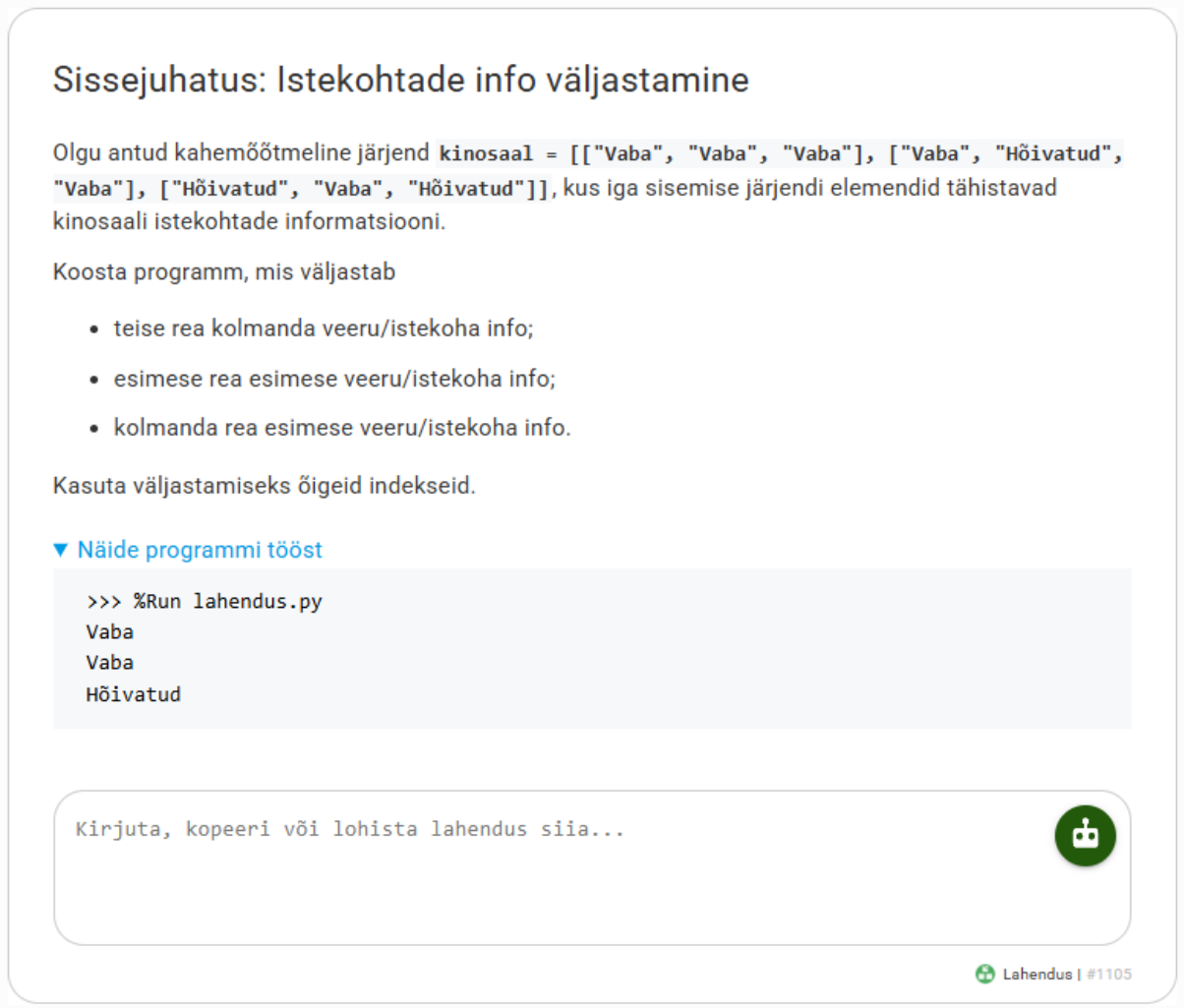
Tänan, et leidsid aega küsimustikule vastata!
Ära unusta vajutada nuppu "Saada ära"!

II. Koostatud ülesanded teemade kaupa

A. Kahemõõtmeline järjend

„Sissejuhatus: Istekohtade info väljastamine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART1A



Sissejuhatus: Istekohtade info väljastamine

Olgu antud kahemõõtmeline järjend `kinosaal = [["Vaba", "Vaba", "Vaba"], ["Vaba", "Hõivatud", "Vaba"], ["Hõivatud", "Vaba", "Hõivatud"]]`, kus iga sisemise järjendi elemendid tähistavad kinosaali istekohtade informatsiooni.

Koosta programm, mis väljastab

- teise rea kolmanda veeru/istekoha info;
- esimese rea esimese veeru/istekoha info;
- kolmanda rea esimese veeru/istekoha info.

Kasuta väljastamiseks õigeid indekseid.

▼ Näide programmi tööst

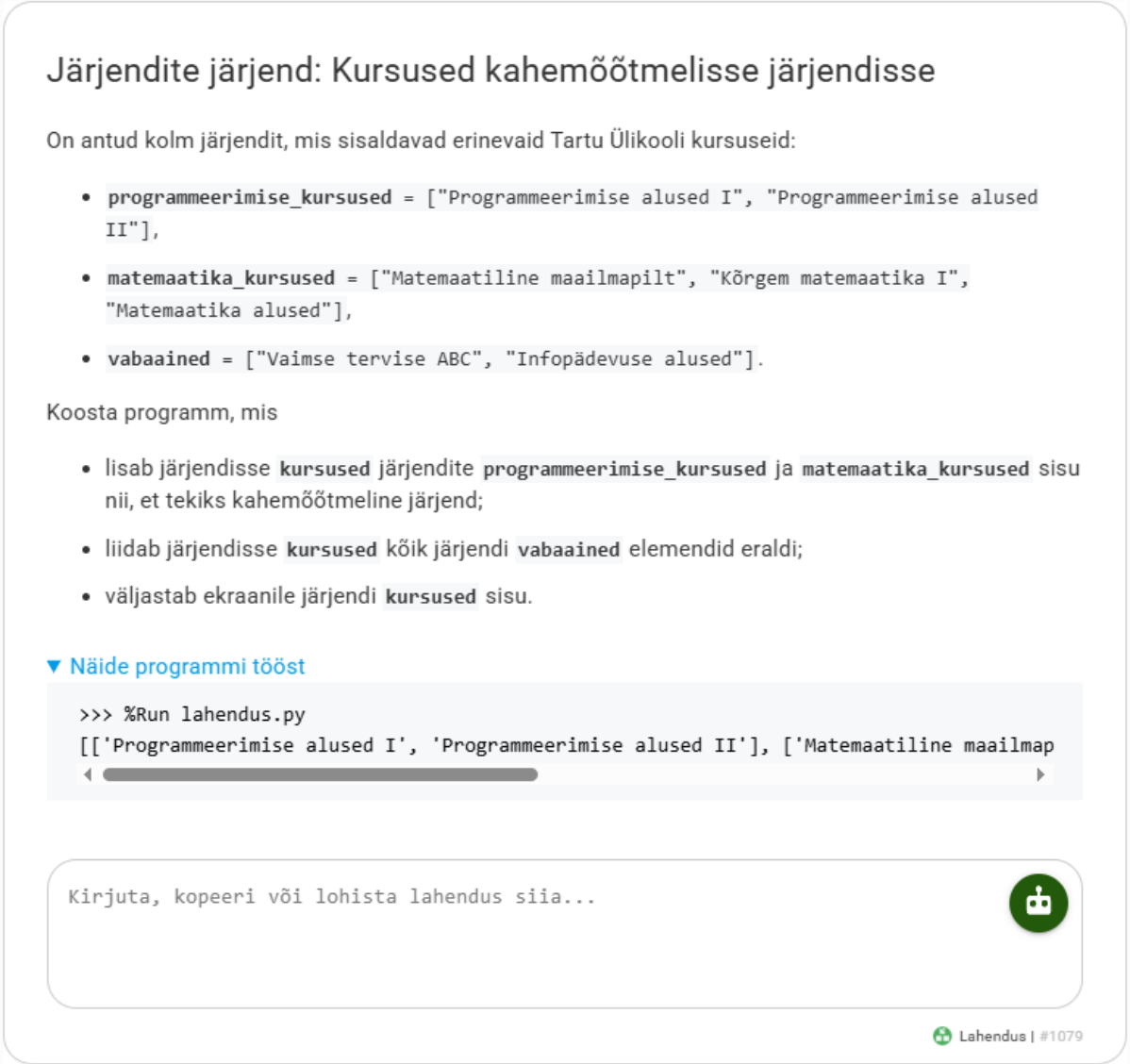
```
>>> %Run lahendus.py
Vaba
Vaba
Hõivatud
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1105

„Järjendite järjend: Kursused kahemõõtmelisse järjendisse“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART1B



Järjendite järjend: Kursused kahemõõtmelisse järjendisse

On antud kolm järjendit, mis sisaldavad erinevaid Tartu Ülikooli kursuseid:

- `programmeerimise_kursused = ["Programmeerimise alused I", "Programmeerimise alused II"]`,
- `matemaatika_kursused = ["Matemaatiline maailmapilt", "Kõrgem matemaatika I", "Matemaatika alused"]`,
- `vabaained = ["Vaimse tervise ABC", "Infopädevuse alused"]`.

Koosta programm, mis

- lisab järjendisse `kursused` järjendite `programmeerimise_kursused` ja `matemaatika_kursused` sisu nii, et tekiks kahemõõtmeline järjend;
- liidab järjendisse `kursused` kõik järjendi `vabaained` elemendid eraldi;
- väljastab ekraanile järjendi `kursused` sisu.

▼ Näide programmi tööst

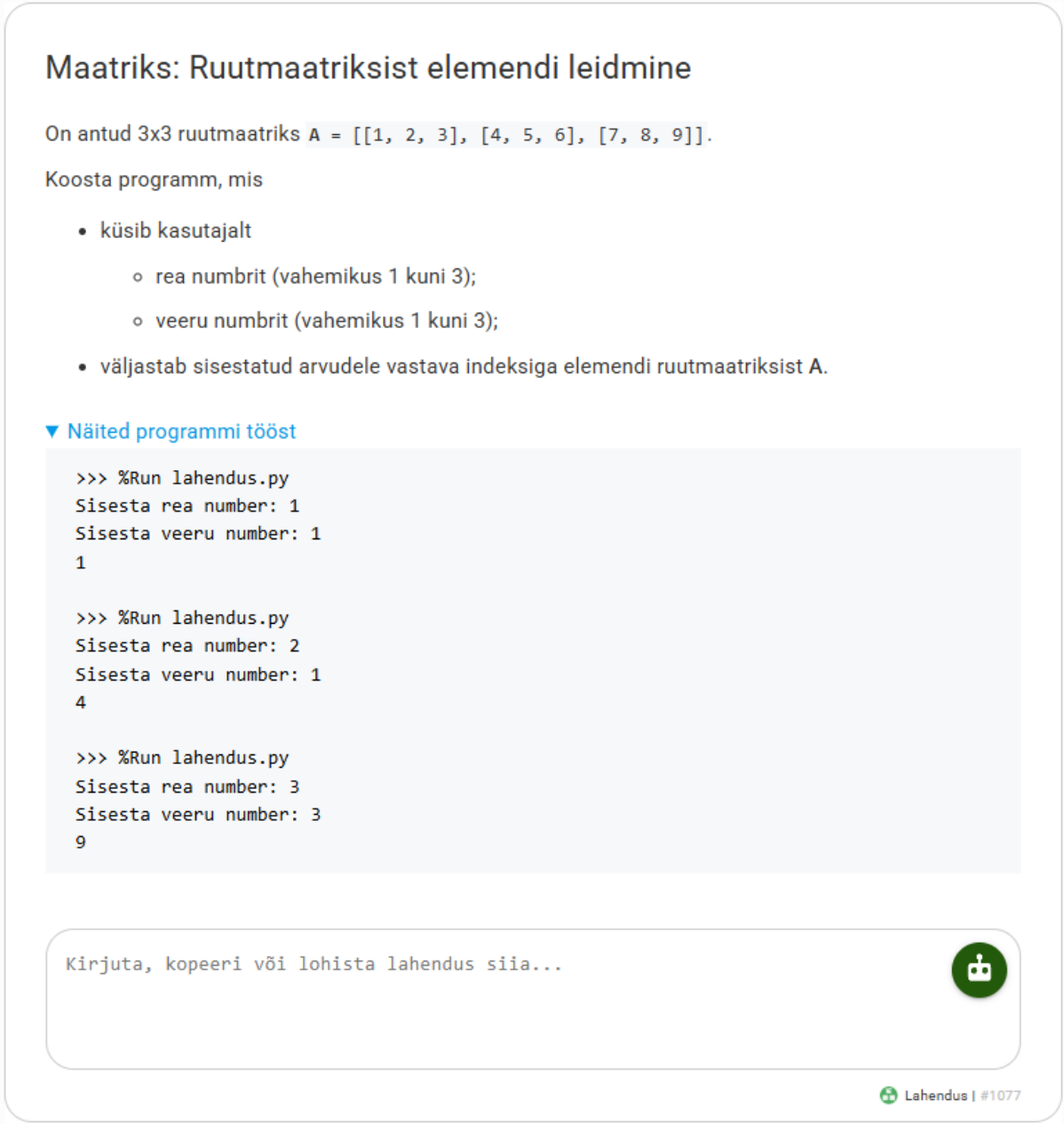
```
>>> %Run lahendus.py
[['Programmeerimise alused I', 'Programmeerimise alused II'], ['Matemaatiline maailmap
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1079

„Maatriks: Ruutmaatriksist elemendi leidmine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART1C



Maatriks: Ruutmaatriksist elemendi leidmine

On antud 3x3 ruutmaatriks `A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`.

