Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

Villem Tõnisson

Programmeerimisülesannete lahendamisel tekkivate Thonny logifailide korrastamine

Bakalaureusetöö (9 EAP)

Juhendajad: Eno Tõnisson, PhD

Heidi Meier, MSc

Tartu 2019

Programmeerimisülesannete lahendamisel tekkivate Thonny logifailide korrastamine

Lühikokkuvõte:

Käesoleva bakalaureusetöö raames valmis Pythoni-programm Thonny programmeerimiskeskkonna koostatud logifailide korrastamiseks. Korrastamise all mõeldakse logifailide lahtipakkimist arhiivifailidest ning ülesannete kaupa eraldamist. Programm loodi eesmärgiga lihtsustada logide kontrollimist ainetes, kus kasutatakse Thonnyt, ja et lihtsustada logide korrastamist nende uurimiseks teadustöös. Käesolev töö annab ülevaate programmi võimalustest ja kasutamisest.

Võtmesõnad:

Logifailid, Thonny, Python

CERCS: P175 Informaatika, süsteemiteooria, S281 Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

Tidying Log Files Created by Solving Programming Tasks in Thonny

Abstract:

As part of this Bachelor's thesis a Python program was developed to help tidy log files created by the Thonny IDE. Tidying in this context means unpacking compressed files and splitting log files based on which Python files were written into. The program was created with the purpose of simplifying checking user logs in courses where Thonny is the standard IDE and to ease use of logs in scientific research. This Bachelor's thesis gives an overview of how to use the program and what can be done with it.

Keywords:

Log files, Thonny, Python

CERCS: P175 Informatics, systems theory, S281 Computer-assisted education

Sisukord

1 Sissejuhatus

Huvi programmeerimise õppimise vastu on praegu suurem kui kunagi varem ja võimalik, et see kasvab veel. Näiteks Tartu Ülikooli aines "Programmeerimine" oli 2018. a sügissemestril ligi 400 üliõpilast. Suur on osalejate arv olnud ka Tartu Ülikooli programmeerimise veebipõhistel masskursustel (MOOC), kus osaleb sadu inimesi. Lisaks sellele on programmeerimine kogumas populaarsust ka väljaspool ülikoole, näiteks gümnaasiumites ning ka põhikoolides. Programmeerimise õpetamisel on suur rõhk programmeerimis-ülesannete lahendamisel. Tavaliselt esitatakse ülesannete puhul lahenduseks vaid lõpptulemus ehk programmi lähtekood, kuid hiljuti on hakatud nõudma ka lahenduse logifaile, et kontrollida lahendamise ausust. Lisaks sellele saab kasutada logifaile õpilaste lahenduskäikude uurimiseks ja sellele vastavalt õppematerjalide täiendamiseks.

Aivar Annamaa loodud Thonny on algajatele mõeldud Pythoni programmeerimiskeskkond, mida korraldajad soovitavad kasutada ja mis on kontrolltööde ajal ka kohustuslik programmeerimiskeskkond mitmes Tartu Ülikooli õppeaines. Thonny salvestab kõik kasutaja tegevused koos ajatemplitega logifaili. Tänu sellele saab täpselt näha, mida ja mis järjekorras kasutaja tegi. Salvestatud logifailides peitub küll palju infot, kuid kahjuks pole hetkel olemas viisi paljude logifailide korraga mugavalt ja lihtsalt töötlemiseks. Juhendajad kasutavad vaid üksiku logifaili lahendusprotsessi taasesitamist, mis võib suuremate logifailidega palju aega võtta. Uus logifail luuakse igal Thonny sulgemisel ja sinna salvestatakse kogu info tegevustest, mis toimusid selle Thonny käivituse ajal. Seega võib olla ühe ülesande lahendus laiali jaotatud mitme logifaili peale ja samas võib ühes logifailis olla ka mitme ülesande lahendus. Kui igas logifailis oleks ühe ülesande lahendus, oleks nende läbivaatamine lihtsam. Selle bakalaureusetöö raames valmis programm, mis võimaldab kasutajal eraldada ühest logifailist mitu logifaili, nii et igas uues logifailis oleks info vaid ühe ülesande lahenduse kohta.

Thonny logifailide kohta on kirjutatud mitu magistritööd ja ka bakalaureusetööd. Magistritöödes on uuritud õppijate käitumismustreid [1], erinevate veateadete tüüpe ja esinemist [2]. Samuti on kirjutatud bakalaureusetöö logifailide analüüsi kohta [3], kus loodi logifailide põhjal CSV-faile, kus oli infot logifailis toimunu kohta, näiteks veateadete arv ja

programmi käivitamiste arv. Lisaks on kirjutatud bakalaureusetöö logifailide koostamise ja koodimuudatuste tuvastamise kohta, mille raames täiendati Thonny logimise funktsiooni [4].

2 Töö eesmärk

Selle bakalaureusetöö eesmärk on luua programm, mis suudaks abistada kasutajaid logifailide korrastamisel, milleks on logide lahtipakkimine arhiivifailidest kellaaegade järgi ja logide eraldamine Pythoni failide järgi. Kuna logifaile esitatakse tihti arhiivifailidena, peab programm olema suuteline lahti pakkima populaarsemates vormingutes arhiivifaile ning suutma eraldada paljude logide seast ette antud kellaajal alustatud logifailid. Kellaaja järgi on logisid vaja eraldada, kuna siis saab teatud olukordades paremini leida asjakohased logifailid. Näiteks, kui kontrolltöö kestis kindlal päeval 12:15-13:45, siis saab lahti pakkida vaid failid, mille tegevus toimub selles vahemikus. Samuti peaks programm suutma mitut logi ühendada ja vajadusel ka eraldada, juhul, kui ühes logifailis on informatsiooni mitme ülesande kohta.

Programmi sisendiks võivad olla kas logifailid txt-vormingus või arhiivifailid, milles on logifailid. Samuti on kasutajal valida, kas eraldada logifaile kellaaegade järgi. Väljundiks on logid, kus on igas failis info vaid ühe ülesande lahenduse kohta.

