TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Andrei Tambovtsev

Interaktiivsete Google Colaboratory töölehtede loomine gümnaasiumi programmeerimise kursusele

Bakalaureusetöö (9 EAP)

Juhendaja:
Tauno Palts, PhD

Tartu 2025

Interaktiivsete Google Colaboratory töölehtede loomine gümnaasiumi programmeerimise kursusele

Lühikokkuvõte:

Selle töö raames koostati interaktiivsed töölehed Gümnaasiumi riikliku õppekava raames loodud programmeerimise kursuse õppematerjalidele. Aluseks võeti olemasolevad tekstitöölehed, mis tavaliselt trükiti paberile ja anti õpilastele kirjalikult täitmiseks. Interaktiivsed töölehed on loodud Google Colaboratory keskkonnas, et võimaldada nii koodi kirjutamist ja käivitamist, kui ka ülesannete teksti vormistust ja piltide kasutamist. Interaktiivsed töölehed sisaldavad samu ülesandeid kui nende paberkandjal versioonid, aga koostati ka lisaülesanded juurde, sest interakatiivse töölehe versioon võtab palju vähem aega kui paberi versioon. Uusi töölehti katsetati gümnaasiumitundides. Pärast esimest uute töölehtede kasutamist jagas gümnaasiumi programmeerimise kursuse õpetaja ka tagasisidet nende nõrkadest ja tugevatest külgedest. Tagasiside alusel töölehti parandati ja antakse soovitused edaspidiseks interaktiivsete töölehtede arendamiseks.

Võtmesõnad: Töölehed, programmeerimise õpetamine, Python

CERCS: S281, S270

Creating Google Colaboratory Interactive Worksheets for the Secondary School Python Programming Course

Abstract:

As part of this thesis, interactive worksheets were developed for a secondary school programming course. These were based on existing worksheets, traditionally printed and completed by hand. The new interactive versions were created in the Google Colaboratory environment, allowing students to both write and execute code directly within the worksheets. Additionally, the environment enabled improved formatting of task descriptions and the integration of visual elements such as images.

The interactive worksheets included all tasks from the original paper versions, with additional exercises added to take advantage of the increased efficiency offered by the digital format. These materials were tested in a secondary school during programming lessons.

Following their use in the classroom, the programming teacher provided feedback to the author regarding the worksheets´ strengths and areas for improvement. Based on this feedback, the worksheets were refined, and successful elements were retained in subsequent iterations. The feedback also informed the conclusions and recommendations presented in this thesis for the future development of interactive programming worksheets.

Keywords: Worksheets, programming teaching, Python

CERCS: S281, S270

Sisukord

Sissejuhatus

Tänapäeval nii Eesti koolides kui ka ülikoolides õpetatakse programmeerimise algkursusi just Pythoni keeles. Programmeerimise õpetamine on IT-valdkonnas väga tähtis, kuna paljude jaoks algab infotehnoloogiaga tutvumine just sellest valdkonnast. Seega on olulised ka õppevahendid, mida kasutatakse õpetamiseks. Üks nendest vahenditest on töölehed ülesannetega, mida õpilased täidavad kirjalikult klassis.

Et muuta esimene kokkupuude programmeerimisega sujuvamaks andes kiiret tagasisidet töölehe vastustele, võeti selle tööeesmärgiks muuta olemasolevad pabertöölehed interaktiivseteks töölehtedeks säilitades väljatöötatud ülesanded ja nende tüübid.

Lõputöö esimeses osas antakse ülevaade teoreetilisest taustast. See peatükk tutvustab informaatika õpetamise ajalooga Eestis, alustades 1959. aastast kuni tänapäevani. Teoreetiline taust on oluline, et näidata, kuidas 60 aastat tagasi õpetati programmeerimist, kirjutades koodi paberile, ning võrrelda seda tänapäevaste töölehtedega. Lisaks antakse ka ülevaade töölehtedest, mis on tänapäeval kasutusel gümnaasiumides, et õpetada programmeerimist Pythoni keeles. Teooria osa lõpus käsitletakse ka interaktiivsete töölehtede kasutuskeskkonda. Samuti selgitatakse, miks valiti just Google Colaboratory keskkond.

Metoodikas kirjeldatakse, kuidas pabertöölehed viiakse interaktiivsesse keskkonda üle: mis sammud on vaja teha, et valmistuda õpilasi uueks keskkonnaks, mis Google Colaboratory sisseehitatud funktsionaalsusi saab kasutada ja mis uuendusi see võimaldab lisada töölehtedesse. Metoodika teises osas kirjeldatakse, kuidas uusi töölehti katsetati gümnaasiumis ning kuidas toimus töölehtede tagasisidestamine gümnaasiumi programmeerimise kursuse õpetaja poolt.

Seejärel on lõputöös tulemused ja nende analüüs. Esiteks tulemuste osas kirjeldatakse lõputöö raames koostatud töölehti. Nende tugevaid ja nõrku külgi uuriti tagasiside alusel, mida andis töölehti kasutatav õpetaja töölehtede autorile. Tagasiside iga katsetatud töölehe kohta on ka kirjeldatud tulemuste järgmises osas, koos järeldustega sellest, mida oli vaja interaktiivsetes töölehtedes parandada. Lisaks annab peatükk infot sellest, kuidas töölehti kasutada ja kuidas on töölehti võimalik edasi arendada.

1. Teoreetiline taust

1.1 Programmeerimise õpetamise ajalugu

Tartu Ülikooli matemaatika-informaatikateaduskonna ajaloo raamatu järgi hakati Eestis arvutiteadust arendama 1959. aastal tänu Ülo Kaasikule [1]. Sama allikas viidab, et Tartu Riikliku Ülikooli arvutuskeskuse avamisega hakati informaatikaga tegelema professionaalsel tasemel, kuid seda ei õpetatud veel koolides või kõrgkoolides.

Tartu Ülikooli haridusteaduste instituudi uurimise raport väidab, et esimest korda hakati infotehnoloogiat õpetama 1962. aastal [2]. See allikas sisaldab ka informatsiooni, et Ülo Kaasik õpetas masinkoodis programmeerimist esimeses matemaatika eriklassis Tartu 1. Keskkoolis, praeguses Hugo Treffneri Gümnaasiumis. Raportis ka öeldud, et pikka aega toimusid programmeerimistunnid vaid vähestes koolides ja eriklassides, kus õppis väike arv õpilasi. Allikas toob esile ka selle, et arvutite puudujäägi tõttu õpetati programmeerimist paberil või tahvlil ja suurem rõhk oli teoorial, kuna kätte saada arvutit, et praktiliselt programmeerida, oli väga raske.

„Matemaatika-Informaatikateaduskond 40" raamatus kirjutatakse ka sellest, et 20 aastat pärast Tartu Riikliku Ülikooli arvutuskeskuse tekkimist, aastal 1979 tehti järgmine suur samm programmeerimise õpetamiseks – avati programmeerimise kateeder [1]. See tähendas, et informaatika muutus iseseisvaks akadeemiliseks distsipliiniks. Allikas lisab, et arvuteid oli veel vähe ja nende võimekus oli madal, selle tõttu oli esimesel õppeaastal väga palju teoreetilisi aineid, aga programmeerimist õpetati nii palju, kui oli võimalik olemasoleva tehnikaga. Matemaatika-informaatika teaduskonna ajaloo raamatus on kirjas, et – „...ressursipuudusel sai iga üliõpilane kontaktaega vaid umbes 40 minutit nädalas". Programmeerimise õpetamise jaoks olid mõeldud spetsiaalsed laboratoorsed tööd. Nende tööde käigus kirjutati programmi Assembler keeles kuni 1980. aastate lõpuni.

Raport „IT oskuste arendamine Eesti koolides" kirjeldab sama ajaperioodi koolides [2]. 1980. aastatel said koolid arvuteid juurde, sest nende kasutusala suurenes. Raportis kirjutatakse ka, et arvutid olid kasulikud mitte ainult programmeerimise õpetamises, vaid ka tekstitöötluses ja tabelitöötluses. Seda kinnitab ka matemaatika-informaatika teaduskonna ajalugu. Isegi kehalise kasvatuse õppejõudude kvalifikatsiooni tõstmisel kasutasid kuulajad programmeerimise kateedri varustust ja õppisid seal andmebaase ja tekstitöötlust. Informaatika aluseid õpetati ka teistes teaduskondades, kuigi mahud erinesid [1]. Allikas toob esile ka selle,

et kõige rohkem tudengeid oli loodus- ja täppisteaduste valdkonnast: keemia-, füüsika-, bioloogia-, geograafiateaduskonnast. Umbes sama palju ka majandusteaduskonnast. Isegi õigus- ja ajalooteaduskonna tudengid said informaatikast ülevaate. 1980. aastate lõpus kogus arvutiteadus veelgi rohkem populaarsust. Kõige kasutatavamaks programmeerimiskeeleks sai Assembleri asemel Pascal [1]. Sellega õpetati programmeerimist rohkem kõrgema taseme keeles. Assembleris programmi kirjutamine nõudis palju aega, sest tuli kasutada rohkelt lihtsaid käske, et programm algusest lõpuni korrektselt üles ehitada. Need operatsioonid on hästi arusaadavad arvutile. Pascal võimaldas rohkem keskenduda tulemusele, muretsemata programmi arhitektuursete detailide pärast. Pascal keeles olid kasutusel lihtsasti loetavad ja inimestele arusaadavad käsud.

Raporti andmete alusel oli kooliõpilasi, keda huvitas informaatika nii palju, et juba 1988. aastal toimus esimene informaatikaolümpiaad, mille raames õpilased lahendasid programmeerimisülesandeid [2]. Edasi väidetakse, et 1989. aastal programmeerimise kateeder sai endale rohkem arvuteid. Anne Villems sai suure vaevaga kateedri jaoks Yamaha arvuteid, ületades bürokraatlike raskusi. Näiteks majandusteaduskonnas oli võimalik kasutada vaid kolme arvutit ühe kuni 15 üliõpilasega õpperühma kohta [1].

Ajaloo raamatu järgi organiseeriti 1993. aastal programmeerimise kateeder ümber ja selle baasil kujunes arvutiteaduse instituut [1]. Allika järgi selgub ka, et selle instituudi lisategevuseks sai koolituste läbiviimine, mille raames informaatika-alased teadmised olid jagatud ka koolide õpetajatega. Näiteks täiend- ja tasemekoolitus TÜ avatud ülikooli raames 1996. aastal, aastatel 1996 - 2006 informaatikaõpetajate kaheaastane kutsekursus, õpitarkvara kursused erinevate õppeainete õpetajatele ja muud. Kuid informaatika ja programmeerimine ei saanud riikliku õppekava kohustuslike ainete nimekirja. Riigiteataja veebilehelt võib näha, et võrreldes vanimat kättesaadavat redaktsiooni (17.01.2011) ja eelviimast redaktsiooni (26.04.2021), ei ole informaatika ja programmeerimisega seotud ained muutunud [3]. Riigiteataja informatsiooni alusel kõik muutused, mis on selle valdkonnaga seotud, on ainult valikkursused loodusainete rühmas. Riikliku õppekavasse lisati Informaatikat valikainena ainult viimases redaktsioonis (11.03.2023).

Paljud koolid annavad praegu võimaluse õppida informaatikat ja programmeerimist. „IT oskuste arendamine Eesti koolides” raporti järgi leidub informaatikaga seotud aineid nii põhikoolides kui ka gümnaasiumides [2]. Sama allikas veel näitab, et suurem roll IT alase õppimisega on huviringidel, mitte tavalistel ainetel. Antud raporti järgi 2017. aastal oli 68% Eesti koolide huviringid seotud informaatikaga ja ainult 38% huviringidest olid

programmeerimisega seotud. Informaatikat ja programmeerimist õpetatakse mitmesuguste materjalidega: loengud, õpikud, praktilised kodutööd, automaatkontrolliga testid ja töölehed. Huviringid ja programmeerimise ained koolides ja ülikoolides kasutavad tihti töölehti tunnitööna. Nendes töölehtedes on tavaliselt erinevad praktilised ja teoreetilised ülesanded, mis on ühe teemaga seotud.