Koosta programm, mis

- küsib kasutajalt
  - rea numbrit (vahemikus 1 kuni 3);
  - veeru numbrit (vahemikus 1 kuni 3);
- väljastab sisestatud arvudele vastava indeksiga elemendi ruutmaatriksist **A**.

▼ Näited programmi tööst

```
>>> %Run lahendus.py
Sisesta rea number: 1
Sisesta veeru number: 1
1

>>> %Run lahendus.py
Sisesta rea number: 2
Sisesta veeru number: 1
4

>>> %Run lahendus.py
Sisesta rea number: 3
Sisesta veeru number: 3
9
```

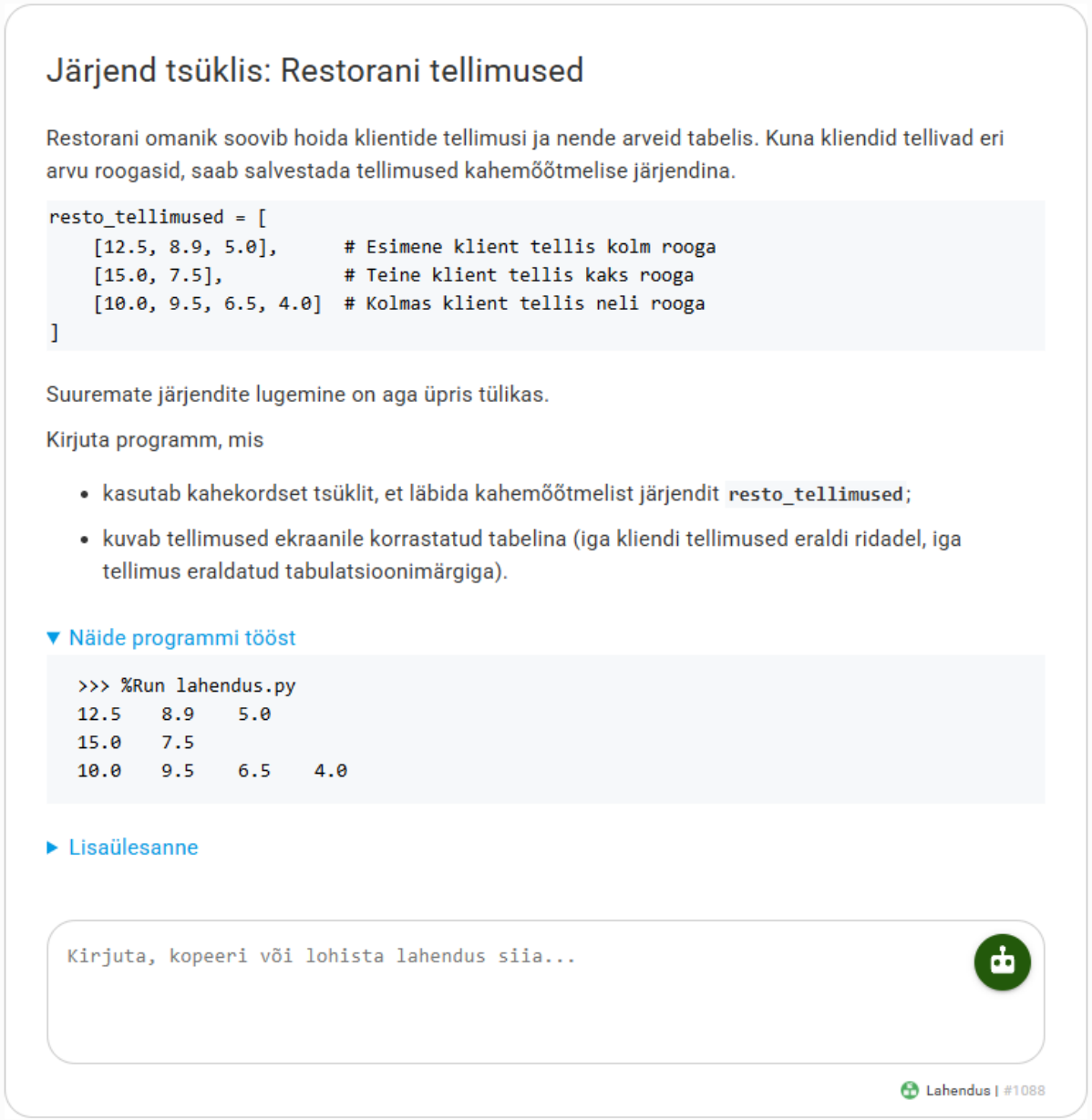
Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1077

B. Kahekordne tsükkel

„Järjend tsüklis: Restorani tellimused“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART2A



Järjend tsüklis: Restorani tellimused

Restorani omanik soovib hoida klientide tellimusi ja nende arveid tabelis. Kuna kliendid tellivad eri arvu roogasid, saab salvestada tellimused kahemõõtmelise järjendina.

```
resto_tellimused = [
    [12.5, 8.9, 5.0],      # Esimene klient tellis kolm rooga
    [15.0, 7.5],           # Teine klient tellis kaks rooga
    [10.0, 9.5, 6.5, 4.0]  # Kolmas klient tellis neli rooga
]
```

Suuremate järjendite lugemine on aga üpris tülikas.

Kirjuta programm, mis

- kasutab kahekordset tsüklit, et läbida kahemõõtmelist järjendit `resto_tellimused`;
- kuvab tellimused ekraanile korrastatud tabelina (iga kliendi tellimused eraldi ridadel, iga tellimus eraldatud tabulatsioonimärgiga).

▼ Näide programmi tööst

```
>>> %Run lahendus.py
12.5    8.9    5.0
15.0    7.5
10.0    9.5    6.5    4.0
```

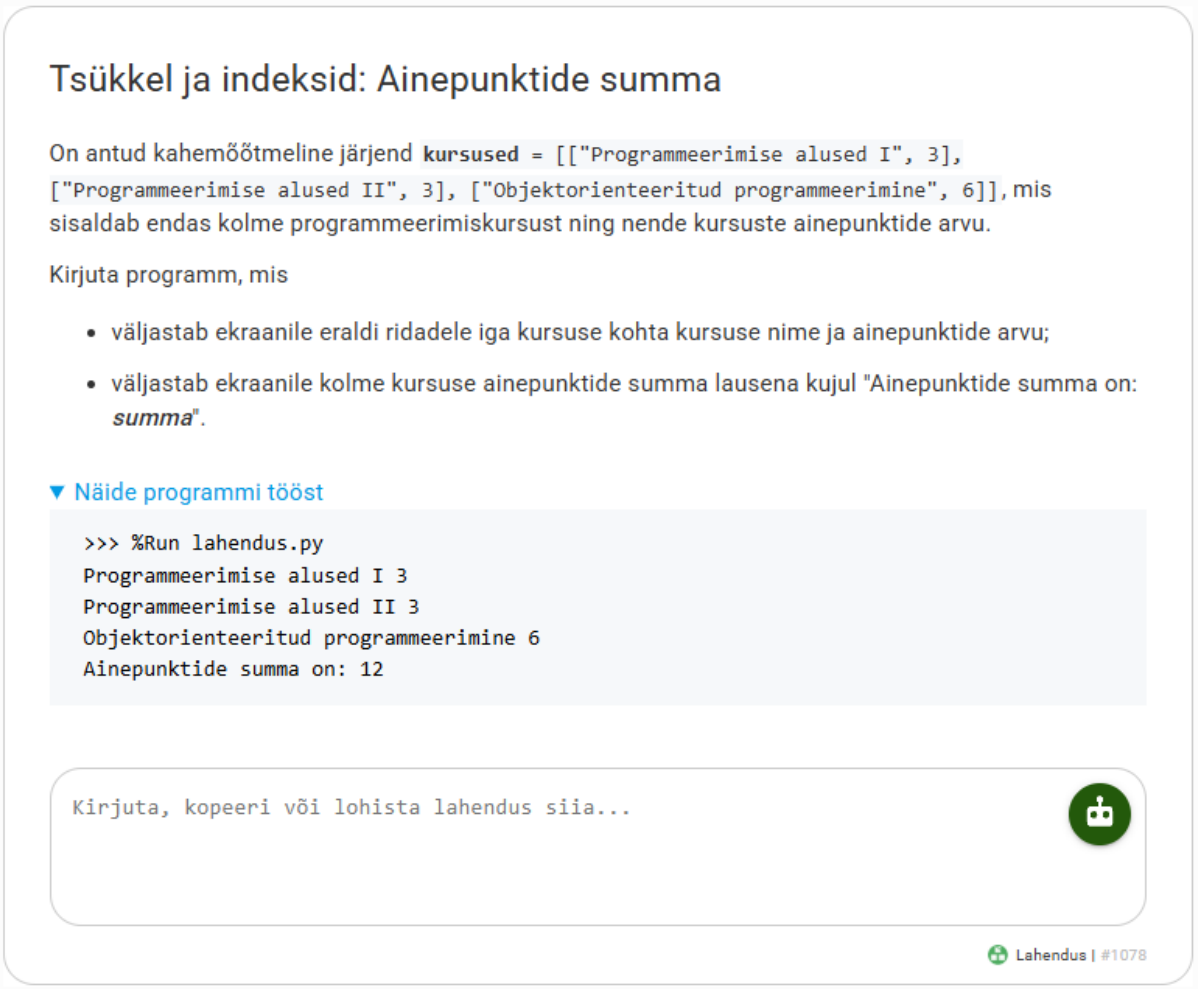
▶ Lisaülesanne

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1088

„Tsükkel ja indeksid: Ainepunktide summa“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART2A



Tsükkel ja indeksid: Ainepunktide summa

On antud kahemõõtmeline järjend `kursused = [["Programmeerimise alused I", 3], ["Programmeerimise alused II", 3], ["Objektorienteeritud programmeerimine", 6]]`, mis sisaldab endas kolme programmeerimiskursust ning nende kursuste ainepunktide arvu.

Kirjuta programm, mis

- väljastab ekraanile eraldi ridadele iga kursuse kohta kursuse nime ja ainepunktide arvu;
- väljastab ekraanile kolme kursuse ainepunktide summa lausena kujul "Ainepunktide summa on: ***summa***".