2018/19. õa sügissemestri jooksul esitati Tartu Ülikooli aines “Programmeerimine” logisid järgmistes andmeformaatides:

- “txt”, mis on kõige tavalisem tekstifail ja kus on sees vaid üks logi,
- “zip”, “rar” ja “7z”, mis on populaarsed arhiiviformaadid.

Nende andmeformaatidega peaks bakalaureusetöö raames valmiv programm hakkama saama.

Programmi potentsiaalseteks kasutajateks on õppeainete “Programmeerimine” ja “Programmeerimise alused” juhendajad ning kõik õpetajad, kes kasutavad oma aines Thonnyt. Nad saavad programmi abil säästa aega, mis muidu kuluks arhiivifailide lahtipakkimiseks ja failide sorteerimiseks. Lisaks sellele saab valminud programmi kasutada ka uurimistöödes, et lihtsustada suures koguses andmetega töötamist ja et luua samas vormingus logifaile, mida saaks uurida.

Selle bakalaureusetöö raames valmis programm, mis suudab arhiivifailidest eraldada nii kõiki logifaile kui ka neid, mida alustati kindlas ajavahemikus. Samuti saab logidest eraldada osasid, nii et igas uues logis oleks info vaid ühe ülesande lahenduse kohta. Programm on autori oma looming, välja arvatud kood, mis on failis *datepicker.py* [8].

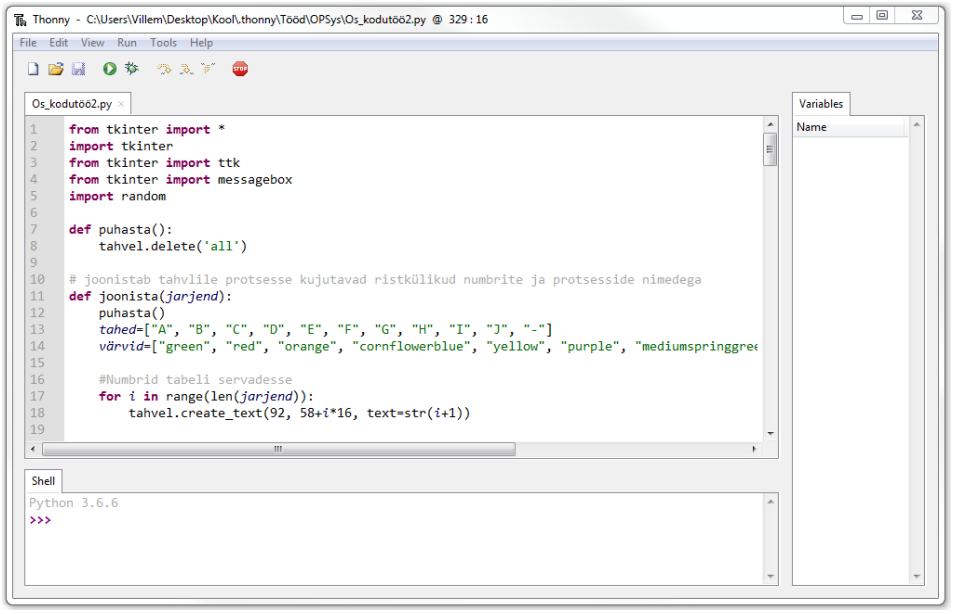
3 Thonny ja logifailide kirjeldus

Selles peatükis on kirjeldatud programmeerimiskeskkonda Thonny ja logifaile, mis Thonny kasutamisel tekivad.

3.1 Thonny

Thonny on eelkõige algajatele ja nende õpetajatele mõeldud tasuta Pythoni programmeerimiskeskkond, mis on loodud Tartu Ülikoolis Aivar Annamaa poolt. Thonny töötab nii Windowsi, Maci, kui ka Linuxi operatsioonisüsteemidel [5]. Tihti on programmeerimiskeskkondadel liiga palju valikuid ja see ajab algajaid segadusse. Thonny on loodud eesmärgiga olla võimalikult lihtne ja tänu sellele on algajatele vajalik funktsionaalsus kergesti leitav.

Thonnys on olemas koodiredaktoriga samas aknas ka Pythoni käsurida. See lihtsustab programmeerijate tööd, kuna seal saab proovida lühikesi koodijuppe, ilma et selleks oleks vaja avada uut akent. Thonnyl on olemas ka sisseehitatud pakettide paigaldaja, mis teeb vajalike pakettide paigaldamise lihtsaks ja intuitiivseks. Samuti on Thonnyl olemas silur, mis võimaldab samm-sammult läbi mängida programmi tööd ja selle abil seda paremini mõista.



Joonis 1. Thonny graafiline liides.

Thonnyl on olemas ka logifailide loomise funktsionaalsus, mis on selle bakalaureusetöö fookuses. Peaaegu kõik kasutaja tegevused jäädvustatakse ja salvestatakse Thonny sulgemisel

faili, mis asetatakse kasutajalogide kausta. Jäädvustamata jäetakse näiteks koodiakna vahetamine. Kuna bakalaureusetöö eesmärk on logifailide korrastamine, siis on neid täpsemalt kirjeldatud järgnevas alapeatükis.

3.2 Logifailid

3.2.1 Logifailide ülesehitus ja vorming

Thonny logid on salvestatud tekstifailidesse (.txt), mis on UTF-8 kodeeringus ja JSON vormingus. JSON on vorming, mille erinevad andmetüübid on kirje, massiiv, väärtus, arv, sõne ja tõeväärtus [6]. Väärtus on üldmõiste, mis hõlmab kõiki teisi andmetüüpe. Arvud võivad olla kas täisarvud või ujukomaarvud. Sõne on jutumärkide vahel asetsev märkide jada. Massiiv on andmestruktuur, mis koosneb järjestikustest väärtustest, mis on eraldatud komadega ja ei pruugi olla sama tüüpi. Kirje on loogeliste sulgude vahel asuv kogum võti-väärtus paaridest, kus võtmeks on sõne. Iga Thonny logifail koosneb kirjete massiivist.