1.2 Programmeerimise töölehed

Gümnaasiumi kursuse Programmeerimine õpetamiseks on Tartu Ülikool loonud materjalid [4]. Materjalid sisaldavad ka õpetajamaterjale[1], kus on kaasas ka väljaprinditavad töölehed, mida saab programmeerimise õpetamisel kasutada. Töölehed trükitakse paberile ja neid antakse õpilastele tunnis lahendamiseks. Töölehed koosnevad nii teoreetilistest küsimustest (vt joonis 1) kui ka praktilistest ülesannetest (vt joonis 2).

Tsükkel. Paaristöö ülesanded

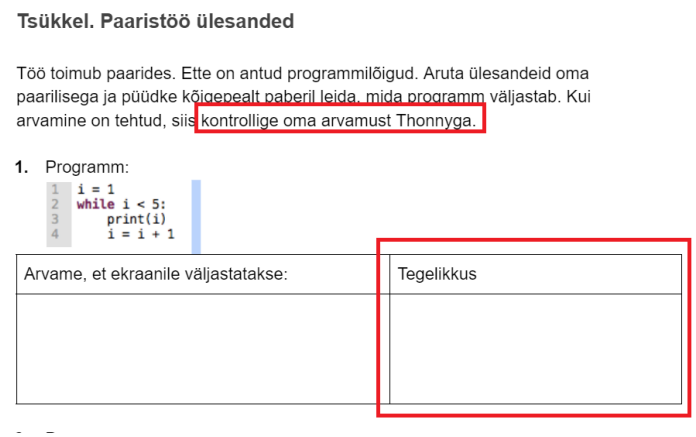
Töö toimub paarides. Ette on antud programmilõigud. Aruta ülesandeid oma paarilisega ja püüdke kõigepealt paberil leida, mida programm väljastab. Kui arvamine on tehtud, siis kontrollige oma arvamust Thonnyga.

1. Programmi:

```
i = 1
while i < 5:
    print(i)
    i = i + 1
```

Arvame, et ekraanile väljastatakse:	Tegelikkus

Joonis 1. Tööleht. Teoreetiline küsimus [5].



Tsükkel. Paaristöö ülesanded

Töö toimub paarides. Ette on antud programmilõigud. Aruta ülesandeid oma paarilisega ja püüdke kõigepealt paberil leida, mida programm väljastab. Kui arvamine on tehtud, siis kontrollige oma arvamust Thonnyga.

1. Programmi:

```
i = 1
while i < 5:
    print(i)
    i = i + 1
```

Arvame, et ekraanile väljastatakse:	Tegelikkus

Joonis 2. Tööleht. Praktiline ülesanne [5].

[1] https://web.htk.tlu.ee/digitaru/programmeerimine/chapter/opetajamaterjal/

Selles töös kasutatud töölehed on praegu kättesaadavad mitmes koolis ja neid kasutatakse aktiivselt õppetöös. Jagatud kaust [5] sisaldab materjale kaheksaks õppenädalaks ja kontrolltööks. Ainult viie nädala jaoks on ette nähtud töölehed, need katavad teemasid nagu „Muutujad ja andmetüübid", „Tingimuslause", „Tsükkel", „Järjend" ja „Funktsioon". Teemadele „Graafika" ja „Andmevahetus" ning viimasele kordamise nädalale töölehti ei ole.

Praeguse prinditava töölehega on mitu ebamugavusi, millest suuremad on:

1. Õpetaja peab enne tundi välja printima suure hulga töölehti, mis raiskab palju aega ja paberit.
2. Õpilased kulutavad palju aega sellele, et koodi maha kirjutada ja seda siis arvutis jooksutada.

Vaadates tagasi ajalukku, ei erine paberilt programmeerimise õppimine 2024. aastal väga 1959. aasta olukorrast. Kuigi koolides on nüüd arvutid ja internetipõhised ressursid testide ning kodutööde jaoks, on töölehtede ülesandeformat endiselt sarnane 60 aasta tagusega.

Samas on ka praegu kasutusel olevatel töölehtedel positiivseid külgi. Näiteks, nagu joonis 1 ja joonis 2 näitavad, on töölehe pealkirjas välja toodud, et see on paaristöö ehk mõeldud kahele õpilasele korraga. Paaristöös programmeerimise ülesannete lahendamine paneb vastutuse suuremale mõttetööle, sest vahepeal tuleb oma seisukohti selgitada ja nende üle argumenteerida. Selline õppetöö vorm valmistab ette ka ülikoolis või töö juures paarisprogrammeerimiseks ja rühmatööks.

Pabertöölehtede kasutamise osas saab tuua nii positiivseid kui ka negatiivseid argumente interaktiivsuse teemal. Programmeerimise töölehtedes on tihti kasutusel ülesande tüüp „kontrolli" mis asub „Tea ja mõista" kategooria all [6]. „Kontrolli" tüüpi ülesanne (vt joonis 1) on mõeldud lahendamiseks ilma arvuti või käivitatava programmita [6]. Seega kood, mis on välja trükitud paberile ja mida ei saa käivitada, sobib väga hästi selleks ülesandeks. Õpilane ei saa vastata küsimusele „Mis väljastatakse ekraanile?" enne koodi lugemist. Nagu metoodika osas saab mainitud (vt ptk 2), kui tegu on interaktiivsete töölehtedega, siis vastust saada on lihtsam, sest koodi saab kopeerida ja jooksutada. Et leida vastus ei ole hädavajalik koodi lugeda. See tähendab, et ülesandetüübi „kontrolli" puhul üks pabertöölehe negatiivne omadus tegelikult aitab säilitada ülesande põhimõtet. See, et koodi paberilt ei saa kohe jooksutada, suunab õpilasi lugema koodi ja mõtlema, mida see kood teeb. Interaktiivsete töölehtedega on olemas risk, et õpilased kohe käivitavad koodi ja leiavad vastuse selle asemel, et hoolikalt analüüsida programmi.

Tartu ülikooli Courses veebilehel „Programmeerimise ülesannete tüübid” [6] on välja toodud 11 ülesannete tüüpi, mis on jagatud kolme kategooriasse, milleks on:

- Tea ja mõista
    - Tuvasta (põhikonstruktsioone teadmine)
    - Kontrolli (programmi tulemuse kontroll)
    - Selgita (programmi analüüs)
- Rakenda ja analüüsi
    - Seosta (väikse koodi osa analüüs suures programmis)
    - Teosta (kirjeldatud programmi kirjutamine)
    - Kohanda (olemasoleva lahenduse muutmine)
    - Transleeri (sama tulemuse saavutamine teises programmeerimis keeles)
- Sünteesi ja hinda
    - Silu (vigade parandamine)
    - Rakenda (suurema ülesande osana antud koodi kasutamine, integreerimine)
    - Modelleeri ja projekteeri (struktuuri või abstraktse lahenduse koostamine)
    - Uuenda (antud koodi muutmine teatud standardile).

Kuna tegu on kursusega, mis on mõeldud õpilastele, kes ei ole varem programmeerinud, siis „Modelleeri ja projekteeri”, „Uuenda” ning „Kohanda” ülesanded ei ole töölehtedes kasutusel. Kursus keskendub ainult programmeerimiskeelele Python, seega „Transleeri” ülesandeid ei ole kasutatud. Olemasolevates pabertöölehtedes on kasutusel järgmised ülesannete tüübid: „Kontrolli” (vt joonis 3), „Selgita” (vt joonis 4), „Teosta” (vt joonis 5), „Silu” (vt joonis 6) ja “Rakenda” (vt joonis 7).

4. Programm:

```
k = 0
while k < 3:
    print("Tere!")
    k += 1
    print(k)
```

Arvame, et ekraanile väljastatakse:	Tegelikkus

Joonis 3. Tööleht. „Kontrolli” ülesanne [5]

Ülesanne 5.

Programm on selline:

```
arv1 = int(input("Maksimum: "))
arv2 = int(input("Tegelik: "))
arv3 = round(arv2 / arv1 * 100)
if arv3 >= 90:
    print("viis")
else:
    if arv3 < 90 and arv3 >= 75:
        print("neli")
    else:
        if arv3 < 75 and arv3 >= 50:
            print("kolm")
        else:
            print("uuesti!")
```

Programmi sisu on:

.....

.....

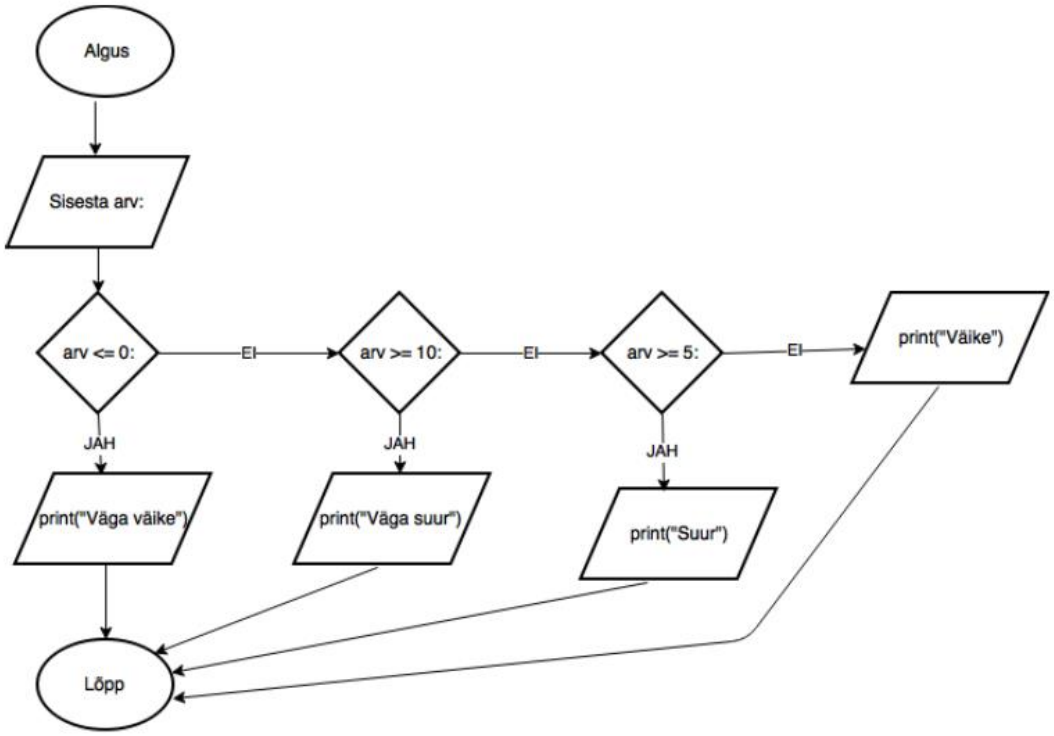
Programmi read	**Mida programmi rida teeb?**
arv1 = int(input("Maksimum: "))	

Joonis 4. Tööleht. „Selgita” ülesanne [5].

Ülesanne 6.

Koostage plokkskeemi järgi programm. Mida see programm teeb? Proovige programmi Thonnyga.

Programmi plokkskeem on selline:



Kirjutage programm siia:

Joonis 5. Tööleht. „Teosta" ülesanne [5].

1. Mati tahtis kirjutada programmi, mis arvutaks ristküliku pindala. Kasutajalt küsitakse ristküliku külgede pikkused (täisarvud). Kuid programm näitab veateadet. Aita Matil viga parandada.

Mati programm	Kus on viga?

`a = input("Sisesta ristküliku pikkus: ")`
`b = input("Sisesta ristküliku laius: ")`
`print("Ristküliku pindala on: " + str(a*b))`	

Joonis 6. Tööleht. „Silu" ülesanne [5].

Mida kirjutada joontele, et ekraanile väljastataks arv 15?
 	
Mida see programm teeb?
	`korrutis = ___`
	`for i in range(1,6,2):`
	`  korrutis = _____*i`
	`print(_____)`

Joonis 7. Tööleht. „Rakenda" ülesanne [5].

Suurem osa ülesannetest on ikka „Kontrolli" tüüpi, need esinevad igal töölehel. „Tea ja mõista" kategooriast ülesandeid on pabertöölehtedes väga palju. Esimeseks põhjuseks on see, et töölehed on praktika ja teadmiste lisakontroll. Oluline on silmas pidada seda, et töölehed on kasutusel koos õpikuga ja koduste töödega. Teiseks põhjuseks on lahenduskeskkonna kirjeldus „Programmeerimise ülesannete tüübid" veebilehel, kus öeldakse, et kategooria „Tea ja mõista" ülesanded ei nõua arvutit lahendamiseks [6].