▼ Näide programmi tööst

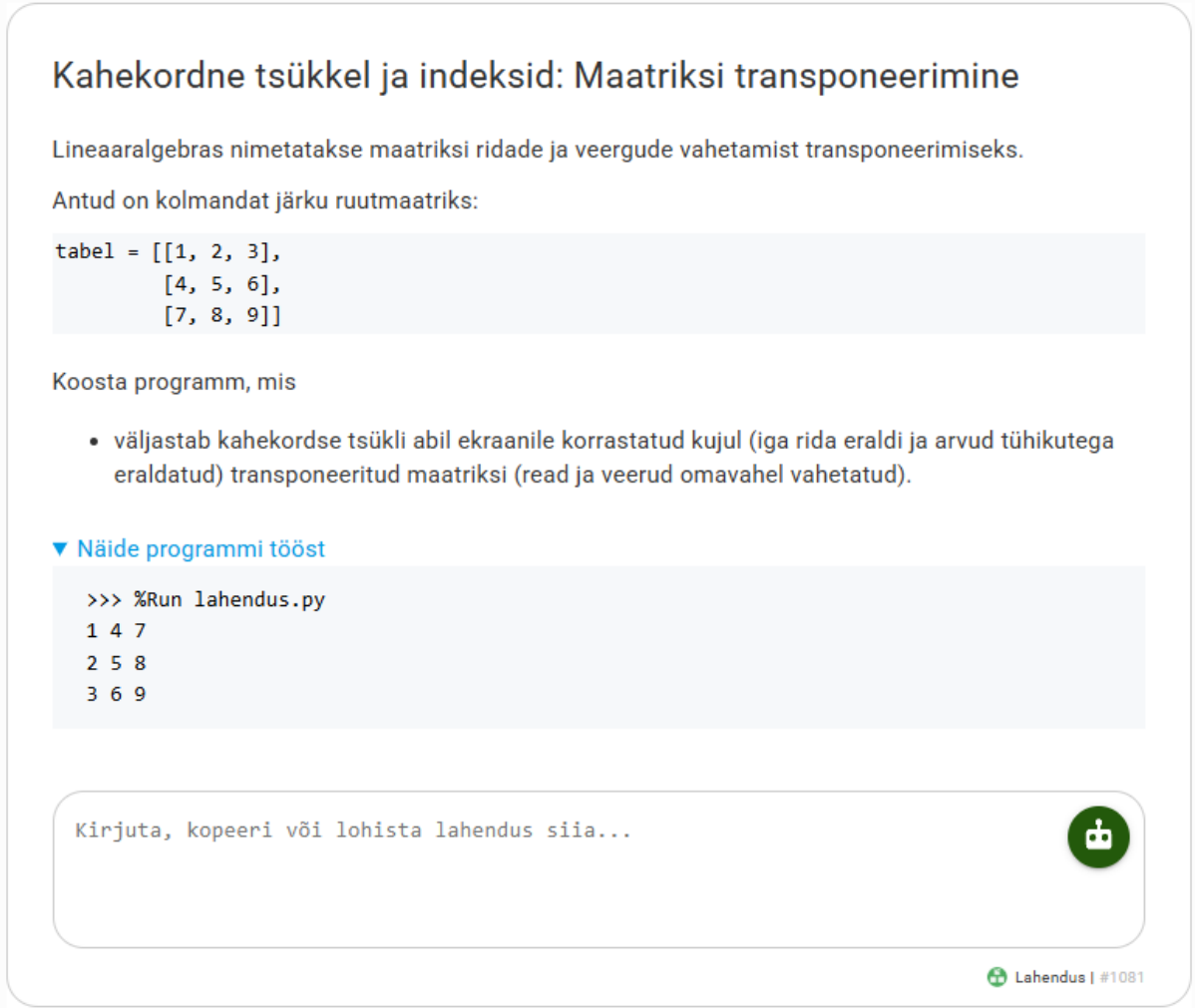
```
>>> %Run lahendus.py
Programmeerimise alused I 3
Programmeerimise alused II 3
Objektorienteeritud programmeerimine 6
Ainepunktide summa on: 12
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1078

„Kahekordne tsükkel ja indeksid: Maatriksi transponeerimine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART2A



Kahekordne tsükkel ja indeksid: Maatriksi transponeerimine

Lineaaralgebras nimetatakse maatriksi ridade ja veergude vahetamist transponeerimiseks.

Antud on kolmandat järku ruutmaatriks:

```
tabel = [[1, 2, 3],
         [4, 5, 6],
         [7, 8, 9]]
```

Koosta programm, mis

- väljastab kahekordse tsükli abil ekraanile korrastatud kujul (iga rida eraldi ja arvud tühikutega eraldatud) transponeeritud maatriksi (read ja veerud omavahel vahetatud).

▼ Näide programmi tööst

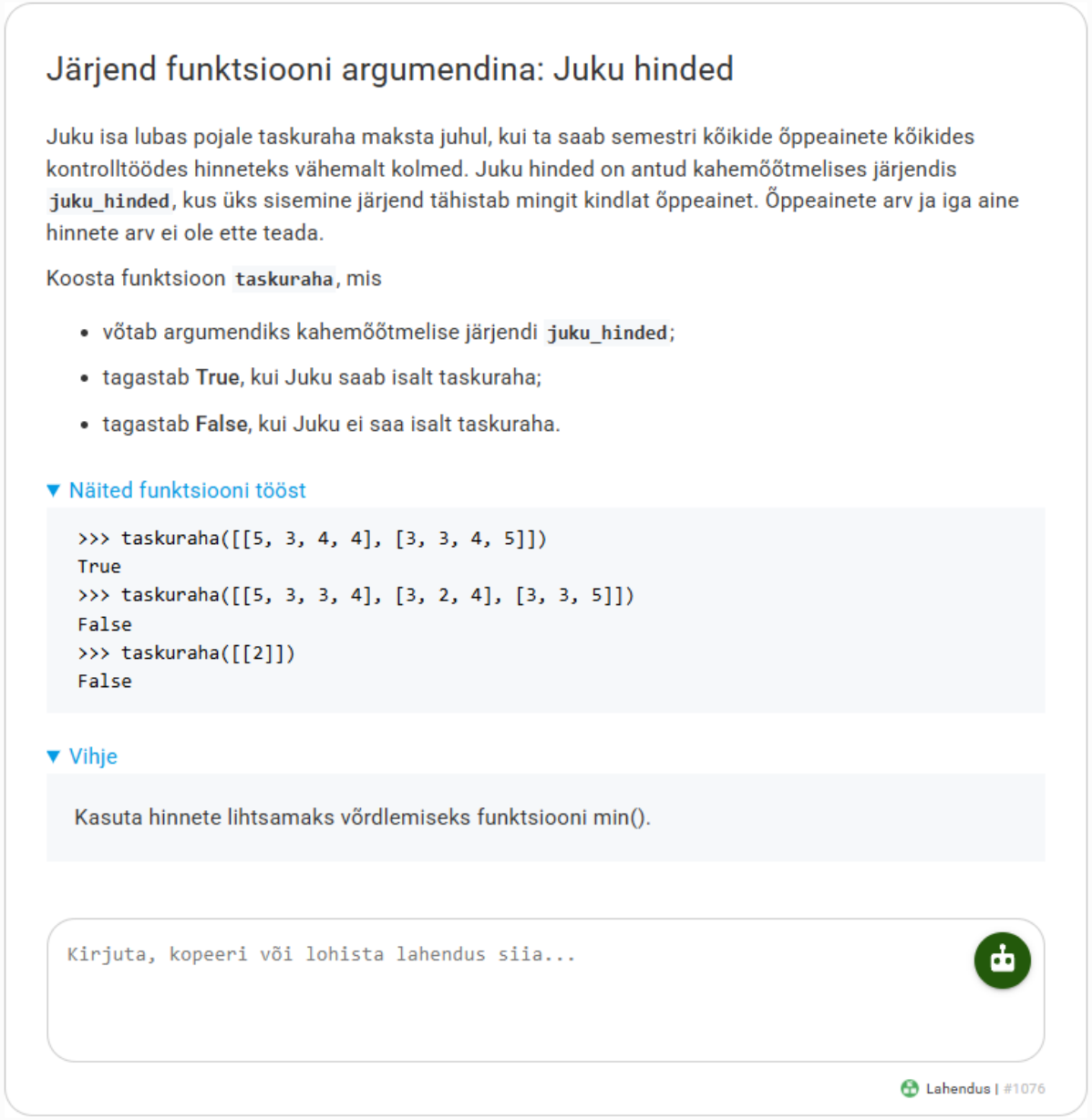
```
>>> %Run lahendus.py
1 4 7
2 5 8
3 6 9
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1081

„Järjend funktsiooni argumendina: Juku hinded“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART2B



Järjend funktsiooni argumendina: Juku hinded

Juku isa lubas pojale taskuraha maksta juhul, kui ta saab semestri kõikide õppeainete kõikides kontrolltöödes hinneteks vähemalt kolmed. Juku hinded on antud kahemõõtmelises järjendis `juku_hinded`, kus üks sisemine järjend tähistab mingit kindlat õppeainet. Õppeainete arv ja iga aine hinnete arv ei ole ette teada.

Koosta funktsioon `taskuraha`, mis

- võtab argumendiks kahemõõtmelise järjendi `juku_hinded`;
- tagastab **True**, kui Juku saab isalt taskuraha;
- tagastab **False**, kui Juku ei saa isalt taskuraha.

▼ Näited funktsiooni tööst

```
>>> taskuraha([[5, 3, 4, 4], [3, 3, 4, 5]])
True
>>> taskuraha([[5, 3, 3, 4], [3, 2, 4], [3, 3, 5]])
False
>>> taskuraha([[2]])
False
```

▼ Vihje

Kasuta hinnete lihtsamaks võrdlemiseks funktsiooni min().

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1076

„Järjend funktsiooni väärtusena: Ruutmaatriks“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART2B

Järjend funktsiooni väärtusena: Ruutmaatriks

Ruutmaatriksi ridade või veergude arvu n nimetatakse selle ruutmaatriksi järguks.

Koosta funktsioon `ruutmaatriks`, mis

- võtab argumendiks ühe positiivse täisarvu **n**;
- tagastab **n** x **n** maatriksi, mille elementideks on arv 0.

▼ Näited funktsiooni tööst

```
>>> ruutmaatriks(3)
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> ruutmaatriks(2)
[[0, 0], [0, 0]]
>>> ruutmaatriks(1)
[[0]]
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1075

„Andmed failist: Punktide kokkulugemine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART2C

Andmed failist: Punktide kokkulugemine

Tekstifailis on igal real tühikuga eraldatult ühe õpilase poolt spordipäeval kogutud punktid kujul:

```
2 5 10
7 6 4
10 5 6
6 9 10
```

Koosta funktsioon `loe_punktid`, mis

- võtab argumendiks tekstifaili nime;
- loeb tekstifailist kokku iga õpilase punktid ning lisab need järjendisse;
- tagastab järjendi, kus on õpilaste kogutud punktide summad.

▼ Näide funktsiooni tööst

```
>>> loe_punktid("andmed.txt")
[17, 17, 21, 25]
```

Koosta programm, mis

- küsib kasutajalt tekstifaili nime;
- kasutab funktsiooni `loe_punktid`, et lugeda eelnevalt sisestatud tekstifailist iga õpilase punktide kogusumma järjendisse;
- väljastab eraldi ridadele saadud järjendi sisu.

▼ Näited programmi tööst

```
>>> %Run lahendus.py
Sisesta tekstifaili nimi: andmed.txt
17
17
21
25
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1080

„Reaalsete andmete kasutamine: Tartu elanike arvu suurimad muutused“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART2C

Reaalsete andmete kasutamine: Tartu elanike arvu suurimad muutused

Eelnevas materjalis oli antud fail `RV0282sm.csv`, kus on andmed Tartu elanike arvu kohta aastatel 2000 kuni 2016.

Võta aluseks eelmise ülesande fail ja kood ning täienda seda nii, et programm väljastaks

- kaks järjestikust aastat, mille vahel oli kõige suurem rahvastiku kasv;
- kaks järjestikust aastat, mille vahel oli kõige suurem rahvastiku langus.

Suurem osa koodist on juba valmis kirjutatud, täiendada tuleks kasvu ja languse leidmise algoritmi. Proovi aru saada koodi loogikast ning loodud muutujate eesmärkidest. Muutma peab ainult neid ridu, kus kood on asendatud kolme plussmärgiga `"+++"`.

▼ Programm, mida peab täiendama

```
fail = open("RV0282sm.csv", encoding="UTF-8")
andmed = [[], []]
for rida in fail:  # loeme ridahaaval
    osad = rida.split(";")  # semikoolonitega eraldatud sõne järjendiks
    andmed[0].append(int(osad[2].strip('"')))
    andmed[1].append(int(osad[3]))
fail.close()

kasv = 0             # salvestatakse elanike arvu kasv, näiteks 290
kasv_aasta_1 = 0     # salvestatakse suurima kasvu algusaasta, näiteks 2001
kasv_aasta_2 = 0     # salvestatakse suurima kasvu lõppaasta, näiteks 2002

langus = 0           # salvestatakse elanike arvu langus, näiteks -3645
langus_aasta_1 = 0   # salvestatakse suurima languse algusaasta, näiteks 2015
langus_aasta_2 = 0   # salvestatakse suurima languse lõppaasta, näiteks 2016

for i in range(len(andmed[0]) - 1):
    muutus = andmed[1][+++] - andmed[1][+++]
    if muutus > kasv:
        kasv_aasta_1 = +++
        kasv_aasta_2 = +++
        kasv = +++
    if muutus < langus:
        langus_aasta_1 = +++
        langus_aasta_2 = +++
        langus = +++

print("Suurim kasv:", "+" + str(kasv), "aastatel", kasv_aasta_1, "->", kasv_aasta_2)
print("Suurim langus:", langus, "aastatel", langus_aasta_1, "->", langus_aasta_2)
```

▶ Näide programmi tööst

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1089

C. Andmestruktuurid

„Järjend ja ennik: Temperatuuride analüüs“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART3A

Järjend ja ennik: Temperatuuride analüüs

Jaagup töötab kohalikus ilmajaamas andmeanalüütikuna. Iga päev mõõdab ilmajaama süsteem automaatselt päeva keskmist temperatuuri. Jaagupile antakse ülesandeks analüüsida ühe nädala jooksul kogutud temperatuure ning koostada neist kokkuvõte. Lõplik kokkuvõte ei tohi olla muudetav.