Joonis 2. Näide väikesest osast logifailist.

3.2.2 Kirjete ja nende väljade kirjeldused

Igal kirjel on kaks välja, mis on alati olemas: “time”, mis näitab aega ja “sequence”, mis näitab, mis tegevust kasutaja tegi. Aeg on vormingus “%Y-%m-%dT%H:%M:%S.%f”, näiteks “2016-12-16T22:27:48.278732”, kus “%Y” on 4-kohaline aastaarv, “%m” on

2-kohaline kuu arv, “%d” on 2-kohaline päeva arv, “%H” on 2-kohaline tundide arv, “%M” on 2-kohaline minutite arv, “%S” on 2-kohaline sekundite arv ja “%f” on 6-kohaline mikrosekundite arv. Väljade arv ei ole fikseeritud ja sõltub sequence väljast.

Järgnevad mõned võimalikud sõned, mis võivad olla väärtuseks sequence väljale: "HideView", "<<Cut>>", "TextInsert", "NewFile", "<FocusIn>", "Save", "<<Copy>>", "<Button-3>", "SaveAs", "<<Paste>>", "<FocusOut>", "Open", "ShellCommand", "ShowView", "TextDelete", "<Button-2>", "Command", "<Button-1>", "EditorTextCreated", "<<Undo>>", "ShellInput".

Järgnevad mõned võtmed, mis on logide kirjetes: "denied", "widget_class", "filename", "index", "editor_id", "index2", "command_id", "widget_id", "tags", "text_widget_id", "view_class", "command_text", "sequence", "view_id", "editor_class", "index1", "input_text", "text_widget_class", "text", "text_widget_context", "time".

Bakalaureusetöö raames valminud programm kasutab võtmeid "filename", "text_widget_id" ja "text_widget_class". "Filename" väljades on kirjas failitee Pythoni failini, millega tegeleti. "Filename" väli on näiteks olemas "SaveAs" ja "Open" ehk siis faili salvestamise ja avamise tegevuste juures. Välja "text_widget_class" kasutatakse selleks, et oleks aru saada, mis alamaknasse kirjutati. Alamaknad on kõik väljad, kuhu saab kasutaja kirjutada, näiteks koodiaknad või Pythoni käsurida. Mõned võimalikud väärtused on “CodeViewText” ning “ShellText”, mis tähendavad sisestamist koodiaknasse ja Pythoni käsureale. Igal alamaknal on oma unikaalne “text_widget_id”. See on 8-kohaline arv, mis genereeritakse suvaliselt iga kord, kui käivitatakse uus alamaken. Ühe käivituse jooksul on akna “text_widget_id” alati sama. Seega, kui me saame teada, et aknas, mille “text_widget_id” on 12345678, tegeleti failiga test.py, teame seda, et iga kirje samas logifailis, mille “text_widget_id” on 12345678, tegeleb sama failiga.

4 Programmi töö

Selles peatükis on kirjeldatud valminud programmi tööd, kuidas seda kasutada ja mida sellega teha saab.

4.1 Programmi töö kirjeldus

Selle bakalaureusetöö raames valmis programm, mis on mõeldud logifailide kontrollimise lihtsustamiseks. Programmiga saab arhiivifailidest eraldada logifaile nende loomise aja järgi. Samuti on võimalik logifailidest eraldada osasid, mis tegelevad vaid kindlatesse Pythoni failidesse kirjutamisega.

4.1.1 Programmi kasutamine

Programmi kasutamiseks tuleb läbida järgmised sammud:

1. Valida töörežiim “Select mode” alt. Täpsem töörežiimide kirjeldus on alampeatükis 4.2.
2. Soovi korral lülitada sisse lisavalikud. Täpsem lisavalikute kirjeldus on alampeatükis 4.3.
3. Kui on valitud aegade järgi eraldamine, tuleb sisestada kuupäevad ja kellaaeg.
4. Valida fail(id), vajutades nupule "Pick File". Kui valitud on mitme logifaili või arhiivi töörežiim, saab valida mitu faili, hoides all “Ctrl” või “Shift” klahve klaviatuuril.
5. Vajutada "Unpack" nuppu.

Seejärel tekivad lahtipakitud failid kausta, millel on sama nimi kui arhiivifailil. Näiteks kui arhiivifaili nimi on *näidis.zip*, siis logid jäävad kausta *näidis*. Kui eraldada logidest osasid, jäävad eraldatud logifailid samasse kausta kui algne logifail.

4.1.2 Programmi käivitamine

Kõige uuema versiooni programmist saab tõmmata GitHubi lehelt (Lisa II). Programmi kasutamiseks tuleb alla laadida ja paigaldada ka 7-Zip. Allalaadimise lingi leiab 7-Zipi kodulehelt [7]. Peale 7-Zipi paigaldamist tuleb muuta programmiga kaasa tulevat

settings.json faili. *Settings.json* failis tuleb “loc_7z” väärtuseks panna *7z.exe* asukoht enda arvutis. Käivitamiseks peavad olema ühes kaustas failid *datepicker.py*, *GUI.py*, *splitter.py* ja *unpacker.py*. Programmi käimapanemiseks tuleb käivitada *GUI.py*. Programmi käivitamise ja kasutamise juhend on olemas ka programm GitHubi lehel (Lisa II).

Joonis 3. Programmi avaaken.

4.2 Töörežiimid

Programmi tööks tuleb valida üks töörežiim. Vaikimisi on valitud ühe arhiivifaili töörežiim.

4.2.1 Üks logifail (*One logfile*)

Ühe logifaili töörežiim on mõeldud ühest logifailist ülesannete eraldamiseks. Selle kasutamiseks peab olema sisse lülitatud “Pythoni failide eraldamise” lisavalik. Eraldatud ülesannete logid tekivad samasse kausta, kus algne logi asub.

