TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Raigo Kodasmaa

Programmeerimiskeele Python veateated programmeerimise algõppes

Magistritöö (30 EAP)

Juhendaja: Eno Tõnisson, MSc

TARTU 2017

Programmeerimiskeele Python veateated programmeerimise algõppes

Lühikokkuvõte:

Magistritöö raames viidi läbi uuring programmeerimiskeele Python veateadetest programmeerimise algõppes ja koostati õppematerjal vastavalt uuringus saadud tulemustele. Tartu Ülikooli vaba juurdepääsuga programmeerimisteemalistel e-kursustel osaleb tuhandeid inimesi, kes lahendavad e-kursuste raames kümneid ülesandeid. Hetkel on programmeerimiskeele Python veateated ingliskeelsed, lühidalt kirjeldatud ja sisaldavad palju tehnilisi detaile, mistõttu on need algajatele programmeerijatele keeruline mõista. Käesolev töö keskendubki veateadete uurimisele ja lahti seletamisele, et algõppe programmeerimise e-kursuslased neid paremini mõistaksid ja seega ka enda programmis tekkinud vigu parandada oskaksid. Läbiviidava uuringu raames koguti tagasisidet e-kursuse *Programmeerimisest maalähedaselt* osalejatelt programmeerimiskeele Python sisseehitatud kompilaatori veateadete kohta küsitluse ja töövahendina kasutusel oleva programmeerimiskeskkonna Thonny logifailide näol. Kogutud andmete põhjal viidi läbi analüüs ja valiti välja sagedamini esinenud veateated, mille kohta koostati eestikeelne õppematerjal. Materjal koosneb 7 levinumast veatüübist, kus omakorda 29 levinumat veateadet on seletatud lahti eesti keeles ja neile on lisatud 35 programminäidet koos võimalike parandusviisidega. Veateadete õppematerjali saab kasutada programmeerimisteemaliste e-kursuste juures abimaterjalina erinevate ülesannete lahendamisel.

Võtmesõnad:

Pythoni veateade, programmeerimise algõpetus, õppematerjal

CERCS: P175 - Informaatika, süsteemiteooria

Python error messages in basic level programming studies

Abstract:

This thesis describes the study of programming language Python error messages in basic level programming studies, which results were the source to an educational material. Thousands of people are taking part in online courses provided by the University of Tartu, where tens of assignments needs to be completed. By default the Python error messages are in English, they are described shortly and consist of many technical details, which makes them difficult to understand for basic level programmers. This master's thesis focuses on studying Python error messages to explain and simplify them to novice programming learners to develop better understanding of error messages. In order to achieve that a valuable feedback was gathered from *Programmeerimisest maalähedaselt* online course by conducting a survey and collecting integrated development environment Thonny log files. The most frequently occurred error messages from those log files were extracted and combined with survey results they were a source to an educational material in Estonian. This material includes 7 most common error types and 29 error messages illustrated by 35 programming examples so it can be used in programming online courses as an auxiliary educational material to help novice programmers to complete various assignments.

Keywords:

Python error message, basic level programming, educational material

CERCS: P175 - Informatics, systems theory

Sisukord

Sissejuhatus

Huvi programmeerimise õppimise vastu on kõrge. Seda näitab asjaolu, et 2014. aastal loodud programmeerimisteemalisel vaba juurdepääsuga e-kursusel *Programmeerimisest maalähedaselt* on 02.01.2017 seisuga kolme aastaga kokku osalenud ligi 5500 huvilist. Lisaks on loodud 2016. aastal samuti vaba juurdepääsuga e-kursus *Programmeerimise alused*, kus käsitletakse programmeerimise teemasid pisut sügavamalt ja detailsemalt. Sellel kursusel osales kahel toimumiskorral kokku 2065 e-kursuslast [1]. Sellise hulga osalejate puhul on õppejõududel keeruline jälgida e-kursuslastele antud ülesannete lahendamiskäike, mistõttu peavad kursusel osalejad enamasti ka programmeerimisel tekkinud vead iseseisvalt üles leidma ja need parandama. Veateated, mida kasutajale vigase programmi käivitamisel kuvatakse, on algajatele enamasti keerulised mõista, mistõttu ei pruugi nende alusel vea parandamine ning tulevikus selle tekkimise vältimine olla lihtne ülesanne [2].

Käesoleva töö eesmärgiks on uurida sagedasemate veateadete kirjeldusi ja nende mõistmist e-kursuste *Programmeerimisest maalähedaselt* ja *Programmeerimise alused* raames [3][4]. Kuna eelpool mainitud e-kursustel kasutatakse programmeerimiskeelt Python, on ka töös kesksel kohal programmeerimiskeele Python veateated. Magistritöö tulemusena valmib õppematerjal, mis koosneb levinumate veateadete eestikeelsetest kirjeldusest, näidetest, kuidas veateade võib tekkida, ja võimalikest parandusviisidest [5]. Materjal on mõeldud kasutamiseks abimaterjalina erinevate programmeerimisülesannete lahendamisel.

Autori personaalne huvi veateadete uurimise vastu programmeerimise algõppes tekkis e-kursustel töövahendina kasutusel oleva programmeerimiskeskkonna Thonny logifailide analüüsimisel, kust selgus, et teatud tüüpi veateateid esines märgatavalt rohkem kui teisi. Kuna veateateid võib lugeda kõige olulisemaks suhtluse lüliks programmi koostaja ja süsteemi vahel, siis see ajendaski mõttele, et programmeerimise õppimisele võib tulla kasuks, kui seletada lahti levinumad veateated ja lisada juurde näiteid ning parandusviise [6].

Käesoleva töö esimeses osas on toodud ülevaade programmeerimise õppimisest ja veateadetest ning nende kohta tehtud uuringutest ning õppematerjalidest. Magistritöö teises osas kirjeldatakse töö autori poolt läbiviidud programmeerimiskeele Python veateadete uuringut, mille raames paluti e-kursusel osalejatel vastata küsitlusele ja üleslaadida õppekeskkonda Moodle programmeerimiskeskkonna Thonny logifaile. Küsitluses uuriti muuhulgas e-kursuslaste

inglise keele oskuse taset, tekkinud veateadete keerukust, kirjeldust, detailsust ja kasulikkust vea parandamisel. Veateadete esinemissageduse jaoks kogutud Thonny logifailid sisaldasid informatsiooni programmi koostamise käigus esinenud veateadete kohta. Nende andmete põhjal loodi õppematerjali struktuur ja koostati valim veateadetest, mis lisati materjali. Töö kolmanda osa moodustab veebikujul olev õppematerjal [5], mis avati kasutamiseks e-kursuse *Programmeerimise alused* osalejatele, et koguda materjalis sisalduvate veateadete kohta tagasisidet. Õppematerjali iga veateate juurde lisati tagasisidevorm, kus paluti e-kursusel osalejal hinnata veateate kirjelduse kasulikkust vea mõistmisel ja näidete abi vea parandamisel. Samuti oli e-kursuslastel võimalus lisada tagasiside lahtrisse veateateid, mida veel materjalis ei leidunud.

Valmivast õppematerjalist võiks olla kasu eelkõige algajatel programmeerimise õppijatel, kellel ei ole piisavalt kogemusi veateadetega kokkupuutumisel. Õppematerjali kasulikkus seisneb peamiselt parema ülevaate andmises erinevate veateadete tekkimise osas, mida illustreerivad näited programmivigade võimalike parandusviisidega. Materjali on plaanis kasutada järgmistel programmeerimise algõppe e-kursuste toimumisaegadel, kus vastava e-kursuse veebilehele on lisatud viide materjali sisule.

1 Programmeerimise algõpe ja veateated

Käesolevas peatükis antakse ülevaade programmeerimise algõppest ja selle õpetamisega seonduvast Tartu Ülikooli vaba juurdepääsuga MOOC (*Massive Open Online Course*) tüüpi programmeerimisteemalistel e-kursustel, mille raames veateadete uuring läbi viidi. Tutvustatakse programmeerimiskeskkonda Thonny ja selle erinevaid võimalusi. Ühtlasi vaadeldakse ka varasemate uuringute tulemusi, millel võib leida seoseid programmeerimiskeele Python veateadete uuringuga.

1.1 Programmeerimise õppimine

Programmeerimise õppimine eeldab abstraktsete kontseptsioonide mõistmist, mistõttu ei ole seda lihtne õppida [7]. Kuna ülesanded on enamasti püstitatud loomulikus keeles, võivad programmi kirjutamisel mõned nüansid jääda tähelepanuta, sest programmi koostaja peab pidevalt analüüsima, millist fragmenti millises programmilõigus kasutada. Samuti tuleb arvesse võtta, et programmeerimiskeel on rangem kui loomulik keel – iga valesti trükitud sümbol võib muuta programmi tähendust või selle kompileerumise õnnestumist [8]. Algajad programmeerijaid lähenevad tihti programmile ridahaaval, kuna nende teadmised programmist võivad olla pinnapealsed ja nad ei oska siduda omavahel programmi erinevaid osi [7].

Uue informatsiooni õppimisel saab eristada kahte selget stiili: sügavõpe (*deep learning*) ja pindmine õpe (*surface learning*). Sügavuti õppijad keskenduvad sellele, et omandada arusaam õpitavast teemast kasutades selleks ka varasemaid teadmisi, pinnapealse õppimisstiiliga inimesed püüavad jätta meelde võimalikult palju informatsiooni [9]. Programmeerimise puhul on vaja leida kesktee nende kahe vahel - pindmise õppimisega saab meelde jätta programmeerimiskeele süntaksilisi eripärasid ja erinevate operaatorite järjekorda, sügavõppega saavutab aga arusaamise algoritmidest, mida programmi arendamisel vaja läheb.

Inimesed eelistavad õppida mitmel erineval moel. Osad omandavad informatsiooni paremini õppides üksinda, teised jällegi arutelu käigus kollektiivselt [10]. Algajad programmeerijad kasutavad tihti ka katseeksitusmeetodit otsides Internetist sobivaid näiteid ja proovides neid modifitseerida. On täheldatud, et olemasoleva programmikoodi muutmine ja parandamine veateadetele tuginedes on oluliselt lihtsam, kui uue kirjutamine nullist. Analoogia põhimõttel on

konkreetsed programminäited ja veaparanduse näited suuremaks abiks õppijale, kui üldisemad abstraktsed probleemi lahendused. [11]

Veateadetel on väga oluline roll programmeerimise algõppes, kuna need on peamised indikaatorid õppijale, et midagi on programmis valesti. Hea veateate puhul ei satu kasutaja esimese asjana segadusse, vaid saab sealt piisavalt informatsiooni, et tekkinud viga parandada. Raskesti loetavad ja mõistetavad veateated võivad tekitada frustratsiooni või viia kasutajat valele teele, mis omakorda võib põhjustada uute veateadete ilmnemist [12].

1.2 Programmeerimiskeel Python õppimiskeelena

Tartu Ülikoolis kasutatakse algõppe e-kursustel programmeerimiskeelt Python. Programmeerimiskeele Python süntaks on fokusseeritud loetavusele ja sidusale programmimudelile, mille erinevad osad funktsioneerivad järjekindlal ja limiteeritud moel [7][13]. Programmi loetavust mõjutab asjaolu, et programmeerimiskeele Python programmikood on lühem, kui sarnane programmikood mõnes teises programmeerimiskeeles (*Java*, *C*, *C++*) [7]. Kuna Python on vabavara, siis saab seda iga programmeerimist õppida sooviv inimene enda personaalsesse arvutisse tasuta alla laadida [13]. Kõik eelpool mainitud aspektid teevad programmeerimiskeelest Python hea õppevahendi algajatele programmeerimise õppimiseks [7].

Programmeerimisel tekib vigu, olenemata programmeerimiskeelest või eelnevast programmeerimise kogemusest. Vea tekkimisel kuvatakse vastav veateade, mis on programmi poolt tagastav informatsioon selle kohta, et programmikoodis on tehtud kasutaja poolt viga. Programmeerimiskeele Python puhul jagunevad veateated kahte suuremasse rühma: süntaksivead (*syntax errors*) ja erindid (*exceptions*) [14].

Süntaksivea puhul on lähtekood (*source code*) kompilaatorile tundmatu, kuna see pole interpreteeritav arvuti protsessorile [12]. Programmi ei käivitata ning kasutajale kuvatakse veateade, mis sisaldab reanumbrit ja vea asukohta lähtekoodis koos vea lühikirjelduse ning veatüübiga (Joonis 1).

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 1
    print "Sisestage arv: "
                          ^
SyntaxError: Missing parentheses in call to 'print'
>>>
```

Joonis 1. Programmeerimiskeele Python süntaksivea näide [5].

Erindi korral programm käivitatakse, kuid mingi konkreetse käsu täitmine ebaõnnestub [8]. Selline käitumine viitab programmi loogikaveale ja iseloomustab näiteks indeksiviga (*index error*), millest sarnaselt süntaksiveale on pikemalt juttu töö teises pooles [12].

1.3 Varasemad veateadetega seotud uuringud

Käesolevas jaotises keskendutakse veateadetega seotud varasemate uuringute tulemustele, et koguda informatsiooni kolmandas peatükis pikemalt kirjeldatava õppematerjali loomiseks.

1.3.1 Uuring veateadete stiilide efektiivsusest

Marie-Hélène Nienaltowski, Michela Pedroni ja Bertrand Meyer viisid läbi uuringu ETH ja Birkbecki Ülikoolis, kus osales vastavalt 43 ja 24 programmeerimise algõppe üliõpilast. Uuringu algul püstitati hüpotees *mida rohkem informatsiooni veateade sisaldab, seda suurem on tõenäosus, et üliõpilane mõistab paremini tekkinud viga ja oskab seda korrektselt parandada*. Kasutati viite erinevat programmeerimiskeele (*C++*, *Java*, *Ada*, *Scheme*, *Eiffel*) kompilaatorit ja kolme erinevat veateate stiili (*short form*, *visual form*, *long form*) (Tabel 1).

Tabel 1. Kompilaatorid ja veateadete stiilid [15].

		Lühivorm	**Visuaalne vorm**	**Pikk vorm**
Programmeerimiskeskkond	***C++***	Digital Mars, Borland, GNU C++, IBM XL C/C++ for AIX		
	Java	SUN JDK6	BlueJ	
	Ada	Gnat-ada95		
	Scheme		Dr. Scheme	
	Eiffel			EiffelStudio

Veateate lühivorm (*Short form*) on kõige tavalisem kompilaatori poolt edastatav veateate stiil, mis koosneb failinimest, tuvastatud vea asukohast, veatüübist ja lühikirjeldusest (Joonis 2) .

Lühivormi näide:

```
ticket_machine.e, line 27: Cannot find identifier.
total := total + price
^
```

Joonis 2. Veateate lühivormi näide [15].

Veateate visuaalse vormi (*Visual form*) puhul värvitakse viga põhjustanud osa programmikoodis teist värvi ja kuvatakse kasutajale koos lühikese veateatega (Joonis 3).

Visuaalse vormi näide:

```
class TICKET_MACHINE
feature {NONE} -- Access
   price: INTEGER
       -- Cost of ticket
   balance: INTEGER
       -- Amount of inserted money
   total: INTEGER
       -- Total value of transaction

feature -- Basic Operations
   print_ticket is
       -- Print ticket.
     local
       an_amount: INTEGER
     do
       if ballance >= price then
         io.put_string ("USD " + price.out)
         total := total + price
         balance := balance - price
       else
         ...
       end
     end
end

Error message:
Cannot find identifier.
```

Joonis 3. Veateate visuaalse vormi näide [15].

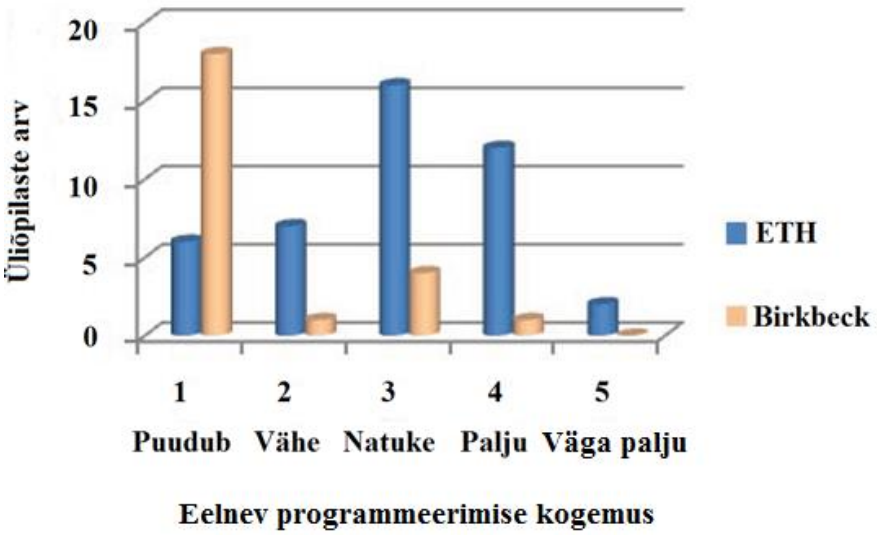
Veateate pika vormi (*Long form*) puhul lisatakse veateatele lisaks lühivormis olevale informatsioonile võimalikke parandusvõimalusi ja kuvatakse veast mõjutatud teisi programmikoodi osasid ning programmiklassi nimi, kus viga esineb (Joonis 4).

Pika vormi näide:

```
Error code: VEEN
Error: unknown identifier.
What to do: make sure that identifier, if needed,
is final name of feature of class, or local entity
or formal argument of routine.
Class: TICKET_MACHINE
Feature: make_tm
Identifier: price
Taking no argument
Line: 10
do
-> price := ticket_cost
balance := 0
```

Joonis 4. Veateate pika vormi näide [15].

Uuringus osalejatel paluti hinnata enda programmeerimisoskust skaalal 1–5, kus 1 tähistab minimaalset eelnevat kogemust ja 5 tähistab suurt kogemust programmeerimise valdkonnas. Jooniselt 5 näeme, et 30% üliõpilastest on programmeerimisega kokkupuutund vähe või üldse mitte (skaalal 1–2), 37% ja 28% on kokkupuutunud natuke või rohkem (skaalal 3–4) ja 5% pidas end kogenud programmeerijaks.



Joonis 5. Üliõpilaste eelnev programmeerimise kogemus skaalal 1–5 [15].

Üliõpilastel paluti vastata üheksale küsimusele, mis hõlmasid kolme veateate stiili (lühivorm, visuaalne vorm ja pikk vorm) ja kolme veatüüpi: tundmatu identifikaator (*unknown identifier)*, vale hulk argumente (*wrong number of arguments)* ja juurdepääsu rikkumine (*private access violation*). Küsimused olid valikvastustega ja üliõpilased pidid veateate puhul arvama õigesti

veatüübi. Eraldi hinnati kahe ülikooli erineva eelneva programmeerimise kogemusega üliõpilasi (Tabel 2).

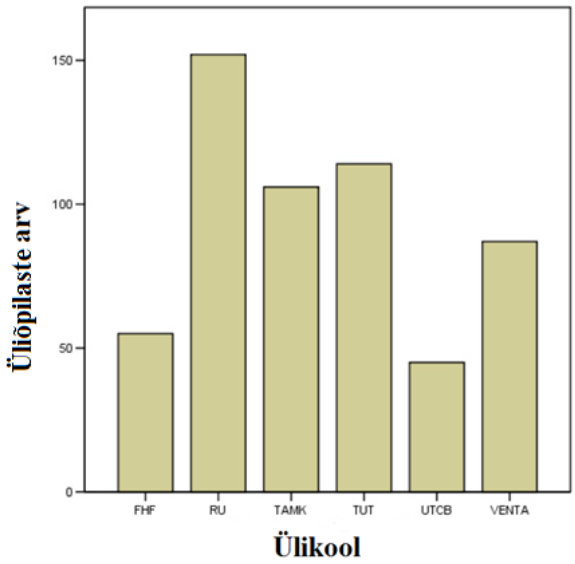
Tabel 2. ETH ja Birkbecki üliõpilaste õiged vastused protsentuaalselt [15].

		Lühivorm	Visuaalne vorm	Pikk vorm
ETH	**Tase 1 & 2**	84.62	71.79	84.62
	Tase 3, 4, 5	95.56	94.44	90.00
Birk-beck	**Tase 1 & 2**	57.89	47.37	56.14
	Tase 3, 4, 5	66.67	60.00	66.67

Tabelist 2 näeme, et uuringu tarbeks püstitatud hüpotees ei pea paika. Veateatele lisainformatsiooni lisamine ei aita ilmtingimata üliõpilastel paremini mõista tekkinud viga. Tulemustest selgub, et visuaalne veateate stiil tekitas uuringus osalejate seas kõige rohkem segadust, kuid pikema või lühema veateate osas olid õigete vastuste protsendid sisuliselt võrdsed [15].

1.3.2 Uuring programmeerimise õppimise ja õpetamise kitsaskohtadest

Essi Lahtinen, Kirsti Ala-Mutka ja Hannu-Matti Järvinen küsitlesid kuue ülikooli 559 üliõpilast (kellel on vähemalt 1 programmeerimiskursus läbitud) ja 34 õppejõudu. 73,4% üliõpilastest kasutas *C++* programmeerimiskeelt kursuse läbimisel, 17,3% programmeerimiskeelt *Java*. Teisi programmeerimiskeeli kasutas vähem kui 10% üliõpilase. Ülikoolidest olid kaasatud Fachhochschule Furtwangen (FHF, Germany), Reykjavik University (RU, Iceland), Tampere Polytechnic (TPU, Finland), Tampere University of Technology (TUT, Finland), Bucharest University of Technology (UTCB, Romania) ja Ventspils University of Technology (VENTA, Latvia). (Joonis 6)



Joonis 6. Üliõpilaste jaotus ülikoolide vahel [16].

Küsitlus koosnes neljast küsimusest, millest kaks tükki hõlmas programmeerimise õpetamist ja õppimist ning kaks tükki kursuse materjalide kasulikkust ja kursuse sisu. Vastata sai skaalal 1–5, kus 1 tähistas kõige lihtsamini ja 5 kõige raskemini õpitavat varianti. Tabelis 3 kujutatud esimeses tulbas on küsimus, teises tulbas küsimuse identifikaator ja tulbad 3–5 sisaldavad üliõpilaste vastuseid (kokku vastuseid, keskmine skoor, standardhälve). Analoogselt on kujutatud tulbad 6–8, mis sisaldavad sarnast informatsiooni, ainult et vastajateks on üliõpilaste asemel õppejõud (kokku vastuseid, keskmine skoor, standardhälve).

Tabel 3. Materjalid ja meetodid, mis aitavad kaasa programmeerimise õppimisele [16].

Question	Code	Students			Teachers		
		N	Avg	Std	N	Avg	Std
THE COURSE CONTENTS							
What kind of issues you feel difficult in learning programming?							
Using program development environment	I1	553	2,43	0,99	33	2,61	0,90
Gaining access to computers/networks	I2	536	2,11	0,95	32	1,97	0,78
Understanding programming structures	I3	556	2,92	1,02	33	3,27	0,67
Learning the programming language syntax	I4	555	2,75	1,01	33	2,70	0,73
Designing a program to solve a certain task	I5	555	3,12	0,98	33	3,97	0,73
Dividing functionality into procedures	I6	543	3,10	1,09	31	4,06	0,63
Finding bugs from my own program	I7	549	3,28	1,03	33	3,91	0,77
Which programming concepts have been difficult for you to learn?							
Variables (lifetime, scope)	C1	541	2,10	0,97	34	2,41	0,70
Selection structures	C2	552	1,98	0,90	34	2,38	0,70
Loop structures	C3	551	2,09	0,97	34	2,79	0,91
Recursion	C4	512	3,22	1,03	31	4,06	0,96
Arrays	C5	526	2,79	1,15	33	3,24	0,71
Pointers, references	C6	518	3,59	1,04	32	4,44	0,56
Parameters	C7	513	2,60	1,09	32	3,47	0,76
Structured data types	C8	496	2,90	1,03	31	3,45	0,81
Abstract data types	C9	499	3,02	1,10	31	4,06	0,81
Input/output handling	C10	519	2,96	1,04	32	3,75	0,88
Error handling	C11	481	3,33	1,01	32	4,13	0,79
Using language libraries	C12	465	3,04	1,09	32	3,88	0,71
LEARNING AND TEACHING PROGRAMMING							
When do you feel that you learn issues about programming?							
In lectures	S1	543	3,01	1,01	33	3,21	1,02
In exercise sessions in small groups	S2	510	3,44	1,10	32	3,84	0,99
In practical sessions	S3	514	3,77	1,03	31	4,35	0,75
While studying alone	S4	546	3,79	1,06	31	3,42	0,72
While working alone on programming coursework	S5	539	3,98	1,09	33	4,00	0,79
What kind of materials have helped/would help you in learning programming?							
Programming course book	M1	515	3,35	1,03	33	3,30	0,88
Lecture notes/copies of transparencies	M2	539	3,39	1,05	34	3,47	0,71
Exercise questions and answers	M3	523	3,33	1,07	34	3,62	1,02
Example programs	M4	551	4,19	0,86	34	4,24	0,65
Still pictures of programming structures	M5	490	3,15	1,00	30	3,70	0,75
Interactive visualizations	M6	315	3,33	1,03	27	4,07	0,87

Tabelist 3 saab järeldada, et üliõpilased hindasid kõige keerulisemaks ülesandeks programmeerimise õppimisel enda programmikoodist vigade leidmist (3,28/5,0). Samuti seda, et programmeerimist eelistatakse õppida üksinda (3,79/5,0) ja töötades läbi kursuse materjale (3,98/5,0). Veel näeme, et kõige kasulikumaks abimaterjaliks programmeerimise õppimisel on programminäited (4,19/5,0) [16].

1.4 Varasemad veateadete materjalid

Käesolev jaotis annab ülevaate programmeerimiskeele Python veateadete kohta tehtud sarnastest õppematerjalidest ja juhenditest, mis annavad ideid magistritöö kolmandas peatükis pikemalt kirjeldatava loodava õppematerjali struktuuri kohta.

1.4.1 Veateadete seletused programmeerimiskursuse abivahendina

California Ülikooli arvutiteaduse osakonna (Center for Computing Education and Diversity in the Universtiy of California) uurimisrühm on loonud veebilehel asuva materjali levinumate veateadete kohta, mis koosneb 17 erinevast veateatest ja hõlmab 11 erinevat veatüüpi [17]:

- *AttributeError- 'str' object has no attribute 'length'*
- *IndentationError- unexpected indent*
- *IndexError string index out of range*
- *IOError: [Errno 2] No such file or directory: 'foo.txt'*
- *IOError: File not open for writing*
- *KeyError- 'a'*
- *NameError: global name 'hello' is not defined*
- *RuntimeError- Set changed size during iteration*
- *SyntaxError invalid syntax*
- *SyntaxError- EOL while scanning string literal*
- *TypeError 'str' object does not support item assignment*
- *TypeError- 'dict' object is not callable*
- *TypeError- expected a character buffer object.xml*
- *TypeError- range() integer end argument expected, got list.*
- *TypeError- unsupported operand type(s) for +- 'int' and 'str'*
- *UnboundLocalError- local variable 'num' referenced before assignment*
- *ZeroDivisionError- integer division or modulo by zero*

Materjali juures on ka tagasiside andmise võimalus, kuhu soovitakse neid veateateid, mida materjalis kirjeldatud veel pole.

Iga veateate puhul on kirjeldatud lühidalt veatüüpi, toodud välja täpne veateade ja kuvatud programminäidet, mis viga põhjustab. Joonisel 7 on kuvatud näiteks indeksiviga, kus kasutaja püüab printida ekraanile elementi indeksiga 11, kuid elementidest koosneva sõne indeksite pikkus on nullist kümneni. Veateade kuvatakse sisuga *IndexError: string index out of range*, mis viitab sellele, et on tehtud indeksiviga ületades lubatud piire.

Error: IndexError: string index out of range

IndexErrors happen when you try to reference an index that doesn't exist for that sequence. Sequences such as lists and strings have indices.

Example: full error message