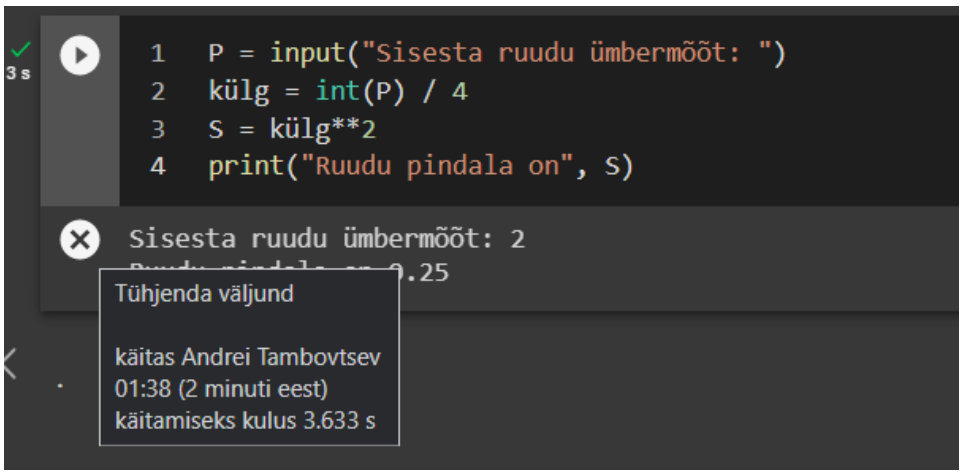
1.3 Google Colaboratory

2017. aastal tutvustas Google kasutajatele uut tasuta Google'isse sisseehitatud tööriista, mille nimeks sai Google Colaboratory või lühendatult Colab. Dave Gershgorn kirjutas peale seda artikli nimega „Nerds rejoice: Google just released its internal tool to collaborate on AI", kus jagas infot Colabi kohta [7]. See on ehitatud Jupyter Project koodi baasil, mis sai avalikuks ja kõigile kättesaadavaks aastal 2014. Colabi kasutamine on mugavam sellepärast, et Google Colabi failid sarnaselt ülejäänute Google Docsi failidega on salvestatud pilves, on lihtsasti jagatavad teiste inimestega ja neid saavad mitu kasutajat muuta samaaegselt. Colab oli mõeldud selleks, et väga lihtsalt ja efektiivselt tegeleda tehisintellekti ja andmeteaduse arendamisega.

Colab, samuti nagu Jupyter, tegeleb failidega laiendusega .ipynb (ingl *interactive python notebook*). Need failid võimaldavad sama dokumendi sees kirjutada koodi, jooksutada koodi, näha tulemusi, lisada tekstiplokke, pilte ja isegi videoid. Koodilahtrid saavad ka teiste käskudega hakkama peale tavalise Python koodi käivitamise. Käsud, mis algavad hüüumärgiga, võimaldavad arvutiga koostoimet, näiteks failide salvestamine, allalaadimine, GitHub repo importimine jne. Suurte andmekogumite töötlemiseks või keeruliste masinõppe mudelite treenimiseks võimaldab Google Colab ka graafikakaardi kasutust, et arvutuslikud protsessid võtaksid vähem aega.

Colab on väga võimekas instrument informaatikute maailmas. Valdavalt kasutatakse seda tehisintellekti ja andmeteaduse valdkondades. Google Marketplace tarkvara kauplus ütleb, et

Colaboratory on andmeanalüüsi ja masinõppe tööriist [8]. Aga tänapäeval on see ka Tartu Ülikoolis õppeainetes suur osa. Sellised ained nagu LTAT.01.003 „Tehisintellekt", LTAT.02.002 „Sissejuhatus andmeteadusesse" ja LTAT.01.002 „Keeletehnoloogia" kasutavad Colab'i kõikide kodutööde ja praktikumide jaoks. Lisaks kõikidele võimalustele on ka väike boonus nendele, kes töötavad rühmana ühe failiga või õpetajatele, kes kontrollivad täidetud dokumente - koodi jooksutamise ajaloos on näha (vt joonis 8), kes ja millal viimasena käivitas koodi lahtri.



Joonis 8. Colab. Käivitamise ajalugu [9].

2. Metoodika

Töö eesmärk oli algseid gümnaasiumi programmeerimis kursuse prinditavad töölehed viia ümber .ipnb laiendusega failidesse kasutades Google Colaboratory keskkonda (vt ptk 2.1). Metoodika osa kirjeldab Colab funktsionaalsust, mida kasutati et säilitada ülesannete põhimõtted ja lisada interaktiivsust.

Töö käigus saadud töölehti katsetati programmeerimise õpetamisel ühes Tartu linna gümnaasiumis (vt ptk 2.2). Katsetamise peatükis kirjeldatud kursuse sisu ja kuidas peale interaktiivsete töölehtede kasutamist jagati tagasiside uute töölehtede autoriga. Katsetamise peatükk sisaldab ka töölehtede parandamise kirjeldust tagasiside alusel.

Bakalaureusetöö tekst parandati grammatiliselt, kasutades suurt keelemudelit ChatGPT-4o.

2.1 Olemasolevate töölehtede täiendamine

Bakalaureusetöö raames tehtud töölehed keskkonnas Google Colab on suuremal määral samade ülesannetega nagu siiamaani kasutatud pabertöölehed. Siinse töö eesmärgi täitmiseks koostati 6 interaktiivset töölehte, üks iga pabertöölehe asemel ja veel viimasele 8. nädalale tehti uus tööleht kordamismaterjaliga.

I. Enne interaktiivsete töölehtede kasutamist oli vaja tutvuda Google Colaboratory võimalustega nii õpilastel kui ka õpetajatel. Selle eesmärgi täitmiseks lõputöö raames tehti .ipynb laiendiga fail nimega „Colab juhend õpilastele.ipynb" [9]. Faili sees on juhised ja seletused keskkonna võimalustest. Juhised teksti abil seletavad, kuidas lisada ja muuta teksti ja koodi Google Colabis. Teksti juurde on lisatud pildid, et keskkonna kasutamine oleks veelgi lihtsamini arusaadav. Juhend on meelega tehtud just .ipynb failis, sest niimoodi juhise lugemise ajal oleks võimalik kohe ka proovida kirjeldatud funktsionaalsust. Nimetatud juhend koos töölehtedega on lisatud ka selle töö lisadesse (vt lisa II Colab juhend õpilasteleColab juhend õpilastele).

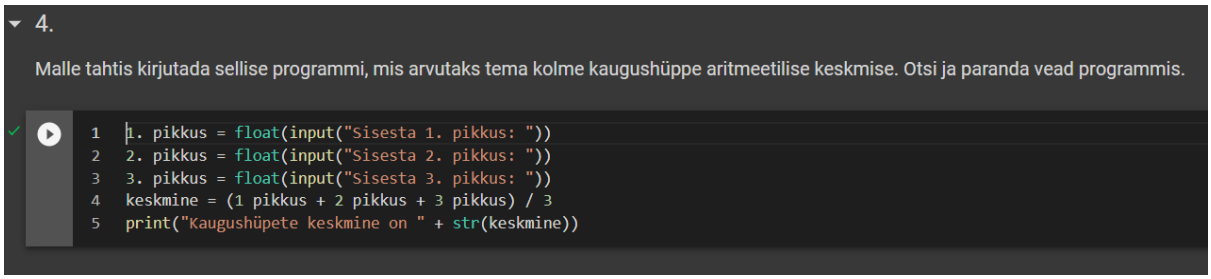
Esimene tööleht oli prototüüp formaadi katsetamiseks. Kõik järgmised töölehed olid koostatud ainult peale eelmise töölehe kohta tagasiside saamist. Esimese töölehe valmistamiseks oli

rohkem aega kui teiste töölehtede jaoks, sellepärast kasutati seal mitmeid tehnilisi uuendusi, aga tööleht sisaldas samu ülesandeid (vt joonis 9 pabertöölehe ülesanne ja joonis 10 interaktiivne loodud tööleht).

4. Malle tahtis kirjutada sellise programmi, mis arvutaks tema kolme kaugushüppe aritmeetilise keskmise. Otsi ja paranda vead programmis.

Malle programm	Kus on viga?
1. pikkus = float(input("Sisesta 1. pikkus: ")) 2. pikkus = float(input("Sisesta 2. pikkus: ")) 3. pikkus = float(input("Sisesta 3. pikkus: ")) keskmine = (1 pikkus + 2 pikkus + 3 pikkus) / 3 print("Kaugushüpete keskmine on " + str(keskmine))	

Joonis 9. Esimene tööleht. Paberlehe ülesanne [5].



Joonis 10. Esimene tööleht. Interaktiivne ülesanne [5].

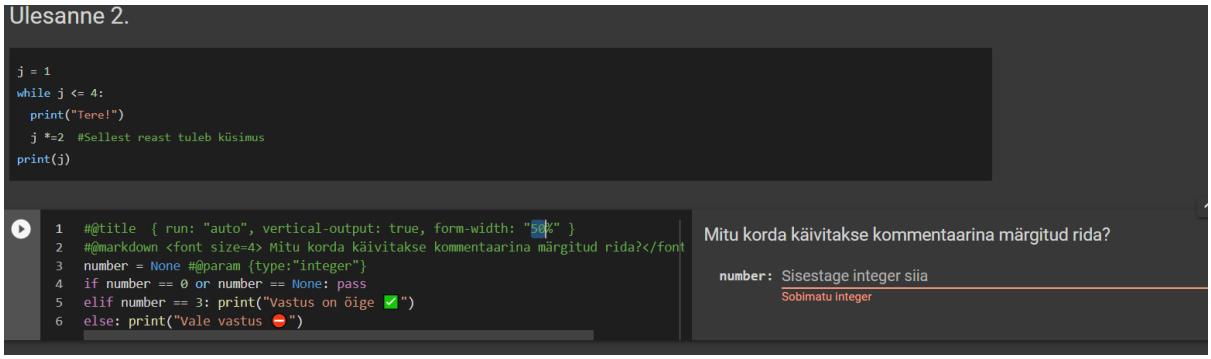
Küsimus jäi samaks, aga nüüd kasutades Colabi koodiploki, on võimalik jooksvalt programmi muuta ja katsetada, nii kaua, kuni see annab oodatud tulemuse, ehk programmi vead saavad parandatud.

Esimeses interaktiivses töölehes prooviti ka automaatselt vastust kontrollida ja õpilasele tagasisidet anda, ehk näidata vastuse all kas see on õige või vale vastus. Selleks esimese ülesande jaoks kasutati Google Colabi funktsionaalsust, mis võimaldab töölehe sisse ehitada interaktiivse küsimustiku. Sellised interaktiivsed küsimustikud töötavad sarnaselt Google Formsi küsimustikega. Colabi küsimustiku loomisel on vaja valida, kuidas sisestatakse vastus ja mis muutuja tüüpi tuleb vastus (vt joonis 11).

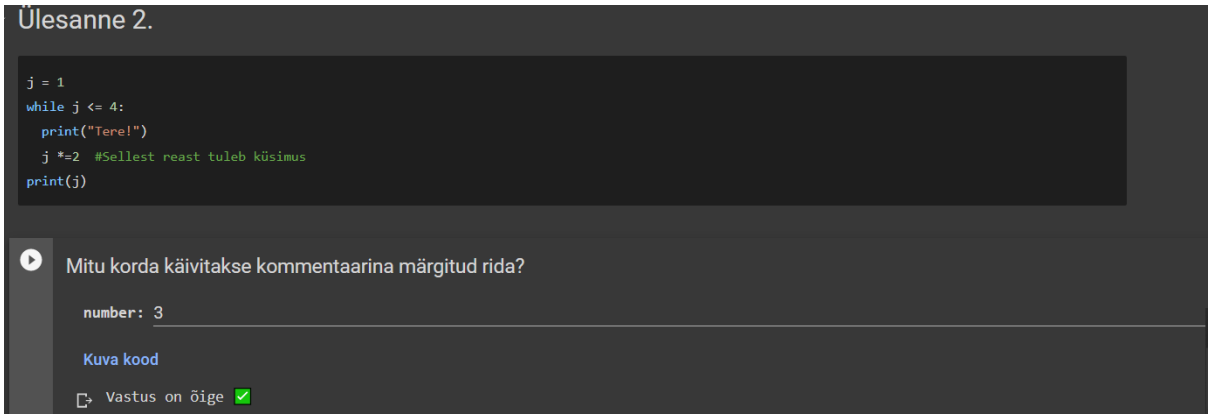
Joonis 11. Google Colaboratory. Interaktiivse vormi koostamine [9].