4.2.2 Üks arhiivifail (*One archive*)

Ühe arhiivifaili töörežiim on mõeldud ühe arhiivifaili lahtipakkimiseks ja sealasuvate logifailide töötlemiseks. Kasutada saab mõlemat lisavalikut. Kaust lahtipakitud failidega tekib samasse kausta, kus on arhiivifail, ja selle nimeks saab arhiivifaili nimi. Näiteks kui on fail *näidis.zip*, siis logid jäävad kausta *näidis*.

4.2.3 Mitu logifaili (*Multiple logfiles*)

Mitme logifaili töörežiim on mõeldud mitmest logifailist ülesannete eraldamiseks. See käitub samamoodi kui ühe logifailiga, kuid faili valides saab võtta mitu faili. Kasutada saab vaid "Pythoni failide eraldamise" lisavalikut.

4.2.4 Mitu arhiivifaili (*Multiple archives*)

Mitme arhiivifaili töörežiim on mõeldud mitme arhiivifaili lahtipakkimiseks. See režiim käitub samamoodi kui ühe arhiivifailiga, luues kausta iga arhiivifaili jaoks. Samuti saab valida mitu faili. Kasutada saab mõlemat lisavalikut.

4.3 Lisavalikud

4.3.1 Lahtipakkimine aega kasutades (*Unpack using dates*)

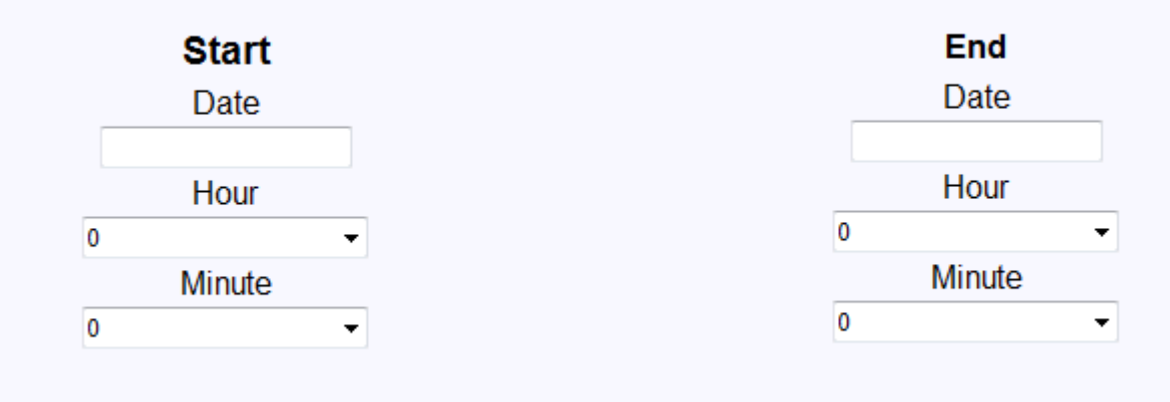
Programm eraldab kõik failid arhiivifailist ja seejärel kustutab failid, mille esimene kirje pole etteantud ajavahemikus. Lahtipakkimine aega kasutades töötab vaid siis, kui kasutatakse ühe või mitme arhiivifaili töörežiimi. Selle kasutamiseks peavad kuupäeva, tunni ja minuti väljad täidetud olema.

4.3.2 Logide eraldamine Pythoni failide järgi (*Separate by .py files*)

Kui see lisavalik on sisse lülitatud, jätab programm meelde kõik logifailid, millega tegeleti. Igast logifailist tehakse mitu logifaili, kus igas uues failis on kaasas informatsioon ühe Pythoni failiga töötamise kohta. Samuti on kaasas kõik käsureal toimuv tegevus. Eraldatud logifailid jäävad samasse kausta kui algne. Kui algse logi nimi on "logi.txt", siis tulemusfailide nimed on formaadis *logi_aaaa.txt*, kus "aaaa" asemel on faili nimi, mida logis muudeti. Kui logis on tegeletud mitme Pythoni failiga, millel on sama nimi, kuid mis asuvad

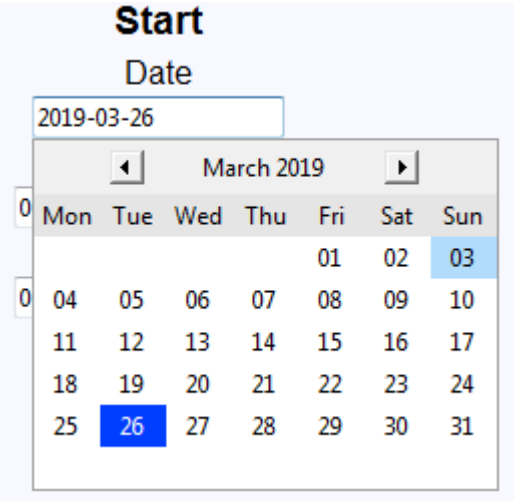
erinevates kaustades, luuakse iga erineva Pythoni faili kohta eraldi logi. Näiteks, kui on avatud kahte *test.py* faili logis *näidis.txt*, tekivad tulemusse *näidis_test.txt* ning *näidis_test_1.py*. Kui on avatud rohkem kui kahte sama nimega faili, on lõppu lisatud suurem arv.

4.4 Väljade kirjeldused



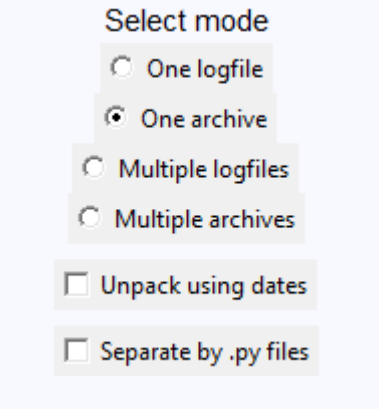
Joonis 4. Aja sisestamise väljad.

Aja järgi logide eraldamiseks tuleb sisestada algus- ja lõppaeg. Ühe aja sisestamiseks tuleb täita kuupäeva, tunni ja minuti väli. Tunni ja minuti sisestamiseks on kaks varianti. Saab vajutada välja peale ja klaviatuuri kasutades sisestada vajaliku arvu. Tunni välja jaoks on sobivad väärtused vahemikus 0-23 ja minuti välja jaoks on sobivad väärtused vahemikus 0-59. Teine valik on välja sees asuva noole peale vajutamine. See avab rippmenüü, kust saab valida sobiva arvu.