```
Traceback (most recent call last):
  File "helloworld.py", line 5, in module
    main()
  File "helloworld.py", line 3, in main
    print foo[11]
IndexError: string index out of range
```

Broken Example Code:

```
1       def main():
2          foo = "hello world"
3          print foo[11]
4
5       main()
```

Joonis 7. Näide programmeerimiskeele Python indeksiveast koos programminäitega [17].

Vigase programmilõigu juurde on toodud ka üldine seletus, miks viga tekkis. Eelpool toodud näite puhul selgitatakse kasutajale, et elementi, mida kasutaja indeksiga otsib, ei eksisteeri. Indeksite loendamine algab programmeerimiskeeles Python nullist ja lõpeb sõne pikkus miinus üks kohal:

„You're trying to access an element of the string that doesn't exist. String indices start at 0, and extend until the length of the string minus one.“

Materjal pakub ka võimalikku lahendust eeldades, et kuna indeksiga 11 elementi sõnes ei leidunud ja sõne pikkus on 11 sümbolit, siis kasutaja soovib printida ekraanile sõne viimast elementi (Joonis 8).

Fixed Example Code:

```
1       def main():
2          foo = "hello world"
3          print foo[10]
4
5       main()
```

Joonis 8. Näide programmeerimiskeele Python parandatud programminäitest [17].

Järgides indekseerimise loogikat tuleb kasutada Joonise 8 näitel indeksit 10, et mitte ületada lubatud piire [17].

1.4.2 Veateadete mõistmise juhend algajatele

Bostoni Ülikooli (Boston University) lektor John Magee on koostanud programmeerimise algõppe kursuse raames juhendi programmeerimiskeele Python veateadete mõistmise ja nende alusel vigade parandamise tarbeks. Veebilehel kättesaadav materjal on mõeldud eelkõige arvutiteadust mitte põhierialana õppivatele üliõpilastele. Materjalis on kirjeldatud süntaksivigu, erindeid ja loogikavigu. Toodud on välja erinevaid programminäiteid ja veateateid, samuti on kirjeldatud vea tekkimise põhjust ja võimalikku parandamisviisi (Joonis 9).

Name Error

This will be a common error you encounter. It will give you a Traceback message like this:

```
Traceback (most recent call last):
  File "C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py", line 7, in
    main()
  File "C:/Users/John/Documents/Teaching-BU/Python-debugging/test.py", line 5, in main
    print hello
NameError: global name 'hello' is not defined
```

These messages can seem hard to understand at first. The first part tells you which file had the error. In the example above, the file is *test.py* and the error occurs on line 7. The next line shows the actual line of code where the error occured. This line executes the `main()` function. Similarly, the next two lines say that the error occurred on line 5, within `main` , and that the line with the error is `print hello`. Lastly, the actual NameError says that *global name 'hello' is not defined*.

A NameError means that Python tried to use a variable or function name, such as `hello` based on a previous definition. If it hasn't been defined at this point, you get the error.

Usual Causes:

- A mistyped variable or function name.
- Using a variable before it is defined.
- The name was intended to be enclosed in quotes.

Examples:

```
print hello
pring "hello"
```

The following example leaves the s off dollars in the second line:

```
dollars = input("Enter dollars: ")
print "You have %d dollars" % dollar
```

Solution:

Usually, you can look at the last line mentioned in the TraceBack error. Place your cursor within idle and move it until you are on the correct line as indicated by the *Ln:* indicator in the bottom right of the editor. You should find the specific line quoted by the error message. In the case of a NameError, you can check if you typed the variable or function name correctly, if it should be in quotes, or if you should have defined it somewhere prior to the line.

Joonis 9. Näide materjalis olevast programmeerimiskeele Python veateatest *NameError: global name 'hello' is not defined* [18].

Joonisel 9 on kuvatud nimevea näide materjalist, mis sisaldab konkreetset veateadet koos selgitusega, levinumaid vea tekke põhjuseid, näiteid ja võimalikke vea parandusviise [18].

1.5 Programmeerimise algkursused Tartu Ülikoolis

Tartu Ülikool pakub veebipõhiseid programmeerimiskursusi, mis on mõeldud neile, kel programmeerimisega varasem kokkupuude vähene või puuduv. E-kursused on MOOC tüüpi ehk vaba ligipääsuga ja suure osalejaskonnaga kursused. Osalejate teadmisi kontrollitakse valikvastustega testide, foorumite ja lahendatud ülesannete puhul automaatselt kontrollitavate programmide näol [19].

1.5.1 E-kursus *Programmeerimisest maalähedaselt*

Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühma korraldatud *Programmeerimisest maalähedaselt* e-kursus toimus esmakordselt pilootkursusena 2014/2015. õppeaasta sügissemestril (01.12.2014 – 28.12.2014), kus osales 32 e-kursuslast. Juba märksa suuremalt reklaamiti kevadsemestril (09.03.2015 – 05.04.2015) uuesti toimuvat MOOC stiilis e-kursust, kus kõrget huvi programmeerimise õppimise vastu näitas 638 e-kursuslase osalemine. Käesoleva hetke (02.01.2017) seisuga on lisaks eelpool mainitud toimumisaegadele korraldatud e-kursust veel kolmel korral – 2015/2016. õppeaasta sügissemestril (05.10.2015 – 01.11.2015), 2015/2016. õppeaasta kevadsemestril (07.03.2016 – 03.04.2017) ja 2016/2017. õppeaasta sügissemestril (10.10.2016 – 07.11.2016), kus osales vastavalt 1534, 1430 ja 1792 osalejat. E-kursusel tutvutakse programmeerimise põhimõistete ja konstruktsioonidega. Vaatluse all on järgmised teemad:

- Algoritm
- Programm
- Muutuja
- Andmetüübid
- Tingimuslause
- Sõned
- Kilpkonnagraafika
- Tsükkel
- Regulaaravaldis
- Funktsioonid
- Andmevahetus
- Failid

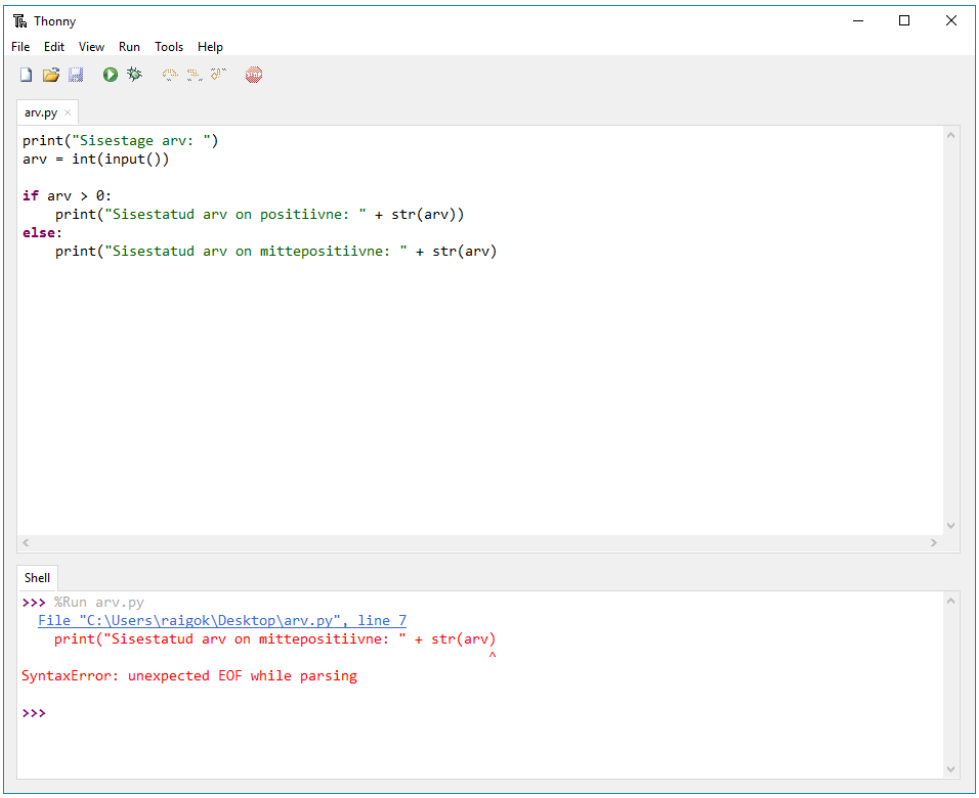
Kursusel osalejatelt eeldatakse 26 tundi iseseisvat tööd nelja nädala vältel, mille jooksul tuleb esitada kaks kohustuslikku ülesannet nädalas ning sooritada e-kursuse lõpufaasis ka lõputest [1] [3].

1.5.2 E-kursus *Programmeerimise alused*

Suur huvi programmeerimise vastu ajendas Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühma looma veel ühte MOOC stiilis e-kursust, mille nimeks sai *Programmeerimise alused*. Pilootkursusel, mis toimus 2015/2016. õppeaasta kevadsemestri esimeses pooles (11.01.2016 – 06.03.2016), osales 295 e-kursuslast. E-kursuse teisel toimumise korral, 2015/2016. õppeaasta kevadsemestri teises pooles (28.03.2016 – 22.05.2016), oli end registreerinud programmeerimist õppima juba 1770 huvilist. E-kursuse raames käsitletakse programmeerimise teemasid põhjalikumalt kui *Programmeerimisest maalähedaselt* e-kursusel ja lisaks tutvustatakse programmeerimise huvilistele järjendeid, kasutajaliideseid ja graafi-karaamistikku tkinter. E-kursus *Programmeerimise alused* on ka mahult suurem ning eeldab kursusel osalejalt 78 tundi iseseisvat tööd kaheksa nädala vältel, kusjuures igal nädalal on kohustuslik esitada neli ülesannet ja sooritada e-kursuse lõppedes lõputest [1].

1.6 Programmeerimiskeskkond Thonny

Thonny on Tartu Ülikooli e-kursustel kasutuses olev vabavaraline programmeerimiskeskkond programmeerimiskeele Python õppimiseks ja õpetamiseks (Joonis 10).

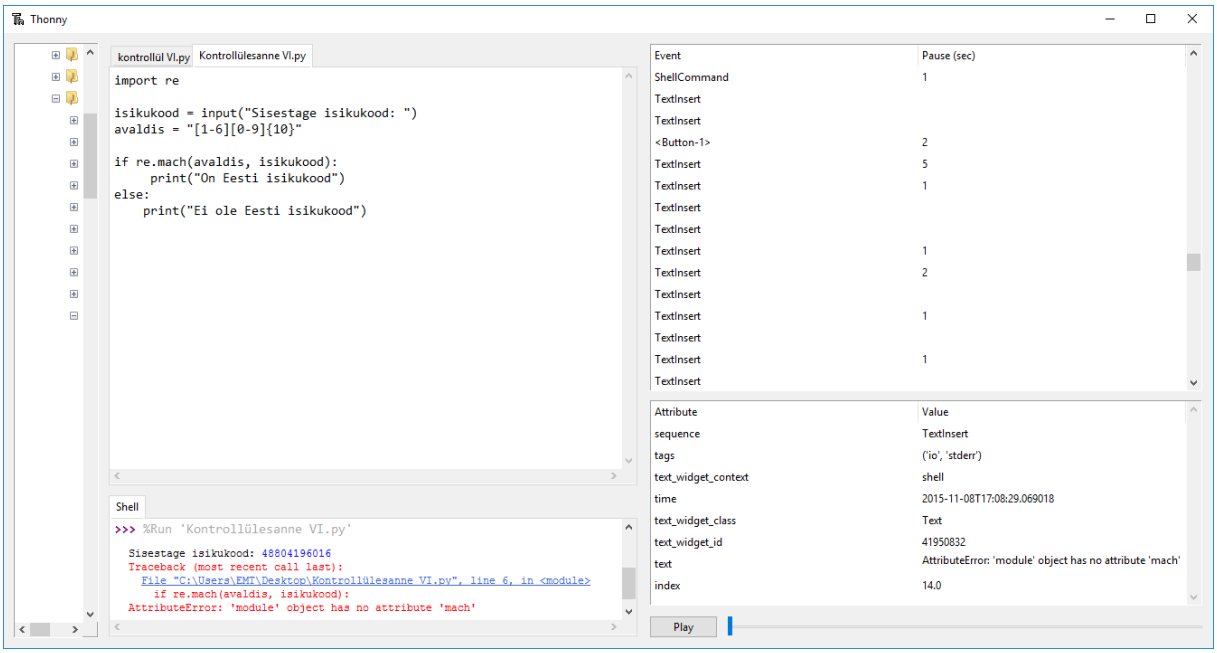


Joonis 10. Programmeerimiskeskkonna Thonny kasutajaliides.

Tegemist on vabavaralise takrvaraga, mille installeerimise kohta on koostatud juhised nii operatsioonisüsteemi Windows, Mac OS kui ka Linux tarbeks [20].

Võrreldes programmeerimiskeele Python peamise programmeerimiskeskkonnaga IDLE (*Integrated Development Environment*), on Thonny eeliseks see, et käsurida asub koodiredaktoriga samas vaates ja on seega lihtsamini jälgitav. Lisaks kuvab Thonny kõik avatud programmikoodid eraldi vahekaartides, mitte ei tekita eraldi programmiaknaid [21]. Thonny üheks oluliseks funktsionaalsuseks on logifaili loomine, mis salvestab JSON (*JavaScript Object Notation*) formaadis programmi loomise käigus kasutaja poolt tehtavad sammud. Thonny programmilogidest on pikemalt juttu peatükis 2.2.3.2 (Thonny logifailide analüüsimine).

Programmeerimiskeskkond Thonny võimaldab veel logifaili alusel kasutussessiooni taasesitada kasutaja enda poolt valitud sobival kiirusel, et leida paremini üles tehtud vigu ja neid parandada (Joonis 11).



Joonis 11. Thonny kasutussessiooni taasesitamise kasutajaliides.

2 Programmeerimiskeele Python veateadete uuring programmeerimise algõppes

Käesolev peatükk annab ülevaate programmeerimiskeele Python veateadete uuringu kohta kirjeldades uuringu kahte suuremat etappi, selles osalejaid, ja tutvustades erinevaid vahendeid, mida uuringu läbiviimisel kasutatakse. Pikemalt kirjeldatakse veebipõhist küsitlust ja Thonny logifailide kogumist. Lõpptulemusena analüüsitakse uuringu tarbeks kogutud tagasisidet programmeerimiskeele Python veateadete kvaliteedi ja esinemissageduse kohta.

Uuringu eesmärgiks on selgitada välja kompilaatori poolt edastatavate veateadete mõistmine ja kasulikkus programmeerimise algõppe e-kursuslaste hulgas. Selle tarbeks kogutakse kasutajatelt arvamusi veateadete kohta küsitluse teel ja tehakse statistikat veateadete esinemissageduse kohta programmilogide analüüsimise tulemusena.

E-kursuse *Programmeerimisest maalähedaselt* osalejatele saadeti e-maili teel kutse osaleda programmeerimiskeele Python veateadete uuringus. Nõusoleku osalemise kohta andis 15 kursusel osalejat, kellel paluti peale iga programmeerimisülesande lõppu täita veebis kättesaadav küsitlus ja laadida õppekeskkonda Moodle üles Thonny logifailid. Uuringus osalevate e-kursuslaste nimesid ja muud tundlikku informatsiooni käesolevas magistritöös ega ka mujal ei avalikustata.

2.1 Küsitlus e-kursusel osalejatele

Veateadete uurimisrühma osalejate seas viidi läbi *online*-küsitlus, et saada informatsiooni selle kohta, kas ja kuidas saaks programmeerimiskeele Python veateateid modifitseerida, et need paremini arusaadavamad oleks. Küsitlus koostati Google Forms vahenditega ja koosnes lisaks kahele sissejuhatavale küsimusele kümnest küsimusest, millest üheksale (kohustuslikule küsimusele) sai vastata valikvastusega skaalal 1–7, kus 1 tähistas täielikku mittenõustumist ja 7 täielikku nõustumist, ning üks (vabatahtlik) küsimus eeldas pikemat vabas vormis vastust.

Lisaks küsimustele sisaldas küsitlus kommentaaride lahtrit, kuhu kasutaja sai soovi korral enda mõtteid veateadete osas jagada.

Küsitluse avalehel kuvati kasutajale lühikirjeldus veateadete uuringu kohta koos küsitluse koostaja kontaktandmetega ja kohustuslikku küsimust, et teada saada, kas küsitlusele vastatakse esimest või mitmendat korda. Selle küsimuse vastusest sõltub, millisesse küsitluse harusse kasutaja hiljem edasi suunatakse, kuna üldisemate küsimuste puhul (näiteks keeleoskus) piisab ühekordsest vastamisest. Järgmine leht sisaldab samuti ühte kohustuslikku küsimust, et saada aimu, kas kasutaja puutus kokku programmeerimiskeele Python veateadetega vähemal kui kolmel korral, kolmel või rohkemal korral, või lahendas programmeerimisülesannet nii, et veateateid ei ilmnenudki. Juhul kui kasutaja vastab, et veateadetega kokkupuude puudus, siis käesoleva uuringu raames rohkem küsimusi temale ei esitatud. Veateadetega kokkupuutunud kasutajatelt küsiti kolmandal ja ühtlasi ka küsitluse viimasel lehel 9 kohustuslikku küsimust ja 1 vabatahtlik küsimus. Küsimused olid järgnevas järjekorras:

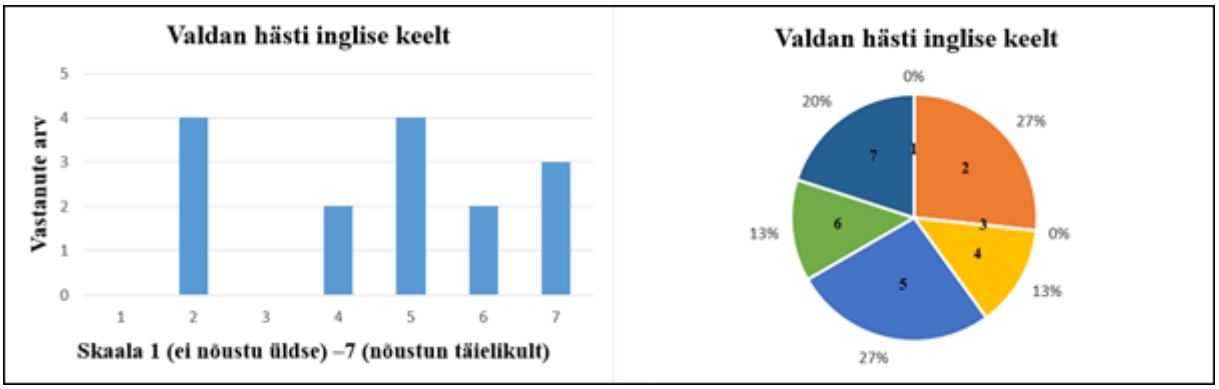
- Valdan hästi inglise keelt. (juhul, kui vastatakse esimest korda)
- Pythoni veateated olid mulle lihtsasti loetavad/arusaadavad.
- Pythoni veateated oleksid võinud olla pikemalt lahtiseletatud.
- Pythoni veateated sisaldasid liialt tehnilisi detaile.
- Pythoni veateated oleksid võinud olla eesti keeles. (juhul, kui vastatakse esimest korda)
- Pythoni veateated selgitasid mulle arusaadavalt vea tekkimise põhjust.
- Kuidas käitusin peale veateate ilmnemist? (vabatahtlik küsimus)
- Pythoni veateated olid mulle abiks vea parandamisel.
- Kardan programmeerimisel vigu teha. (juhul, kui vastatakse esimest korda)
- Veateadete ilmnemine mõjutas minu motivatsiooni ülesande lahendamisel.

2.2 Küsitluse tagasiside e-kursusel osalejatelt

Küsitlusele vastas kokku 15 kursusel osalejat, kellest 5 olid meessoost ja 10 naissoost isikud. 14 e-kursuslast puutus veateadetega kokku kolmel või rohkemal korral, 1 kursusel osaleja vähem kui kolmel korral. Veateateid täielikult vältida ei õnnestunud ühelgi veateadete uuringus osalenud e-kursuslasel.

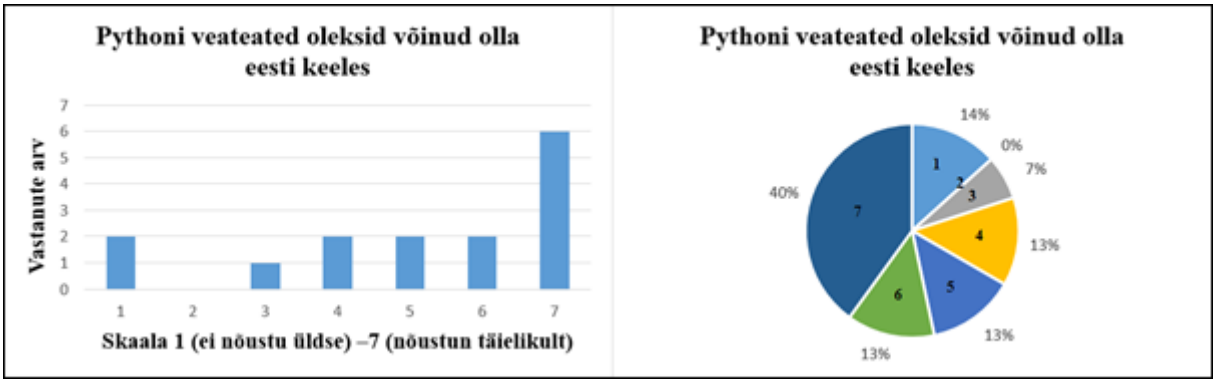
Üks peamisi eesmärke küsitluse koostamisel oli selgitada välja, kas eesti keelt emakeelena kõnelevatel kursusel osalejatel on tekkinud raskusi programmeerimiskeele Python veateadete mõistmisel, kuna need on kirjeldatud inglise keeles. Joonis 12 kirjeldab e-kursuslaste inglise

keele oskust järgmiselt: 27% kursusel osalejatest hindab enda inglise keele oskust madalamaks kui neli palli seitsmest, 60% osalejatest kõrgemaks kui neli palli seitsmest. 13% arvas, et nende inglise keele tase on keskmine.



Joonis 12. Inglise keele oskuse hindamine skaalal 1–7.

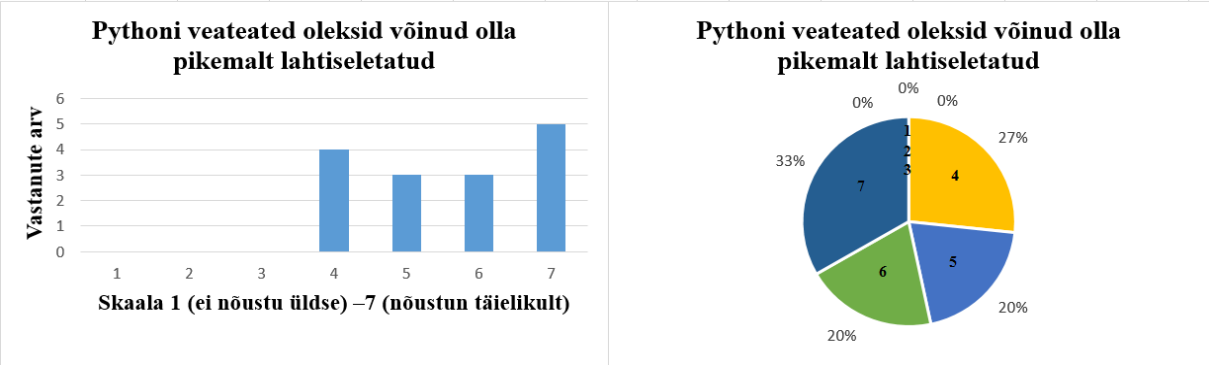
Jooniselt 13 saab järeldada, et enamik kursusel osalejaid (66% vastanutest) soovib veateateid lugeda eesti keeles ja 21% inglise keeles. 13% vastanud e-kursuslastest on rahul mõlema variandiga.



Joonis 13. Programmeerimiskeele Python veateadete tõlkimise vajaduse hindamine skaalal 1–7.

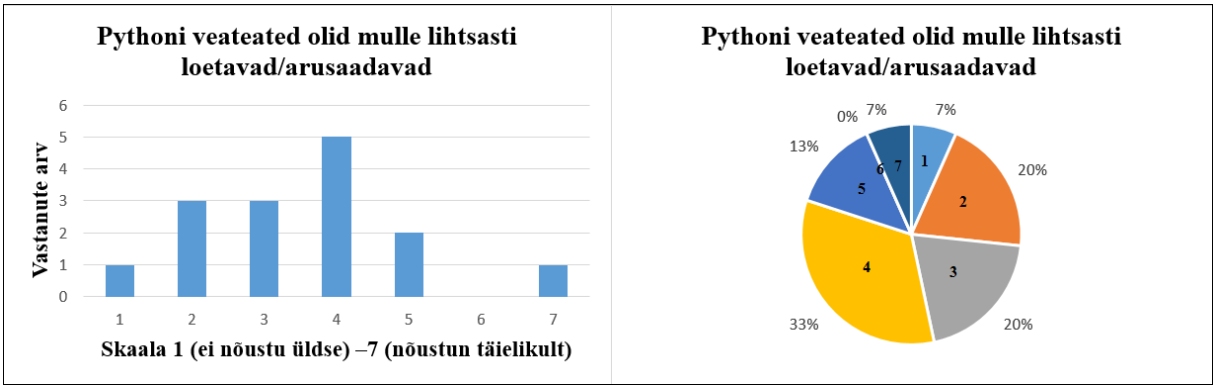
Kursusel osalejatelt uuriti veel arvamust programmeerimiskeele Python veateadete kirjelduse pikkuse, loetavuse ja arusaadavuse kohta. Selgus, et tuntakse puudust veateadete pikemast lahtiseletamisest. Kuna programmeerimiskeele Python sisseehitatud veateated kasutavad veateate lühivormi stiili, mida kasutab ka e-kursustel kasutusel olev programmeerimiskeskkond Thonny, siis Jooniselt 14 näeb, et tervelt 73% küsitlusele vastanud e-kursuslastest on arvamusel, et lühikirjeldus pole piisav veateate lahtiseletamisel. Sealjuures 27% kursusel osalejatest

ei oma kindlat arvamust sel teemal. Ükski e-kursuslane pole arvamusel, et veateated ei peaks olema pikemalt lahtiseletatud, kui nad vaikimisi on.



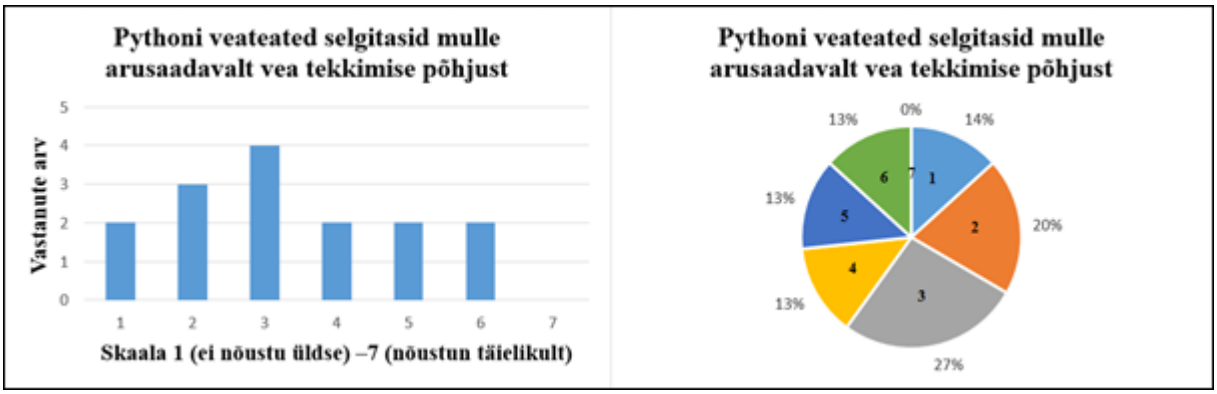
Joonis 14. Programmeerimiskeele Python veateadete pikema lahtiseletamise vajadus skaalal 1–7.

Veateadete loetavuse ja arusaadavuse kohta saab järeldusi teha Jooniselt 15 kus kõige populaarsem vastus jääb täpselt skaala keskele. 20% vastanutest hindab programmeerimiskeele Python veateateid lihtsasti loetavateks ja arusaadavateks, samas 47% arvates on olukord vastupidine.



Joonis 15. Programmeerimiskeele Python veateadete loetavus/arusaadavus skaalal 1–7.

Veel uuriti kursusel osalejatelt programmeerimiskeele Python veateadete selgituse kvaliteeti vea tekkimise põhjuse välja selgitamisel. Joonis 16 annab ülevaate, et 26% e-kursuslaste arvamustest langeb graafiku paremale poolde ehk nõusoleku osale ja 61% arvamustest graafiku vasakule poolde ehk mittenõustumise osale.



Joonis 16. Programmeerimiskeele Python veateadete selgitus vea tekkimise põhjuse mõistmisel skaalal 1–7.

Tagasiside põhjal saab järeldada, et enamik e-kursusel osalejaid hindab enda inglise keele oskust keskmisest paremaks, kuid samas soovib veateateid eestikeelsetena. Pea kolmveerand e-kursusel osalejaid soovib pikemat seletust veateatele, mis suurendab vea loetavust, arusaamist ja tekke põhjuse paremat mõistmist.

2.3 Thonny logid e-kursusel osalejatelt

Selles peatükis kirjeldatakse meetodeid ja abivahendeid, millega juhendati e-kursuslasi salvestama ja üleslaadima Thonny logifaile. Samuti antakse ülevaade logifailide analüüsimiseks kasutatud vahenditest, mille tulemusena valmib veateadete abimaterjali loomiseks vajalik statistika.

Thonny salvestab automaatselt ajatempliga eristatavad programmilogid lokaalsele kõvakettale sisselogitud kasutaja profiili alla *.thonny/user_logs* kausta .txt-laiendiga tekstifailina. Veateadete uurimisrühma osalejatelt paluti Thonny programmilogisid, et koguda infot selle kohta, mis tüüpi ja kui palju veateateid konkreetse programmeerimisülesande lahendamisel tekkis. Logifaile laekus e-kursustelt *Programmeerimisest maalähedaselt* (10 logifaili) ja *Programmeerimise alused* (15 logifaili). E-kursusel osalejatele valmistati ette teksti- ja videopõhine juhend, kuidas Thonny logifaile lokaalselt kõvakettalt üles leida, kokkupakkida ühtseks failiks ja õppekeskkonda Moodle õige programmeerimisülesande sektsiooni alt serverisse üleslaadida [22][23].

2.3.1 Video- ja tekstipõhine juhend Thonny logifailide üleslaadimiseks

Tekstipõhine juhend programmeerimiskeskkonnaga Thonny salvestatud logifailide üleslaadimise kohta on kättesaadav e-kursusel osalejatele Google Drive pilvepõhises teenuses (*cloud storage service*) [24]. Juhend koosneb seitsmest A4 formaadis leheküljest ja kirjeldab detailselt vajalikke samme Thonny logifaili üleslaadimiseni (koos illustreerivate näidetega) õppekeskkonda Moodle veateadete uuringu katserühma teema alla. [23].

Videopõhise juhendi koostamisel kasutati Camtasia tarkvara, mis võimaldas nii salvestada arvutiekraanil tehtavad toimingud kui ka sünkroniseerida need helifailiga luues 7 minutilise ja 10 sekundilise pikkusega videofaili, mis e-kursusel osalejatele kättesaadav videokeskkonnas YouTube [22][25]. Arvutiekraanil toimuva jäädvustamiseks kasutati programmi Camtasia Recorder (versioon 8.0.4), mis võimaldas salvestada kogu ekraani ulatuses pilti (Joonis 17).

Joonis 17. Camtasia Recorder tarkvara kasutajaliides [25].

Camtasia Recorder pakub lisaks ka võimalust salvestada arvuti veebikaamerast tulevat pilti ja mikrofoni ühendamisel helisalvestust [25]. Käesoleva töö tarbeks loodud videopõhise juhendi jaoks kasutati ainult ekraanisalvestuse funktsiooni, kuna video autori pildi näitamine veebi-kaamerast ei olnud vajalik ja heli salvestamiseks kasutati parema kvaliteedi saavutamiseks digitaalset stereodiktofoni Zoom H1 Handy Recorder.

Lõpptulemus jäädvustati .camrec-laiendiga faili, mille töötlemiseks sobis samuti Camtasia poolt pakutav tarkvara Camtasia Studio (versioon 8.0.4) (Joonis 18).

Joonis 18. Camtasia Studio tarkvara kasutajaliides [26].

Tarkvara Camtasia Recorder poolt salvestatud videofaili monteerimiseks kasutati Camtasia Studio vahendeid, millega sünkroniseeriti ekraanisalvestus ja helifail ning salvestati tulemus AVI (*Audio Video Interleaved*) videoformaati [26].

2.3.2 Thonny logifailide analüüsimine

Thonny logifailid on salvestatud formaati JSON formaati ja sisaldavad palju detailselt infot programmi looja poolt tehtud sammude kohta. Programmeerimiskeskkond Thonny salvestab kasutaja iga tegevuse täpses järjekorras koos kirjelduse ja ajatempliga formaadi JSON objektiks. Käesoleva uuringu jaoks kõige olulisem informatsioon logifailis on programmi loomise vältel tekkinud veateated, mida on võimalik tänu kindlale formaadile lihtsasti välja selekteerida ja kokku lugeda (Joonis 19).

```
{
    "sequence": "TextInsert",
    "tags": "('io', 'stderr')",
    "index": "6.0",
    "time": "2015-11-08T17:06:05.325796",
    "text_widget_id": 41950832,
    "text_widget_class": "Text",
    "text": "    if re.mach(avaldis):\n",
    "text_widget_context": "shell"
},
{
    "sequence": "TextInsert",
    "tags": "('io', 'stderr')",
    "index": "7.0",
    "time": "2015-11-08T17:06:05.325796",
    "text_widget_id": 41950832,
    "text_widget_class": "Text",
    "text": "AttributeError: 'module' object has no attribute 'mach'\n",
    "text_widget_context": "shell"
},
{
    "sequence": "TextInsert",
    "tags": "('toplevel', 'prompt', 'vertically_spaced')",
    "index": "8.0",
    "time": "2015-11-08T17:06:05.351798",
    "text_widget_id": 41950832,
    "text_widget_class": "Text",
    "text": ">>> ",
    "text_widget_context": "shell"
},
```


Joonis 19. Näide Thonny logifailis sisalduvast programmeerimiskeele Python veateatest: *AttributeError: 'module' object has no attribute 'mach'*\n.

Thonny programmilogide analüüsimiseks kasutati operatsioonisüsteemi UNIX käske, et filtreerida välja ja sorteerida veateated ülejäänud informatsioonist programmilogist.

- Logifailide lugemiseks kasutati standardkäsku *cat*:
  `cat [/teekond logifailide kataloogini/]*.txt`

- Kõikide veateadete kättesaamiseks kasutati standardkäsku *grep*:
  `grep '\"text\"\: \"[[:upper:]][[:lower:]]*Error\:'`

- Ebavajalike sümbolite (jutumärgid, koolonid, tühikud) eemaldamiseks kasutati standardkäsku *sed*:
  `sed 's/\"text\"\: \"\(.*\)$/\1/g'`
  `sed 's/[\][n]["],//g'`
  `sed 's/[\][n]["]//g'`
  `sed 's/^        //g'`

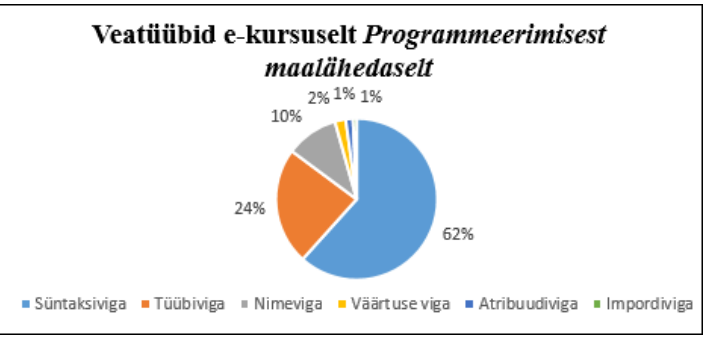
- Veateadete sorteerimiseks ja esinemissageduse koostamiseks kasutati standardkäske *sort* ja *uniq*:

```
sort | uniq -c | sort -nr
```

Operatsioonisüsteemi UNIX käske rakendati eraldi *Programmeerimisest maalähedaselt* ja *Programmeerimise alused* e-kursuselt laekunud logifailidele [27].

E-kursuse *Programmeerimisest maalähedaselt* puhul osutus kõige levinumaks veatüübiks süntaksiviga olles esindatud logifailides 682 korral, mis on 62% kõigist veateadetest (Lisa 1). Tüübiviga ilmnes e-kursusel osalejatel 260 korral ehk 24% juhtudel. Nimeviga esines 10%, teisi veatüüpe (nimeviga, väärtuse viga, atribuudiviga ja impordiviga) esines vähem kui 5% kõigist veateadetest (Joonis 20).

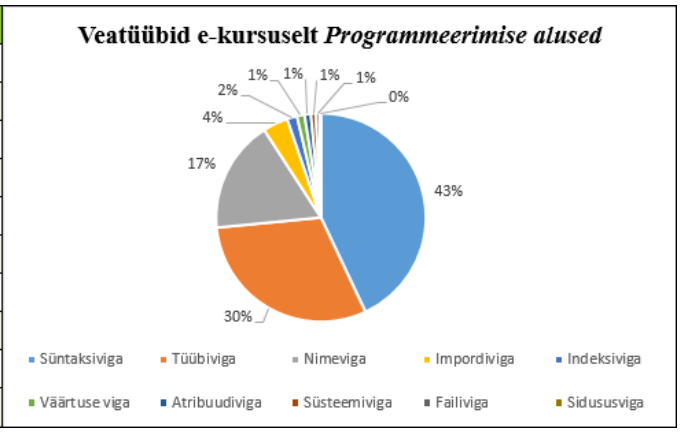
Vea tüüp	Esinemissagedus
Süntaksiviga	682
Tüübiviga	260
Nimeviga	116
Väärtuse viga	24
Atribuudiviga	16
Impordiviga	8



Joonis 20. E-kursusel *Programmeerimisest maalähedaselt* esinenud veatüüpide esinemissagedustabel (koos graafikuga).

E-kursuse *Programmeerimise alused* veateadete esinemissagedustabeli kolm esimest veatüüpi on samas järjekorras, mis *Programmeerimisest maalähedaselt* e-kursuselgi. Protsentuaalselt esineb süntaksiviga, tüübiviga ja nimeviga vastavalt 43%, 30% ja 17% kõigist e-kursuslastel ilmnenud veateadetest (Lisa 2). Vähem kui 5% puututi kokku järgmiste veateadetega: impordiviga (tkinter-moodul), indeksiviga, väärtuse viga, atribuudiviga, süsteemiviga, failiviga ja sidususviga (Joonis 21).

Vea tüüp	Esinemissagedus
Süntaksiviga	222
Tüübiviga	157
Nimeviga	90
Impordiviga	20
Indeksiviga	8
Väärtuse viga	6
Atribuudiviga	5
Süsteemiviga	4
Failiviga	3
Sidususviga	1



Joonis 21. E-kursusel *Programmeerimise alused* esinenud veatüüpide esinemissagedustabel (koos graafikuga).

Jooniselt 20 ja Jooniselt 21 võib näha, et erinevaid veatüüpe leidus Thonny logifailides vastavalt 6 (*Programmeerimisest maalähedaselt*) ja 10 (*Programmeerimise alused*), millest sagedasemad olid süntaksiviga, tüübiviga ja nimeviga. *Programmeerimisest maalähedaselt* e-kursusel esinenud veatüübid arvati kõik järgmises peatükis pikemalt kirjeldatava veateadete abimaterjali hulka. E-kursuse *Programmeerimise alused* puhul ei võetud arvesse veatüüpe, mis esinesid vähem kui 5 korda, kuna nende esinemissagedus on väga madal.

3 Programmeerimiskeele Python veateadete õppematerjali loomine

Käesolev peatükk kirjeldab veateadete õppematerjali loomist, tagasiside kogumise protsessi ja materjali täiendamist laekunud informatsiooni põhjal. Lühidalt kirjeldatakse ka materjali kasutamist edaspidiselt.

3.1 Õppematerjali kirjeldus

Veateadete õppematerjali eesmärgiks on olla abiks programmeerimiskeele Python veateadete mõistmisel ja tekkinud programmivigade parandamisel. Materjal on mõeldud kasutamiseks eel-kõige programmeerimise algõppe e-kursuslastele. Veateadete materjali loomisel võeti aluseks veateadete uuringus osalejatelt saadud tagasiside nii küsitluse kui Thonny logifailide näol - inspiratsiooni koguti ka varasemalt loodud sarnastelt veateadete veebilehtedelt. Õppematerjali struktuuri koostamisel vaadeldi eelnevalt koostatud veateadete materjale.

Nii California Ülikooli arvutiteaduse osakonna poolt loodud programmeerimiskeele Python veateadete seletuste veebilehelt kui ka Bostoni Ülikooli poolt loodud veateadete juhendist võeti idee kuvada iga veateate puhul lühikirjeldus, vigase programmi näide, vea tekke põhjus ja parandatud programminäide [17][18]. Algselt oli plaanis õppematerjalis kuvada veateadet põhjustanud rida teist värvi, kuid tuginedes veateadete stiilide uuringule, siis Tabelist 2 paistab see kõige rohkem segadust tekitav veateate stiil, mistõttu ei kasutata seda loodavas materjalis [15]. Veateadete materjali loomise kasuks eelpool mainitud struktuuriga saab kinnitust vaadates programmeerimise õppimise ja selle õpetamise kitsaskohtade kohta tehtavat uuringut (Tabel 3), kus jõuti tulemuseni, et üliõpilased hindasid kõige keerulisemaks ülesandeks program-meerimisel iseenda vigadest õppimist. Siinkohal osutub veateadete õppematerjal kasulikuks neile, kes eelistavad õppida üksinda ja neile, kes soovivad materjalist leida programminäiteid, sest loodav programmeerimiskeele Python veateadete õppematerjal erinevaid näiteid ohtralt sisaldab [16].

Materjal on veebis kõigile kättesaadav lisatuna e-kursuse *Programmeerimise alused* kodulehele eraldi sektsiooni *Veateated* alla [5].

Veateadete õppematerjal koosneb sissejuhatavast lehest ja seitsmest veatüübi alamlehest, mis omakorda jagunevad 27 erinevaks veateate kirjelduseks. Materjali sisu koosneb programmeerimise algõppe levinumate veatüüpide eestikeelsest kirjeldusest ja illustreerivatest näidetest - iga veateate tüüp asub eraldi lehel ja selle lõppu on lisatud ka väike küsitlus (Lisa 3). Kõik programminäited on programmeerimiskeskkonnaga Thonny käivitatavad - nii vigased näited kui ka ühe võimaliku variandina parandatud näited [5]. Materjal käsitleb järgmisi veatüüpe ja -teateid:

1. veatüüp. Atribuudiviga

1.1 *AttributeError: 'list' object has no attribute 'split'*

2. veatüüp. Impordiviga

2.1 *ImportError: no module named 'Tkinter'*

3. veatüüp. Indeksiviga

3.1 *IndexError: list assignment index out of range*

3.2 *IndexError: list index out of range*

4. veatüüp. Nimeviga

4.1 *NameError: name 'summa' is not defined*

5. veatüüp. Süntaksiviga

5.1 *SyntaxError: 'break' outside loop*

5.2 *SyntaxError: 'return' outside function*

5.3 *SyntaxError: EOL while scanning string literal*

5.4 *SyntaxError: expected an indented block*

5.5 *SyntaxError: invalid syntax*

5.6 *SyntaxError: missing parentheses in call to 'print'*

5.7 *SyntaxError: unexpected EOF while parsing*

5.8 *SyntaxError: unexpected indent*

5.9 *SyntaxError: unindent does not match any outer indentation level*

6. veatüüp. Tüübiviga

6.1 *TypeError: 'int' object is not subscriptable*

6.2 *TypeError: 'list' object is not callable*

6.3 *TypeError: can't convert 'int' object to str implicitly*

6.4 *TypeError: can't convert 'NoneType' object to str implicitly*

6.5 *TypeError: can't multiply sequence by non-int of type 'float'*

6.6 *TypeError: float() argument must be a string or a number, not 'list'*

6.7 *TypeError: list indices must be integers or slices, not str*

6.8 *TypeError: unorderable types: str() > int()*

6.9 *TypeError: unsupported operand type(s) for +: 'int' and 'str'*

7. veatüüp. Väärtuse viga

7.1 *ValueError: could not convert string to float*

7.2 *ValueError: empty separator*

7.3 *ValueError: I/O operation on closed file*

7.4 *ValueError: invalid literal for int() with base 10: '7.5'*

Iga veateate puhul on kirjeldatud lühidalt veateate olemust ja kõige tõenäolisemat tekke põhjust [5]. Näiteks süntaksivea puhul:

Süntaksiviga SyntaxError tekib siis, kui programmis on tehtud õigekirjaviga, kus eksitud keelekonstruktsiooni vastu. Märgiga ^ tähistatakse veateates viga põhjustanud rea kõige varasemat punkti, kus viga programmis tuvastati.

Need kirjeldused on kokkupandud analüüsides Thonny programmilogisid ja leides sealt täpseid veateadet puudutavaid näiteid. Lisaks kirjeldusele koosneb iga konkreetne veateade ühest või kahest programminäitest (olenevalt veateate ulatuslikkusest) (Joonis 22).

```
1  print("Sisestage arv: ")
2  arv = int(input())
3
4  if arv > 0:
5      print("Sisestatud arv on positiivne: " + str(arv))
6  else:
7      print("Sisestatud arv on mittepositiivne: " + str(arv)
```

Joonis 22. Näide programmeerimiskeele Python programmist [5].

Ülal toodud programminäite puhul kuvab Thonny kasutajale veateate selle kohta, et käsu *print* lõpust on puudu lõpetav sulg (Joonis 23).

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 7
    print("Sisestatud arv on mittepositiivne: " + str(arv)
                                                          ^
SyntaxError: unexpected EOF while parsing
>>>
```

Joonis 23. Näide veateatest: *SyntaxError: unexpected EOF while parsing* [5].

Kasutajale parema ülevaate andmiseks tekkinud veateate kohta on materjali lisatud üks võimalik programmi parandusviis, mille rakendamisel enam veateadet kasutajale ei kuvata. Eelpool toodud vigase programminäite puhul soovitatakse kasutajal lisada puuduolev sulg *print* käsu lõppu, et tekitada sulgudele paarsust [5] (Joonis 24).

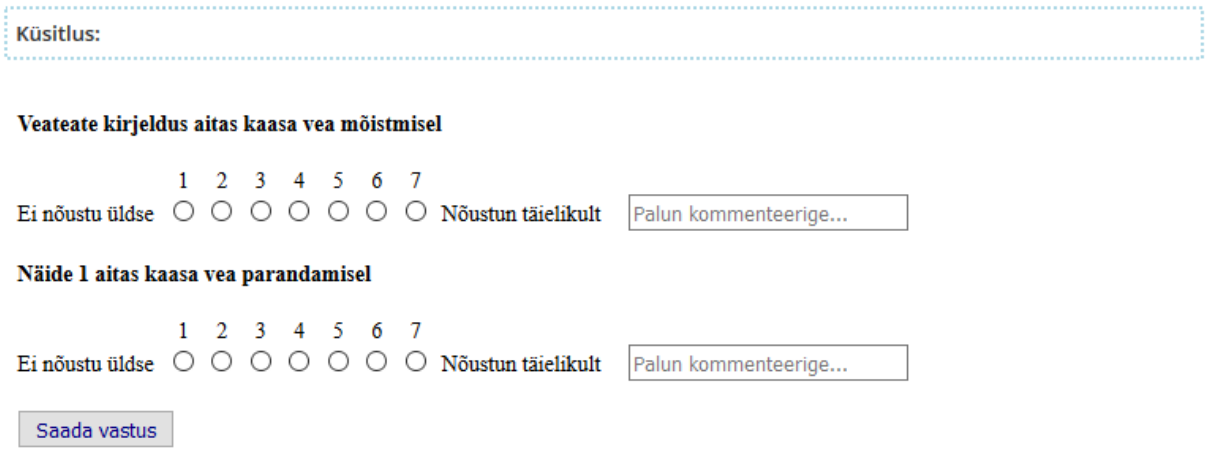
```
1  print("Sisestage arv: ")
2  arv = int(input())
3
4  if arv > 0:
5      print("Sisestatud arv on positiivne: " + str(arv))
6  else:
7      print("Sisestatud arv on mittepositiivne: " + str(arv))
```

Joonis 24. Näide programmeerimiskeele Python parandatud programmist [5].

Iga veateate lehekülje lõpus on ka lühike tagasiside vorm, millest pikemalt juttu järgmises peatükis.

3.2 Tagasiside õppematerjali kohta e-kursusel osalejatelt

Materjali kohta tagasiside saamiseks paigutati materjali veebilehel iga veateate kirjelduse ja näite kohta eraldi tagasiside vorm, kus saab hinnata veateadet skaalal 1–7 ning jätta vabas vormis kommentaare. Skaala madalaim hinne 1 tähistab absoluutselt mittenõustumist ja skaala kõrgeim hinne 7 täielikku nõustumist (Joonis 25).



Joonis 25. Tagasiside vorm veateate kirjelduse ja programminäite kohta [5].

Tagasiside väljade koostamisel kasutati HTML <*form*>-elementi, mille sees omakorda *radio*-tüüpi <*input*>-elementi, mis võimaldas tekitada valikvastustega küsimustiku. Kommentaaride jätmiseks lisati *text*-tüüpi <*input*>-element [28]. Vastuste töötlemiseks loodi PHP skript, mis loeb ja töötleb nii valikvastuseid kui tekstikujul antud vastuseid ja salvestab tulemused serveris asuvasse faili [29].

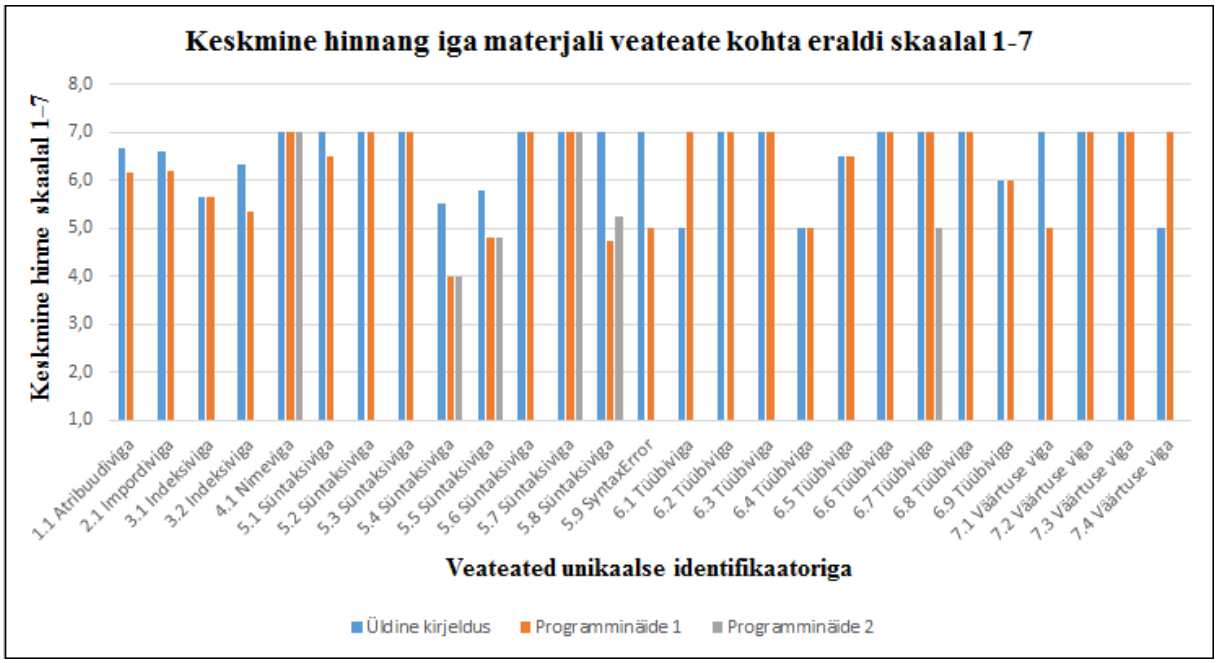
Õppematerjali valmides paluti selle esialgse versiooni kohta kommentaare Tartu Ülikooli arvutiteaduse instituudi programmeerimise õpetamise töörühma liikmetelt, et saada tagasisidet enne materjali avalikustamist. Kolmelt töörühma liikmelt laekus materjali kohta kokku 85 kommentaari, mille põhjal õppematerjalis ka täiendusi tehti.

Materjal avati e-kursuse *Programmeerimise alused* osalejatele palvega hinnata ja kommenteerida programmeerimiskeele Python veateadete kirjeldusi ja programminäiteid. Info saadeti e-kursusel osalejatele meili teel, mis sisaldas veebilinki materjalile, kontaktandmeid ja selgitust, et tegemist on materjali esialgse versiooniga, mis kuulub ühe osana käesoleva magistritöö juurde. Tagasisidet sai anda 27 erineva veateate eestikeelsele kirjeldusele ja 33 programminäitele. Materjal koos tagasiside vormiga oli avatud kogu e-kursuse toimumisaja (28.03.2016 – 22.05.2016), mille jooksul laekus 75 vastust. Üldist kirjeldust hinnati 74 korda, programminäidet 1 hinnati 73 ja programminäidet 2 hinnati 18 korda (Tabel 4).

Tabel 4. Keskmine hinnang iga materjali veateate kohta eraldi skaalal 1–7.

Veateade	Üldine kirjeldus	Näide 1	Näide 2
1.1 *AttributeError: 'list' object has no attribute 'split'*	6,7	6,2	
2.1 *ImportError: no module named 'Tkinter'*	6,6	6,2	
3.1 *IndexError: list assignment index out of range*	5,7	5,7	
3.2 *IndexError: list index out of range*	6,3	5,3	
4.1 *NameError: name 'summa' is not defined*	7	7	7
5.1 *SyntaxError: 'break' outside loop*	7	6,5	
5.2 *SyntaxError: 'return' outside function*	7	7	
5.3 *SyntaxError: EOL while scanning string literal*	7	7	
5.4 *SyntaxError: expected an indented block*	5,5	4	4
5.5 *SyntaxError: invalid syntax*	5,8	4,8	4,8
5.6 *SyntaxError: missing parentheses in call to 'print'*	7	7	
5.7 *SyntaxError: unexpected EOF while parsin* g	7	7	7
5.8 *SyntaxError: unexpected indent*	7	4,8	5,3
5.9 *SyntaxError: unindent does not match any outer indentation level*	7	5	
6.1 *TypeError: 'int' object is not subscriptable*	5	7	
6.2 *TypeError: 'list' object is not callable*	7	7	
6.3 *TypeError: can't convert 'int' object to str implicitly*	7	7	
6.4 *TypeError: can't convert 'NoneType' object to str implicitly*	5	5	
6.5 *TypeError: can't multiply sequence by non-int of type 'float'*	6,5	6,5	
6.6 *TypeError: float() argument must be a string or a number, not 'list'*	7	7	
6.7 *TypeError: list indices must be integers or slices, not str*	7	7	5
6.8 *TypeError: unorderable types: str() > int()*	7	7	
6.9 *TypeError: unsupported operand type(s) for +: 'int' and 'str'*	6	6	
7.1 *ValueError: could not convert string to float*	7	5	
7.2 *ValueError: empty separator*	7	7	
7.3 *ValueError: I/O operation on closed file*	7	7	
7.4 *ValueError: invalid literal for int() with base 10: '7.5'*	5	7	

Tabeli 4 põhjal saab järeldada, et keskmiselt alla nelja palli seitsmest ei saanud ükski veateate kirjeldus ega programminäide, mis tähendab, et veateate kirjeldused aitavad kaasa vea mõistmisel ja programminäited vea parandamisel. Kõige madalama hinde sai veateade süntaksiviga 5.4 ehk *SyntaxError: expected an indent block*. Kõige kõrgemat ja ühtlasi ka maksimaalset hinnet omistati koguni 16 veateate kirjeldusele ja 16 programminäitele (Joonis 26). Kõrge keskmise hinde põhjal võib järeldada, et California Ülikooli arvutiteaduse osakonna ja Bostoni Ülikooli poolt loodud õppematerjalide struktuur oli mõistlik valik veateadete kirjeldamisel ja programminäidete toomisel [17][18]. Samuti ei leidunud tagasisides ühtegi kommentaari veateadete stiilide kohta, mis annab kinnitust, et lühivormis olev veateade koos seletusega on algõppe programmeerijatele piisavalt arusaadav [15].



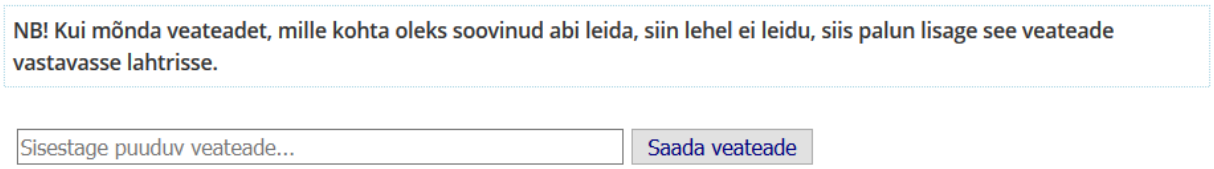
Joonis 26. Keskmine hinnang iga materjali veateate kohta eraldi skaalal 1–7.

Nii veateadete üldiste kirjelduste kui ka programminäidete kohta avaldati vabas vormis arvamust kolmel korral:

- **Üldine kirjeldus**: „Ometigi on taanetega kõik korras, aga ikkagi selline veateade.“
- **Üldine kirjeldus**: „Sain teada, et sulu paarilise puudumine annab sellise veateate ja püüdsin seda viga leida.“
- **Üldine kirjeldus**: „Sisestasin 1, mitte ujukomaarvu.“
- **Programminäide**: „Kahjuks oma viga selle järgi parandada ei osanud.“
- **Programminäide**: „Taanded on korras, aga ikka viskab sellise teate.“
- **Programminäide**: „Sulg oli puudu eelmisel real (nt rida 3), mitte seal, kust viga näidati (rida 4) ja seetõttu ei suutnud väga kaua viga leida, sest otsisin seda vale rea pealt.“

Tagasisidet kommentaaride kujul laekus vähe ja selle põhjal midagi järeldada on keeruline, kuna enamasti jagati arusaamatust enda programmis tekkinud vea kohta, mitte niivõrd konkreetse veateate kirjelduse või programminäite hinnet puudutavat lisakommentaari.

Lisaks juba olemasolevate veateadete tagasiside kogumisele lisati iga veatüübi lehe juurde vabas vormis sisendit lubav tekstiväli, kuhu saab sisestada materjalist puuduoleva veateate, mille kohta kirjeldust ja programminäiteid soovitakse (Joonis 27).



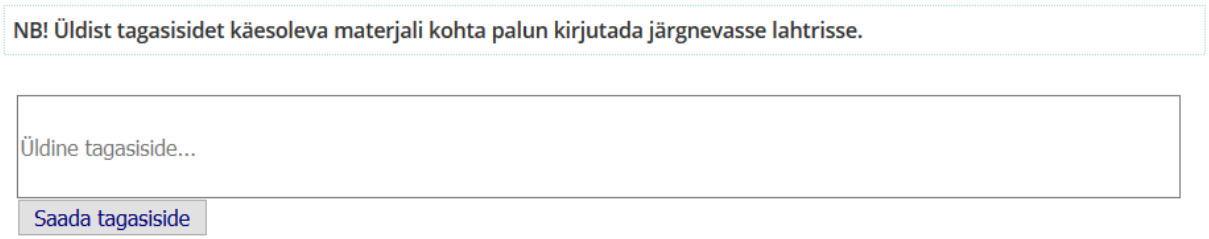
Joonis 27. Materjalist puuduoleva veateate kohta tagasiside andmise vorm [5].

Veateateid, mille kohta kirjeldusi ja programminäiteid materjalis ei leidunud, saadeti kokku 25 tükki, millest 15 olid erinevad (Tabel 5).

Tabel 5. E-kursusel osalejate poolt välja pakutud veateated, mille kohta materjali informatsiooni lisada.

Esinemissagedus	Veateade
7	*TypeError: 'float' object is not iterable*
5	*TypeError: 'int' object is not iterable*
1	*AttributeError: 'str' object has no attribute 'txt'*
1	*ImportError: No module named '_socket'*
1	*SyntaxError: positional argument follows keyword argument*
1	*TypeError: '_io.TextIOWrapper' object is not subscriptable*
1	*TypeError: Can't convert 'function' object to str implicitly*
1	*TypeError: float() takes at most 1 argument (2 given)*
1	*TypeError: round() takes at most 2 arguments (3 given)*
1	*TypeError: teleri_diagonaal() takes 0 positional arguments but 1 was given*
1	*TypeError: unorderable types: list() >= float()*
1	*TypeError: unsupported operand type(s) for *: 'float' and 'function'*
1	*TypeError: unsupported operand type(s) for /: 'tuple' and 'int'*
1	*TypeError:list indices must be integeres, not tuple*
1	*UnicodeDecodeError: 'utf-8' codec can't decode byte 0xf5 in position 5: invalid start byte*

Materjali avalehele lisati veel üks tagasiside andmise väli, kuhu sooviti vabas vormis üldist tagasisidet materjali kohta (Joonis 28).



Joonis 28. Vabas vormis materjali kohta üldise tagasiside andmise väli [5].