Aita Jaagupil koostada funktsioon `temp_analyys`, mis

- võtab argumendiks järjendi või enniku, mille elementideks on keskmised temperatuurid (täisarvud);
- tagastab teate "**Puudulikud andmed!**", kui
  - järjendis või ennikus on vähem kui seitsme päeva andmed.
- tagastab kolme elemendiga enniku ehk kolmiku, kus
  - esimene element on esimese seitsme päeva kõige kõrgem temperatuur;
  - teine element on esimese seitsme päeva kõige madalam temperatuur;
  - kolmas element on tõeväärtus **True** või **False**, vastavalt sellele, kas järjendi või enniku esimese seitsme päeva andmetes leidub temperatuur 0.

NB! Võib juhtuda, et argumendina antud järjendis või ennikus on rohkem kui seitsme päeva temperatuurid. Seega peaks leidma tagastatavad väärtused vaid esimese seitsme päeva andmete põhjal (kasuta viilutamist).

▼ Näide funktsiooni tööst

```
>>> temp_analyys([-2, 0, 3, -1, 5])
Puudulikud andmed!

>>> temp_analyys([-2, 0, 3, -1, 5, 6, 7])
(7, -2, True)

>>> temp_analyys((14, 16, 14, 17, 19, 21, 18))
(21, 14, False)

>>> temp_analyys((-1, 1, 3, -1, 5, 6, 7, 8, 6, 2, 0, -2, 1, 4))
(7, -1, False)
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1096

„Hulk: Ostunimekirja uuendaja“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART3B

Hulk: Ostunimekirja uuendaja

Mardi peres on vanad harjumused visad kaduma – ostunimekirja peetakse ikka veel paberi peal. Kui pereliikmed midagi poest vajavad, tuleb see nimekirja enne poeskäiku käsitsi üles kirjutada. Selle asemel, et alati paberit raisata, otsustab Mart nimekirja koostamiseks luua programmi.

Koosta funktsioon `lisa_nimekirja`, mis

- võtab argumendiks ostunimekirja ehk hulga;
- kasutab `while True` tsüklit, et küsida kasutajalt ostunimekirja lisatavate esemete nimetusi;
- lisab sisestatud esemed ostunimekirja;
- tagastab tühja sisendi puhul (ehk kui kasutaja vajutab `enter`) ostunimekirja hetkeseisu.

Selleks, et sisendit küsitaks mistahes arv kordi, tuleks kasutada eelmisest kursusest tuttavat `while True` tsüklit. Tsükkel tuleks `break` abil lõpetada, kui sisendiks on tühisõne (`if sisend == ""`).

▶ Näide funktsiooni tööst

▼ Lisaülesanne

Proovi koostada ka funktsioon `eemalda_nimekirjast`, mis

- võtab argumendiks ostunimekirja ehk hulga;
- kasutab `while True` tsüklit, et küsida eseme nimetust, mida soovitakse nimekirjast eemaldada;
- küsib sisendit uuesti, kui sisestatud ese pole nimekirjas;
- eemaldab sisestatud esemed nimekirjast;
- tagastab tühja sisendi puhul ostunimekirja hetkeseisu.

▶ Näide funktsiooni tööst

```
Kirjuta, kopeeri või lohista lahendus siia...
```

Lahendus | #1098

„Hulk: Ostunimekirja kontrollija“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART3B

Hulk: Ostunimekirja kontrollija

Mart vastutab enamasti oma peres selle eest, et vajalikud asjad saaksid toidupoest ostetud. Üldiselt on Mart säästlik ning ostab ainult neid esemeid, mis eelnevalt ostunimekirja said lisatud. Tihti käivad aga temaga koos poes ka teised pereliikmed, kes alati ostunimekirja ei jälgi. Võib juhtuda, et pereliikmed teevad ettekavatsemata oste või jääb mõni ese üldse ostmata.

Aita Mardil luua programm, mis koostaks ülevaate ostetud, ostmata jäänud ning üleliigsetest esemetest.

Koosta funktsioon `ostunimekirja_kontroll`, mis

- võtab argumentideks kaks hulka, millest üks on ostunimekirja ja teine ostetud esemete hulk;
- tagastab enniku, mille elementideks on hulgad, nii et
  - esimene element on hulk ostetud asjadest, mis olid nimekirjas;
  - teine element on hulk asjadest, mis olid nimekirjas, aga jäid ostmata;
  - kolmas element on hulk ostetud asjadest, mis osteti ettekavatsemata (osteti, aga ei olnud nimekirjas).

▶ Näide funktsiooni tööst

Võid leitud hulkade põhjal proovida väljastada ekraanile ka poeskäigu ülevaate loetaval kujul (automaatkontroll seda ei kontrolli).

Näiteks:

```
Ostunimekiri: {'sai', 'riis', 'mahl', 'kanaliha', 'leib'}
Ostetud asjad: {'kohuke', 'limonaad', 'krõpsud', 'leib'}
Nimekirjast osteti: {'leib'}
Ostmata jäi: {'riis', 'sai', 'kanaliha', 'mahl'}
Üleliigsed ostud: {'kohuke', 'limonaad', 'krõpsud'}
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1097

„Sõnastik: Autode hooldusajalugu“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART3C

Sõnastik: Autode hooldusajalugu

Andres töötab autoremodnitöökojas mehaanikuna. Igapäevaselt käivad tema juures kliendid, kes toovad oma auto korralisse hooldusse. Peale hooldust märgitakse auto number ja hoolduse teostamise kuupäev süsteemi. Kuna töökoda alustas tegevusega alles hiljuti, puudub neil päringute süsteem, mis tuletaks meelde auto viimase hoolduse kuupäeva, mille alusel saaks hiljem kliendile saata meeldetuletuse järgmise korralise hoolduse kohta.

Aita luua programm, mis tuletab auto registreerimisnumbri põhjal meelde viimase hoolduse kuupäeva.

Koosta funktsioon `hooldus`, mis

- võtab argumendiks sõnastiku, mille võtmeteks on autode registreerimisnumbrid (sõnedena) ning väärtusteks viimased hoolduskuupäevad (sõnedena);
- küsib kasutajalt sisendina auto registreerimisnumbrit;
- tagastab sõnumi `"Auto numbrimärgiga numbrimärk pole töökojas varem käinud."`, kui numbrimärki pole sõnastikus;
- tagastab enniku, mille esimene element on kasutaja sisestatud auto numbrimärk ja teine element on sama numbrimärgiga auto viimase hoolduse kuupäev.

Funktsiooni argumendina võid kasutada järgmist sõnastikku:

```
hooldusajalugu = {
    "123ABC": "15/02/2024",
    "456DEF": "20/11/2024",
    "789GHI": "10/01/2024",
    "321JKL": "05/09/2023",
    "654MNO": "01/03/2025"
}
```

▼ Näide funktsiooni tööst

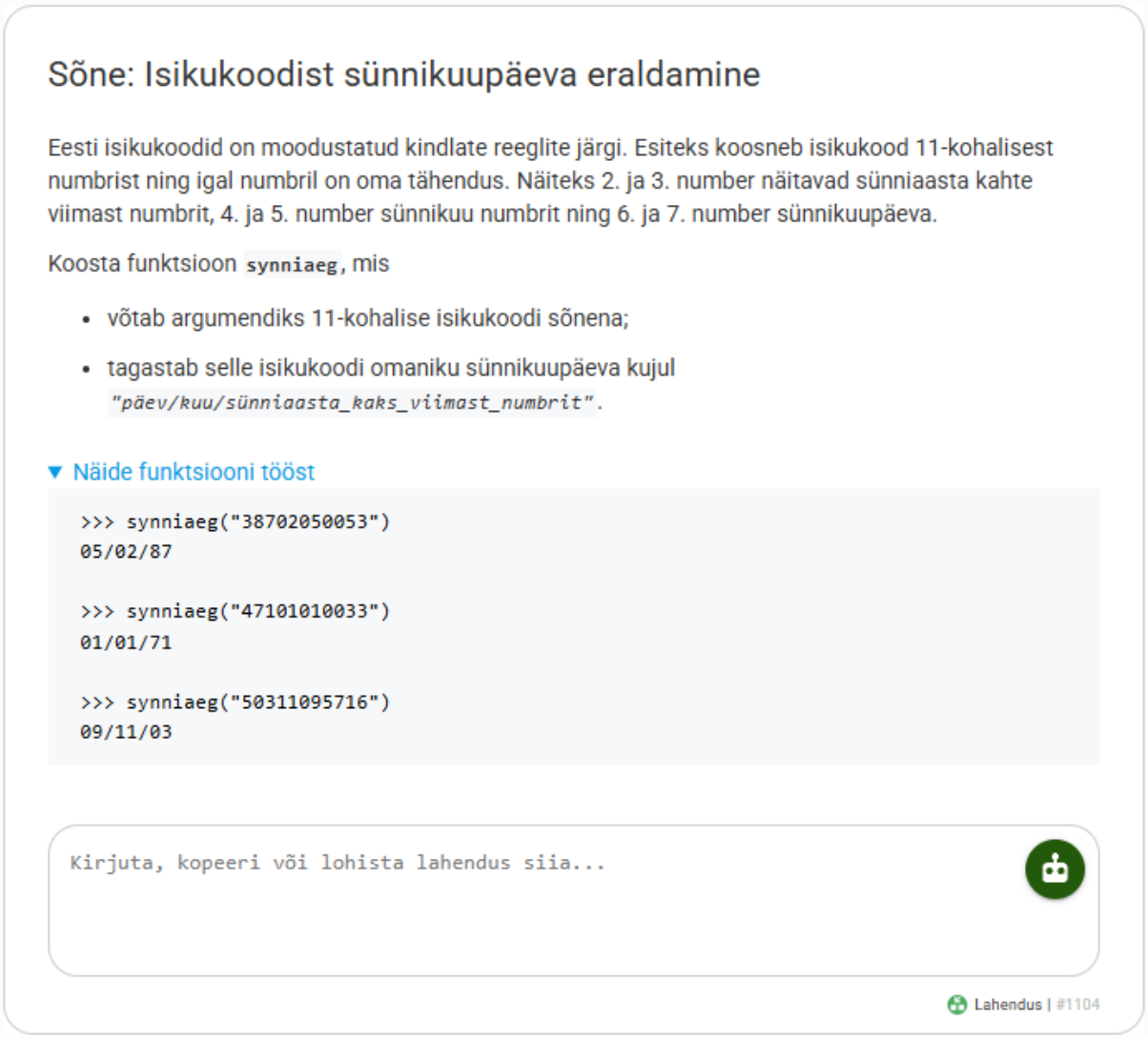
```
>>> hooldus({"123ABC": "15/02/2024", "456DEF": "20/11/2024", "789GHI": "10/01/2024", "
Sisesta auto number: 123ABC
('123ABC', '15/02/2024')

>>> hooldus({"123ABC": "15/02/2024", "456DEF": "20/11/2024", "789GHI": "10/01/2024", "
Sisesta auto number: 486HJZ
Auto numbrimärgiga 486HJZ pole töökojas varem käinud.
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1103

„Sõne: Isikukoodist sünnikuupäeva eraldamine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART3D



Sõne: Isikukoodist sünnikuupäeva eraldamine

Eesti isikukoodid on moodustatud kindlate reeglite järgi. Esiteks koosneb isikukood 11-kohalisest numbrist ning igal numbril on oma tähendus. Näiteks 2. ja 3. number näitavad sünniaasta kahte viimast numbrit, 4. ja 5. number sünnikuu numbrit ning 6. ja 7. number sünnikuupäeva.

Koosta funktsioon `synniaeg`, mis

- võtab argumendiks 11-kohalise isikukoodi sõnena;
- tagastab selle isikukoodi omaniku sünnikuupäeva kujul *`"päev/kuu/sünniaasta_kaks_viimast_numbrit"`*.

▼ Näide funktsiooni tööst