Joonis 5. Kalender kuupäeva sisestamiseks.

Kuupäeva sisestamiseks tuleb vajutada kuupäeva välja peale. Seejärel avaneb kalender, kus saab kuupäeva valida. Esimesel avamisel avaneb kalender praeguse kuu peal. Nooltega saab kalendrit ühe kuu kaupa edasi või tagasi liigutada. Hetkel valitud kuupäev on tumesinisel taustal valge tekstiga. Kuupäev, mille kohal on hiir, on helesinise taustaga ja musta tekstiga. Kalendri kasutamise asemel võib kuupäeva ka käsitsi sisestada. Selle jaoks tuleb kirjutada kuupäev formaadis "YYYY-MM-DD", kus "YYYY" on neljakohaline aastaarv, "MM" on kahekohaline kuu arv, "DD" on kahekohaline päeva arv ning nende vahel on sidekriipsud.



Joonis 6. Töörežiimi ja lisavalikute menüü.

Töörežiimi valimiseks on olemas neli nuppu. Korraga saab valida vaid ühe töörežiimi ja alati peab olema mõni töörežiim valitud. Programmi käivitades on valitud ühe arhiivifaili töörežiim. Töörežiimi valimiseks saab vajutada ringi sisse või vastava teksti peale. Kui valida uus töörežiim, tühjendatakse valitud faili väli.

Lisavalikutest saab valida kas mõlemad, ühe või mitte kumbagi. Selleks, et lisavalikut sisse lülitada, tuleb vajutada vastava kasti või teksti peale. Lisavaliku välja lülitamiseks tuleb vajutada samamoodi kas kasti või teksti peale.

Töötlemiseks failide valimiseks tuleb vajutada nuppu "Pick File(s)". Seejärel avaneb faili valimise menüü. Vastavalt töörežiimile saab valida kas ühe või mitu faili. Mitme faili valimisel saab valida mitu erinevat faili, mis asuvad samas kaustas. Selle jaoks, et mitut faili korraga valida, tuleb hoida all "Shift" või "Ctrl" klahvi klaviatuuril. Kui all on "Ctrl" klahv, saab valida ühe faili kaupa neile eraldi vajutades. Kui all on "Shift" klahv, saab valida esimese ja viimase faili ning valituks saavad kõik nende vahele jäävad failid.

4.5 Kasutatud tehnoloogiad

Python on laiaotstarbeline programmeerimiskeel, milles on kirjutatud ka Thonny. Kuna Thonnys kirjutatakse Pythonit, tundus sobiv valida ka selle bakalaureusetöö programmi jaoks Pythoni keel. Samuti on Pythonis lihtne JSON failide töötlemine, mis on suur osa valmivast programmist.

Tkinter on moodul, millega saab Pythoni keeles luua graafilisi liideseid. Kuna Tkinteri kasutamine on üsnagi lihtne ja seda kasutatakse tihti lihtsate rakenduste loomiseks, on Tkinter ka Python 3-ga kaasas. Tkinterit on kasutatud ka Thonny graafilise liidese loomiseks. Tkinter valiti programmi graafilise liidese loomise jaoks, kuna selle kasutamine on lihtne ja autoril oli olemas vähene eelnev kogemus.

7-Zip on vabavaraline avatud lähtekoodiga arhiveerimisprogramm [7]. Sellega saab kokku ja lahti pakkida enamikku populaarseimatest arhiivifailide formaatidest, näiteks zip ja 7z. 7-Zipi kasutatakse valminud programmis arhiivifailis asuvate failide loendi saamiseks ning selle jaoks, et arhiivifaile lahti pakkida. 7-Zipil on käsurea kasutusjuhend, mille abil pandi kokku käsud, mida programmis kasutatakse [9].

Kuna Tkinteris pole sisse ehitatud kuupäeva sisestamise valikut ja kuupäevade sisestamine kirjutades ei pruugi olla kõige mugavam ega kiirem variant, oli vaja mõnda muud viisi kuupäevade lisamiseks. Sobivaks osutus Miguel Martinez Lopezi kirjutatud moodul [8], mis asub failis *datepicker.py*, mida kasutati kuupäevade valimise funktsionaalsuse jaoks.

5 Programmi struktuur

Selles peatükis on kirjeldatud selle bakalaureusetöö raames valminud programmi ja selle tööpõhimõtteid.

5.1 Üldine tööpõhimõte

Käivitatakse *GUI.py*, mis loob kasutajaliidese akna. Kogu graafika ja kõik täidetavad väljad on tehtud Tkinterit kasutades. Valitakse töörežiim ja täidetakse sellele vastavalt väljad. Seejärel vajutatakse “Unpack” nuppu ja programm teeb oma tööd. Funktsioonid failide lahtipakkimiseks ja logifailide töötlemiseks on failides *unpacker.py* ja *splitter.py*. *GUI.py* impordib need funktsioonid ja kasutab neid kasutaja sisestatud andmetega.