Kommentaare laekus kuue e-kursusel osaleja poolt, mis kõik ka järgnevas loetelus kajastatud saavad:

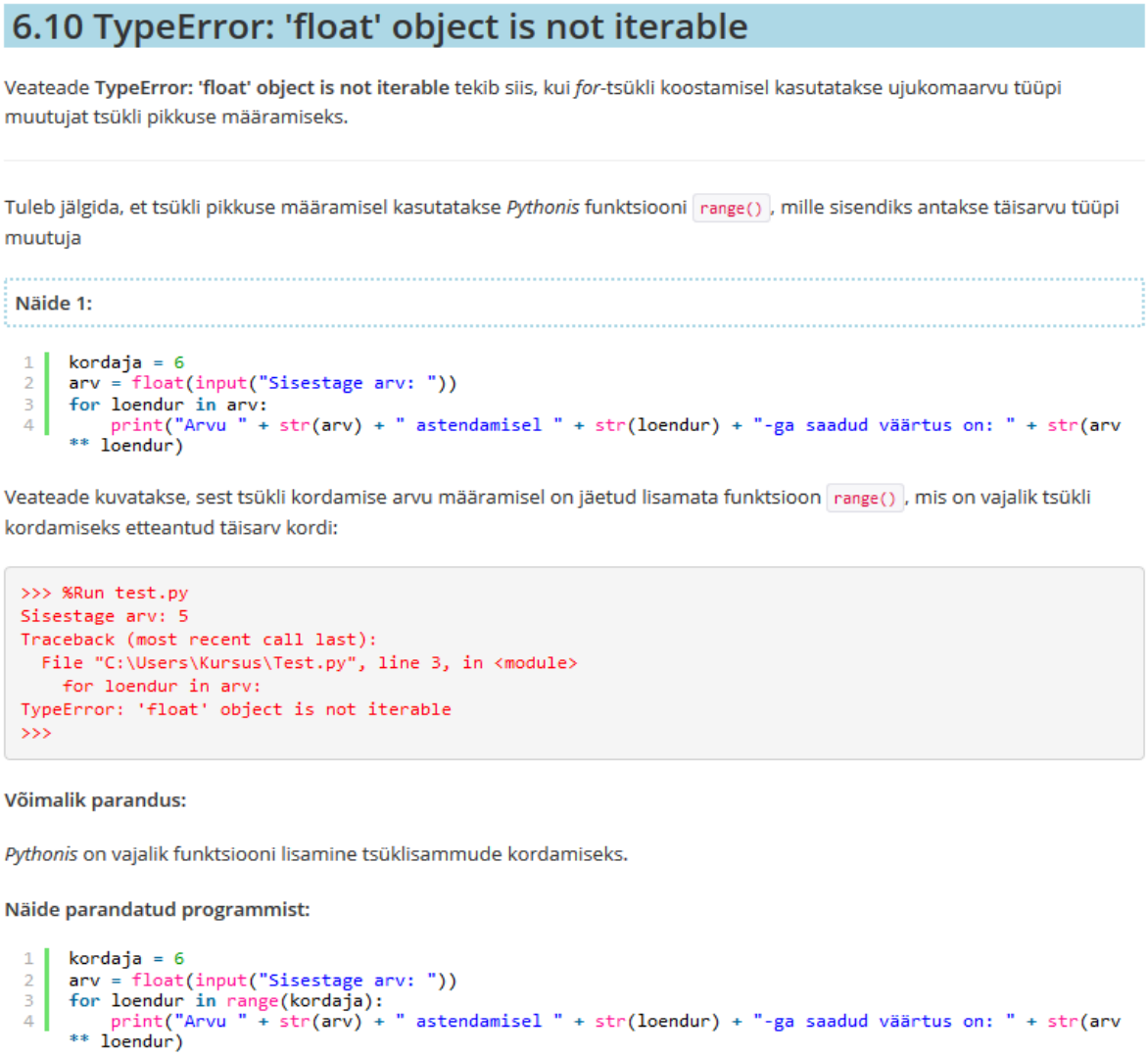
- **Üldine kommentaar:** „Jah, väga hea ja kasulik materjal :)“
- **Üldine kommentaar:** „Minu veateated olid kõik kajastatud. Väga kasulik rubriik.“
- **Üldine kommentaar:** „Päris hästi alustatud ja soovin jõudu jätkamisel.“
- **Üldine kommentaar:** „ValueError: could not convert string to float: 'ļ»æ5\n', esineb kui teksti dokument on salvestatud UTF-8ga, ANSIga salvestatult töötab.“
- **Üldine kommentaar:** „Kindlasti tarvilik materjal.“
- **Üldine kommentaar:** „Võib-olla võiks seda lauset üritada selgemana väljendada: Kõik programminäited on Thonny-ga käivitatavad nii vigasel moel kui ka ühe võimaliku variandina parandatud programmina.“

Tagasiside põhjal selgus, et materjal on kasulik algõppe programmeerijatele ja soovitakse täiendusi. Kuna programminäited moodustavad väga suure osa õppematerjalist, siis tagasiside põhjal saab järeldada, et nende alusel vigade mõistmine ja parandamine osutus e-kursusel osalejatele jõukohaseks [16]. Kasulikku informatsiooni laekus ühe konkreetse veateate kohta, mille puhul programmeerimise ülesande tekst nõudis, et programm loeb lähteandmed sisse teksti-failist. Sellisel juhul tuleb jälgida, et tekstidokument oleks salvestatud ANSI kodeeringuga failiks. Samuti selgus, et ühe pikema lause puhul materjali avalehel jääb segaseks programminäidete kohta olev informatsioon.

3.3 Õppematerjali täiendamine ja kasutamine edaspidi

Materjali kohta koguti tagasisidet eelkõige eesmärgiga teha sellest järeldusi ja parandusi materjali täiendamiseks. Tagasisidet koguti konkreetsete veateadete kirjelduste ja programminäidete kohta hinnete ja kommentaaride näol. Lisaks oli võimalus e-kursuslastel jagada arvamust vabas vormis kogu materjali kohta ja saata veateateid, mille kohta materjalis informatsioon puudus.

Joonisel 30 üksikult esinevaid programmeerimiskeele Python veateateid materjali juurde lisama ei hakata (kuna nende esinemissagedus on väga madal). Küll aga seitse korda esinev veateade *TypeError: 'float' object is not iterable* ja viis korda esinev veateade *TypeError: 'int' object is not iterable* kajastatakse materjali täiustatud versioonis kirjelduste ja programminäidete kujul [30] (Joonis 29 ja Joonis 30).



6.10 TypeError: 'float' object is not iterable

Veateade **TypeError: 'float' object is not iterable** tekib siis, kui *for*-tsükli koostamisel kasutatakse ujukomaarvu tüüpi muutujat tsükli pikkuse määramiseks.

Tuleb jälgida, et tsükli pikkuse määramisel kasutatakse *Pythonis* funktsiooni `range()`, mille sisendiks antakse täisarvu tüüpi muutuja

Näide 1:

```
1  kordaja = 6
2  arv = float(input("Sisestage arv: "))
3  for loendur in arv:
4      print("Arvu " + str(arv) + " astendamisel " + str(loendur) + "-ga saadud väärtus on: " + str(arv
   ** loendur)
```

Veateade kuvatakse, sest tsükli kordamise arvu määramisel on jäetud lisamata funktsioon `range()`, mis on vajalik tsükli kordamiseks etteantud täisarv kordi:

```
>>> %Run test.py
Sisestage arv: 5
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 3, in <module>
    for loendur in arv:
TypeError: 'float' object is not iterable
>>>
```

Võimalik parandus:

Pythonis on vajalik funktsiooni lisamine tsüklisammude kordamiseks.

Näide parandatud programmist:

```
1  kordaja = 6
2  arv = float(input("Sisestage arv: "))
3  for loendur in range(kordaja):
4      print("Arvu " + str(arv) + " astendamisel " + str(loendur) + "-ga saadud väärtus on: " + str(arv
   ** loendur)
```

Joonis 29. Veateate *TypeError: 'float' object is not iterable* kirjeldus materjalis [5].

6.11 TypeError: 'int' object is not iterable

Veateade **TypeError: 'int' object is not iterable** tekib siis, kui *for*-tsükli koostamisel kasutatakse täisarvu tüüpi muutujat tsükli pikkuse määramiseks.

Tuleb jälgida, et tsükli pikkuse määramisel kasutatakse *Pythonis* funtsiooni `range()`, mille sisendiks antakse täisarvu tüüpi muutuja

Näide 1:

```
1  arv = int(input("Sisestage tsükli pikkus: "))
2  for loendur in arv:
3      print(loendur)
```

Veateade kuvatakse, sest *for*-tsükli pikkuse määramisel on jäetud lisamata funktsioon `range()`, mis on vajalik tsükli kordamiseks etteantud täisarv kordi:

```
>>> %Run test.py
Sisestage arv: 5
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 2, in <module>
    for loendur in arv:
TypeError: 'int' object is not iterable
```

Võimalik parandus:

Pythonis on vajalik funktsiooni lisamine tsüklisammude kordamiseks.

Näide parandatud programmist:

```
1  arv = int(input("Sisestage tsükli pikkus: "))
2  for loendur in range(arv):
3      print(loendur)
```

Joonis 30. Veateate *TypeError: 'int' object is not iterable* kirjeldus materjalis [5].

Olemasolevatest veateadetest tagasisides kõige madalama skoori saanud süntaksiviga 5.4 ehk *SyntaxError: expected an indent block* puhul muudeti kirjeldust.

Veateate *SyntaxError: expected an indented block* kirjeldus enne muutmist:

Veateade SyntaxError: expected an indented block tekib siis, kui Pythonis pole pärast tsükli, tingimuslause, funktsiooni või klassi esimest rida lisatud taandega programmiplokki.

Veateate *SyntaxError: expected an indented block* kirjeldus pärast muutmist:

Veateade SyntaxError: expected an indented block tekib siis, kui Pythonis pole kasutatud korrektselt taanet tsükli, tingimuslause või funktsiooni sees.

Veateate *SyntaxError: expected an indented block* tekke põhjuse kirjeldus enne muutmist:

Veateade kuvatakse, sest esimene print käsk peaks alustama tingimuslause sees taandega koodiplokki, kuid on sama taandega, mis tingimuslause esimene rida.

Veateate *SyntaxError: expected an indented block* tekke põhjuse kirjeldus pärast muutmist:

Veateade kuvatakse, sest esimese print käsu puhul ei kasutata tingimuslauses taanet.

E-kursusel osalejate üldisest tagasisidest lähtudes modifitseeriti ka materjali avalehel olevat järgmist lauset (parema loetavuse ja arusaamise tarbeks):

Kõik programminäited on Thonnyga käivitatavad nii vigasel moel kui ka ühe võimaliku variandina parandatud programmina.

Lause täiustamiseks seletati pisut pikemalt lahti vigase programminäite ja parandatud programminäite erinevus ning muudeti sõnastust. Uus versioon sellest lauses näeb materjalis välja järgmiselt:

Kõik materjalis esinevad programminäited on käivitatavad Thonnyga – vigasel kujul olevad näited väljastavad veateateid, parandatud kujul olevate näidete puhul on vead parandatud.

Tagasiside põhjal võib järeldada, et õppematerjal sobib kasutusse Tartu Ülikooli e-kursuste *Programmeerimisest maalähedaselt* ja *Programmeerimise alused* juures abimaterjalina programmeerimisülesannete lahendamisel. Erinevate programmeerimisülesannete puhul veateadete esinemissagedused varieeruvad, seega uute programmeerimisteemade lisamisel e-kursustele tuleks materjali täiustada uute veateadete kirjelduste ja programminäidetega. Selleks on tarvis koguda e-kursusel osalejatelt kogu kursuse vältel Thonny logifaile ja nende põhjal statistikat teha.

Kokkuvõte

Käesolevas magistritöös uuriti programmeerimiskeele Python veateadete kirjeldusi ja arusaadavust programmeerimise algõppe e-kursuste *Programmeerimisest maalähedaselt* ja *Programmeerimise alused* raames. Töö tulemina valmis eestikeelne veebikujul olev õppematerjal sagedasemate veateadete kirjelduste ja programminäidetega.

Magistritöö esimeses osas anti ülevaade programmeerimise õppimisest ja programmeerimiskeele Python veateadetest. Kirjeldati ka veateadetega seotud varasemalt tehtud uuringuid ja õppematerjale. Lisaks tutvustati pikemalt Tartu Ülikooli programmeerimisteemalisi e-kursuseid ja nendes kasutusel olevat programmeerimiskeskkonda Thonny.

Töö teises osas kirjeldati detailselt läbiviidud programmeerimiskeele Python veateadete uuringut. E-kirja teel saadeti välja kutse e-kursusele registreerunutele palvega osaleda veateadete uuringus. Nõusoleku osaleda andis 15 e-kursuslast. Uuringu raames küsitleti e-kursuslasi veateadete loetavuse, mõistmise, detailsuse, keelevaliku, tehnilise kirjelduse ja veateate kasulikkuse kohta vea parandamise osas. Samuti koguti Thonny logifaile, mille põhjal valmisid veateadete esinemissagedustabelid.

Käesoleva töö kolmanda peatüki moodustas veateadete kohta loodud õppematerjal. Materjali loomisel võeti aluseks varasemate uuringute tulemused ja õppematerjalide struktuur, mis kombineeriti läbiviidud uuringu tulemustega. Esialgse õppematerjali sisu koosnes 27 programmeerimiskeele Python erineva veateate kirjeldusest ja 33 programminäitest ning selle eesmärgiks oli olla abiks veateadete mõistmisel ja tekkinud vigade parandamisel. Materjali esialgne versioon, mis avati ka e-kursuse *Programmeerimise alused* osalejatele, sisaldas iga veateate juures tagasisidevormi, mille põhjal analüüsiti materjali kasulikkust ja puudujääke. Laekunud tagasiside põhjal täiustati õppematerjali kahe uue veateatega ja tehti järeldus, et vähemalt tagasisidet andnud e-kursusel osalejate arvates on sellisel kujul valminud veateadete materjal abiks veateadete mõistmisel ja sobib seega kasutusse programmeerimisteemaliste e-kursuste juures abimaterjalina.

Magistritöö tulemusi võib kasutada tulevaste uuringute tarbeks, et uurida põhjalikumalt veateadete õppematerjali seost veateadete tekkimise hulgaga. Saab uurida, kuivõrd oli kasu mõne konkreetse veateate kirjeldusest või programminäitest vea parandamisel. Veateadete

õppematerjali saab täiendada lisades uusi veateadete kirjeldusi ja programminäiteid, mis Thonny logifaile analüüsides uute programmeerimisülesannete lahenduste puhul sagedasemateks osutuvad.

Kasutatud kirjandus

[1] Tartu Ülikooli programmeerimisteemaliste e-kursuste võrdlustabel: https://courses.cs.ut.ee/2016/progmaa/spring/Main/Maalahealusedvordlus (08.01.2017)

[2] Traver, V. Javier. 2010. On Compiler Error Messages: What They Say and What They Mean. *Advances in Human-Computer Interaction.*

[3] E-kursuse *Programmeerimisest maalähedaselt* kirjeldus: https://courses.cs.ut.ee/2016/progmaa/spring (08.01.2017)

[4] E-kursuse *Programmeerimise alused* kirjeldus: https://courses.cs.ut.ee/2016/eprogalused/spring (08.01.2017)

[5] Programmeerimiskeele Python veateadete õppematerjal: https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Veateated (08.01.2017)

[6] Marceau, G., Fisler, K., Krishnamurthi, S. 2011. Measuring the Effectiveness of Error Messages Designed for Novice Programmers. *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM.

[7] Lutz, M. 2007. Learning Python, 3rd Edition. *O'Reilly Media*.

[8] Programmeerimise õpik: https://programmeerimine.cs.ut.ee/01_sissejuhatus.html (08.01.2017)

[9] Garrison, D. R. 2011. E-learning in the 21st century: A framework for reasearch and practise. *Taylor & Francis.*

[10] Jenkins, T. 2002. On the difficulty of learning to program. *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*. Vol. 4.

[11] Hartmann, B., MacDougall, D., Brandt, J., Klemmer, S. R. 2010. What Would Other Programmers Do: Suggesting Solutions to Error Messages. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM.

[12] Schliep, P. A. 2015. Usability of Error Messages for Introductory Students. *Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal* 2.2.

[13] Leping, V., Lepp, M., Niitsoo, M., Tõnisson, E., Vene, V., Villems, A. 2009. Python Prevails. *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*. ACM.

[14] Programmeerimiskeele Python veateadete klassifikatsioon: https://docs.python.org/2.7/tutorial/errors.html (02.01.2017)

[15] Nienaltowski, M. H., Pedroni, M., Meyer, B. 2008. Compiler error messages: What can help novices? *ACM SIGCSE Bulletin*. Vol. 40. No. 1. ACM.

[16] Lahtinen, E., Ala-Mutka, K., Järvinen, H. M. 2005. A Study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*. Vol. 37. No. 3. ACM.

[17] California Ülikoolis kasutusel oleva programmeerimiskeele Python veateadete materjal: https://discover.cs.ucsb.edu/commonerrors/pythonerrors.html (27.12.2016)

[18] Bostoni Ülikoolis kasutusel oleva programmeerimiskeele Python veateadete materjal: http://www.cs.bu.edu/courses/cs108/guides/debug.html (02.01.2017)

[19] Hollo, K. 2016. Programmeerimise e-kursusel osalejate küsimuste analüüs ja selle põhjal murelahendajate koostamine. *Tartu Ülikool, arvutiteaduse instituut.*

[20] Annamaa, A. 2015. Thonny, a Python IDE for Learning Programming. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM.

[21] Pedel, K. 2016. E-kursuse *Programmeerimise alused* logifailide analüüs. *Tartu Ülikool, arvutiteaduse instituut.*

[22] Programmeerimiskeskkonna Thonny programmilogide üleslaadmise videopõhine juhend: https://youtu.be/aliHOcLDNjY (11.12.2016)

[23] Programmeerimiskeskkonna Thonny programmilogide üleslaadimise tekstipõhine juhend: https://drive.google.com/file/d/0B6CV7K7kS00IenRFbFdCSXVVeEk/view (11.12.2016)

[24] Google Drive tarkvara kirjeldus: https://www.google.com/drive/ (27.12.2016)

[25] Camtasia Recorder tarkvara kirjeldus: https://www.techsmith.com/camtasia.html (27.12.2016)

[26] Camtasia Studio tarkvara kirjeldus: https://www.techsmith.com/tutorial-camtasia-8.html (27.12.2016)

[27] Operatsioonisüsteemi UNIX tekstitöötlemise käsud: http://www.tldp.org/LDP/abs/html/textproc.html (12.12.2016)

[28] Märgistuskeele HTML vahendite kirjeldus: http://www.w3schools.com/html/html_forms.asp (02.01.2017)

[29] Skriptimiskeele PHP vahendite kirjeldus: http://www.w3schools.com/php/php_forms.asp (02.01.2017)

[30] Python *range*() funktsiooni kirjeldus: http://pythoncentral.io/pythons-range-function-explained/ (01.01.2017)

Lisa 1

E-kursusel *Programmeerimisest maalähedaselt* esinenud veatüüpide esinemissagedustabel töötlemata kujul

Esinemissagedus	Veateade
496	SyntaxError: invalid syntax\n,
80	TypeError: expected string or buffer\n,
68	SyntaxError: EOL while scanning string literal\n,
54	SyntaxError: unexpected EOF while parsing\n,
28	SyntaxError: expected an indented block\n,
26	TypeError: match() missing 1 required positional argument: 'string'\n,
16	SyntaxError: unexpected indent\n,
16	TypeError: 'str' object does not support item assignment\n,
16	TypeError: a float is required\n,
16	TypeError: Can't convert 'int' object to str implicitly\n,
16	TypeError: search() missing 1 required positional argument: 'string'\n,
12	AttributeError: 'module' object has no attribute 'mach'\n,
12	TypeError: 'int' object does not support item assignment\n,
12	TypeError: unsupported operand type(s) for -: 'str' and 'int'\n,
10	TypeError: object of type 'int' has no len()\n,
8	ImportError: cannot import name 'randiant'\n,
8	NameError: name 'aasta' is not defined\n,
8	NameError: name 'parool' is not defined\n,
8	NameError: name 'red' is not defined\n,
8	SyntaxError: keyword can't be an expression\n,
8	TypeError: ruut() missing 1 required positional argument: 'k\u00fclg'\n,
8	TypeError: unsupported operand type(s) for /: 'tuple' and 'int'\n,
8	ValueError: invalid literal for int() with base 10: ''\n,
6	NameError: name 'suurus' is not defined\n,
6	NameError: name 'vanaema_vanus' is not defined\n,
6	SyntaxError: Missing parentheses in call to 'print'\n,
6	TypeError: 'int' object is not subscriptable\n,
6	TypeError: unsupported operand type(s) for +: 'int' and 'str'\n,
6	ValueError: could not convert string to float: '18,5'\n,

4	AttributeError: 'str' object has no attribute 'lowwer'\n,
4	NameError: name 'aaaaaaaaaa' is not defined\n,
4	NameError: name 'aDu' is not defined\n,
4	NameError: name 'ADu' is not defined\n,
4	NameError: name 'ADU' is not defined\n,
4	NameError: name 'afi' is not defined\n,
4	NameError: name 'arvude_summa' is not defined\n,
4	NameError: name 'begin_full' is not defined\n,
4	NameError: name 'extonclick' is not defined\n,
4	NameError: name 'foward' is not defined\n,
4	NameError: name 'intput' is not defined\n,
4	NameError: name 'number' is not defined\n,
4	NameError: name 'pakk_ceil' is not defined\n,
4	NameError: name 'Print' is not defined\n,
4	SyntaxError: can't assign to operator\n,
4	TypeError: 'type' object is not subscriptable\n,
4	TypeError: bad operand type for abs(): 'str'\n,
4	TypeError: first argument must be string or compiled pattern\n,
4	TypeError: len() takes exactly one argument (0 given)\n,
4	TypeError: len() takes exactly one argument (2 given)\n,
2	NameError: name '\u00fcletundide_arv' is not defined\n,
2	NameError: name 'arv' is not defined\n,
2	NameError: name 'floa' is not defined\n,
2	NameError: name 'isikukood' is not defined\n,
2	NameError: name 'jalalaba_pikkus_cm' is not defined\n,
2	NameError: name 'jalalaba_suurus_cm' is not defined\n,
2	NameError: name 'jalan\u00f5udesuurus' is not defined\n,
2	NameError: name 'kkk' is not defined\n,
2	NameError: name 'korrutis' is not defined\n,
2	NameError: name 'kuni' is not defined\n,
2	NameError: name 'normtunnid' is not defined\n,
2	NameError: name 't\u00e4ht' is not defined\n,
2	NameError: name 'tr\u00fckiAB' is not defined\n,
2	NameError: name 'x' is not defined\n,
2	SyntaxError: invalid token\n,

2	TypeError: Can't convert 'float' object to str implicitly\n,
2	TypeError: float() argument must be a string or a number, not 'builtin_function_or_method'\n,
2	TypeError: int() argument must be a string, a bytes-like object or a number, not 'builtin_function_or_method'\n,
2	TypeError: unorderable types: str() > int()\n,
2	ValueError: invalid literal for int() with base 10: '269080227AM'\n,
2	ValueError: invalid literal for int() with base 10: '2690802788AM'\n,
2	ValueError: invalid literal for int() with base 10: '3h546879547'\n,
2	ValueError: invalid literal for int() with base 10: '5,5'\n,
2	ValueError: invalid literal for int() with base 10: '5.5'\n,
1	Search "Error" (553 hits in 55 files)

Lisa 2

E-kursusel *Programmeerimise alused* esinenud veatüüpide esinemissagedustabel töötlemata kujul.