```
>>> synniaeg("38702050053")
05/02/87

>>> synniaeg("47101010033")
01/01/71

>>> synniaeg("50311095716")
09/11/03
```

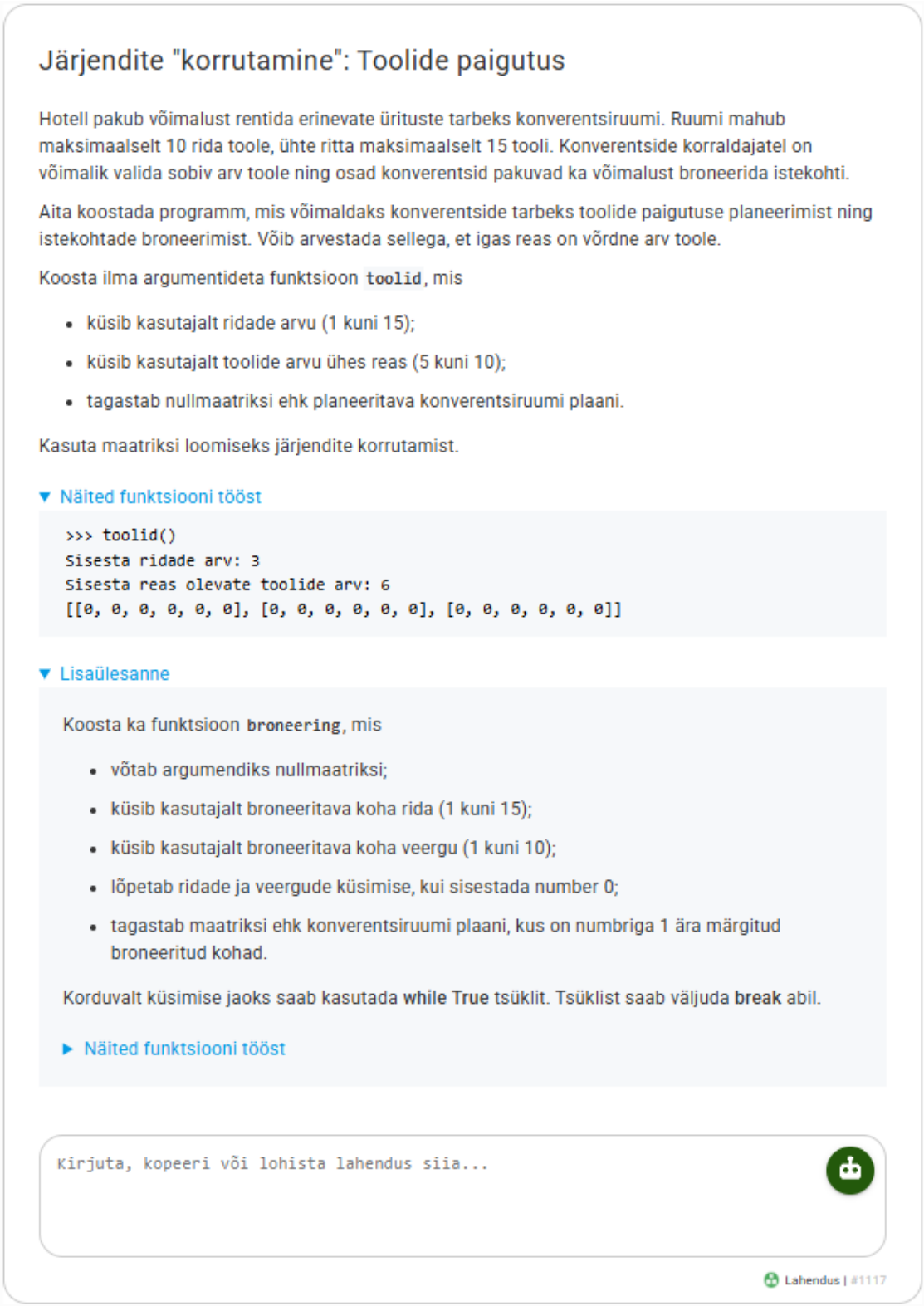
Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1104

D. Viitamine ja muteerimine

„Järjendite "korrutamine": Toolide paigutus“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART4C



Järjendite "korrutamine": Toolide paigutus

Hotell pakub võimalust rentida erinevate ürituste tarbeks konverentsiruumi. Ruumi mahub maksimaalselt 10 rida toole, ühte ritta maksimaalselt 15 tooli. Konverentside korraldajatel on võimalik valida sobiv arv toole ning osad konverentsid pakuvad ka võimalust broneerida istekohti.

Aita koostada programm, mis võimaldaks konverentside tarbeks toolide paigutuse planeerimist ning istekohtade broneerimist. Võib arvestada sellega, et igas reas on võrdne arv toole.

Koosta ilma argumentideta funktsioon `toolid`, mis

- küsib kasutajalt ridade arvu (1 kuni 15);
- küsib kasutajalt toolide arvu ühes reas (5 kuni 10);
- tagastab nullmaatriksi ehk planeeritava konverentsiruumi plaani.

Kasuta maatriksi loomiseks järjendite korrutamist.

▼ Näited funktsiooni tööst

```
>>> toolid()
Sisesta ridade arv: 3
Sisesta reas olevate toolide arv: 6
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

▼ Lisaülesanne

Koosta ka funktsioon `broneering`, mis

- võtab argumendiks nullmaatriksi;
- küsib kasutajalt broneeritava koha rida (1 kuni 15);
- küsib kasutajalt broneeritava koha veergu (1 kuni 10);
- lõpetab ridade ja veergude küsimise, kui sisestada number 0;
- tagastab maatriksi ehk konverentsiruumi plaani, kus on numbriga 1 ära märgitud broneeritud kohad.

Korduvalt küsimise jaoks saab kasutada **while True** tsüklit. Tsüklist saab väljuda **break** abil.

▶ Näited funktsiooni tööst

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1117

„Funktsioon ja muteerimine: Ostukorvi soodustus“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART4D

Funktsioon ja muteerimine: Ostukorvi soodustus

Käimas on Must Reede ning e-poes on hetkel kogu kaup 25% soodsam. Selles e-poes rakendatakse soodustus alles ostukorvi sisu kuvades.

Koosta funktsioon `soodustus`, mis

- võtab argumendiks järjendi ehk ostukorvi;
- rakendab ostukorvile 25% soodustuse (iga hind ümardada pärast soodustuse rakendamist kahe komakohani);
- tagastab ostukorvi lõpphindade summa.

Ümardamiseks võib Pythonis kasutada sisseehitatud funktsiooni **round**.

```
>>> round(1.475, 2)
1.48
```

Oluline on, et funktsioon muudaks soodustust rakendades etteantud argumenti (järjendit), mitte ei tagastaks uut järjendit.

▼ Näide funktsiooni tööst

```
>>> soodustus([20.00, 15.50, 9.99, 30.00])
56.61
```

Koosta põhiprogramm, mis

- väljastab funktsiooni rakendades ekraanile ostukorvi lõpphindade summa;
- väljastab ekraanile järjendina ostukorvi, millele on rakendatud 25% soodustus.

Kuna programmi mõte on muuta olemasoleva järjendi sisu, peaksid väljastama ekraanile selle muutuja sisu, mille andsid funktsioonile argumendiks.

Näiteks nii:

```
ostukorv = [20.00, 15.50, 9.99, 30.00]
print(soodustus(ostukorv))
print(ostukorv)
```

▼ Näide programmi tööst

```
>>> %Run lahendus.py
56.61
[15.0, 11.62, 7.49, 22.5]
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1120

„Töövoost, mida ikka kasutame: Maakondande kilometraaž“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART4K

Töövoost, mida ikka kasutame: Maakondande kilometraaž

Eelnevas materjalis on antud fail **maakonnad.txt**, kus on andmed Tartu laulupeole tulijate kohta. Igal real on kirjas maakond, selle keskuse kaugus Tartust, mitu meest on tulemas ja mitu naist. Vaja on leida maakond, mille inimesed läbivad laulupeole edasi-tagasi minekuga kokku kõige rohkem kilometreid. See tähendab, et näiteks kui 10 inimest lähevad laulupeople ning vahemaa oli 30 kilometrit, siis järelikult selle maakonna inimesed läbisid kokku 10 * 30 * 2 = 600 kilometrit.

▶ Faili **maakonnad.txt** sisu

Koosta funktsioon `maakondade_km`, mis

- võtab argumendiks faili nime;
- loeb failist `maakonnad.txt` vajalikud andmed;
- koostab sõnastiku, mille võtmeteks on maakondade nimed ning väärtusteks nende maakondade inimeste edasi-tagasi kilometraaž Tartusse;
- tagastab eelnevalt kirjeldatud sisuga sõnastiku.

▼ Näide funktsioonist

```
def maakondade_km(failinimi):
    fail = open(failinimi, encoding="UTF-8")
    maakondade_sonastik = {}

    for rida in fail:
        """
        Lisa siia andmete töötlemise loogika
        """

    fail.close()
    return maakondade_sonastik
```

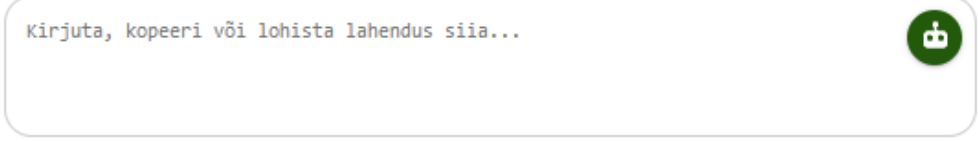
▶ Näide funktsiooni tööst

Koosta põhiprogramm, mis

- koostab maakondade nimede ja nende maakondade inimeste edasi-tagasi klometraažide sõnastiku, rakendades funktsiooni `maakondade_km`;
- leiab ja väljastab ekraanile suurima kilometraažiga maakonna nime ja kilometrite arvu.

Tuleta meelde, kuidas sõnastikust võtmete väärtusi kätte saada. Vajadusel saad abi peatükist "Andmestruktuurid".

▶ Näide programmi tööst



Lahendus | #1121

„Sama programm hoopis funktsiooniga: Ülepinge kontroll“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART4K

Sama programm hoopis funktsiooniga: Ülepinge kontroll

Kohalikust alajaamast jõuab elektrienergia tarbijani pingel 230 V. Tänaval asuvad äsja ehitatud majad, mis on jagatud kolmeks sektsiooniks. Sektsioonid on antud kahemõõtmelises järjendis, kus üks rida tähistab ühte sektsiooni ning ühe sektsiooni element tähistab ühe maja elektrienergia pinget, mis peaks tavaliselt olema 230 V. Kui pinge on kasvõi ühes majas suurem kui 230 V, tuleb tehnikutel see sektsioon üle vaadata.

Selleks on koostatud programm, mis peaks kontrollima sektsioonide üldist seisu ning väljastama, mitmes sektsioonis esineb ülepinge.

```
sektsioonid = [
    [230, 230, 230],
    [250, 230, 230],
    [230, 230, 250]
]

ylepingega_sektsioonid = 0
for i in range(len(sektsioonid)):
    for j in range(len(sektsioonid[i])):
        if sektsioonid[i][j] >= 230:
            ylepinge = True
        else:
            ylepinge = False
    if ylepinge == True:
        ylepingega_sektsioonid += 1