Google Colab pakub 3 vastuse sisestamise võimalust: valik nimekirjast (ingl *dropdown*), teksti väli, et kirjutada vaba vastus (ingl *input*) ja teatud intervallis liikuv liugur (ingl *slider*). Lihtteksti välja (ingl *markdown*) kasutatakse küsimuse või kommentaari, ehk mitte muutuva teksti sisestamiseks, seda ei saa kasutada vastuse sisestamiseks. Selle töö raames kasutati Lihtteksti välja küsimuste kirjutamiseks ja vastuste jaoks teksti välja. Vastus salvestatakse muutujasse, mille nime saab ka valida. Muutuja tüübiks saab valida üht kuuest tüübist. Selle bakalaureusetöö raames on kasutatud ainult sõne-tüüpi (ingl *string*) ja täisarvulisi (ingl *integer*) muutujaid.

Lõpuks on võimalik koostada vorm, mis esitab küsimuse ja omistab muutujasse vastuse. Seda on võimalik kasutada näiteks kiire tagasiside andmiseks, kui kirjutada vormi juurde väike automaatkontroll (vt joonis 12). Selleks, et õpilased ei saaks näha, mida kontrollitakse automaatkontrollis, on võimalik kood vormi alla peita. Selleks on vaja muuta *form-width* parameetrit, et ta oleks 100% (vt joonis 13).



Joonis 12. Interaktiivne tööleht. Vorm koos koodiga [9].



Joonis 13. Esimene tööleht. Vorm peidetud koodiga ja töötava automaatkontrolliga [9].

Ka sel juhul on võimalik vormi sätetest leida nupp, mis kuvab vormi koodi. Et õpilased ei näeks õiget vastust, tuleb küsimus nutikalt koostada. Õpilase sisendi ja õige vastuse otsese võrdlemise asemel võib õige vastuse salvestada muutujasse eraldi koodiplokis. Seejärel saab vormi koodis sisendit võrrelda selle muutujaga. Seda prooviti saavutada esimeses töölehes. See tööleht on ainuke Colabi tööleht, mis kasutab niinimetatud „Küsimuste tehase" koodi. See on suur koodiväli, mis asub töölehe alguses ja vaikimisi see väli on ahendatud ehk ei ole kohe nähtav. Malli alusel koostati küsimused kasutades juhunumbrite generaatorit. Need küsimused on vastavuses esimese töölehe esimesele ülesannele (vt joonis 14 ja 15).

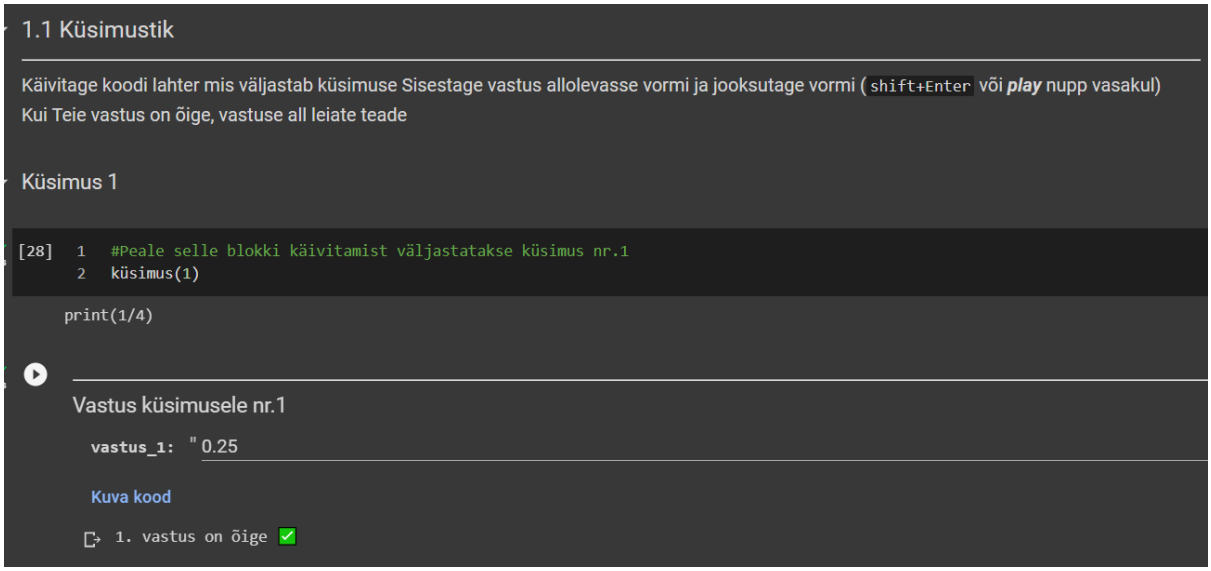
Töö toimub paarides. Ette on antud programmid, mis ei tööta ja vajavad parandamist. Arutage paarilisega ja püüdke paberil leida programmis olevad vead. Proovige oma lahendusi Thonnyga.

Kirjutage juurde vastus:

```
print(type(5.0))
print(type(5,0))
print(type("5"))
print(5/2)
print(5//2)
```

```
print(5+2)
print(int(1.7)+int(1.8))
print(str(1.7)+str(1.8))
print(int(5.999))
```

Joonis 14. Esimene tööleht. Esimene ülesanne [5].



Joonis 15. Esimene interaktiivne tööleht. Esimene ülesanne [9].

Joonise 15. mall on „print(x/y)". Selleks, et vormi koodis ei oleks näha õiget vastust numbrina oli tehtud selline algorütm:

1. „Küsimuste tehas" esiteks genereerib kaks juhuslikku täisarvu.
2. Siis kontrollib, et nende üksteisega jagamise tulemus ei ole lõpmatu ja et seal ei ole rohkem kui 2 komakohta.
3. Kui kõik need tingimused on täidetud, siis programm salvestab küsimuse küsimuste järjendisse.
4. Nüüd Google Colabis jookseb kood, mis salvestab vastuste järjendisse õige vastuse.

Niimoodi isegi, kui õpilane avab vormi taha peidetud koodi, ta näeb ainult koodi järgmisel kujul:

if (vastus(1) == vastus_1):

Siin vastus(1) on õige vastus salvestatud siis, kui küsimuse loodi „küsimuste tehases". vastus_1 on muutuja, mida sisestab õpilane. Kui need väärtused on omavahel võrdsed, siis õpilane näeb tulemuseks „1. vastus on õige". Muul juhul väljastatakse sõnum, et vastus on vale. Küsimuste järjekord on ka igal jooksutamisel juhuslik, et ei oleks liiga lihtne saada klassikaaslaselt valmis vastust.

Interaktiivsed vormid on kasutusel 1. ja 3. töölehel. Kuid „küsimuste tehas" 3. töölehel ei ole kasutusel. Ülejäänud originaalsed töölehed olid rohkem suunatud koodi kirjutamisele ja analüüsi poole, ehk ei sisaldanud väga palju teoreetilisi küsimusi, et tekitada interaktiivse vormi nende asemel. Nendest kahest töölehest, kus on interaktiivsed vormid kasutusel ainult

esimesel, oli rakendatud „küsimuste tehase” funktsionaalsus. „Küsimuste tehase” kasutamine ja küsimuste juhuslik järjekord jäid ainult esimeses töölehes. Katsed gümnaasiumis näitasid, et see tekitab väga palju segadust õpilaste seas.

2.2 Katsetamine gümnaasiumis ja parandused

Kõiki bakalaureusetöö raames tehtud töölehti katsetati 2022. aasta sügisel ühes Tartu gümnaasiumi sissejuhataval programmeerimise kursusel. Kursus kestis 9 nädalat ja kursusel osales 36 õpilast. Iga nädal oli ühele teemale pühendatud. Teemad [5] olid järgmised:

1. Muutujad ja andmetüübid
2. Tingimuslause
3. Tsükkel
4. Sõned ja graafika
5. Järjend
6. Funktsioon
7. Andmevahetus. Lihtne kasutajaliides
8. Kordamine
9. Kontrolltöö

Iga teema kohta toimus kaks tundi kestvusega 75 minutit. Esimeses tunnis lahendati ülesandeid ja teise tunni ajal kasutati töölehti. Nendest üheksast teemast ei olnud töölehti teemadele: „Sõned ja graafika”, „Andmevahetus. Lihtne kasutajaliides”, „Kordamine” ja Kontrolltöö.

Pärast iga töölehe kasutamist andis kooli õpetaja tagasisidet e-kirja teel vabas vormis töölehe tugevatest ja nõrkadest külgedest. Tagasiside sisaldas ajalise kulukuse hinnangut, tehnilisi soovitusi ja ka õigekeelsuse või vormistamise parandusi.

Esimese töölehe tagasiside alusel otsustati, et töölehe lõppu tuleb panna samale teemale lisaülesanded, mis oleksid natuke keerulisemad. Iga järgnev tööleht sisaldas ka 1 või 2 lisaülesannet.

3. Tulemused

3.1 Koostatud interaktiivsed töölehed

Selle töö raames koostati 7 .ipynb formaadis töölehte: üks Google Colaboratory kasutamise juhend, 5 töölehte, mis asendavad olemasolevaid töölehti ja 1 uus tööleht kordamisülesannetega. Kokku lisati töölehtede juurde 10 ülesannet: 6 lisaülesannet olemasolevatele töölehtedele ja 4 uut ülesannet uuele töölehele. Töölehti teemadel „Muutujad ja andmetüübid", „Tingimuslause", „Tsükkel", „Järjend" ja „Funktsioon" katsetati gümnaasiumis aine Programmeerimine raames.

Peamised muudatused võrreldes pabertöölehtedega seisnevad selles, et:

1. Koodi saab jooksutada nüüd töölehe sees.
2. Töölehti saab õpilastele jagada kasutades linki Google Drive'is.
3. Õpetajad saavad hinnata töölehte ja anda tagasisidet kommentaaridena töölehe juurde, nii nagu see on võimalik teha Google'i Docsi failidega, mida õpilased saavad lugeda nendele sobival ajal.
4. Töölehe jaoks kasutatud aeg on väiksem, kuna õpilased säästavad aega koodi mahakirjutamise asemel.
5. Juurde tekkinud aeg võib kasutada lisaülesannete lahendamiseks.
6. Töölehed sisaldavad testküsimusi koos automaatkontrolliga.
7. Mõned küsimused on juhusliku arvu genereerimise kasutamisega, et õpilastel oleks raskem üksteise lahendustest maha kirjutada.
8. Töölehti saab jätta ka koduseks tööks (nt puudujatele).

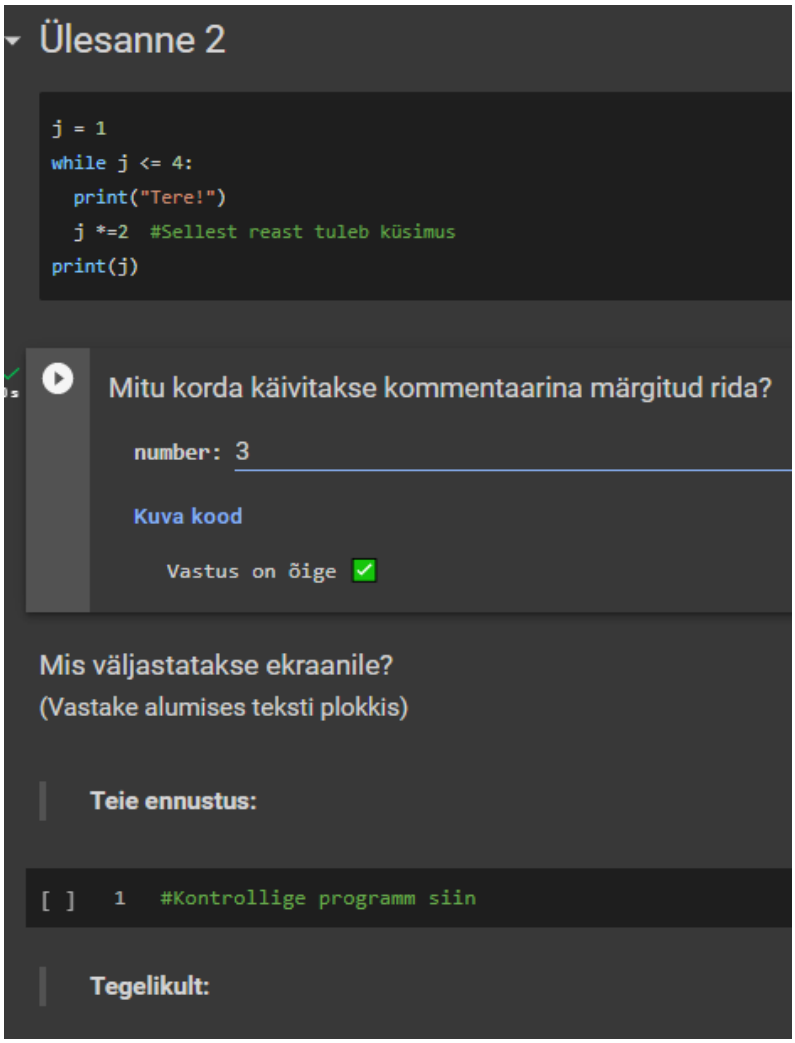
Ülesannete tüübid püüti jätta samaks, sest need on mitmekesised ja töötasid varasemalt hästi. Olemasolevate ülesannete juurde lisati automaatkontrollid ja uued küsimused (vt joonis 16 ja 17).