Esinemissagedus	Veateade
154	SyntaxError: invalid syntax\n,
25	SyntaxError: unexpected indent\n,
16	TypeError: unsupported operand type(s) for /: 'function' and 'int'\n,
15	TypeError: can't multiply sequence by non-int of type 'float'\n,
14	TypeError: list indices must be integers or slices, not float\n,
10	SyntaxError: expected an indented block\n,
10	SyntaxError: unexpected EOF while parsing\n,
9	NameError: name 'summa' is not defined\n,
8	IndexError: list index out of range\n,
7	TypeError: bad operand type for unary -: 'list'\n,
7	TypeError: float() argument must be a string or a number, not 'list'\n,
6	TypeError: can only concatenate list (not \tuple\") to list\n",
5	SyntaxError: invalid token\n,
5	TypeError: list indices must be integers or slices, not str\n,
5	TypeError: unorderable types: str() >= int()\n,
5	TypeError: unsupported operand type(s) for *: '_io.TextIOWrapper' and 'float'\n,
4	NameError: name 'canvas' is not defined\n,
4	NameError: name 'k' is not defined\n,
4	NameError: name 'kmtaHind' is not defined\n,
4	OSError: [Errno 22] Invalid argument: 'Sisestage failinimi : '\n,
4	SyntaxError: can't assign to function call\n,
4	TypeError: Can't convert 'float' object to str implicitly\n,
4	TypeError: input() takes no keyword arguments\n,
4	TypeError: unsupported operand type(s) for -: 'str' and 'int'\n,
4	TypeError: unsupported operand type(s) for +=: 'int' and 'list'\n
4	TypeError: unsupported operand type(s) for +=: 'int' and 'str'\n,
3	_tkinter.TclError: wrong # coordinates: expected at least 4, got 2\n,
3	NameError: name 'hind' is not defined\n,

3	NameError: name 'm\u00fcndid' is not defined\n,
3	NameError: name 'math' is not defined\n,
3	NameError: name 'Math' is not defined\n,
3	NameError: name 'new_outer_RAD' is not defined\n,
3	NameError: name 'raam' is not defined\n,
3	NameError: name 'randint' is not defined\n,
3	NameError: name 'start_y' is not defined\n,
3	NameError: name 'summaKMga' is not defined\n,
3	SyntaxError: invalid syntax\n
3	TypeError: float() argument must be a string or a number, not '_io.TextIOWrapper'\n,
3	TypeError: unsupported operand type(s) for +: 'int' and 'str'\n,
3	TypeError: unsupported operand type(s) for +: 'int' and 'tuple'\n,
2	_tkinter.TclError: couldn't recognize data in image file \Panel.jpg\"\n",
2	_tkinter.TclError: unknown color name \sinine\"\n",
2	_tkinter.TclError: unknown option \pyimage1\"\n",
2	FileNotFoundError: [Errno 2] No such file or directory: '20'\n,
2	NameError: name 'hind_k\u00e4ibemaksuga' is not defined\n,
2	NameError: name 'km' is not defined\n
2	NameError: name 'kmgaHind' is not defined\n,
2	NameError: name 'kokku1' is not defined\n,
2	NameError: name 'maksalates' is not defined\n
2	NameError: name 'start_x' is not defined\n,
2	NameError: name 'tagasi' is not defined\n,
2	NameError: name 'triibud' is not defined\n,
2	SyntaxError: 'return' outside function\n,
2	SyntaxError: Missing parentheses in call to 'print'\n,
2	SyntaxError: positional argument follows keyword argument\n,
2	SyntaxError: unexpected EOF while parsing\n
2	SyntaxError: unindent does not match any outer indentation level\n,
2	TypeError: can only concatenate list (not \int\") to list\n",
2	TypeError: Can't convert 'int' object to str implicitly\n,
2	TypeError: float() argument must be a string or a number, not 'list'\n
2	TypeError: float() takes at most 1 argument (2 given)\n,
2	TypeError: int() argument must be a string, a bytes-like object or a number, not 'list'\n

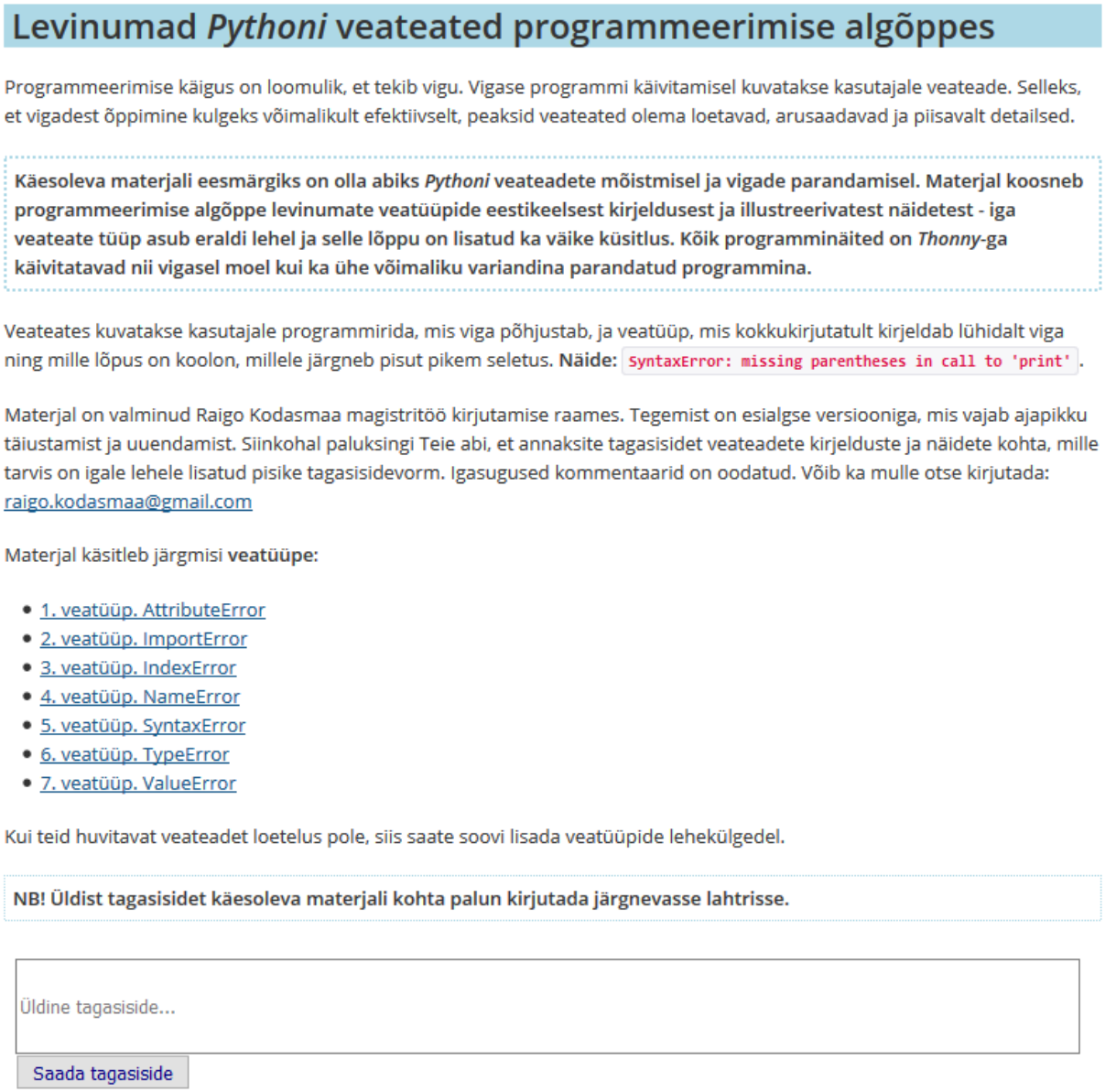
2	TypeError: makeStar() missing 2 required positional arguments: 'o' and 'i'\n,
2	TypeError: str() argument 2 must be str, not int\n,
2	TypeError: tetrakoloor() missing 1 required positional argument: 'v\u00e4rv4'\n,
2	TypeError: triibud() missing 1 required positional argument: 'v\u00e4rv2'\n,
2	TypeError: type str doesn't define __round__ method\n,
2	TypeError: unsupported operand type(s) for *: 'NoneType' and 'float'\n,
2	TypeError: unsupported operand type(s) for /: 'str' and 'int'\n,
2	ValueError: invalid literal for int() with base 10: '2.6'\n,
1	_tkinter.TclError: bad screen distance \<class 'range'>\"\n",
1	_tkinter.TclError: font \\" doesn't exist\n",
1	_tkinter.TclError: image \Knob.gif\" doesn't exist\n",
1	_tkinter.TclError: unknown color name \green50\"\n",
1	_tkinter.TclError: unknown color name \whitw\"\n",
1	_tkinter.TclError: unknown option \-bd\"\n",
1	_tkinter.TclError: unknown option \-file\"\n",
1	_tkinter.TclError: unknown option \-heigth\"\n",
1	_tkinter.TclError: unknown option \100\"\n",
1	_tkinter.TclError: unknown option \165\"\n",
1	_tkinter.TclError: wrong # coordinates: expected an even number, got 7\n,
1	AttributeError: 'Canvas' object has no attribute 'Button'\n,
1	AttributeError: 'Canvas' object has no attribute 'makeStar'\n,
1	AttributeError: 'float' object has no attribute 'close'\n,
1	AttributeError: 'float' object has no attribute 'strip'\n,
1	AttributeError: 'function' object has no attribute 'strip'\n,
1	FileNotFoundError: [Errno 2] No such file or directory: "\n
1	NameError: name 'ainult_k\u00e4ibemaks' is not defined\n,
1	NameError: name 'arv' is not defined\n,
1	NameError: name 'create_text' is not defined\n,
1	NameError: name 'file' is not defined\n,
1	NameError: name 'font' is not defined\n,
1	NameError: name 'hindx' is not defined\n,
1	NameError: name 'i' is not defined\n,
1	NameError: name 'intfloat' is not defined\n,
1	NameError: name 'intput' is not defined\n,

1	NameError: name 'kindkm' is not defined\n,
1	NameError: name 'km' is not defined\n,
1	NameError: name 'kmsumma' is not defined\n,
1	NameError: name 'kmSumma' is not defined\n,
1	NameError: name 'm\u00fcnt' is not defined\n,
1	NameError: name 'master' is not defined\n,
1	NameError: name 'nput' is not defined\n,
1	NameError: name 'number' is not defined\n,
1	NameError: name 'o' is not defined\n,
1	NameError: name 'ostud' is not defined\n,
1	NameError: name 'pi' is not defined\n,
1	NameError: name 'random' is not defined\n,
1	NameError: name 'ridadekaupa' is not defined\n,
1	NameError: name 'roand' is not defined\n,
1	NameError: name 'tk' is not defined\n,
1	NameError: name 'true' is not defined\n,
1	NameError: name 'y' is not defined\n,
1	SyntaxError: EOL while scanning string literal\n,
1	TypeError: 'float' object is not callable\n,
1	TypeError: 'float' object is not iterable\n,
1	TypeError: 'type' object is not subscriptable\n,
1	TypeError: an integer is required (got type str)\n,
1	TypeError: bad operand type for unary +: 'list'\n,
1	TypeError: can only concatenate list (not \float\") to list\n",
1	TypeError: can't concat bytes to str\n,
1	TypeError: can't multiply sequence by non-int of type 'float'\n
1	TypeError: float() argument must be a string or a number, not '_io.TextIOWrapper'\n
1	TypeError: float() argument must be a string or a number, not 'NoneType'\n,
1	TypeError: float() argument must be a string or a number, not 'tuple'\n,
1	TypeError: hind_k\u00e4ibemaksuga() missing 1 required positional argument: 'k\u00e4ibemaks'\n,
1	TypeError: hk() missing 2 required positional arguments: 'hind' and 'km'\n,
1	TypeError: k\u00e4ibemaks_maha_summast() missing 1 required positional argument: 'km'\n

1	TypeError: makeStar() missing 2 required positional arguments: 'start_x' and 'start_y'\n,
1	TypeError: randint() missing 1 required positional argument: 'b'\n,
1	TypeError: str() argument 2 must be str, not int\n
1	TypeError: tetrakoloor() missing 2 required positional arguments: 'v\u00e4rv3' and 'v\u00e4rv4'\n,
1	TypeError: type function doesn't define __round__ method\n,
1	TypeError: type tuple doesn't define __round__ method\n,
1	TypeError: unsupported operand type(s) for +: 'float' and 'str'\n,
1	TypeError: unsupported operand type(s) for +: 'int' and 'list'\n,
1	TypeError: unsupported operand type(s) for +=: 'float' and 'list'\n,
1	TypeError: unsupported operand type(s) for +=: 'int' and 'list'\n,
1	UnboundLocalError: local variable 'tagasi' referenced before assignment\n,
1	ValueError: could not convert string to float: 'EUR 2'\n,
1	ValueError: invalid literal for int() with base 10: "\n
1	ValueError: invalid literal for int() with base 10: '2.6\\n'\n,
1	ValueError: invalid literal for int() with base 10: '2.75'\n,

Lisa 3

Programmeerimiskeele Python veateadete õppematerjal kogu mahus.



Levinumad *Pythoni* veateated programmeerimise algõppes

Programmeerimise käigus on loomulik, et tekib vigu. Vigase programmi käivitamisel kuvatakse kasutajale veateade. Selleks, et vigadest õppimine kulgeks võimalikult efektiivselt, peaksid veateated olema loetavad, arusaadavad ja piisavalt detailsed.

Käesoleva materjali eesmärgiks on olla abiks *Pythoni* veateadete mõistmisel ja vigade parandamisel. Materjal koosneb programmeerimise algõppe levinumate veatüüpide eestikeelsest kirjeldusest ja illustreerivatest näidetest - iga veateate tüüp asub eraldi lehel ja selle lõppu on lisatud ka väike küsitlus. Kõik programminäited on *Thonny*-ga käivitatavad nii vigasel moel kui ka ühe võimaliku variandina parandatud programmina.

Veateates kuvatakse kasutajale programmirida, mis viga põhjustab, ja veatüüp, mis kokkukirjutatult kirjeldab lühidalt viga ning mille lõpus on koolon, millele järgneb pisut pikem seletus. **Näide:** `SyntaxError: missing parentheses in call to 'print'`.

Materjal on valminud Raigo Kodasmaa magistritöö kirjutamise raames. Tegemist on esialgse versiooniga, mis vajab ajapikku täiustamist ja uuendamist. Siinkohal paluksingi Teie abi, et annaksite tagasisidet veateadete kirjelduste ja näidete kohta, mille tarvis on igale lehele lisatud pisike tagasisidevorm. Igasugused kommentaarid on oodatud. Võib ka mulle otse kirjutada: raigo.kodasmaa@gmail.com

Materjal käsitleb järgmisi **veatüüpe:**

- 1. veatüüp. AttributeError
- 2. veatüüp. ImportError
- 3. veatüüp. IndexError
- 4. veatüüp. NameError
- 5. veatüüp. SyntaxError
- 6. veatüüp. TypeError
- 7. veatüüp. ValueError

Kui teid huvitavat veateadet loetelus pole, siis saate soovi lisada veatüüpide lehekülgedel.

NB! Üldist tagasisidet käesoleva materjali kohta palun kirjutada järgnevasse lahtrisse.

Üldine tagasiside...

Saada tagasiside

Allikas 1. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Veateated

Allikas 2. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Attribute

1.1 AttributeError: 'list' object has no attribute 'split'

Veateade **AttributeError: 'list' object has no attribute 'split'** tekib, sest järjendite puhul ei saa *Pythonis* kasutada tükeldamisfunktsiooni `split`, mis on mõeldud ainult sõnetüüpi väärtustele.

Tuleb jälgida, et tükeldamisfunktsiooni `split` ei saa kasutada järjendite, vaid sõnede puhul.

Näide 1:

```
1  tooted = ['kartul', 'kaalikas', 'porgand']
2
3  print("Tooted on: " + str(tooted.split(",")))
```

Veateade kuvatakse, sest funktsioon `split` ei ole võimeline *Pythonis* järjendit tükeldama:

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 3, in <module>
    print("Tooted on: " + str(tooted.split(",")))
AttributeError: 'list' object has no attribute 'split'
>>>
```

Võimalik parandus:

Muutuja `tooted` tuleks muuta sõnetüüpi andmetüübiks, mis sobib funktsioonile `split` sisendiks.

Näide parandatud programmist:

```
1  tooted = 'kartul,kaalikas,porgand'
2
3  print("Tooted on: " + str(tooted.split(",")))
```

Allikas 3. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Attribute1

Allikas 4. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Import

2.1 ImportError: no module named 'Tkinter'

Veateade **ImportError: no module named 'Tkinter'** tekib siis, kui üritatakse importida sellise nimega moodulit, mis ei vasta konkreetsele *Pythoni* versioonile või on kättesaamatu.

Tuleb jälgida, et imporditavad moodulid oleks vastava *Pythoni* versiooni poolt toetatavad ja programmile kättesaadavad.

Näide 1:

```
1  from Tkinter import *
2
3  root = Tk()
4  tekst = Text(root, height=2, width=30)
5  tekst.pack()
6  tekst.insert(END, "Tere!")
7  root.mainloop()
```

Veateade kuvatakse, sest alates Python 3 versioonist tuleb `Tkinter`'i asemel kasutada `tkinter` moodulit:

```
line 1
    from Tkinter import *
ImportError: No module named 'Tkinter'
```

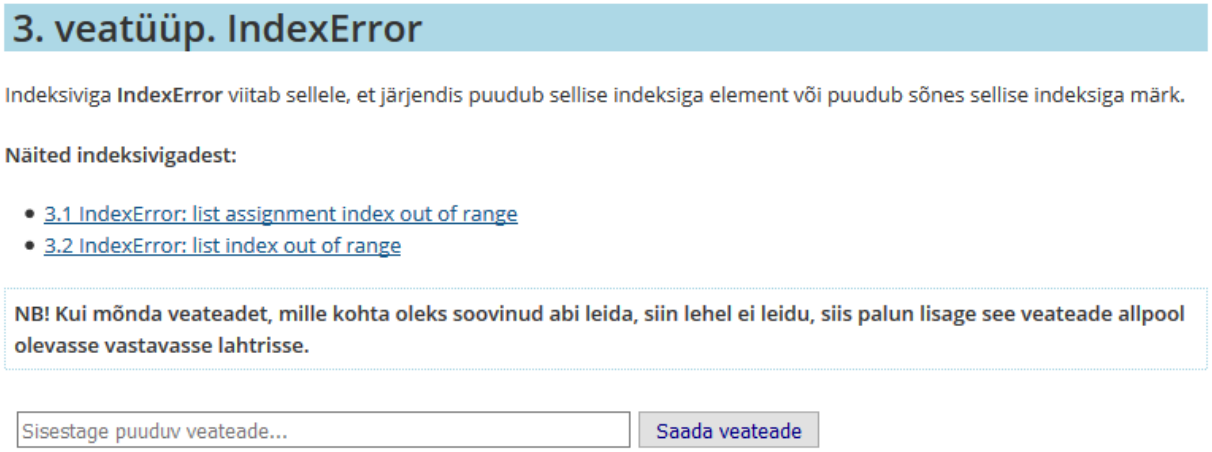
Võimalik parandus:

Imporditava mooduli nimi tuleks viia vastavusse *Pythoni* kasutatava versiooni poolt toetatava mooduli nimega.

Näide parandatud programmist:

```
1  from tkinter import *
2
3  root = Tk()
4  tekst = Text(root, height=2, width=30)
5  tekst.pack()
6  tekst.insert(END, "Tere!")
7  root.mainloop()
```

Allikas 5. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Import1



Allikas 6. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Index

3.1 IndexError: list assignment index out of range

Veateade **IndexError: list assignment index out of range** tekib siis, kui üritatakse järjendis väärtustada olematu indeksiga elementi.

Tuleb jälgida, et elemendi väärtustamisel eksisteeriks vastava indeksiga element.

Näide 1:

```
1  tulemus = []
2  tulemus[0] = 5
3
4  print("Tulemus on: " + str(tulemus[0]))
```

Veateade kuvatakse, sest järjend `tulemus` on tühi (indeksiga element puudub):

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 2, in <module>
    tulemus[0] = 5
IndexError: list assignment index out of range
>>>
```

Võimalik parandus:

Järjendisse tuleks lisada käsuga `append` element, et seda indeksiga väärtustada.

Näide parandatud programmist:

```
1  tulemus = []
2  tulemus.append(5)
3
4  print("Tulemus on: " + str(tulemus[0]))
```

Allikas 7. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Index1

3.2 IndexError: list index out of range

Veateade **IndexError: list index out of range** viitab sellele, et järjendist üritatakse otsida sellise indeksiga elementi, mida seal ei eksisteeri.

Tuleb jälgida, et järjendite puhul algab indekseerimine 0-st, mitte 1-st.