print("Leitud", ylepingega_sektsioonid, "ülepingega sektsiooni.")
```

Programm on aga hetkel vigane ning konkreetse probleemi jaoks ka liiga keerulise lahendusega. Ülesande mõtteks on koostada uus funktsioon, mis lahendaks probleemi võimalikult lihtsalt ja väheste ridadega. Sellest tulenevalt ei ole siin ka kindlat "õiget" lahendust. Näiteks on võimalik ülesannet lahendada ilma kahekordset tsüklit kasutamata. Proovi varasemalt õpitu põhjal leida kõige sobivam ja mõistlikum lahendus.

Koosta funktsioon `ylepinged`, mis

- saab argumendiks kahemõõtmelise järjendi, milles on sektsioonide pinged;
- kontrollib, kas sektsioonides esineb ülepingeid;
- tagastab sektsioonide üldseisu ehk arvu, mitmes sektsioonis esineb ülepinge (ülepinge puudumisel tagastab 0).

▶ Näide funktsiooni tööst

Koosta põhiprogramm, mis

- rakendab funktsiooni `ylepinged` ja väljastab tulemuse ekraanile arusaadava lausena.

▶ Näide programmi tööst

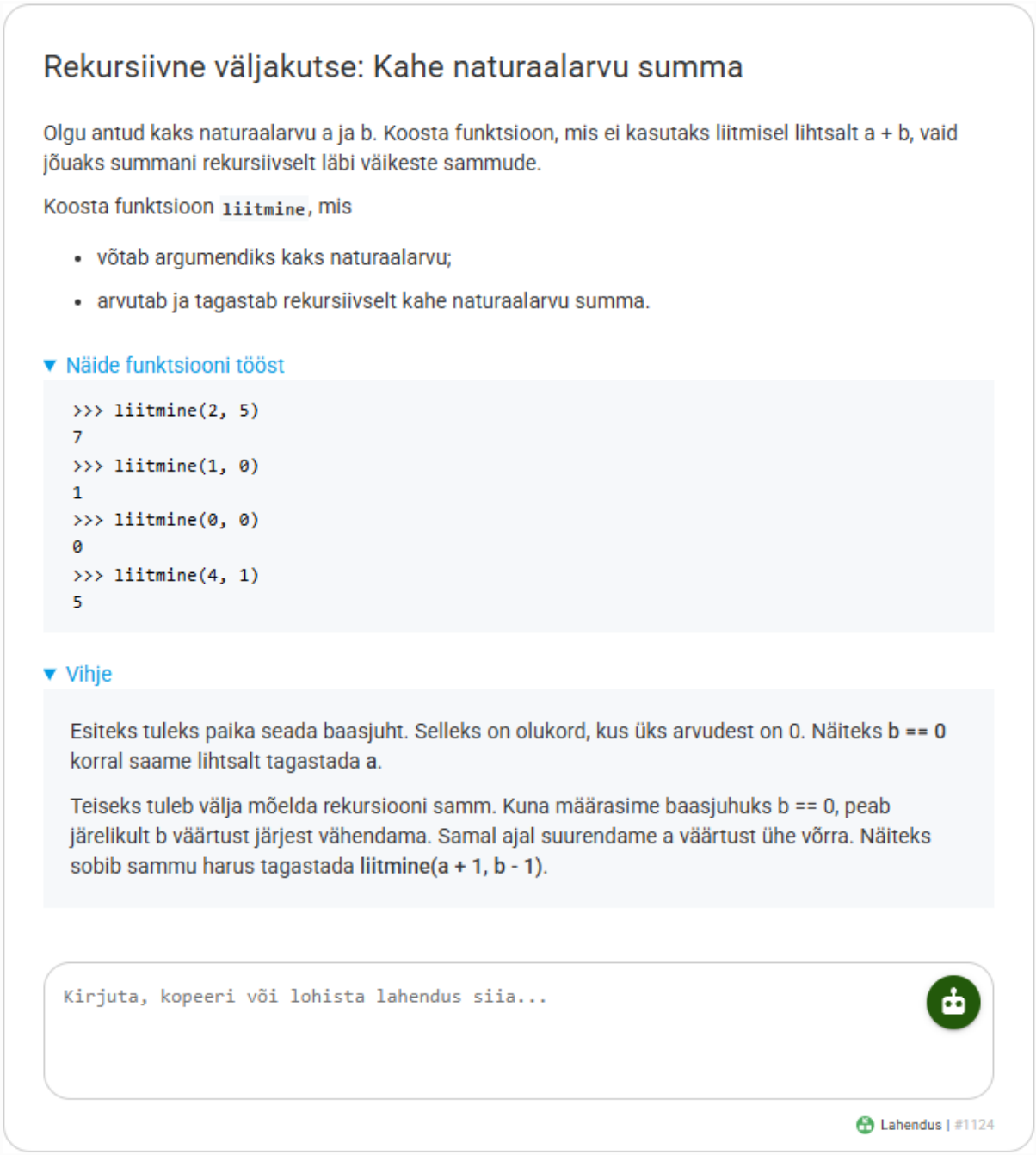
Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1122

E. Testimine ja silumine. Rekursioon

„Rekursiivne väljakutse: Kahe naturaalarvu summa“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART5B



Rekursiivne väljakutse: Kahe naturaalarvu summa

Olgu antud kaks naturaalarvu a ja b. Koosta funktsioon, mis ei kasutaks liitmisel lihtsalt a + b, vaid jõuaks summani rekursiivselt läbi väikeste sammude.

Koosta funktsioon `liitmine`, mis

- võtab argumendiks kaks naturaalarvu;
- arvutab ja tagastab rekursiivselt kahe naturaalarvu summa.

▼ Näide funktsiooni tööst

```
>>> liitmine(2, 5)
7
>>> liitmine(1, 0)
1
>>> liitmine(0, 0)
0
>>> liitmine(4, 1)
5
```

▼ Vihje

Esiteks tuleks paika seada baasjuht. Selleks on olukord, kus üks arvudest on 0. Näiteks **b == 0** korral saame lihtsalt tagastada **a**.

Teiseks tuleb välja mõelda rekursiooni samm. Kuna määrasime baasjuhuks b == 0, peab järelikult b väärtust järjest vähendama. Samal ajal suurendame a väärtust ühe võrra. Näiteks sobib sammu harus tagastada **liitmine(a + 1, b - 1)**.

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1124

„Rekursiivne väljakutse: Kahe naturaalarvu vahe“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART5B

Rekursiivne väljakutse: Kahe naturaalarvu vahe

Olgu antud kaks naturaalarvu a ja b. Koosta funktsioon, mis ei kasutaks lahutamisel lihtsalt a - b, vaid jõuaks vaheni rekursiivselt läbi väikeste sammude.

Koosta funktsioon `lahutamine`, mis

- võtab argumendiks kaks naturaalarvu;
- arvutab ja tagastab rekursiivselt kahe naturaalarvu vahe.

▼ Näide funktsiooni tööst

```
>>> lahutamine(5, 2)
3
>>> lahutamine(3, 4)
-1
>>> lahutamine(1, 0)
1
>>> lahutamine(0, 0)
0
```

▼ Vihje

Esiteks tuleks paika seada baasjuht. Selleks on olukord, kus üks arvudest on 0. Näiteks **b == 0** korral saame lihtsalt tagastada **a**.

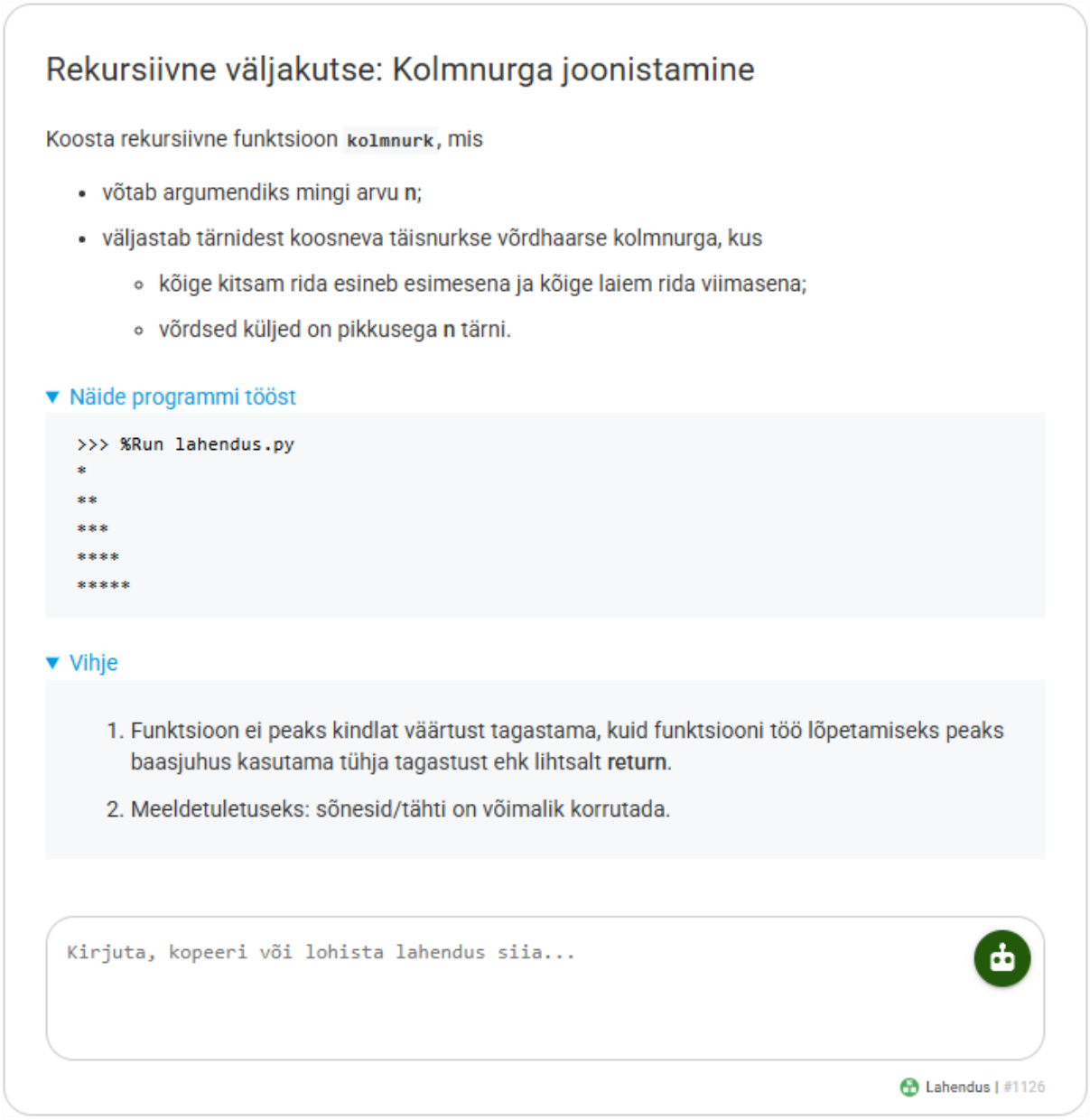
Teiseks tuleb välja mõelda rekursiooni samm. Kuna määrasime baasjuhuks b == 0, peab järelikult b väärtust järjest vähendama. Samal ajal vähendame ka a väärtust ühe võrra. Näiteks sobib sammu harus tagastada **lahutamine(a - 1, b - 1)**.

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1125

„Rekursiivne väljakutse: Kolmnurga joonistamine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART5C



Rekursiivne väljakutse: Kolmnurga joonistamine

Koosta rekursiivne funktsioon `kolmnurk`, mis

- võtab argumendiks mingi arvu **n**;
- väljastab tärnidest koosneva täisnurkse võrdhaarse kolmnurga, kus
  - kõige kitsam rida esineb esimesena ja kõige laiem rida viimasena;
  - võrdsed küljed on pikkusega **n** tärni.

▼ Näide programmi tööst

```
>>> %Run lahendus.py
*
**
***
****
*****
```

▼ Vihje

1. Funktsioon ei peaks kindlat väärtust tagastama, kuid funktsiooni töö lõpetamiseks peaks baasjuhus kasutama tühja tagastust ehk lihtsalt **return**.
2. Meeldetuletuseks: sõnesid/tähti on võimalik korrutada.

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1126

„Rekursioon sõnede ja järjenditega: Järjendi arvude summa leidmine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART5C

Rekursioon sõnede ja järjenditega: Järjendi arvude summa leidmine

Koosta funktsioon `rek_summa`, mis

- võtab argumendiks järjendi, mille elementideks on arvud;
- leiab rekursiivselt järjendis olevate elementide summa ja tagastab selle.

▼ Näide funktsiooni tööst