2. Programm:

```
j = 1
while j <= 4:
    print("Tere!")
    j *=2
print(j)
```

Arvame, et ekraanile väljastatakse:	Tegelikkus

Joonis 16. Kolmas pabertööleht. „Kontrolli" ülesanne tsükli teemal [5].

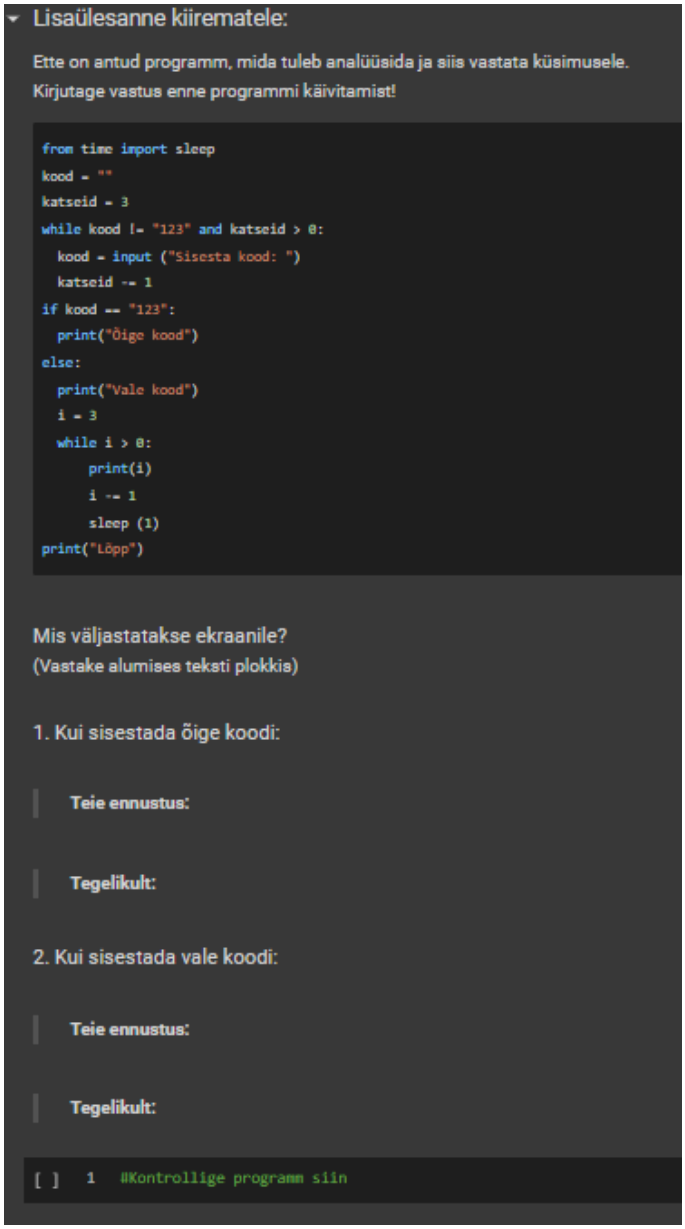


Joonis 17. Kolmas interaktiivne tööleht. „Kontrolli" ülesanne tsükli teemal ja lisaküsimus automaatkontrolliga [9].

Vaatame koostatud töölehtede sisu näidet tsükli teema ülesannete põhjal. Sellel töölehel oli alguses 11 „kontrolli" tüüpi ülesannet, kus tuleb ennustada, mida väljastab antud programm. Interaktiivses töölehes igale küsimusele lisati küsimus mingist konkreetsest reast tsükli sees või tsüklilt väljas. Kuna küsimusele „mitu korda käivitub see koodi rida?" saab vastata ühe

konkreetse arvuga ja saab olla ainult üks õige vastus, siis see tõi sisse ka võimaluse panna juurde automaatkontrolli. Samas selline ülesanne paneb hoolikalt analüüsima koodi ja mõtlema tsükli kordamistingimusest.

Sama töölehe lõppu oli pandud ka lisaülesanne kiirematele, mis on sama „kontrolli" tüüpi ülesanne. Lisaülesanne tingimuses on pikem koodiplokk ja esineb ka eelmise teema kontroll. Teise töölehe teemaks oli tingimuslause, kolmanda oma on tsüklid. Lisaülesanne esitab küsimuse koodist, kus on kaks tsüklit, aga mis nendest läheb käima, sõltub tingimuslausest. Küsimused on esitatud mõlema tsükli kohta, ehk et õpilane peab kaks korda programmi peas läbi käima (vt joonis 18).



Joonis 18. Kolmas interaktiivne tööleht. Lisaülesanne [9].

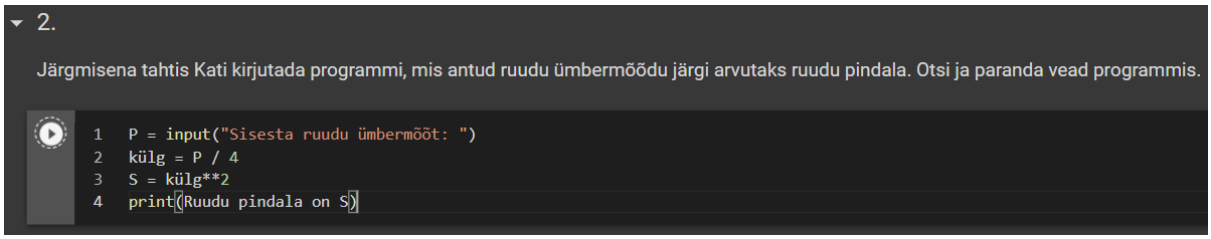
„Kontrolli” tüüpi ülesanded interaktiivses töölehes võtavad rohkem ruumi kui paberil. „Silu” tüüpi ülesanne vastupidi näeb välja kompaktsem. Pabertööleht oli vaja kirjutada koodi ümber vigadeta (vt joonis 19).

2. Järgmisena tahtis Kati kirjutada programmi, mis antud ruudu ümbermõõdu järgi arvutaks ruudu pindala. Otsi ja paranda vead programmis.

Kati programm	Kus on viga?
1 P = input("Sisesta ruudu ümbermõõt: ") 2 külg = P / 4 3 S = külg**2 4 print(Ruudu pindala on S) 5	

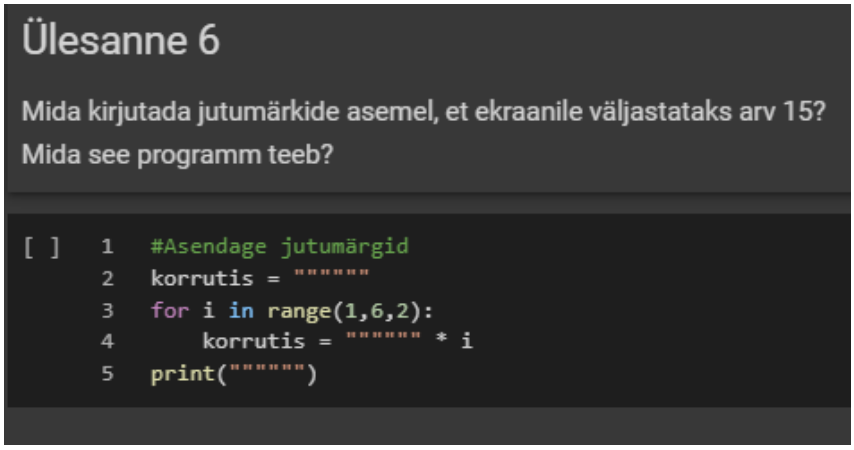
Joonis 19. Pabertööleht. „Silu” ülesanne [5].

Google Colaboratory töölehel võib selle tüüpi ülesannet vormistada nii, et koodiplokis antakse vigane programm ja õpilane võib seal koodi parandada ja jooksutada (vt joonis 20).



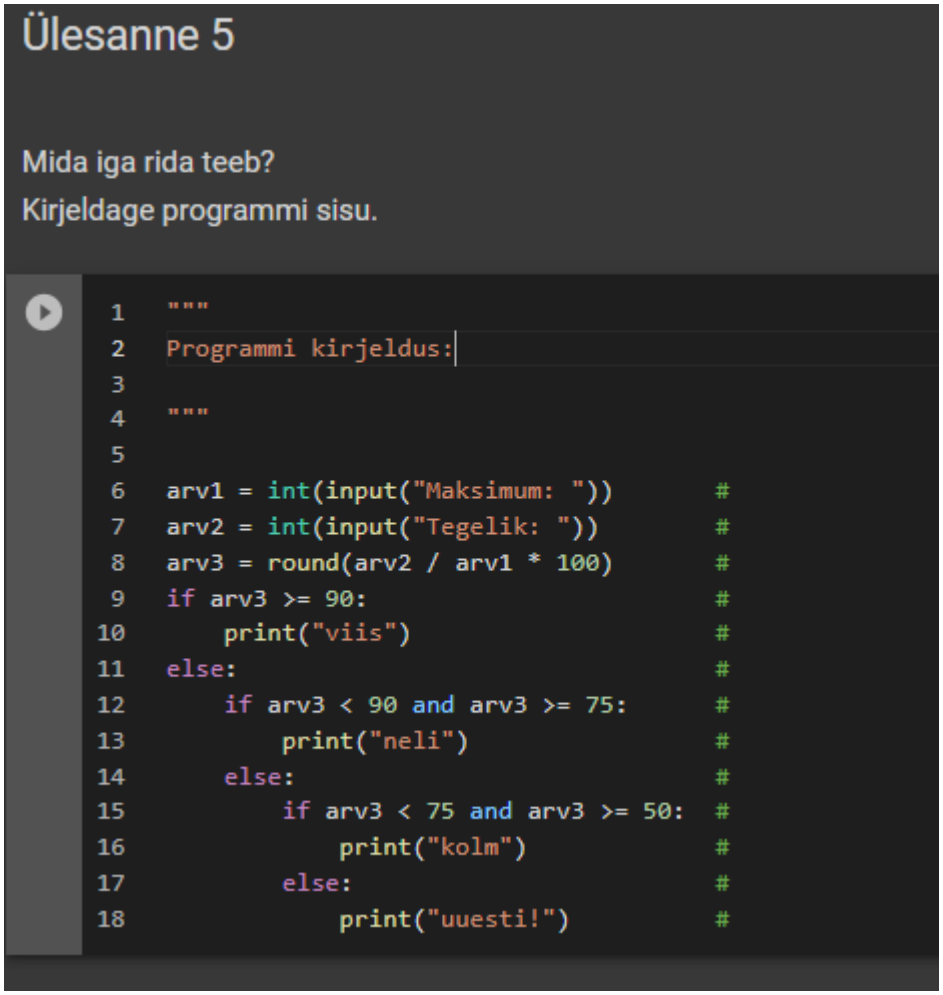
Joonis 20. Interaktiivne tööleht. „Silu” ülesanne [9].