5.2 Koodifailide kirjeldused

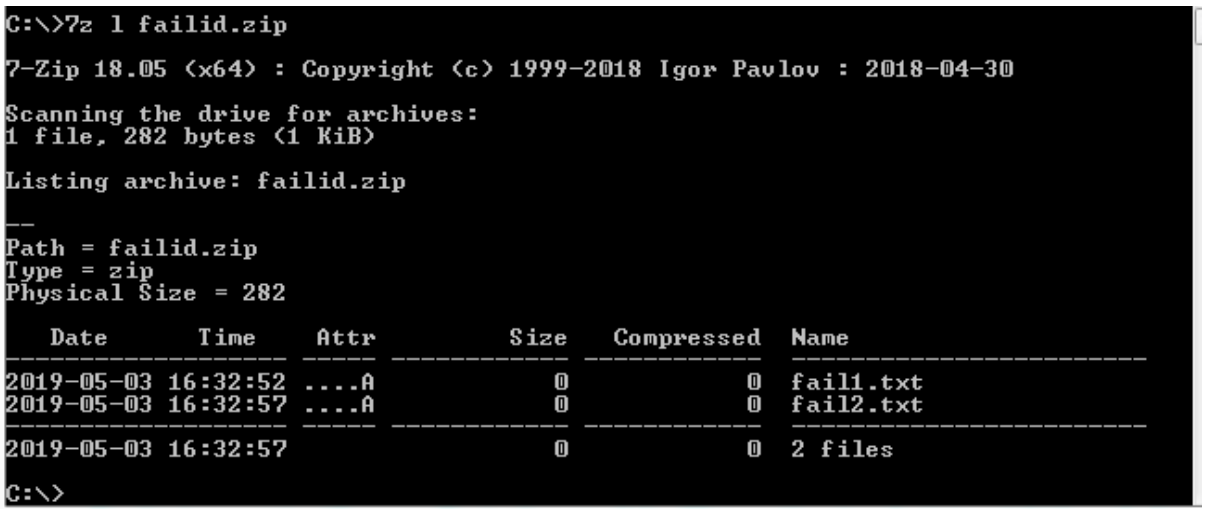
5.2.1 Failide lahtipakkimine (Unpacker.py)

Unpacker.py loodi eesmärgiga hoida meetodeid, mis tegelevad failide lahti pakkimisega. Faili alguses avatakse *settings.json* fail ja sealt loetakse sisse 7-Zipi asukoht ning kasutatav kodeering. Seda osa pole vaja eraldi välja kutsuda, kuna see käivitatakse alati *unpacker.py* importimisel.

Funktsioon `run_process` võtab ühe argumendi, milleks on operatsioonisüsteemi käsurea käsk ning käivitab selle. Funktsioon tagastab jooksvalt rea kaupa kogu teksti, mis muidu ilmuks käsureale peale etteantud käsu täitmist. Seda funktsiooni kasutatakse programmi 7-Zipi käivitamiseks ja tulemuste kätte saamiseks.

Funktsioon `stringify` on selleks, et oleks lihtne viis sõnede jutumärkide vahele panekuks. Argumendiks on üks sõne ning tagastatakse see sama sõne jutumärkide vahel. Seda on vaja, kuna muidu tõlgendaks käsurida käske valesti. Kuna teatud sümbolitel nagu sidekriips on käsureal kindel funktsioon, tuleb panna failiteede ümber jutumärgid, et oleks selge, mis on failitee ning mis on lisavalikud. Näiteks kui on logifail nimega "logi -KT2", tõlgendaks käsurida seda kui "logi" lisavalikuga "KT2".

Arhiivist kõigi seal asuvate failide nimede kättesaamiseks on funktsioon nimega `get_filenames`. Sellel funktsioonil on üks argument, milleks on arhiivifaili failitee. 7-Zipi käsureal käivitades saab arhiivifailis olevaid faile käsuga “7z l arhiiv.zip” [10].



Joonis 7. 7-Zipi failinimede loetelu käsu tulemus.

Kuna väljundis pole kohe kirjas failinimesid, peab leidma üles õige tulba ja reavahemiku, kus on failinimed. Selle jaoks loeb funktsioon ridu seni, kuni leiab rea, mis algab sümbolitega “----” ning tulba “*Name*”. Seni kuni jõutakse uuesti reani, mis algab sümbolitega “----”, salvestatakse kõik tulbas “*Name*” olevad sõned järjendisse, mis tagastatakse funktsiooni lõpus.

Funktsioon `unpack` on arhiivifailide lahtipakkimiseks. Argumentideks on arhiivi failitee, järjend failinimedega, mida tahetakse lahti pakkida, ning valikuline argument väljundkausta määramiseks. Vaikimisi pakitakse failid lahti samasse kausta, kus on arhiivifail. Kui tahta lahti pakkida kõiki faile, tuleb failinimedeks ette anda tühi järjend. Funktsioon koostab käsu, mis käivitatakse `run_process` funktsiooni abil. Vaikimisi kirjutab programm lahti pakkides üle kõik failid, millel on sama nimi, kui mõnel lahtipakitaval failil. Käsk koostati 7-Zipi käsurea kasutusjuhendi abil [11, 12].

Arhiivifailide lahtipakkimiseks on ka funktsioonid `unpack_all` ning `unpack_between_infile`. Funktsiooni `unpack_all` argumentideks on arhiivi failitee ning valikuliselt ka kaust, kuhu failid lahti pakkida. Kui mitte määrata väljundkausta, pakitakse failid lahti kausta, mis on sama nimega kui arhiivifail. Tagastatakse järjend sõnedest, kus on failiteed kõigi lahtipakitud failideni.

Funktsioon `unpack_between_infile` on selleks, et saaks eraldada vaid logifaile, mis jäävad etteantud ajavahemikku. Sarnaselt teistele lahtipakkimise jaoks mõeldud funktsioonidele on argumentideks arhiivi asukoht ja valikuline argument väljundkausta määramiseks. Lisaks sellele on ka alg- ja lõppaeg, mille vahel alustatud logifaile tahetakse. Programm pakib lahti kõik failid etteantud arhiivifailis. Seejärel proovitakse avada kõiki faile ning lugeda nende esimesi kirjeid. Kui kirje lugemine ei ole võimalik, näiteks siis, kui fail on valel kujul, või kui esimese kirje aeg ei kuulu etteantud ajavahemikku, kustutatakse see fail. Seega jäävad alles vaid failid, mille esimene kirje on õiges ajavahemikus. Funktsioon tagastab failiteed kõigile failidele, mis alles jäid.