```
>>> rek_summa([1, 2, 3])
6
>>> rek_summa([15])
15
>>> rek_summa([5, 7, -8])
4
>>> rek_summa([])
0
```

▼ Vihje

Baasjuhuks on olukord, kus järjend on tühi.

Rekursiooni sammul peaks liitma järjendi esimese elemendi rekursiivsele väljakutsele, mille argumendiks on ülejäänud järjendi elemendid. Siin võiks abi olla järjendi viilutamisest.

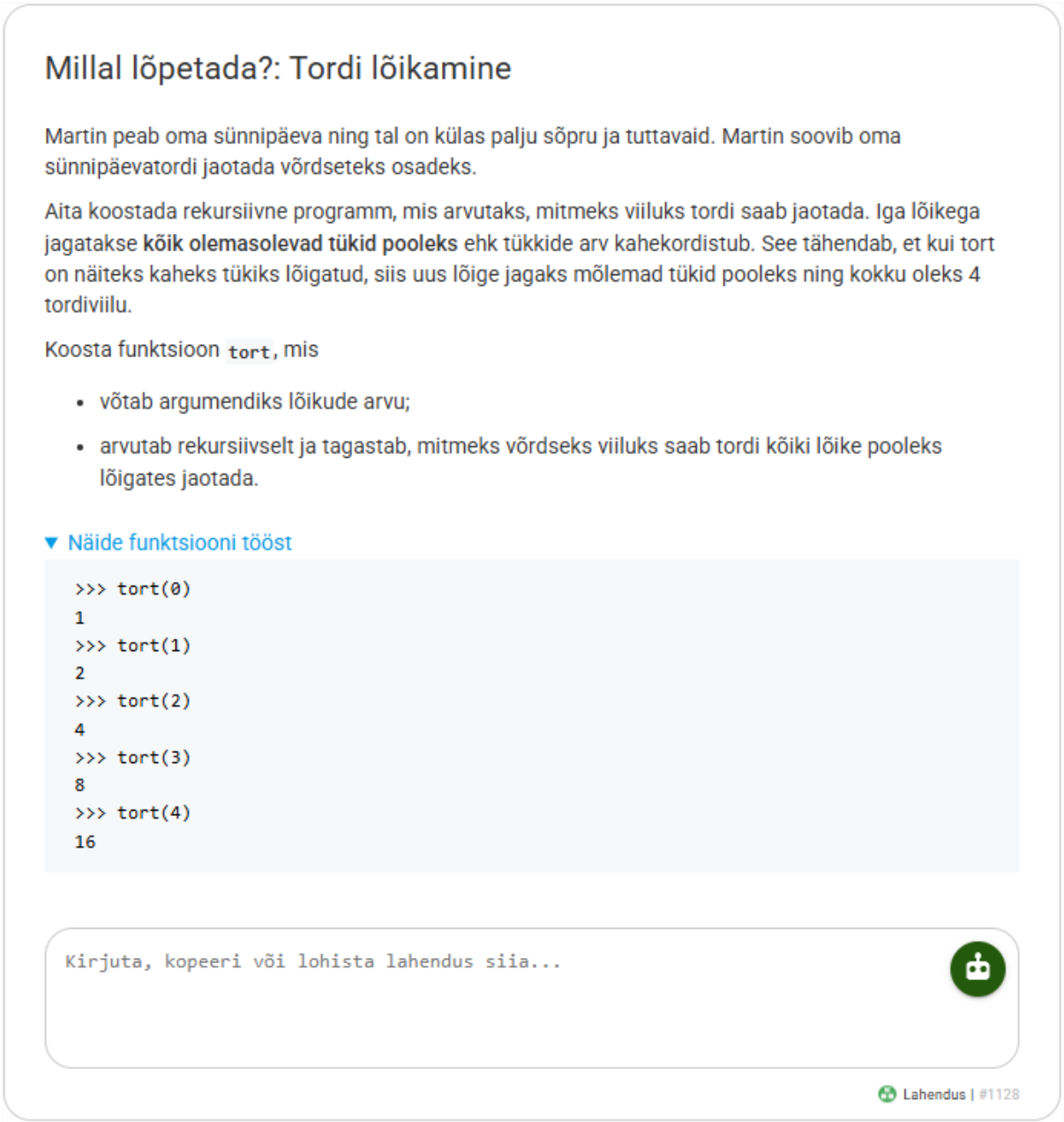
Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1127

F. Rekursioon II

„Millal lõpetada?: Tordi lõikamine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6A



Millal lõpetada?: Tordi lõikamine

Martin peab oma sünnipäeva ning tal on külas palju sõpru ja tuttavaid. Martin soovib oma sünnipäevatordi jaotada võrdseteks osadeks.

Aita koostada rekursiivne programm, mis arvutaks, mitmeks viiluks tordi saab jaotada. Iga lõikega jagatakse **kõik olemasolevad tükid pooleks** ehk tükkide arv kahekordistub. See tähendab, et kui tort on näiteks kaheks tükiks lõigatud, siis uus lõige jagaks mõlemad tükid pooleks ning kokku oleks 4 tordiviilu.

Koosta funktsioon `tort`, mis

- võtab argumendiks lõikude arvu;
- arvutab rekursiivselt ja tagastab, mitmeks võrdseks viiluks saab tordi kõiki lõike pooleks lõigates jaotada.

▼ Näide funktsiooni tööst

```
>>> tort(0)
1
>>> tort(1)
2
>>> tort(2)
4
>>> tort(3)
8
>>> tort(4)
16
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1128

„Akumulaator: Astendamine rekursiivselt“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6A

Akumulaator: Astendamine rekursiivselt

Pythonis on olemas operaator **, mille abil on võimalik arve astendada. Astendamist saab teostada ka rekursiivselt, ilma eelnevat operaatorit kasutamata.

Koosta rekursiivne funktsioon `astenda`, mis

- võtab argumendiks kolm muutujat
  - arv, mida astendatakse;
  - arvu astendaja;
  - akumulaator, mille vaikeväärtus on 1 (`aku = 1`).
- astendab arvu ning tagastab lõpptulemuse.

▼ Näide funktsiooni tööst

```
>>> astenda(2, 0)
1
>>> astenda(1, 2)
1
>>> astenda(2, 2)
4
>>> astenda(2, 10)
1024
```

▼ Vihje

Baasjuht on see, kui arvu astendaja on 0. Tagastama peaks akumulaatori väärtuse.

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1129

„Sabarekursioon: Arvude 1 kuni n summa“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6A

Sabarekursioon: Arvude 1 kuni n summa

Allpool on antud tsükliga funktsioon, mis arvutab arvude 1 kuni **n** summa. Teisenda see funktsioon sabarekursiooniga funktsiooniks.

```
def summa(n):
    tulemus = 0
    for i in range(1, n + 1):
        tulemus += i
    return tulemus
```

▼ Näide funktsiooni tööst

```
>>> summa(3)
6
>>> summa(5)
15
```

Koosta rekursiivne funktsioon `summa_saba`, mis

- võtab argumendiks kaks muutujat
  - arv, milleni arve kokku liidetakse;
  - akumulaator, mille vaikeväärtus on 0.
- tagastab arvude 1 kuni **n** summa

▼ Näide funktsiooni tööst

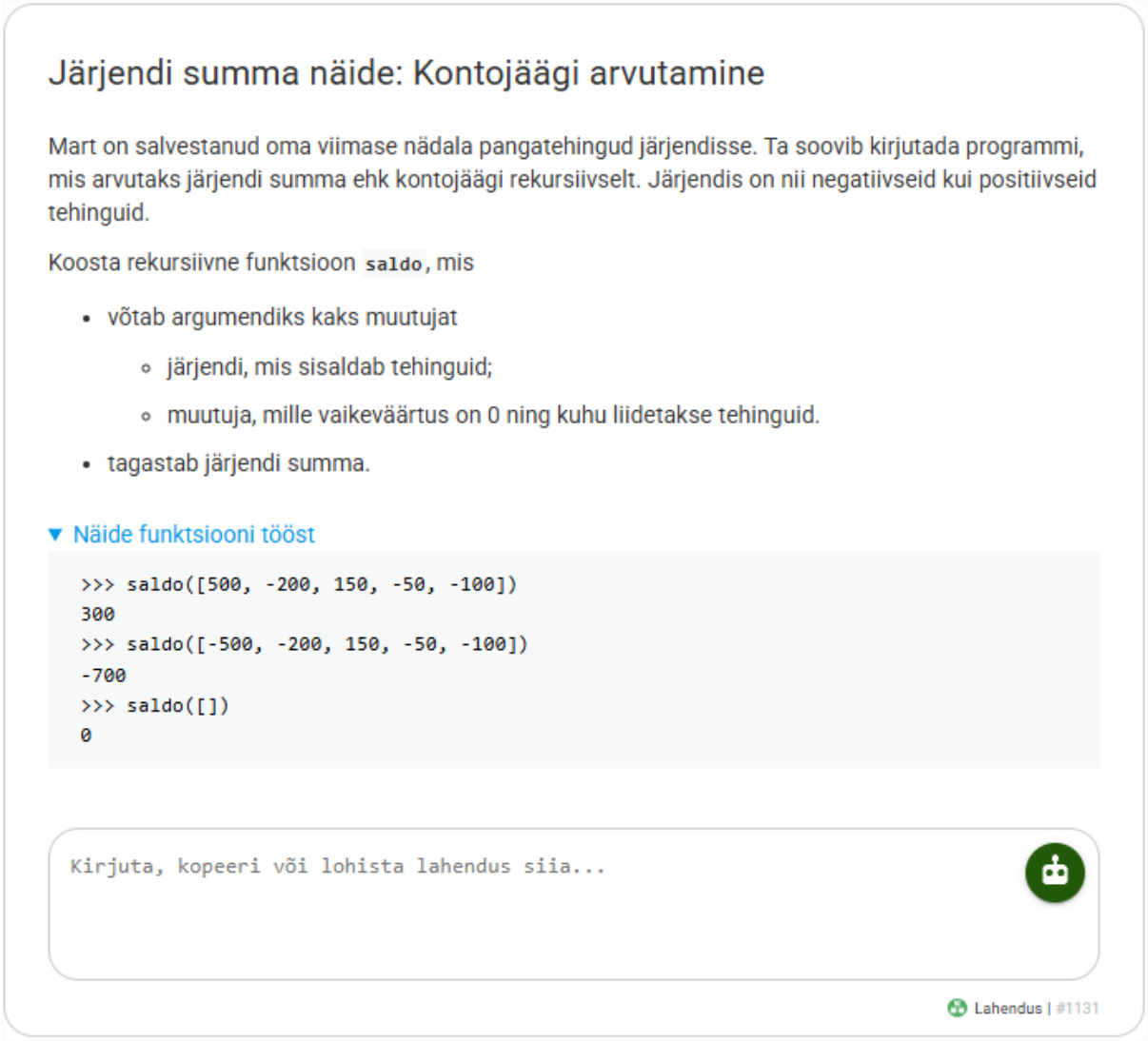
```
>>> summa_saba(0)
0
>>> summa_saba(1)
1
>>> summa_saba(3)
6
>>> summa_saba(5)
15
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1130

„Järjendi summa näide: Kontojäägi arvutamine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6A



Järjendi summa näide: Kontojäägi arvutamine

Mart on salvestanud oma viimase nädala pangatehingud järjendisse. Ta soovib kirjutada programmi, mis arvutaks järjendi summa ehk kontojäägi rekursiivselt. Järjendis on nii negatiivseid kui positiivseid tehinguid.

Koosta rekursiivne funktsioon `saldo`, mis

- võtab argumendiks kaks muutujat
  - järjendi, mis sisaldab tehinguid;
  - muutuja, mille vaikeväärtus on 0 ning kuhu liidetakse tehinguid.
- tagastab järjendi summa.

▼ Näide funktsiooni tööst

```
>>> saldo([500, -200, 150, -50, -100])
300
>>> saldo([-500, -200, 150, -50, -100])
-700
>>> saldo([])
0
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1131

„Kahendotsing: Lennupiletite otsimine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6B

Kahendotsing: Lennupiletite otsimine

Joosep soovib minna soojale maale puhkama. Ta on endale seadnud kindla eelarve ning plaanib esialgu osta üheotsapileti. Lennupileti hind ei tohi ületada Joosepi eelarvet, kuid see võib olla sellega võrdne või madalam.

Allpool on toodud poolik rekursiivne funktsioon `sobiv_pilet`, mis

- võtab argumentideks neli muutujat
  - järjend, mis sisaldab piletite hindasid;
  - muutuja, mille väärtuseks on eelarve suurus;
  - muutujad algus ja lõpp, mis näitavad, millisest järjendi osast piletit otsida tuleks.
- tagastab vastavalt eelarvele ja eeltoodud reeglitele sobiva pileti hinna (kui sobivat ei leidu, tagastab `None`).

Proovi aru saada, kuidas kood peaks töötama. Täiendada tuleks funktsioonis neid ridu, kus kood on asendatud kolme plussmärgiga `"+++"`.

▼ Programm, mida peab täiendama

```
def sobiv_pilet(hinnad, eelarve, algus, lõpp):
    if algus > lõpp:
        if algus > 0:
            return hinnad[+++]  # Tagastus, kui eelarvele vastav pilet leidub.
        else:
            return +++  # Tagastus, kui eelarvele vastavat piletit ei leidu.

    kesk = (+++) // 2   # Keskel asuva elemendi indeks.

    if hinnad[kesk] <= eelarve:
        return sobiv_pilet(+++)     # Keskelt lõpuni otsimine.
    else:
        return sobiv_pilet(+++)     # Keskelt alguseni otsimine.

print(sobiv_pilet([80, 100, 120, 140, 160, 200], 130, 0, len([80, 100, 120, 140, 160,
```

▶ Näide funktsiooni tööst

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1134

„Hargnev rekursioon e puurekursioon: Mündivisete kombinatsioonid“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6B

Hargnev rekursioon e puurekursioon: Mündivisete kombinatsioonid

Kaisa ja Joosep mängivad mündiviske mängu. Iga kord, kui münti visatakse, võib tulla kas kull või kiri. Nad tahavad teada, millised visete kombinatsioonid on võimalikud, kui münti visata **n** korda. Samuti huvitab neid, mitu erinevat järjestust saab tekkida.

Koosta rekursiivne funktsioon `myndiviske_kombinatsioonid`, mis

- võtab argumendiks kaks muutujat
  - arv **n**, mis näitab mündivisete arvu;
  - akumulaator, mille vaikeväärtuseks on tühi sõne ja kuhu saab liita mündivisete kombinatsioonid.
- väljastab ekraanile kõik mündiviske kombinatsioonid (sõned tühikuga eraldatud), mis on **n** viskega võimalikud;
- tagastab erinevate järjestuste koguarvu.

▼ Näide funktsiooni tööst

```
>>> myndiviske_kombinatsioonid(1)
kull
kiri
2
>>> myndiviske_kombinatsioonid(2)
kull kull
kull kiri
kiri kull
kiri kiri
4
>>> myndiviske_kombinatsioonid(3)
kull kull kull
kull kull kiri
kull kiri kull
kull kiri kiri
kiri kull kull
kiri kull kiri
kiri kiri kull
kiri kiri kiri
8
```

▶ Vihje

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1133

„Fibonacci arvud: Trepiastmed“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6B

Fibonacci arvud: Trepiastmed

Oletame, et kortermajas on trepp, millel on **n** astet. Igal sammul võib inimene astuda kas üks või kaks astet korraga. Olenevalt trepiastmete arvust võib tekkida palju erinevaid viise, kuidas trepist üles saada.