„Rakenda” tüüpi ülesandeid on nüüd võimalik lahendada ka ilma lisaruumita vastuse jaoks. Selle ülesande tüübi jaoks on osa koodist juba antud õpilasele ja ta peab ainult lünkade täitmisega tegelema. Seega interaktiivse töölehe koodiplokis on kirjutatud programm ja õpilane integreerib sinna oma koodi (vt joonis 21). Selleks, et koodiplokk ei näitaks süntaktilist vigu, on programmi lüngad märgitud tühjade sõnedega. Python programmeerimiskeel võimaldab sõne märkida nii kahe jutumärgi vahel, kui ka kolme jutumärgiga. Kolme jutumärgiga sõned sobivad selle ülesande jaoks väga hästi lünkade täitmiseks.



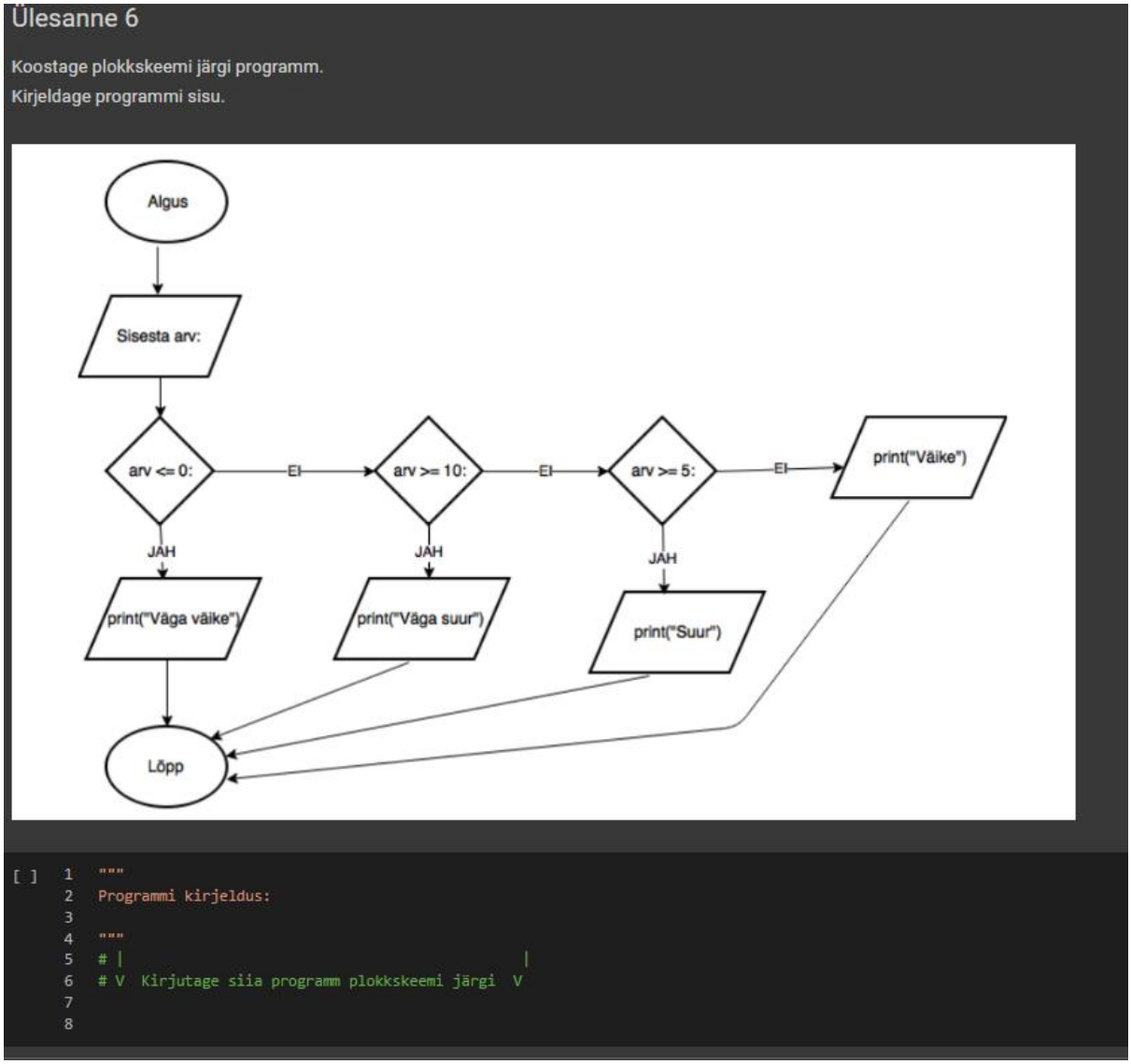
Joonis 21. Interaktiivne tööleht. „Rakenda" ülesanne [9].

Pabertöölehtedel „Selgita" ülesannete jaoks oli kirjas kogu programm ja siis veel kord dubleeritud iga rida, et kirjutada, mis toimub vastava rea käivitamisel. Interaktiivse töölehe puhul otsustati kirjutada programm koodiplokis, nagu „silu" ja „rakenda" ülesannete jaoks. „Selgita" ülesande vastus on võimalik kirjutada ka koodiplokis. Selleks on võimalik kasutada kommentaare. Iga programmi rea lõppu pannakse kommentaari märk, mis Python keeles on #. Enne programmi on olemas ka mitmerealine sõne, piiratud kolme jutumärgiga, mida tihti kasutatakse suure kommentaari jaoks. Õpilased võivad kirjutada nii vastuse küsimusele „Mida see rida teeb" kui ka harjutada kommentaaride kirjutamist programmile, kirjutades programmi üldise kirjelduse (vt joonis 22).



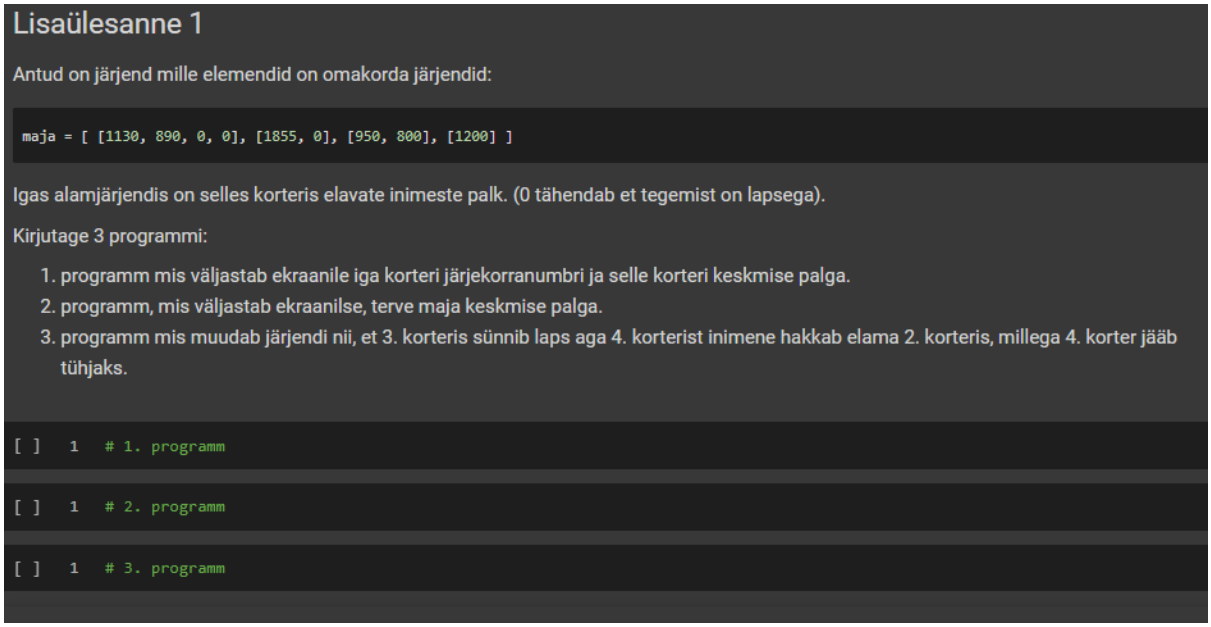
Joonis 22. Interaktiivne tööleht. „Selgita” ülesanne [9].

„Teosta” ülesannet oli alguses ainult üks. Teises töölehes oli vaja skeemi järgi kirjutada programm. Sama ülesanne ilma muutusteta on pandud ka interaktiivse töölehe sisse. Kuna Google Colaboratory võimaldab teksti plokkidesse lisada pilte, anti õpilastele skeemi pilt koos koodiplokiga vastuse kirjutamiseks (vt joonis 23).



Joonis 23. Interaktiivne tööleht. „Teosta" ülesanne [9].

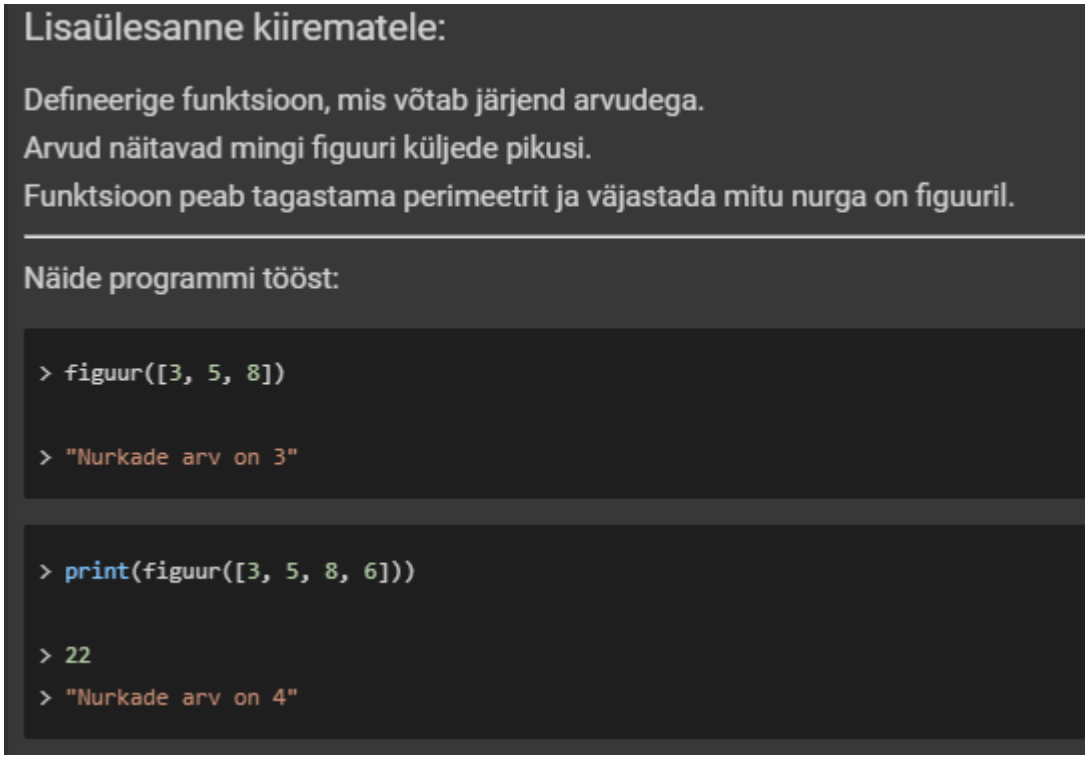
Viienda ja kuuenda interaktiivse töölehe jaoks olid tehtud „teosta" tüüpi lisaülesanded. Kaks lisaülesannet viienda töölehe jaoks olid tehtud sõne ja järjendi kohta. Kuna viies tööleht katab kahte teemat, siis ka lisaülesannet oli tehtud kaks. Õpilased peaksid lugema programmi nõudeid ja kirjutama koodi, mis neid rahuldaks (vt joonis 24 ja 25).



Joonis 24. Interaktiivne tööleht. Järjendi jaoks „teosta" lisaülesanne [9].

Joonis 25. Interaktiivne tööleht. Sõne jaoks „teosta" lisaülesanne [9].

Kuuenda töölehe teemaks oli funktsioon, ja lisaülesanne sai ka „teosta" tüüpi. Antud kirjelduse alusel oleks vaja kirjutada funktsioon, mis nii tagastab ühe väärtuse, kui ka väljastab teise (vt joonis 26).



Joonis 26. Interaktiivne tööleht. Funktsiooni jaoks „teosta" lisaülesanne [9].