Näide 1:

```
1  nädalapäevad = ["E", "T", "K", "N", "R", "L", "P"]
2
3  print("Puhkepäevad on:\n" + nädalapäevad[6] + "\n" + nädalapäevad[7])
```

Veateade kuvatakse, sest indeksiga 7 element järjendis puudub (võimalikud indeksid 0-6):

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 3, in <module>
    print("Puhkepäevad on:\n" + nädalapäevad[6] + "\n" + nädalapäevad[7])
IndexError: list index out of range
>>>
```

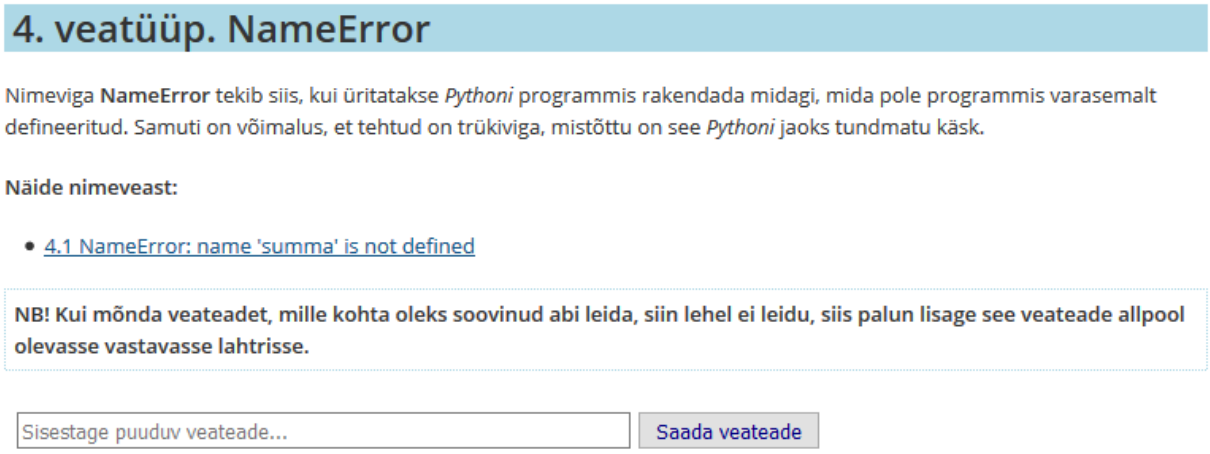
Võimalik parandus:

Indeksid tuleks asendada ühe võrra väiksematega, kuna indeksite loendamine algab 0-st.

Näide parandatud programmist:

```
1  nädalapäevad = ["E", "T", "K", "N", "R", "L", "P"]
2
3  print("Puhkepäevad on:\n" + nädalapäevad[5] + "\n" + nädalapäevad[6])
```

Allikas 8. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Index2



Allikas 9. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Name

4.1 NameError: name 'summa' is not defined

Veateade **NameError: name 'summa' is not defined** tekib siis, kui programmis kutsutakse välja muutujat või funktsiooni, mis ei ole eelnevalt defineeritud, on tehtud väljakutsumisel trükiviga või on kasutatud sama nime, mis programmil endal.

Tuleb jälgida, et muutuja oleks defineeritud enne, kui seda kasutatakse.

Näide 1:

```
1  arv1 = 5
2  arv2 = 3
3
4  print ("Summa on: " + str(summa))
5  summa = arv1 + arv2
```

Veateade kuvatakse, sest muutujat `summa` tahetakse väljastada enne, kui see on defineeritud programmis:

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 4, in <module>
    print ("Summa on: " + str(summa))
NameError: name 'summa' is not defined
>>>
```

Võimalik parandus:

Muutuja tuleks defineerida, alles seejäral saab seda väljastada.

Näide parandatud programmist:

```
1  arv1 = 5
2  arv2 = 3
3  summa = arv1 + arv2
4
5  print ("Summa on: " + str(summa))
```

Tuleb jälgida, et muutuja väljastamisel ei oleks tehtud viga.

Näide 2:

```
arv1 = 5
arv2 = 3
kokku = arv1 + arv2

print ("Summa on: " + str(summa))
```

Veateade kuvatakse, sest väljastada tahetakse sellist muutujat, mis pole programmis eelnevalt defineeritud:

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 5, in <module>
    print ("Summa on: " + str(summa))
NameError: name 'summa' is not defined
>>>
```

Võimalik parandus:

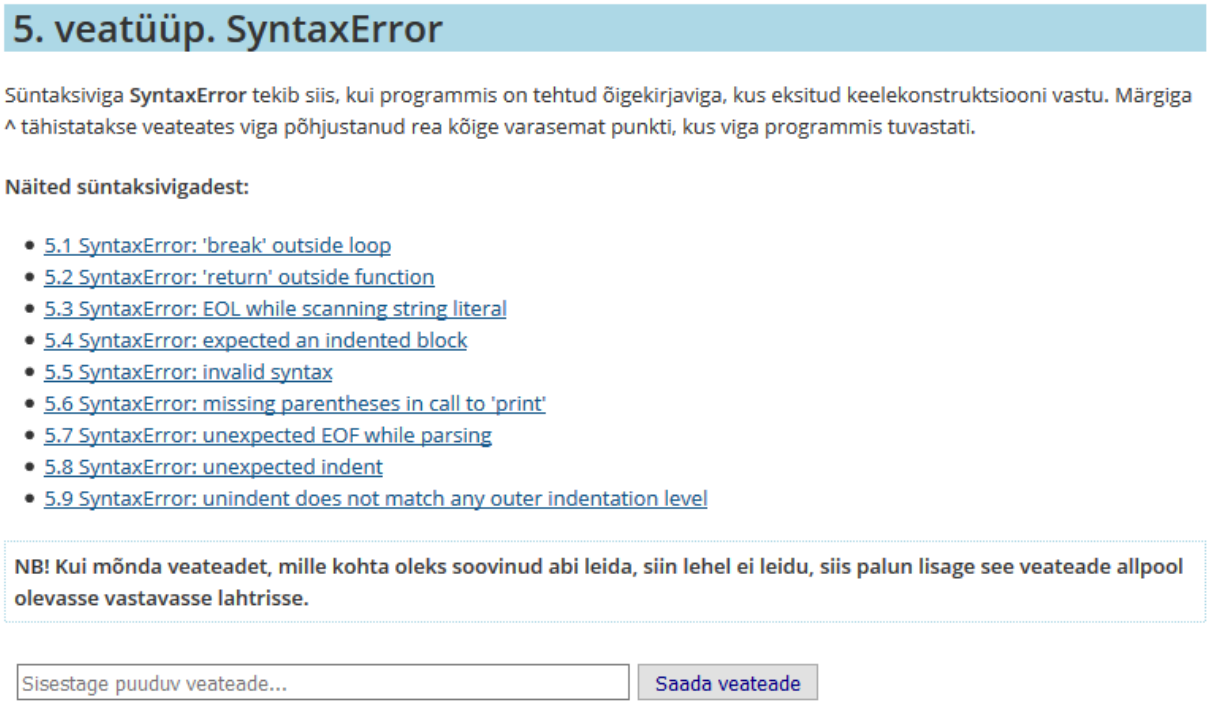
Väljastavtava muutuja nimi tuleks muuta vastavalt programmis defineeritud muutuja nimele.

Näide parandatud programmist:

```
arv1 = 5
arv2 = 3
kokku = arv1 + arv2

print ("Summa on: " + str(kokku))
```

Allikas 10. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Name1



Allikas 11. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax

5.1 SyntaxError: 'break' outside loop

Veateade **SyntaxError: 'break' outside loop** tekib siis, kui tsükli tööd üritatakse katkestada väljaspool tsüklit. Käsk `break` peab asuma tsükliploki sees.

Tuleb jälgida, et `break` käsk asuks tsükli sees.

Näide 1:

```
print("Sisestage arv: ")
arv = int(input())

while True:
    arv = arv + 1
    print("loendur: " + str(arv))
    if arv > 100:
        print("Arv " + str(arv) + " ületab 100 piiri, katkestan töö")
break;
```

Veateade kuvatakse, sest `break` käsk asub väljaspool `while` -tsüklit:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 9
    break;
        ^
SyntaxError: 'break' outside loop
>>>
```

Võimalik parandus:

Käsk `break` tuleks asendada õige taandega, et ta asuks tsükli sees.

Näide parandatud programmist:

```
print("Sisestage arv: ")
arv = int(input())

while True:
    arv = arv + 1
    print("loendur: " + str(arv))
    if arv > 100:
        print("Arv " + str(arv) + " ületab 100 piiri, katkestan töö")
        break;
```

Allikas 12. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax1

5.2 SyntaxError: 'return' outside function

Veateade **SyntaxError: 'return' outside function** tekib siis, kui käsuga `return` üritatakse tagastada midagi väljaspool funktsiooni.

Tuleb jälgida, et käsk `return` oleks funktsiooniploki sees.

Näide 1:

```
1  def tervitus():
2      lause = 'Võõrustaja: "Tere!" \nKülaline: "Tere!"'
3  return lause
4
5  print(tervitus())
```

Veateade kuvatakse, sest käsk `return` on vale taandega ja seega funktsioonist väljaspool:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 3
    return lause
               ^
SyntaxError: 'return' outside function
>>>
```

Võimalik parandus:

Käsk `return` tuleks asendada õige taandega, et ta asuks funktsiooni sees.

Näide parandatud programmist:

```
1  def tervitus():
2      lause = 'Võõrustaja: "Tere!" \nKülaline: "Tere!"'
3      return lause
4
5  print(tervitus())
```

Allikas 13. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax2

5.3 SyntaxError: EOL while scanning string literal

Veateade **SyntaxError: EOL while scanning string literal** tekib siis, kui *Pythoni* programm on alustanud sõne lugemist real, kuid ei leia rea lõppu jõudes sõne lõpetavat sümbolit.

Tuleb jälgida, et sõne oleks alati ümbritsetud jutumärkide või ülakomadega.

Näide 1:

```
1  print("Sisestage arv: ")
2  arv = int(input())
3
4  if arv > 0:
5      print("Sisestatud arv on positiivne:  + str(arv))
6  else:
7      print("Sisestatud arv on mittepositiivne: " + str(arv))
```

Veateade kuvatakse, sest sõne lõpetavad jutumärgid on puudu:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 5
    print("Sisestatud arv on positiivne:  + str(arv))
                                                    ^
SyntaxError: EOL while scanning string literal
>>>
```

Võimalik parandus:

Jutumärgid tuleks lisada sõne lõppu.

Näide parandatud programmist:

```
1  print("Sisestage arv: ")
2  arv = int(input())
3
4  if arv > 0:
5      print("Sisestatud arv on positiivne: " + str(arv))
6  else:
7      print("Sisestatud arv on mittepositiivne: " + str(arv))
```

Allikas 14. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax3

5.4 SyntaxError: expected an indented block

Veateade **SyntaxError: expected an indented block** tekib siis, kui *Pythonis* pole pärast tsükli, tingimuslause, funktsiooni või klassi esimest rida lisatud taandega programmiplokki.

Tuleb jälgida, et tingimuslause sees kasutatakse igal real taanet, mis oleks sama pikkusega.

Näide 1:

```
print("Sisestage arv: ")
arv = int(input())

if arv > 0:
print("Sisestatud arv on positiivne")
else:
    print("Sisestatud arv on mittepositiivne")
```

Veateade kuvatakse, sest esimene `print` käsk peaks alustama tingimuslause sees taandega koodiplokki, kuid on sama taandega, mis tingimuslause esimene rida:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 5
    print("Sisestatud arv on positiivne")
        ^
SyntaxError: expected an indented block
>>>
```

Võimalik parandus:

Käsk `print` tuleks asendada õige taandega, et ta asuks tingimuslause sees.

Näide parandatud programmist:

```
print("Sisestage arv: ")
arv = int(input())

if arv > 0:
    print("Sisestatud arv on positiivne")
else:
    print("Sisestatud arv on mittepositiivne")
```

Tuleb jälgida, et tsüklile eelnev rida ja tsüklit alustav rida kasutaks sama pikkusega taanet, samuti peab tsükli sees kasutama ühtlast taanet.

Näide 2:

```
print("Sisestage positiivne arv: ")
arv = int(input())
i = 0

while i < arv:
    i = i + 1
    print("loendur: " + str(i))
    if arv % 2 == 0:
        print("Sisestatud arv on paarisarv")
    else:
    print("Sisestatud arv on paaritu arv")
```

Veateade kuvatakse, sest `while`-tsükli sees ei ole tingimuslause sama pikkuse taandega:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 11
    print("Sisestatud arv on paaritu arv")
        ^
SyntaxError: expected an indented block
>>>
```

Võimalik parandus:

Käsk `print` tuleks asendada õige taandega, et ta asuks tsükli sees.

Näide parandatud programmist:

```
print("Sisestage positiivne arv: ")
arv = int(input())
i = 0

while i < arv:
    i = i + 1
    print("loendur: " + str(i))
if arv % 2 == 0:
    print("Sisestatud arv on paarisarv")
else:
    print("Sisestatud arv on paaritu arv")
```

Allikas 15. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax4

5.5 SyntaxError: invalid syntax

Veateade **SyntaxError: invalid syntax** tekib siis, kui programmis on tehtud õigekirjaviga keelekonstruktsioonides. Suure tõenäosusega on puudu või vales kohas mõni sümbol (koolon, sulg, jutumärk) või on tehtud trükiviga.

Tuleb jälgida, et sulud, jutumärgid ja ülakomad ei oleks üksikult ega vales järjestuses programmi teiste sümbolitega: programmis eksisteerivad paarikaupa nii alustavad sulud/jutumärgid/ülakomad kui ka lõpetavad sulud/jutumärgid/ülakomad ning asuvad õigetel kohtadel.

Näide 1:

```
1  print("Sisestage number: ")
2  number = int(input())
3
4  print("Sisestasite numbri: + " str(number))
```

Veateade kuvatakse, sest lõpetavad jutumärgid on asetatud valesse kohta. Plussmärk, mis ühendab kahte sõne, peab asuma jutumärkidest väljaspool:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 4
    print("Sisestasite numbri: + " str(number))
                                    ^
SyntaxError: invalid syntax
>>>
```

Võimalik parandus:

Kahte sõne ühendav plussmärk tuleks lisada jutumärkidest väljaspoole.

Näide parandatud programmist:

```
1  print("Sisestage number: ")
2  number = int(input())
3
4  print("Sisestasite numbri: " + str(number))
```

Tuleb jälgida, et kooloniga tähistakse taandega programmiploki (näiteks tingimuslause või tsükli) algust.

Näide 2:

```
print("Sisestage arv: ")
arv = int(input())

if (arv > 0)
    print("Sisestatud arv on positiivne")
else:
    print("Sisestatud arv on mittepositiivne")
```

Veateade kuvatakse, sest `if` -tingimuslause lõpust on puudu koolon:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 4
    if (arv > 0)
               ^
SyntaxError: invalid syntax
>>>
```

Võimalik parandus:

Tingimuslause lõppu tuleks lisada koolon.

Näide parandatud programmist:

```
print("Sisestage arv: ")
arv = int(input())

if (arv > 0):
    print("Sisestatud arv on positiivne")
else:
    print("Sisestatud arv on mittepositiivne")
```

Allikas 16. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax5

5.6 SyntaxError: missing parentheses in call to 'print'

Veateade **SyntaxError: missing parentheses in call to 'print'** tekib siis, kui Pythoni programmis (alates Python 3 versioonist) on käsu `print` puhul puudu sisu ümbritsevad sulud.

Tuleb jälgida, et sõne oleks `print` käsu puhul ümbritsetud sulgudega.

Näide 1:

```
print "Sisestage arv: "
arv = int(input())

if arv > 0:
    print("Sisestatud arv on positiivne: " + str(arv))
else:
    print("Sisestatud arv on mittepositiivne: " + str(arv))
```

Veateade kuvatakse, sest sõne ümber ei ole `print` käsu puhul kasutatud sulge:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 1
    print "Sisestage arv: "
                          ^
SyntaxError: Missing parentheses in call to 'print'
>>>
```

Võimalik parandus:

Käsu `print` puhul tuleks sõne ümber lisada sulud.

Näide parandatud programmist:

```
print("Sisestage arv: ")
arv = int(input())

if arv > 0:
    print("Sisestatud arv on positiivne: " + str(arv))
else:
    print("Sisestatud arv on mittepositiivne: " + str(arv))
```

Allikas 17. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax6

5.7 SyntaxError: unexpected EOF while parsing

Veateade **SyntaxError: unexpected EOF while parsing** tekib siis, kui *Pythoni* programm on alustanud sisu lugemist, kuid ei leia faili lõppu jõudes käsku lõpetavat sümbolit.

Tuleb jälgida, et kõik jutumärgid/sulud oleks alati paarikaupa.

Näide 1:

```
1  print("Sisestage arv: ")
2  arv = int(input())
3
4  if arv > 0:
5      print("Sisestatud arv on positiivne: " + str(arv))
6  else:
7      print("Sisestatud arv on mittepositiivne: " + str(arv)
```

Veateade kuvatakse, sest `print` käsu lõpust on puudu lõpetav sulg:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 7
    print("Sisestatud arv on mittepositiivne: " + str(arv)
                                                          ^
SyntaxError: unexpected EOF while parsing
>>>
```

Võimalik parandus:

Sõne alustavale sulule tuleks lisada paarsuse saavutamiseks ka lõpetav sulg.

Näide parandatud programmist:

```
1  print("Sisestage arv: ")
2  arv = int(input())
3
4  if arv > 0:
5      print("Sisestatud arv on positiivne: " + str(arv))
6  else:
7      print("Sisestatud arv on mittepositiivne: " + str(arv))
```

Tuleb jälgida, et väljakommenteerimisel ei jääks mõni oluline rida tühjaks.

Näide 2:

```
print("Sisestage arv: ")
arv = int(input())

if arv > 0:
    print("Sisestatud arv on positiivne: " + str(arv))
else:
    #print("Sisestatud arv on mittepositiivne: " + str(arv))
```

Veateade kuvatakse, sest tingimuslause sees on üks rida väljakommenteeritud, seega `else`-lause on tühi:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 7
    #print("Sisestatud arv on mittepositiivne: " + str(arv))
                                                           ^
SyntaxError: unexpected EOF while parsing
>>>
```

Võimalik parandus:

`else`-lause sisuks tuleks midagi lisada, tühjaks ei tohi jätta.

Näide parandatud programmist:

```
print("Sisestage arv: ")
arv = int(input())

if arv > 0:
    print("Sisestatud arv on positiivne: " + str(arv))
else:
    #print("Sisestatud arv on mittepositiivne: " + str(arv))
    print("Siin peab midagi olema")
```

Allikas 18. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax8

5.8 SyntaxError: unexpected indent

Veateade **SyntaxError: unexpected indent** tekib siis, kui *Pythoni* programmis on teatud kohtades ebaühtlase pikkusega taane. Taanet vajavad need read, millele eelnev rida lõpeb kooloniga. Levinumad on *Pythonis* tingimuslaused, tsüklid ja funktsioonid.

Tuleb jälgida, et tingimuslause sees kasutatakse igal real sama pikkusega taanet.

Näide 1:

```
1  import re
2
3  print("Sisestage isikukood: ")
4  isikukood = int(input())
5
6  if re.match("^[1-6][0-9]{10}$", str(isikukood)):
7             print("sobib")
8                print("korrektne")
9  else:
10          print("ei sobi")
```

Veateade kuvatakse, sest `print` käskude puhul ei ole kasutatud ühtlast taanet:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 8
    print("korrektne")
    ^
SyntaxError: unexpected indent
>>>
```

Võimalik parandus:

`print` käsud tuleks asendada ühtlase taandega.

Näide parandatud programmist:

```
1  import re
2
3  print("Sisestage isikukood: ")
4  isikukood = int(input())
5
6  if re.match("^[1-6][0-9]{10}$", str(isikukood)):
7          print("sobib")
8          print("korrektne")
9  else:
10         print("ei sobi")
```

Tuleb jälgida, et tsüklit alustav rida kasutaks üleliigset taanet, samuti peab tsükli sees kasutama ühtlast taanet.

Näide 2:

```
i = 10
    while i > 0:
        print(i)
        i = i - 1
```

Veateade kuvatakse, sest `while`-tsükkel ei pea algama taandega:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 2
    while i > 0:
    ^
SyntaxError: unexpected indent
>>>
```

Võimalik parandus:

`while`-tsükli rea algusest tuleks eemaldada taane.

Näide parandatud programmist:

```
i = 10
while i > 0:
    print(i)
    i = i - 1
```

Allikas 19. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax9

5.9 SyntaxError: unindent does not match any outer indentation level

Veateade **SyntaxError: unindent does not match any outer indentation level** tekib siis, kui taane on *Pythonis* lisatud tingimuslause sisse, kuid ei sobi ülejäänud tingimuslause osadega.

Tuleb jälgida, et tingimuslause sisu oleks sama pikkuse taandega.

Näide 1:

```
print("Sisestage arv: ")
arv = int(input())

if arv < 0:
 print("Sisestatud arv on mittepositiivne")
elif arv > 0:
  print("Sisestatud arv on positiivne")
 else:
  print("Sisestatud arv on null")
```

Veateade kuvatakse, sest `else` -lause ei ole sama pikkuse taandega kui `if` -lause või `elif` -lause:

```
>>> %Run Test.py
  File "C:\Users\Kursus\Test.py", line 8
    else:
       ^
SyntaxError: unindent does not match any outer indentation level
>>>
```

Võimalik parandus:

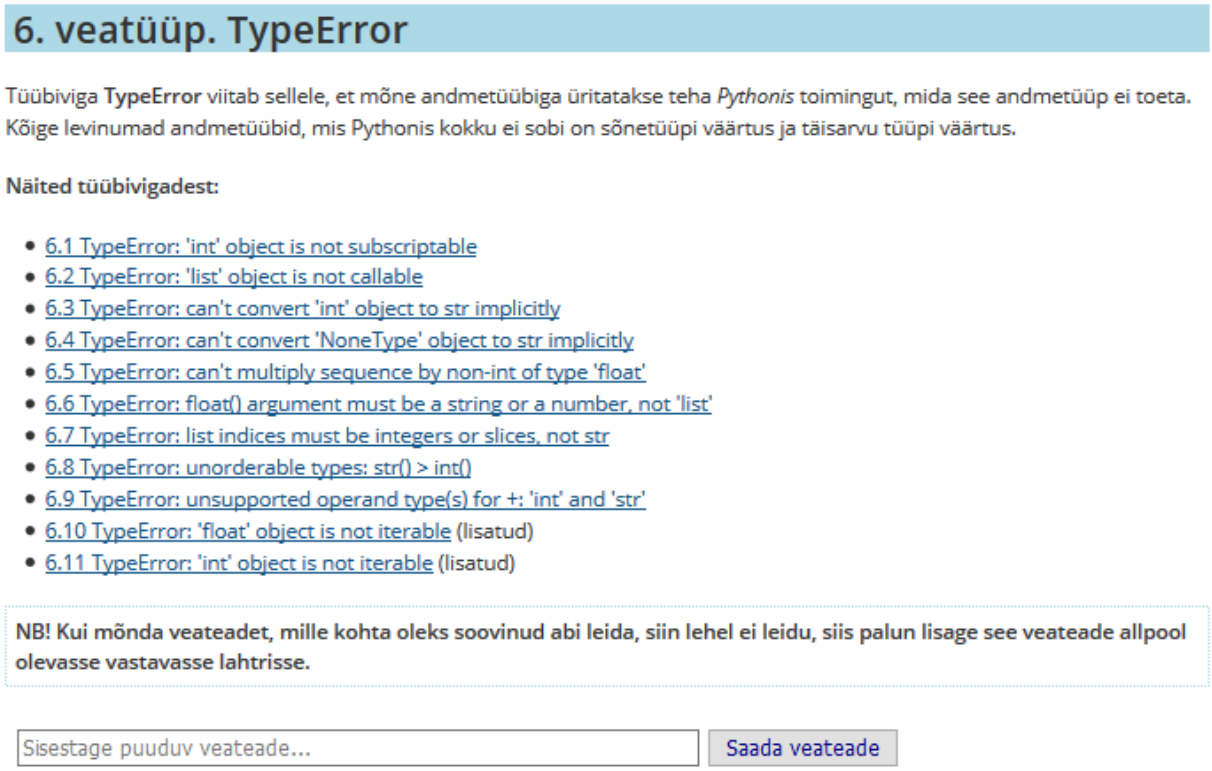
`else` -lause rea algusest tuleks eemaldada taane.

Näide parandatud programmist:

```
print("Sisestage arv: ")
arv = int(input())

if arv < 0:
 print("Sisestatud arv on mittepositiivne")
elif arv > 0:
  print("Sisestatud arv on positiivne")
else:
  print("Sisestatud arv on null")
```

Allikas 20. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Syntax10



6. veatüüp. TypeError

Tüübiviga **TypeError** viitab sellele, et mõne andmetüübiga üritatakse teha *Pythonis* toimingut, mida see andmetüüp ei toeta. Kõige levinumad andmetüübid, mis Pythonis kokku ei sobi on sõnetüüpi väärtus ja täisarvu tüüpi väärtus.

Näited tüübivigadest:

- 6.1 TypeError: 'int' object is not subscriptable
- 6.2 TypeError: 'list' object is not callable
- 6.3 TypeError: can't convert 'int' object to str implicitly
- 6.4 TypeError: can't convert 'NoneType' object to str implicitly
- 6.5 TypeError: can't multiply sequence by non-int of type 'float'
- 6.6 TypeError: float() argument must be a string or a number, not 'list'
- 6.7 TypeError: list indices must be integers or slices, not str
- 6.8 TypeError: unorderable types: str() > int()
- 6.9 TypeError: unsupported operand type(s) for +: 'int' and 'str'
- 6.10 TypeError: 'float' object is not iterable (lisatud)
- 6.11 TypeError: 'int' object is not iterable (lisatud)

NB! Kui mõnda veateadet, mille kohta oleks soovinud abi leida, siin lehel ei leidu, siis palun lisage see veateade allpool olevasse vastavasse lahtrisse.

Sisestage puuduv veateade... Saada veateade

Allikas 21. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type

6.1 TypeError: 'int' object is not subscriptable

Veateade **TypeError: 'int' object is not subscriptable** tekib siis, kui täisarvu tüüpi muutuja korral kasutatakse indeksit.

Tuleb jälgida, et indeksit saab kasutada järjendite ja sõnede puhul, kuid mitte täisarvu tüüpi muutujate puhul.

Näide 1:

```
1  käärikul = [401, 604, 547]
2  kohilas = [900, 0, 333]
3  tulemus = 0
4
5  if tulemus[2] > tulemus[2]:
6      tulemus = käärikul[2]
7  else:
8      tulemus = kohilas[2]
9
10 print("Parim tulemus viimasel katsel oli " + str(tulemus))
```

Veateade kuvatakse, sest Python ei võimalda indeksiga viidata täisarvu tüüpi muutujale:

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 5, in <module>
    if tulemus[2] > tulemus[2]:
TypeError: 'int' object is not subscriptable
>>>
```

Võimalik parandus:

Indeksiga tuleks viidata näiteks järjendi elemendile.