5.2.2 Logide eraldamine Pythoni failide järgi (Splitter.py)

Selles failis on vaid üks funktsioon, milleks on `separate_by_ids`, mille argumendiks on logifaili failitee. Fail avatakse, sealt loetakse andmed mälusse ja fail suletakse. Seejärel toimuvad järgnevad tegevused:

- Pythoni failidega seotud “text_widget_id”-de leidmine
- Väljundfaili nimede loomine
- Tühjade järjendite loomine, kuhu lähevad kirjed väljundfailide jaoks
- Kõigi kirjete läbi vaatamine ja õigesse järjendisse panemine
- Failidesse kirjutamine

Pythoni failidega seotud “text_widget_id”-de leidmiseks käib funktsioon läbi kõik kirjed, mis logifailis olid. Igast kirjest otsitakse välja nimega “filename” ning kui selline väli on olemas, salvestatakse selle sama kirje “text_widget_id”. “Filename” väli tekib näiteks faili salvestamisel või avamisel. Nii saab kätte kõik “text_widget_id”-d, millega on mõni fail seotud.

Kuna “filename” väljades on terve failitee Pythoni failideni, on hea, kui väljundfaili nimeks tuleb vaid Pythoni faili nimi enne .py-laiendit. Samas võib tekkida olukord, kus kahel Pythoni failil on sama nimi, kuid need asuvad erinevates kaustades. Kui on mitu faili sama nimega, tuleb väljundfaili nimesse numbriga laiend.

Peale tühjade järjendite loomist käib funktsioon taaskord läbi kõik kirjed. Kui kirje “text_widget_class” on “CodeViewText”, pannakse ta vaid kindlasse väljundfaili järjendisse, millega oli selle kirje “text_widget_id”-ga seotud. Kui kirje “text_widget_class” ei ole

“CodeViewText”, pannakse ta kõigisse väljundfailide järjenditesse. Seejärel kirjutatakse kirjed väljundfailide järjenditest failidesse.

5.2.3 Graafiline kasutajaliides (GUI.py)

Selles failis on Tkinteri abil loodud graafiline kasutajaliides ning funktsioonid teistest koodifailidest funktsioonide kasutamiseks. Funktsioonid on `pick_file` ja `unpack`, millel pole argumente, kuna nad saavad kogu info globaalsetelt muutujatelt. `Pick_file` on valitud faili välja täitmiseks ning avab selleks ühe lisaakna vastavalt tööreżiimile, kus saab valida soovitud faile. `Unpack` funktsioon kutsutakse välja, kui vajutatakse “Unpack” nuppu ja see funktsioon teostab vajadusel arhiivifailide lahtipakkimist ning logifailide eraldamist Pythoni failide kaupa. Selle jaoks tegeletakse kõigepealt lahtipakkimisega, mille käigus jäetakse meelde kõik logifailid, mis alles jäid, ning seejärel eraldatakse vajadusel logidest Pythoni faili kaupa uusi logifaile.

5.3 Võimalikud edasiarendused

Selle bakalaureusetöö raames valmis programm, mis töötab vaid teatud kindlatel juhtudel. Praegusel kujul on programm küll kasutatav, kuid mõnede edasiarendustega muutuks programmi kasutamine mugavamaks ja sellest oleks rohkem kasu. Neid arendusi pole veel realiseeritud, kuna nad ei mahu bakalaureusetöö skoopi. Selles peatükis on loetletud võimalikke viise, kuidas programmi edasi arendada.

Nagu mainitud peatükis 6.2.1, on 7-Zipi käsureal võimalus määrata väljundkausta lahtipakkides. Selle jaoks on funktsioonidel ka argumendid olemas, kuid kasutajaliideses neid muuta ei saa. Samuti ei ole piisavalt testitud lahtipakkimiseks loodud funktsioone, kui nende argument “output” on määratud. Väljundkausta määramine lahtipakkides lubaks kõiki logifaile hoida ühes kaustas ning teeks pärast logide töötlemist nende analüüsimise lihtsamaks. Selle implementeerimiseks tuleks teha graafilisse liidesesse üks väli juurde, kus saaks määrata väljundkausta.

Praeguses seisus ei kontrollita mitmes kohas kasutaja sisendit. Selle tõttu võib programmi töö käigus tekkida erindeid, mis peatavad programmi. Selle parandamiseks tuleks kas sisestamisel või “Unpack” nupu vajutamisel kontrollida kõiki sisendeid ja anda kasutajale teada sellest, kui ta on midagi valesti sisestanud.

Samuti peaks programm andma rohkem tagasisidet selle kohta, kas mõni operatsioon õnnestus. Praegusel hetkel paneb kasutaja programmi käima ja tulemusest annab aimu vaid see, kas faile tekkis juurde. Programm peaks näiteks teada andma sellest, kui mõni fail oli vigane või ükski logi arhiivis ei ole etteantud ajavahemikus.

Hetkel on programmi kasutajaliides väga algeline ja ei paista hea välja. Selle parandamiseks tuleks väljade asukohta muuta ning teha nende täitmine lihtsamaks. Probleeme võib tekkida ka arvutite peal, mille monitori resolutsioon on väike, kuna akna suurus on fikseeritud 800x600 pikslit. Samuti võiks programmi sees olla kasutusjuhend, nii et ei peaks seda iga kord GitHubist otsima.

Ühe programmeerimisülesande lahendamise ajal võidakse Thonny sulgeda ning uuesti käivitada. Seega, võib tekkida olukord, kus ühe ülesande lahenduse kohta on infot mitmes logifailis. Hetkel suudab programm küll ühest logifailist mitme Pythoni faili kaupa eraldada logisid, kuid peaks olema ka võimalik mitmest logist ühte kokku panna, kui mitmes logis kirjutatakse sama faili. Lisaks sellele võiks programm anonümiseerida logisid. Kuna logides on tihti sees failitee, kus on kirjas autori nimi, oleks nõusolekuta nende logide kasutamine teadustöös ebaeetiline.

On võimalus, et inimesed ei esita oma logifaile üheski formaadis, millega selle bakalaureusetöö raames valminud programm hakkama saaks. Näiteks võidakse esitada arhiivifail, mille sees on arhiivifail, mille sees on logid. Seetõttu oleks hea, kui programm teostaks lahtipakkimist ja logide kontrolli rekursiivselt. See tähendab seda, et kui arhiivifailist leitakse teine arhiivifail, pakitakse ka sisemine lahti, et sealt logisid otsida.