Selle töö raames interaktiivsete töölehtede juurde ei ole muid ülesannete tüüpe lisatud. Interaktiivsete töölehtede koostamisel kasutati ainult neid tüüpe, mis olid originaalsetes töölehtedes. Aga oli tehtud ka uus tööleht kordamisküsimustega. Seal oli iga ülesande tüübiks „kontrolli". Kordamisküsimuste tööleht sisaldab 4 pikka teoreetilist ülesannet, mis katavad kõik teemad teistest töölehtedest:

1. ülesanne - tingimuslause
2. ülesanne - tsükkel ja tingimuslause
3. ülesanne - järjend ja tsükkel
4. ülesanne - funktsioon ja sõne

Seda töölehte ei testitud õpilastega, mistõttu ei olnud võimalik koguda tagasisidet selle praktilise rakendamise kohta. Vaatamata sellele programmeerimise kursuse õpetaja vaatas läbi ja andis oma tagasisidet.

3.2 Töölehtede tagasiside ja parandused

Uute interaktiivsete töölehtede reaalse kasutuselevõtu kohta info kogumiseks katsetati neid gümnaasiumis ja koguti nende kohta õpetajalt tagasisidet. Kui töölehte kasutati tunnis, siis pärast tundi õpetaja jagas infot, mis olid probleemsed kohad töölehel, kuidas oli ajakulu ja andis muid kommentaare.

Esimese töölehe kohta teemal „Muutujad ja andmetüübid" oli tagasiside pigem positiivne. Tööleht oli õpetaja sõnade kohaselt hästi koostatud, aga välja tulid ka mõned aspektid, mis vajasid parandamist. Esiteks esimesed küsimused, mis olid juhuarvudega genereeritud ja juhuslikus järjekorras esitatud, tekitasid segadust õpilaste seas. Teiseks oli väike mure sellega, et kui õpetaja näitas esimest töölehte õpilastele, õpetaja sisestas sinna mõned vastused ja oma nime. Tahtmatult õpetaja muutis just originaal faili, mida hiljem jagas õpilastega. Seega esimene näiteülesanne oli juba lahendatud. Viimane negatiivne asi oli seotud ajakuluga. Interaktiivne tööleht vähem aega kui pabertööleht. Seda võib lugeda ka positiivseks, sest siis tekkis võimalus järgmiste töölehtede koostamisel arvestada, et õpilased jõuavad rohkem ülesandeid lahendada. Seega esimene tööleht, kus ei olnud veel lisaülesandeid, võttis maksimumina 30 minutit tunnist, mis kestab 75 minutit. Kiiremad õpilased said hakkama isegi 15 minutiga. Vaatamata sellele läks töölehe katsetamine tehniliselt hästi. Õpilased tagastasid lahendatud töö nii failide kui ka linkidena. Õpetaja mainis, et linkidena jagamine oli mugavam. Esimese töölehe tagasiside alusel sai selgeks, et interaktiivse vormiga küsimused juhuslikult genereeritavate järjekorra ja numbritega tekitab segadust õpilaste seas. Selle tõttu järgmistes töölehtedes juhunumbrite ja järjekorra genereerimist ei kasutatud.

Teise töölehe tegemisel võeti arvesse esimese töölehe tagasisidet. Seega nüüd ei olnud töölehel segadust tekitavaid Colab funktsionaalsusi kasutusel, loodi lisaülesanne ja õpetaja ei muutnud originaalfaili. Tööleht töötas hästi ja ajaliselt sobis tundi koos lisaülesandega. Õpetaja tõi välja, et ühes ülesandes oleks parem programmi ümber kirjutada nii, et kasutada mittevõrduse märgi (!=) asemel võrdusmärki (==), sest nii on lihtsam seletada, mis programm teeb. Aga kuna see oli niimoodi rõhutatult ka originaalses pabertöölehes, jäi see muutmata. Teine kommentaar oli Colaboratory keskkonna koodiploki kohta. Tuli välja, et tavaliselt, kui programmeerija vajutab TAB nuppu klaviatuuril, et taandada koodi, siis Colab paneb kaks tühikut. Arenduskeskkond nimega Thonny, mida õpilased kasutavad kodutööde lahendamiseks, sama nupuga paneb 4 tühikut. Selle erinevuse tõttu mõned õpilased eelistasid skeemi alusel programmi kirjutada Thonny keskkonnas ja siis kopeerida Colabi. Viimane kommentaar oli seotud lisaülesandega.

Töölehe koostamisel lisaülesanne lahendamiseks oli pakutud draw.io veebileht, kus on võimalik teha skeeme ja jooniseid. Mõnedele õpilastele see keskkond ei olnud mugav ja nad eelistasid joonistada skeemi kasutades Paint tarkvara. Aga ülesanne sai ikka lahendatud ja suuri probleeme ei tekkinud. Seega teise töölehe tagasiside oli veelgi positiivsem kui esimese oma.

Kolmas tööleht jõudis selle autori vea tõttu õpetajani halvas seisundis. Link töölehele viis prototüübile, mitte viimasele versioonile. Seega töölehel oli palju vigu. Näiteks 3 ülesannet olid vale automaatkontrolliga. Õigele vastusele anti teade „vale vastus" ja vastupidi. Kaks ülesannet olid halvasti esitatud küsimusega. Õpilased pidid pakkuma, mitu korda käivitatakse ühe programmi rea tsüklis ja pärast kontrollima oma vastust, aga read ei väljastanud midagi ekraanile, ehk otseselt kontrollida oma vastust ei olnud võimalik. Viimase ülesanne jaoks oli kopeeritud vale kood, mille tõttu küsimus ja programm ei olnud kooskõlas. Need probleemsed kohad said parandatud peale katsetamist ja edaspidi töölehtedes ette ei tulnud. Tagasisides oli ka üks keeleline parandus. Ajaliselt sai tööleht mõnedel õpilastel liiga kiiresti tehtud, sest nad ei proovinud programmi lugeda ja ennustada tulemust, aga kohe kopeerisid koodi, jooksutasid ja panid vastuse kirja. Õpetaja kirjutas, et ülesannete formaat on ikka hea, aga loogilised vead olid ikka problemaatilised. Samuti ütles õpetaja, et mõne ülesande vigane tagasiside pani ka õpilasi omakorda koodiõigsusesse rohkem süvenema. Tööleht parandati pärast tundi.

Järgmine tööleht oli tehtud viienda nädala teema jaoks. Tagasiside alusel see tööleht võttis kõige rohkem aega, aga tunni jooksul sai lahendatud. Tööleht koosnes ainult praktilistest ülesannetest, ei olnud teoreetilisi küsimusi ega automaatkontrolliga teste. Õpilased kirjutasid programme, mis vastavad nõuetele põhiosas ja lisaülesanded olid sama tüüpi, aga keerulisemad. Õpetaja tagasisides on ka öeldud, et need lisaülesanded olid hästi koostatud.

Viimane katsetatud tööleht kuuenda nädala jaoks oli ka väga positiivselt hinnatud õpetaja poolt. Tema sõnade järgi läks tööleht väga hästi. Ajaline kulu oli paras ja ülesanded ise pakkusid häid väljakutseid õpilastele. Nii viimane kui ka eelviimane tööleht said keelelisi parandusi õpetaja käest, aga muid probleeme nendega ei olnud.

3.3 Arutelu

Pärast katsetamist ja tagasiside lugemist tehti töölehtede parandused. Viimase ja eelviimase töölehe tagasiside alusel võib järeldada, et interaktiivse töölehe koostamise protsessis tuleb väga hoolikalt jälgida teoreetilisi ja praktilisi ülesannete osakaalu. Viienda nädala tööleht, mis koosnes ainult praktilistest ülesannetest, võttis kõige rohkem aega. Seal oli 17 küsimust, millele

oli vaja vastata kirjutades natuke koodi järjendi manipuleerimiseks ning oli ka 2 lisaülesannet. Kuuenda nädala tööleht sisaldas 11 teoreetilist küsimust koodi analüüsimiseks ja 5 ülesannet koodi kirjutamiseks ning 1 suurem praktiline lisaülesanne. Ajaline kulu näitab, et puhtalt praktiline tööleht võtab väga palju aega. Interaktiivse töölehe koostamisel tuleb lisada teoreetilisi küsimusi ka, kuid siin tuleb õpetajal olla väga ettevaatlik, et mitte lasta teoreetilistele küsimustele vastata peale programmi katsetamist. On oluline rõhutada, et õpilased võtaksid aega koodi analüüsimiseks, mitte ainult juhuslikuks katsetamiseks.

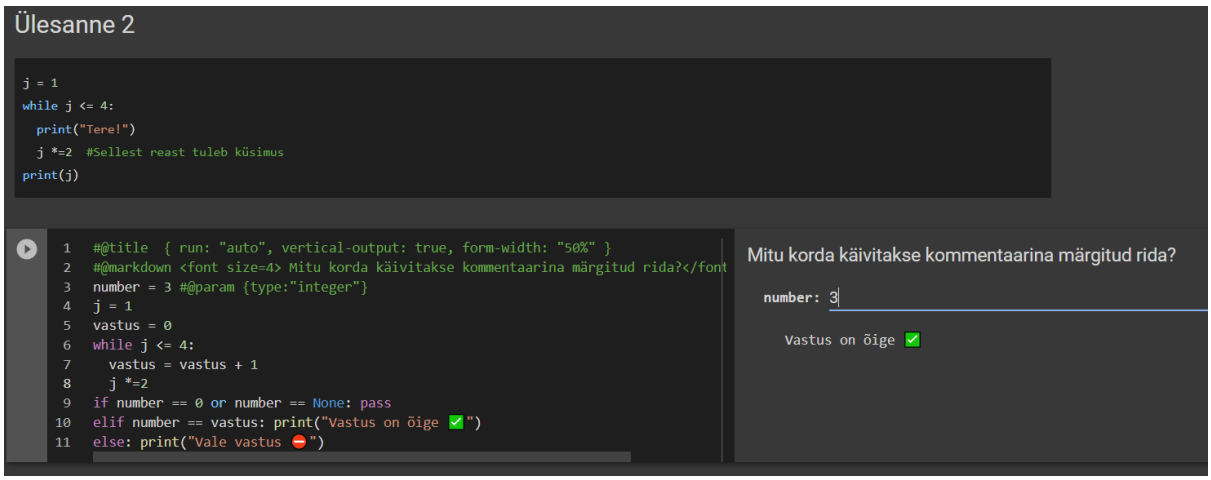
Ajalise sobivuse jaoks on ka oluline ülesannete arv ja töölehe arusaadavus. Esimene tööleht näitas, et vaatamata keerulisele struktuurile „küsimuste tehase" kasutamise tõttu, ajaliselt tööleht läks ikka liiga kiiresti. 10 teoreetilist küsimust automaatkontrolliga, kus vastuseks on üks sõna, ja 6 koodi parandamise ülesannet on kiiresti lahendatavad. Seoses sellega oleks vaja esimese töölehe juurde lisada ülesandeid, mis vajavad rohkem koodi kirjutamist või parandamist, koostama peaks ka lisaülesande ja võib „küsimuste tehase" funktsiooni eemaldada. Liiga palju ülesandeid lisada ei ole vaja, selle asemel võib tunni alguses koos õpilastega uurida juhendit ja esitamise võimalusi.

„Küsimuste tehase" funktsionaalsust võiks katsetada hilisemate tundide töölehtedes. Esimese töölehe kasutamine õpilaste poolt on tõenäoliselt esimene kokkupuude Google Colaboratory keskkonnaga. Hiljem, kui õpilased on juba harjunud töölehti täitma Colabis, siis võib tuua rohkem interaktiivsust ja korraldada töölehe alguses kiiret teadmiste kontrolli või läbitud materjali kordamist, kasutades „küsimuste tehase" juhuarvude generaatori. Aga „tehase" koodi kirjutamiseks läheb palju aega ja tulemus ei ole nii hädavajalik, et kasutada seda iga töölehe jaoks. Kõige parem koht, kus võib kasutada juhuslikult genereeritavaid küsimusi, on viimane tööleht kursuse materjali kordamiseks. See ka nõuab õpetajalt päris keerulise koodi kirjutamiseks, aga võib olla tehtud ainult üks kord ja pakutud õpilastele mitmekordseks kasutamiseks, et valmistada viimaseks kontrolltööks.