Näide parandatud programmist:

```
1  käärikul = [401, 604, 547]
2  kohilas = [900, 0, 333]
3  tulemus = 0
4
5  if käärikul[2] > kohilas[2]:
6      tulemus = käärikul[2]
7  else:
8      tulemus = kohilas[2]
9
10 print("Parim tulemus viimasel katsel oli " + str(tulemus))
```

Allikas 22. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type1

6.2 TypeError: 'list' object is not callable

Üldine:

Veateade **TypeError: 'list' object is not callable** tekib siis, kui järjendi elementi üritatakse väärtustada keelekonstruktsiooniliselt valesti.

Tuleb jälgida, et järjendite puhul kasutatakse indeksiga elementide otsimisel nurksulge, mitte ümarsulge.

Näide 1:

```
1  tooted = input("Sisestage tooted: ").split(',')
2  toode = int(input("Valige toote järjekorra number: "))
3
4  if 0 < toode <= len(tooted):
5      print(tooted(toode - 1))
6  else:
7      print("Tooteid pole piisavalt!")
```

Veateade kuvatakse, sest järjendi elemendi otsimisel kasutatakse ümarsulge:

```
>>> %Run Test.py
Sisestage tooted: leib, sai, piim
Valige toote järjekorra number: 2
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 5, in <module>
    print(tooted(toode - 1))
TypeError: 'list' object is not callable
>>>
```

Võimalik parandus:

Järjendi elemendi otimisel tuleks kasutada nurksulge.

Näide parandatud programmist:

```
1  tooted = input("Sisestage tooted: ").split(',')
2  toode = int(input("Valige toote järjekorra number: "))
3
4  if 0 < toode <= len(tooted):
5      print(tooted[toode - 1])
6  else:
7      print("Tooteid pole piisavalt!")
```

Allikas 23. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type2

6.3 TypeError: can't convert 'int' object to str implicitly

Veateade **TypeError: can't convert 'int' object to str implicitly** tekib siis, kui *Pythonis* üritatakse ühendada täisarvu tüüpi muutujat sõnetüüpi tekstiga.

Tuleb jälgida, et täisarvu tüüpi muutuja oleks teisendatud sõnetüüpi muutujaks enne tekstiga ühendamist.

Näide 1:

```
1  print("Sisestage nädala palk: ")
2  palk = int(input())
3
4  print ("Teie selle nädala palk on " + palk + " eurot")
```

Veateade kuvatakse, sest `palk` on täisarvu tüüpi muutuja ja tekst, mida muutujaga siduda tahetakse, on sõnetüüpi:

```
>>> %Run Test.py
Sisestage nädala palk:
200
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 4, in <module>
    print ("Teie selle nädala palk on " + palk + " eurot")
TypeError: Can't convert 'int' object to str implicitly
>>>
```

Võimalik parandus:

Muutuja `palk` tuleks teisendada sõneks `print` käsu sees.

Näide parandatud programmist:

```
1  print("Sisestage nädala palk: ")
2  palk = int(input())
3
4  print ("Teie selle nädala palk on " + str(palk) + " eurot")
```

Allikas 24. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type3

6.4 TypeError: can't convert 'NoneType' object to str implicitly

Veateade **TypeError: can't convert 'NoneType' object to str implicitly** tekib siis, kui funktsioonil puudub `return` käsk. Sellisel juhul on selle funktsiooni väärtus Pythonis tühi (*NoneType*) ja seega ei ole võimalik funktsiooni ühendada sõnetüüpi väärtusega.

Tuleb jälgida, et funktsiooni puhul oleks olemas ka tagastuskäsk `return`.

Näide 1:

```
1  def vali_toode(sisend):
2      print("Tere!")
3
4  sisend = input("Millist toodet soovite? ")
5
6  print("Väljastan toote: " + vali_toode(sisend) + "!")
```

Veateade kuvatakse, sest funktsioon ilma `return` käsuta on *Pythoni* jaoks tühi andmetüüp:

```
>>> %Run Test.py
Millist toodet soovite? jahu
Tere!
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 6, in <module>
    print("Väljastan toote: " + vali_toode(sisend) + "!")
TypeError: Can't convert 'NoneType' object to str implicitly
>>>
```

Võimalik parandus:

Funktsioonile tuleks lisada ka tagastuskäsk `return`.

Näide parandatud programmist:

```
1  def vali_toode(sisend):
2      print("Tere!")
3      return sisend
4
5  sisend = input("Millist toodet soovite? ")
6
7  print("Väljastan toote: " + vali_toode(sisend) + "!")
```

Allikas 25. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type4

6.5 TypeError: can't multiply sequence by non-int of type 'float'

Veateade **TypeError: can't multiply sequence by non-int of type 'float'** tekib siis, kui üritatakse *Pythonis* omavahel korrutada sõne/järjendit ja ujukomaarvu.

Tuleb jälgida, et Pythonis on võimalik korrutada sõne ja täisarvu.

Näide 1:

```
1  kaugus = input("Sisestage kaugus televiisorini (meetrites): ")
2
3  print("Televiisori diagonaal peaks olema " + str(round(kaugus * 100 * 0.39 / 2.5)) + " tolli")
```

Veateade kuvatakse, sest sisend on sõnetüüpi muutuja ja korrutatav tehte liige ujukomaarvu tüüpi muutuja:

```
>>> %Run Test.py
Sisestage kaugus televiisorini (meetrites): 3
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 3, in <module>
    print("Televiisori diagonaal peaks olema " + str(round(kaugus * 100 * 0.39 / 2.5)) + " tolli")
TypeError: can't multiply sequence by non-int of type 'float'
>>>
```

Võimalik parandus:

Muutuja `kaugus` tuleks teisendada täisarvu tüüpi väärtuseks.

Näide parandatud programmist:

```
1  kaugus = int(input("Sisestage kaugus televiisorini (meetrites): "))
2
3  print("Televiisori diagonaal peaks olema " + str(round(kaugus * 100 * 0.39 / 2.5)) + " tolli")
```

Allikas 26. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type5

6.6 TypeError: float() argument must be a string or a number, not 'list'

Veateade **TypeError: float() argument must be a string or a number, not 'list'** tekib siis, kui funktsiooni `float` argumendiks antakse ette järjend, mis ei ole sobiv andmetüüp sellele funktsioonile. Sobivad on näiteks sõne või täisarv.

Tuleb jälgida, et järjendi puhul ei võimalda Python kasutada `float` funktsiooni.

Näide 1:

```
1  arvud = [5, 33, 2.6, 18, 35, 105.55, 99.99]
2  summa = 0
3
4  for arv in float(arvud):
5      if arv > 38:
6          summa += arv
7
8  print(summa)
```

Veateade kuvatakse, sest `float` funktsiooni argumendiks antakse järjend:

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 4, in <module>
    for arv in float(arvud):
TypeError: float() argument must be a string or a number, not 'list'
>>>
```

Võimalik parandus:

Järjendi puhul tuleks eemaldada ujukomaarvuks tehtav teisendus.

Näide parandatud programmist:

```
1  arvud = [5, 33, 2.6, 18, 35, 105.55, 99.99]
2  summa = 0
3
4  for arv in arvud:
5      if arv > 38:
6          summa += arv
7
8  print(summa)
```

Allikas 27. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type6

6.7 TypeError: list indices must be integers or slices, not str

Veateade **TypeError: list indices must be integers or slices, not str** tekib siis, kui järjendi indeksiks antakse sõnetüüpi väärtus. Sobivad indeksi väärtused on *Pythonis* täisarvu tüüpi väärtus ja lõigutüüpi väärtus.

Tuleb jälgida, et järjendite puhul ei kasutataks indeksina sõnetüüpi väärtusi.

Näide 1:

```
1  tooted = input("Sisestage tooted: ").split(',')
2  toode = input("Valige toode: ")
3
4  print(tooted[toode])
```

Veateade kuvatakse, sest `toode` on sõnetüüpi väärtus ja ebasobiv andmetüüp järjendi elementidele viitamisel:

```
>>> %Run Test.py
Sisestage tooted: leib, sai, piim
Valige toode: sai
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 4, in <module>
    print(tooted[toode])
TypeError: list indices must be integers or slices, not str
>>>
```

Võimalik parandus:

Muutuja `toode` tuleks teisendada täisarvu tüüpi muutujaks.

Näide parandatud programmist:

```
1  tooted = input("Sisestage tooted: ").split(',')
2  toode = int(input("Valige toote järjekorra number: "))
3
4  print(tooted[toode - 1])
```

Tuleb jälgida, et järjendi lõikude märgistamise puhul ei kasutataks jutumärke.

Näide 2:

```
1  tooted = ['sai','või','leib','piim','keefir']
2
3  print(tooted[':'])
```

Veateade kuvatakse, sest järjendi kõikide elementide kuvamiseks ei ole tarvis koolonile jutumärke ümber lisada:

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 3, in <module>
    print(tooted[':'])
TypeError: list indices must be integers or slices, not str
>>>
```

Võimalik parandus:

Listi kõigi elementide välja printimisel tuleks eemaldada kooloni ümbert jutumärgid.

Näide parandatud programmist:

```
1  tooted = ['sai','või','leib','piim','keefir']
2
3  print(tooted[:])
```

Allikas 28. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type7

6.8 TypeError: unorderable types: str() > int()

Veateade **TypeError: unorderable types: str() > int()** tekib siis, kui üritatakse *Pythonis* võrrelda sõnetüüpi väärtust ja täisarvu tüüpi väärtust.

Tuleb jälgida, et arvuliste väärtuste võrdluse puhul ei kasutataks sõnetüüpi väärtusi.

Näide 1:

```
1  tunnid = input("Sisesta nädala tundide arv: ")
2  tasu = input("sisesta tunnitasu: ")
3  palk = int(tunnid) * int(tasu)
4
5  if tunnid > 40:
6       lisa = tunnid - 40
7       lisa = int(lisa) * tasu * 0.5
8       palk = palk + lisa
9
10 print(int(palk))
```

Veateade kuvatakse, sest funktsioon `input` tagastab sõnetüüpi väärtuse ja seda üritatakse võrrelda täisarvuga:

```
>>> %Run Test.py
Sisesta nädala tundide arv: 40
sisesta tunnitasu: 5
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 5, in <module>
    if tunnid > 40:
TypeError: unorderable types: str() > int()
>>>
```

Võimalik parandus:

Muutuja `tunnid` tuleks teisendada täisarvu tüüpi muutujaks.

Näide parandatud programmist:

```
1  tunnid = int(input("Sisesta nädala tundide arv: "))
2  tasu = int(input("sisesta tunnitasu: "))
3  palk = tunnid * tasu
4
5  if tunnid > 40:
6       lisa = tunnid - 40
7       lisa = int(lisa) * tasu * 0.5
8       palk = palk + lisa
9
10 print(int(palk))
```

Allikas 29. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type8

6.9 TypeError: unsupported operand type(s) for +: 'int' and 'str'

Veateade **TypeError: unsupported operand type(s) for +: 'int' and 'str'** tekib siis, kui *Pythonis* üritatakse ühendada sõnetüüpi muutujat ja täisarvu tüüpi muutujat.

Tuleb jälgida, et kahe muutuja ühendamisel oleks nende andmetüüp sama.

Näide 1:

```
1  aasta = 2016
2  eesnimed = "Leonardo Wilhelm"
3  perenimi = "DiCaprio"
4
5  print(aasta + " " + eesnimed + " " + perenimi)
```

Veateade kuvatakse, sest Python ei võimalda ühendada täisarvu tüüpi muutujat sõnetüüpi muutujaga:

```
>>> %Run Test.py
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 5, in <module>
    print(aasta + " " + eesnimed + " " + perenimi)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

Võimalik parandus:

Muutuja `aasta` tuleks teisendada `print` käsu sees sõnetüüpi muutujaks.

Näide parandatud programmist:

```
1  aasta = 2016
2  eesnimed = "Leonardo Wilhelm"
3  perenimi = "DiCaprio"
4
5  print(str(aasta) + " " + eesnimed + " " + perenimi)
```

Allikas 30. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type9

6.10 TypeError: 'float' object is not iterable

Veateade **TypeError: 'float' object is not iterable** tekib siis, kui *for*-tsükli koostamisel kasutatakse ujukomaarvu tüüpi muutujat tsükli pikkuse määramiseks.

Tuleb jälgida, et tsükli pikkuse määramisel kasutatakse *Pythonis* funktsiooni `range()`, mille sisendiks antakse täisarvu tüüpi muutuja

Näide 1:

```
1  kordaja = 6
2  arv = float(input("Sisestage arv: "))
3  for loendur in arv:
4      print("Arvu " + str(arv) + " astendamisel " + str(loendur) + "-ga saadud väärtus on: " + str(arv
   ** loendur)
```

Veateade kuvatakse, sest tsükli kordamise arvu määramisel on jäetud lisamata funktsioon `range()`, mis on vajalik tsükli kordamiseks etteantud täisarv kordi:

```
>>> %Run test.py
Sisestage arv: 5
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 3, in <module>
    for loendur in arv:
TypeError: 'float' object is not iterable
>>>
```

Võimalik parandus:

Pythonis on vajalik funktsiooni lisamine tsüklisammude kordamiseks.

Näide parandatud programmist:

```
1  kordaja = 6
2  arv = float(input("Sisestage arv: "))
3  for loendur in range(kordaja):
4      print("Arvu " + str(arv) + " astendamisel " + str(loendur) + "-ga saadud väärtus on: " + str(arv
   ** loendur)
```

Allikas 31. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type10

6.11 TypeError: 'int' object is not iterable

Veateade **TypeError: 'int' object is not iterable** tekib siis, kui *for*-tsükli koostamisel kasutatakse täisarvu tüüpi muutujat tsükli pikkuse määramiseks.

Tuleb jälgida, et tsükli pikkuse määramisel kasutatakse *Pythonis* funtsiooni `range()`, mille sisendiks antakse täisarvu tüüpi muutuja

Näide 1:

```
1  arv = int(input("Sisestage tsükli pikkus: "))
2  for loendur in arv:
3      print(loendur)
```

Veateade kuvatakse, sest *for*-tsükli pikkuse määramisel on jäetud lisamata funktsioon `range()`, mis on vajalik tsükli kordamiseks etteantud täisarv kordi:

```
>>> %Run test.py
Sisestage arv: 5
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 2, in <module>
    for loendur in arv:
TypeError: 'int' object is not iterable
```

Võimalik parandus:

Pythonis on vajalik funktsiooni lisamine tsüklisammude kordamiseks.

Näide parandatud programmist:

```
1  arv = int(input("Sisestage tsükli pikkus: "))
2  for loendur in range(arv):
3      print(loendur)
```

Allikas 32. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Type11

Allikas 33. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Value

7.1 ValueError: could not convert string to float

Veateade **ValueError: could not convert string to float** viitab sellele, et Python ei võimalda teisendada sõnetüüpi väärtust ujukomaarv tüüpi väärtuseks.

Tuleb jälgida, et ujukomaarvu puhul kasutatakse *Pythonis* punkti.

Näide 1:

```
1  pikkus = float(input("Sisestage pikkus: "))
2
3  print("Sisestatud pikkus: " + str(pikkus))
```

Veateade kuvatakse, sest funktsioon `float(input())` eeldab sisendiks ujukomaarvu tüüpi väärtust, kuid saab sõnetüüpi väärtuse:

```
>>> %Run Test.py
Sisestage pikkus: 172,5
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 1, in <module>
    pikkus = float(input("Sisestage pikkus: "))
ValueError: could not convert string to float: '172,5'
>>>
```

Võimalik parandus:

Programm on tegelikult korrektne, kasutajal tuleks sisestada arv, mille eraldajaks on punkt, mitte koma.

Näide parandatud programmist:

```
1  pikkus = float(input("Sisestage pikkus (kasutage eraldajana punkti): "))
2
3  print("Sisestatud pikkus: " + str(pikkus))
```

Allikas 34. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Value1

7.2 ValueError: empty separator

Veateade **ValueError: empty separator** tekib siis, kui sõne tükeldamisel ei ole määratud eraldajat, mis määrab, millistest kohtadest tükeldada.

Tuleb jälgida, et sõne tükeldamisel `split` funktsiooniga oleks määratud ka eraldaja.

Näide 1:

```
tooted = input("Sisestage tooted: ").split('')
toode = int(input("Valige toote järjekorra number: "))

if 0 < toode <= len(tooted):
    print(tooted[toode - 1])
else:
    print("Tooteid pole piisavalt!")
```

Veateade kuvatakse, sest funktsiooni `split` puhul on eraldaja tühi (määramata):

```
>>> %Run Test.py
Sisestage tooted: leib, piim, sai
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 1, in <module>
    tooted = input("Sisestage tooted: ").split('')
ValueError: empty separator
>>>
```

Võimalik parandus:

Funktsioonile `split` tuleks määrata ka eraldaja (näiteks koma).

Näide parandatud programmist:

```
tooted = input("Sisestage tooted: ").split(',')
toode = int(input("Valige toote järjekorra number: "))

if 0 < toode <= len(tooted):
    print(tooted[toode - 1])
else:
    print("Tooteid pole piisavalt!")
```

Allikas 35. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Value2

7.3 ValueError: I/O operation on closed file

Veateade **ValueError: I/O operation on closed file** tekib siis, kui *Pythoni* programm üritab etteantud faili sisu lugeda, kuid fail on selleks hetkeks juba sulgemiskäsu saanud.

Tuleb jälgida, et faili sulgemisfunktsioon `close` oleks tsükli sees õige taandega.

Näide 1:

```
1  nimi = input("Sisestage faili nimi koos laiendiga: ")
2  fail = open(nimi, encoding="UTF-8")
3  for rida in fail:
4      print(rida)
5      fail.close()
```

Veateade kuvatakse, sest fail sulgetakse peale tsükli esimest sammu:

```
>>> %Run Test.py
Sisestage faili nimi koos laiendiga: andmed.txt
must, kollane, punane, valge, roheline
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 3, in <module>
    for rida in fail:
ValueError: I/O operation on closed file.
>>>
```

Võimalik parandus:

Faili sulgemisfunktsioon `close` tuleks tsüklist väljapoole liigutada.

Näide parandatud programmist:

```
1  nimi = input("Sisestage faili nimi koos laiendiga: ")
2  fail = open(nimi, encoding="UTF-8")
3  for rida in fail:
4      print(rida)
5  fail.close()
```

Allikas 36. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Value3

7.4 ValueError: invalid literal for int() with base 10: '7.5'

Veateade **ValueError: invalid literal for int() with base 10: '7.5'** tekib siis, kui funktsioonile `int` üritatakse anda argumendiks ujukomaarvu tüüpi argumenti.

Tuleb jälgida, et täisarvu tüüpi väärtuse kitsendamisel ei sisestataks mõnda muu andmetüübi väärtust.

Näide 1:

```
1  arv = int(input("Sisestage arv: "))
2
3  print(arv)
```

Veateade kuvatakse, sest funktsioon `int(input())` nõuab täisarvu tüüpi argumenti, kuid antud näites on sisestatud ujukomaarvu tüüpi argument:

```
>>> %Run Test.py
Sisestage arv: 7.5
Traceback (most recent call last):
  File "C:\Users\Kursus\Test.py", line 1, in <module>
    arv = int(input("Sisestage arv: "))
ValueError: invalid literal for int() with base 10: '7.5'
>>>
```

Võimalik parandus:

Programm on tegelikult korrektne, kasutajal tuleks sisestada täisarvu tüüpi sisend.

Näide parandatud programmist:

```
1  arv = int(input("Sisestage arv: "))
2
3  print(arv)
```

Allikas 37. https://courses.cs.ut.ee/2016/eprogalused/spring/Main/Value4

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Raigo Kodasmaa**,

(*autori nimi*)

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

Programmeerimiskeele Python veateated programmeerimise algõppes,

(*lõputöö pealkiri*)

mille juhendaja on **Eno Tõnisson**,

(*juhendaja nimi*)

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

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

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

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

Tartus, 09.01.2017