Koosta rekursiivne funktsioon `trepiastmed`, mis

- võtab argumendiks trepiastmete arvu;
- arvutab ja tagastab, mitu erinevat viisi on **n** astet läbida, kui astuda tohib kas üks või kaks astet korraga.

▼ Näide funktsiooni tööst

```
>>> trepiastmed(1)
1
>>> trepiastmed(2)
2
>>> trepiastmed(3)
3
>>> trepiastmed(4)
5
>>> trepiastmed(5)
8
>>> trepiastmed(6)
13
```

▼ Vihje

Funktsioonis peaks olema kaks baasjuhtu. Esiteks, kui n == 0, tuleks tagastada 1. Teiseks, kui n < 0, tuleks tagastada 0.

Rekursiooni samm peaks hargnema. Sammu põhimõte on väga sarnane Fibonacci arvude arvutamise rekursiivse väljakutsega.

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1132

„Vastastikune rekursioon: Palgatõus“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6B

Vastastikune rekursioon: Palgatõus

Tööandja pakub töötajatele vahelduva skeemiga palgatõusu. Igal paaritul aastal tõuseb palk 5% ning igal paarisaastal tõuseb palk 40 €.

Koosta funktsioon `protsent`, mis

- võtab argumentideks kaks muutujat
  - muutuja, mis sisaldab töötaja praegust palka (näiteks 1000);
  - muutuja, mis sisaldab aastate arvu (näiteks 4).
- arvutab uue palga, kui tõus on 5%;
- sammul kutsub välja funktsiooni `fikseeritud`;
- baasjuhul tagastab palganumbri `x` aasta pärast.

Koosta funktsioon `fikseeritud`, mis

- võtab argumentideks kaks muutujat
  - muutuja, mis sisaldab töötaja praegust palka (näiteks 1000);
  - muutuja, mis sisaldab aastate arvu (näiteks 4).
- arvutab uue palga, kui tõus on 40 €;
- sammul kutsub välja funktsiooni `protsent`;
- baasjuhul tagastab palganumbri `x` aasta pärast.

Kuna praegu on aasta 2025 ehk paaritu aasta, peaks välja kutsuma funktsiooni `protsent`.

▼ Näide funktsiooni tööst

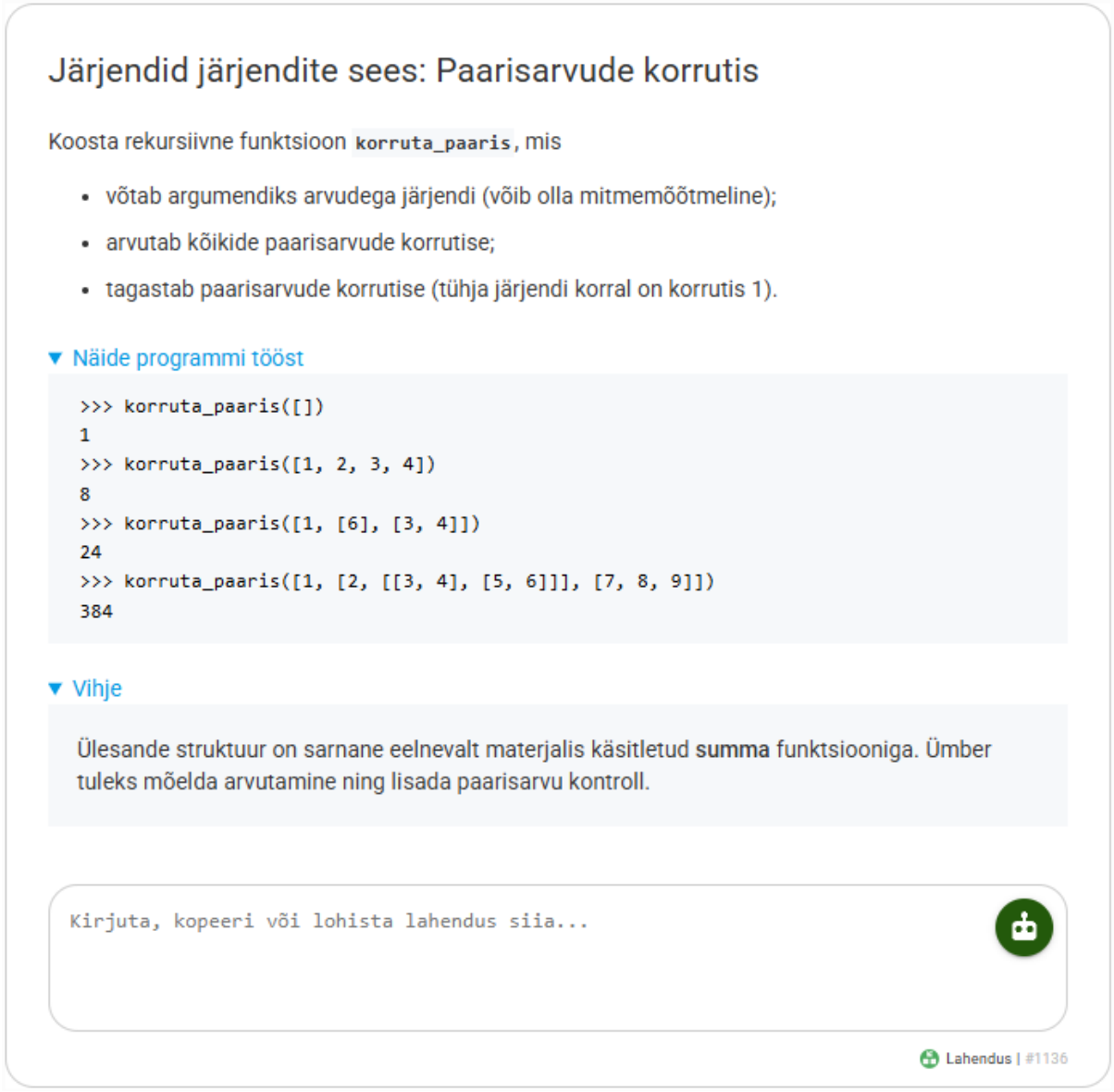
```
>>> protsent(1000, 1)
1050.0
>>> protsent(1000, 2)
1090.0
>>> protsent(1000, 3)
1144.5
>>> protsent(1000, 4)
1184.5
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1135

„Järjendid järjendite sees: Paarisarvude korrutis“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6C



Järjendid järjendite sees: Paarisarvude korrutis

Koosta rekursiivne funktsioon `korruta_paaris`, mis

- võtab argumendiks arvudega järjendi (võib olla mitmemõõtmeline);
- arvutab kõikide paarisarvude korrutise;
- tagastab paarisarvude korrutise (tühja järjendi korral on korrutis 1).

▼ Näide programmi tööst

```
>>> korruta_paaris([])
1
>>> korruta_paaris([1, 2, 3, 4])
8
>>> korruta_paaris([1, [6], [3, 4]])
24
>>> korruta_paaris([1, [2, [[3, 4], [5, 6]]], [7, 8, 9]])
384
```

▼ Vihje

Ülesande struktuur on sarnane eelnevalt materjalis käsitletud **summa** funktsiooniga. Ümber tuleks mõelda arvutamine ning lisada paarisarvu kontroll.

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1136

„Failide ja kaustade kuvamine: Järjend failisüsteemina“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART6C

Failide ja kaustade kuvamine: Järjend failisüsteemina

Allpool on toodud poolik rekursiivne funktsioon `kuva_failid`, mis

- võtab argumendiks kaks muutujat
  - mitmemõõtmelise järjendi, kus on failide nimed;
  - vaikeväärtusega muutuja, mida saab kasutada taande jaoks.
- läbib rekursiivselt argumendiks antud mitmemõõtmelise järjendi kõik elemendid;
- järjendi puhul väljastab ekraanile "Kaust" ja järjekorra numbri;
- sõne puhul väljastab ekraanile "Fail:" ja failinime;
- eraldab erinevate kaustade failid taandega "\t".

Proovi aru saada, kuidas kood peaks töötama. Täiendada tuleks funktsioonis neid ridu, kus kood on asendatud kolme plussmärgiga `"+++"`.

▼ Programm, mida peab täiendama

```
def kuva_failid(struktuur, taane=""):
    kausta_nr = 1
    for element in struktuur:
        if isinstance(element, list):
            print(+++, +++, +++)
            kuva_failid(+++, taane + "\t")
            kausta_nr += +++
        else:
            print(+++, +++, +++)
```

▼ Näide funktsiooni tööst

```
>>> kuva_failid([["Fail11"], ["Fail21"], [["Fail311"], "Fail31", "Fail32", "Fail33"],
 Kaust 1:
        Fail: Fail11
 Kaust 2:
        Fail: Fail21
 Kaust 3:
        Kaust 1:
                Fail: Fail311
        Fail: Fail31
        Fail: Fail32
        Fail: Fail33
 Fail: Fail1
 Fail: Fail2
```

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1137

G. Objektorienteeritud programmeerimine

„Meetodid: Raamatute haldamine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART7

Meetodid: Raamatute haldamine

Martenil on kodus palju vanu raamatuid. Ta soovib neid vanuse alusel riiulitesse sorteerida. Selleks soovib ta koostada programmi, mis oskaks arvutada raamatu vanust.

Koosta klass `Raamat`, mis

- defineerib konstruktoris kaks isendimuutujat **pealkiri** ja **aasta**;
- sisaldab meetodit `vanus`, mis
  - arvutab konkreetse raamatu vanuse (2025 - ilmumisaasta);
  - tagastab raamatu vanuse.

NB! Ülesande automaatkontroll jälgib ainult ekraanile väljastamist, seega tuleks täpselt järgida alumise näite vormistust ning juhendit.

▼ Näide programmi tööst

```
>>> %Run lahendus.py
Tõde ja õigus
99
Läänerindel muutuseta
96
Loomade farm
80
```

Juhend automaatkontrolliks:

- loo kolm raamatut:
  - "Tõde ja õigus", 1926;
  - "Läänerindel muutuseta", 1929;
  - "Loomade farm", 1945;
- väljasta ekraanile iga raamatu kohta pealkiri ja vanus eraldi ridadel (kasuta arvutamisel meetodit `vanus`).

Kirjuta, kopeeri või lohista lahendus siia...

Lahendus | #1155

„Lõpetuseks: Pangakonto haldamine“:

https://courses.cs.ut.ee/2025/progalused2/spring/Main/PART7

Lõpetuseks: Pangakonto haldamine

Anna ja Juhan soovivad oma taskurahal lihtsamini silma peal hoida. Nad soovivad koostada programmi, mis võimaldaks luua kontod, teha sisse- ja väljamakseid ning kuvada kontode hetkeseisu. Aita koostada vastav programm.

Koosta klass `Konto`, mis

- defineerib konstruktoris kaks isendimuutujat **omanik** ja **saldo**;
- sisaldab meetodit `sissemakse`, mis
  - võtab argumendiks summa;
  - lisab isendi kontole raha juurde;
  - väljastab õnnestunud tehingu puhul sõnumi kujul "**Sissemakse õnnestus!**".
- sisaldab meetodit `väljamakse`, mis
  - võtab argumendiks summa;
  - eemaldab isendi kontolt raha;
  - väljastab õnnestunud tehingu puhul sõnumi kujul "**Väljamakse õnnestus!**";
  - väljastab ekraanile "**Kontol pole piisavalt raha!**", kui isendi kontol on tehingu jaoks liiga väike summa.
- sisaldab meetodit `__str__`, mis
  - tagastab isendi informatsiooni kujul "**Konto omanik: x, saldo: x €**".

NB! Ülesande automaatkontroll jälgib ainult ekraanile väljastamist, seega tuleks täpselt järgida alumise näite vormistust ning juhendit.

▶ Näide programmi tööst

Juhend automaatkontrolliks:

- loo kaks kontot - Anna konto saldoga 100 € ja Juhani konto saldoga 50 €;
- tee mõlemale kontole 50 € suurune sissemakse;
- tee Anna kontolt väljamakse summas 30 € ja Juhani kontolt summas 125 €;
- väljasta ekraanile mõlema isendi informatiivne kirjeldus (kui defineerisid `__str__` meetodi, saad lihtsalt väljastada isendimuutuja).

```
Kirjuta, kopeeri või lohista lahendus siia...
```

Lahendus | #1154

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Markus Leivo**

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Enesekontrolliks mõeldud programmeerimisülesannete loomine kursusele „Programmeerimise alused II“**, mille juhendaja on **Heidi Taveter**, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, 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.

Markus Leivo

14.05.2025