Ilma „küsimuste tehase" kasutamist on ka võimalik teha häid teoreetilisi küsimusi automaatkontrolliga. Need võivad kasutusel olla iga töölehe alguses, sest võtavad palju vähem aega, et koostada, aga toovad sisse kiire ja interaktiivse teadmiste kontrolli. Aga nende küsimustega võivad ka probleemid tekkida. Kõige suurem mure seisneb selles, et õpilased võivad juhuslikult või meelega teha interaktiivse vormi koodi lahti ja saada kätte vastused. See ei olnud liiga suur oht koolis katsetamise ajal, kuna töölehtede täitmine ei hinnata. Need töölehed olid ainult selleks et õpilased saaksid rohkem praktikat ja vajadusel küsiksid nõu

õpetaja käest. Kui interaktiivseid töölehti on plaanis kasutada õpilaste hindamiseks, siis automaatkontrolliga küsimused ei sobi üldse selleks eesmärgiks. Vaatamata sellele võib proovida peita vastust. Esiteks vormi tuleb kindlasti laiendada 100% võimalikust ruumist (vt ptk 2.1, joonis 12 ja 13). Niimoodi on raske juhuslikult näha vastust. Teiseks on võimalik kirjutada automaatkontrolli, kus õpilaste poolt sisestatud vastus ei võrrelda kõvakodeeritud õige vastusega (vt joonis 27 halva automaatkontrolliga ja joonis 28 parema versiooniga).

Joonis 27. Interaktiivne tööleht. Vorm nähtava vastusega [9].



Joonis 28. Interaktiivne tööleht. Vorm mitte nähtava vastusega [9].

Vormi sees võib jooksutada sama koodi, mis on küsimuses ja vastust kirjutada muutujasse. Niimoodi, isegi kui vormi kood tehakse lahti, ei ole vastus lihtsasti arusaadav. Aga vormi lahtitegemise probleem ei olnud kasutuse ajal märgatud õpetaja poolt.

Teine mure tekkis sellega, et õpilased kopeerisid koodi, mida tuleb analüüsida ja jooksutaksid seda, et saada vastus. Selle vastu ei ole mingit konkreetset strateegiat. Pabertöölehtede täitmisel võivad õpilased käituda samamoodi, kuid kindlasti koodi kopeerimine on kiirem kui ümberkirjutamine paberilt. See tähendab, et interaktiivsed töölehed tekitavad soodsaid

tingimusi, aga kui õpetajal on tahet ja aega, siis võib proovida vaadata faili muutuste ajalugu. Juhul, kui õpilane muutis koodiploki ja käivitas selle varem, kui teksti vastuse ploki ja nende kahe tegevuse vahel läks piisavalt palju aega, siis seda võiks näha ka faili ajaloos. See ei ole usaldusväärne viis sellise käitumise vastu võitlemiseks, aga mainimist väärt.

Vaatamata sellele, et on veel väga palju ruumi parandamiseks, näitasid töölehed isegi praeguses seisundis ennast hea alternatiivina pabertöölehtedele. Kasutades neid õppevahendeid päriselus tunni jooksul, oli võimalik panna rohkem ülesandeid töölehe sisse kasutades sama aja nende lahendamiseks. Töölehti oli mugav jagada ja lahendatult kätte saada. Sama vormi töölehti on võimalik ja oleks kasulik ka teha teistele gümnaasiumi kursustele või isegi ülikoolile. Informaatika programmeerimise kursuse jaoks on võimalik ka koostada interaktiivseid töölehti samas keskkonnas. Kuna Google Colaboratory on kasutusel ka teiste ülikooli kursuste raames, oleksid interaktiivsed töölehed Colab keskkonnas kasulikud informaatika bakalaureuse programmis. Colabi töölehti võib kasutada ka magistriõppes, näiteks infotehnoloogia mitteinformaatikutele õppekava raames.

Paljudes õppekavades, koolides ja ülikoolides integreerimiseks on kindlasti vaja õpetada õpetajaid neid töölehti koostama, muutma ja kasutama. Praegu olemasolevate töölehtede kasutamiseks ei ole vaja palju kogemust või teadmisi sellest, kuidas nad toimivad, aga põhiarusaam sellest, mis on Google Colaboratory ja mis võiksid olla kasutamisega seotud raskused, peab õpetajatel olema. Sarnaselt õpilastele tehtud juhendiga on võimalik teha ka sarnane .ipynb laiendusega fail õpetajetele. Teoreetiliselt see peaks sisaldama sama infot, mis on vaja õpilastele ja mõningaid vihjeid õpetaja jaoks: kuidas jagada, tagasi saada, hinnata kommentaaridega, kontrollida käivitamise ja muudatuste ajalugu ja nii edasi.

Uute töölehtede koostamine või olemasolevate muutmine on õpetajatele raskem ülesanne. Selleks oleks võimalik korraldada töötubasid ja kursusi. Näiteks informaatika õpetajate konverentsi ajal on võimalik korraldada töölehtedega tuvustamise tööruum ja jagada vajalikke teadmisi interaktiivsetest töölehtedest. 2023. aasta jaanuaris toimunud informaatika õpetajate konverentsil selle lõputöö autor korraldas töötoa, et tutvuda interaktiivsete töölehtede ideed, tol ajal valmisolevaid töölehti ja näidata Google Colaboratory põhifunktsionaalsust [10]. Töötoa ajal õpetajad ise kasutasid valmis töölehti õpilaste rollis ja proovisid koostada uusi töölehti nullist. Ühe töötoa jooksul ei olnud piisavalt aega, et näidata kõiki võimalikke viise töölehe koostamiseks, aga see tekitas huvi. Mitu gümnaasiumi õpetajat tundsid huvi interaktiivsete töölehtede kasutamise vastu.

Kokkuvõtte

Selle bakalaureusetöö eesmärk oli muuta olemasolevad pabertöölehed interaktiivseteks töölehtedeks säilitades väljatöötatud ülesanded ja nende tüübid. Töölehed on väga levinud õppematerjal programmeerimise õpetamisel, aga nende paberil kasutamine ei tundu kaasaegne.

Interaktiivsete töölehtede loomiseks valiti Google Colaboratory keskkond, mis võimaldab ilma allalaadimiseta või installeerimiseta kasutada .ipynb faili redaktorit. Google Colab võimaldab koostada, muuta ja vahetada interaktiivseid märkmikke, mis võivad sisaldada tekste, pilte, videoid ja Python keeles koodi. See on sobilik tarkvara koolide jaoks, sest see on tasuta ressurss, mis vajab ainult Google'i kontot ja ühendust internetiga.

Töö raames said 5 pabertöölehte interaktiivse versiooni. Lisaks ka keskkonna kasutamisjuhise õpilastele ja uue töölehe materjali kordamiseks ning koostati lisaülesanded. Uusi interaktiivseid töölehti katsetati gümnaasiumi programmeerimise kursusel 10. klassi õpilastega. Iga töölehe kohta andis programmeerimise õpetaja, kes kasutas uut materjali, kvalitatiivset tagasisidet.

Tagasiside alusel tehti jooksvalt parandusi olemasolevates interaktiivsetes töölehtedes ja püüti järgmises töölehes mitte lasta samal probleemil tekkida. Tagasiside sisaldas mitut olulist aspekti: ajaline sobivus, ülesannete arusaadavus, ülesannete ülesehitus, ülesannete vorm ja õigekeelsus.

Kuna interaktiivsete töölehtede täitmine võtab vähem aega, kui paberlehtede täitmine, siis kosati töölehtedele ka lisaülesandeid. Aega jäi üle esimese töölehe puhul, mis sisaldab ka töölehtede formaadiga tutvumist. Seega on edaspidi võimalik ajaliselt esimese tunni raames anda rohkem ülesandeid. Järgmised töölehed sobisid ajaliselt hästi.

Katsetamine õpilastega näitas, et töölehed on hästi kasutatavad ja ei ole vaja töölehti ainult paberil trükkida. Interaktiivsed töölehed võimaldavad teha sama tööd ja lahendada rohkem ülesandeid sama aja jooksul. Koostatud töölehed saab praeguses seisundis kasutusele võtta ka teistes koolides, kus varem kasutati pabertöölehti programmeerimise kursuse raames. Interaktiivseid töölehti on võimalik koostada ka teiste kursuste jaoks, kui nad on seotud Python keeles programmeerimisega. Samas formaadis töölehed võivad olla kasulikud ka kõrgematel õppetasemel, näiteks bakalaureusetaseme informaatika õppekava või infotehnoloogia mitteinformaatikutele magistri õppekava raames. Interaktiivsete töölehtede kasutamine pakkus huvi mitmele informaatikaõpetajatele.

Kasutatud kirjandus

[1] M. Abel, E. Kolk, Matemaatika-Informaatikateaduskond 40, lk. 159-166, 2007.

[2] K. Kori, P. Beldman, E. Tõnisson, P. Luik, R. Suviste, L. Siiman, M. Pedaste. 2019. IT oskuste arendamine Eesti koolides Uuringu raport. https://wise.com/documents/IT%20oskuste%20arendamine%20Eesti%20koolides.pdf (15.12.2022)

[3] Vabariigi Valitsus. 2011. Gümnaasiumi riiklik õppekava. https://www.riigiteataja.ee/akt/123042021011 (16.12.2022)

[4] E. Tõnisson, T. Palts, M. Säde, K. Tõnisson jt. Programmeerimine. Informaatika valikkursus gümnaasiumile.

https://web.htk.tlu.ee/digitaru/programmeerimine/

(11.04.2023)

[5] Programmeerimine. Tunnimaterjalid.

https://drive.google.com/drive/folders/1-x9LkMGk0kddTFFKyhVvi6iS20UbFOG-?usp=share_link (07.01.2023)

[6] R. Schihalejev, T. Palts. 2021. Programmeerimise ülesannete tüübid. https://courses.cs.ut.ee/t/progylTyybid/Main/HomePage (05.04.2023)

[7] Dave Gershgorn. 2017. Nerds rejoice: Google just released its internal tool to collaborate on AI.

https://qz.com/1113999/nerds-rejoice-google-just-released-its-internal-tool-to-collaborate-on-ai (07.04.2023)

[8] Colaboratory team. 2022. Google Colaboratory Marketplace description. https://workspace.google.com/marketplace/app/colaboratory/1014160490159 (07.04.2023)

[9] Andrei Tambovtsev. Programmeerimine. Interaktiivsed töölehed. https://drive.google.com/drive/folders/1eek6VLFVLByxi790Iz1mjuOFX7-q99Ci?usp=sharing (08.04.2023)

[10] Informaatika õpetajate konverents 2023. https://didaktika.cs.ut.ee/sundmused/konverents-2023/ (09.05.2023)

Lisad

I. Algsed töölehed

I. Programmeerimine. Töölehed.

https://drive.google.com/drive/folders/1-x9LkMGk0kddTFFKyhVvi6iS20UbFOG-?usp=share_link

II. Valminud töölehed

II. Colab juhend õpilastele

https://colab.research.google.com/drive/1BeSFR4uSDL01cXYyBkyRSkx-XjaRoRbK?usp=sharing

II. Muutujad ja andmetüübid

https://colab.research.google.com/drive/12CXw7V_SsmoDzaxvuvJgKxWq3KUuRmkN?usp=sharing

III. Tingimuslause

https://colab.research.google.com/drive/1SkdaycOzsB0l8vZDrYbipAA7LrUzvq6L?usp=sharing

IV. Vastus lisaülesannele:

https://drive.google.com/file/d/1iSM8QWHm3dk4Udv5ueZYokuweM_RZFXd/view?usp=share_link

V. Tsükkel

https://colab.research.google.com/drive/1zoLgC-zkfdTfamVVRMMjDW7vKZ0ZUKrC?usp=sharing

VI. Järjend

https://colab.research.google.com/drive/18G_2Y2RvGd2rqrNN56TQtF0qo_jQKHXA?usp=sharing

VII. Funktsioon

https://colab.research.google.com/drive/1cEzi4sGDMmzISm9NBPiyQ-RTfoAivcv_?usp=sharing

VIII. Kordamine

https://colab.research.google.com/drive/15o6fz6ygKARWzkx1DkgMzPYUMG6Njjar?usp=sharing

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Andrei Tambovtsev ,

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

   Interaktiivsete Google Colaboratory töölehtede loomine gümnaasiumi programmeerimise kursusele ,

   mille juhendaja(d) on Tauno Palts ,

   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.

Andrei Tambovtsev

14.05.2025