6 Tagasiside ja testimine

Selles peatükis on kirjeldatud tagasisidet, mida saadi programmi erinevatele versioonidele ning testimismeetodeid, millega veenduti programmi korrektses töös.

6.1 Juhendajate tagasiside

Pärast programmi esimese kasutatava versiooni valmimist saadeti programm juhendajatele, et nad seda testida saaksid. Kuna esimest versiooni ei olnud enne seda põhjalikult testitud, leiti mitmeid vigu, mida tuli parandada.

Esimene leitud viga oli seotud käsuga, mis koostati arhiivifailide lahtipakkimiseks. Kuna mõnedel sümbolitel, näiteks sidekriips, on eriline tähendus operatsioonisüsteemi käsureal, tekkis käsureal probleeme mõistmisega, mis osa käsust on failitee ning mis osa on lisavalikuna mõeldud. Nagu näha joonisel 8, tekkis probleem, kui failiteel oli sidekriips. Selle parandamiseks saab panna jutumärgid ümber iga failitee. Probleemi lahendamiseks sai loodud funktsioon `stringify`.

```
>>> %Run GUI.py
  C:\Users\X5\Documents\aa ATI\Logid - MOOC 10 esimest tähestikus\15122018.zip


  Command Line Error:
  Unknown switch:
  -
  []
```

Joonis 8. Sidekriipsuga tekkiv veateade.

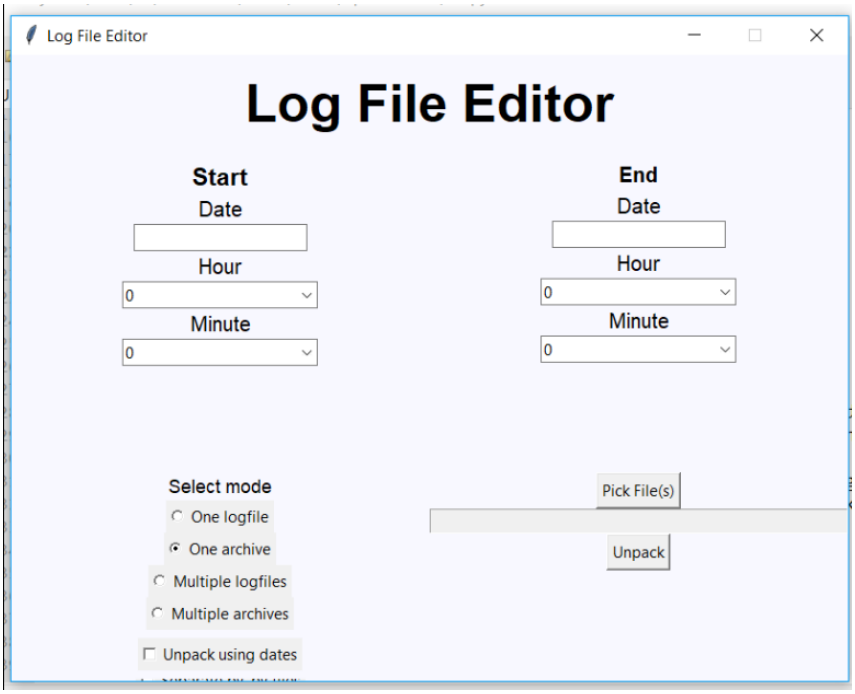
Ka teine leitud viga oli seotud käsuga, mis koostati arhiivifailide lahtipakkimiseks. Seekord oli programmil probleeme täpitähtedega. Algselt kasutati funktsioonis run_process käsurealt lugemiseks kodeeringut UTF-8. Kuna operatsioonisüsteemide käsuridadel ei pruugi vaikimisi kodeeringuks olla UTF-8, tekkis probleem, kui üritati mõnes teises kodeeringus teksti lugeda UTF-8 tekstina. Nagu näha joonisel 9, tuli veateade, kui anda käsureale failitee, kus sisaldus ä-täht. Selle probleemi parandamiseks, tuli leida, mis kodeeringut kasutab operatsioonisüsteem. Kuna arendusel kasutati Windowsi, tuli leida Windowsi käsurea kodeering. Selle jaoks on käsk “chcp” (*change code page*) [13], mis kuvab, millist kooditabelit hetkel kasutatakse. Kooditabelid on tabelid, kus kindlatele arvudele vastavad kindlad sümbolid.

Selle käsuga leiti, et eesti arvutites kasutatakse koodilehte 775, mis vastab Pythonis kodeeringule cp775. Kui muuta dekodeerimisel kodeering UTF-8 pealt cp775 peale, sai programm ka hakkama täpitähtedega. See lahendus ei ole universaalne ning ei pruugi töötada Unixi operatsioonisüsteemidel. Selle jaoks lisati faili *settings.json* kodeeringu muutmise jaoks väli. Kui kodeeringuga cp775 tekib probleeme, tuleks proovida kodeeringut UTF-8. Kui failiteedes pole regionaalseid sümboleid, saab kasutada ka UTF-8 kodeeringut.

```
C:\Users\X5\Documents\aa ATI\Lägid\16-AS-logid-20190120T144748Z-001.zip
"C:\Program Files\7-Zip\7z.exe" l "C:\Users\X5\Documents\aa ATI\Lägid\16-AS-logid-20190120T144748Z-001.zip"
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\X5\AppData\Local\Programs\Thonny\lib\tkinter\__init__.py", line 1705, in __call__
    return self.func(*args)
  File "C:\Users\X5\Documents\aa ATI\Villem\L-put--master\GUI.py", line 96, in unpack
    filenames=unpack_all(loc_7z, chosen_file_field.get())
  File "C:\Users\X5\Documents\aa ATI\Villem\L-put--master\unpacker.py", line 69, in unpack_all
    all_filenames=get_filenames(loc_7z, arch_loc)
  File "C:\Users\X5\Documents\aa ATI\Villem\L-put--master\unpacker.py", line 39, in get_filenames
    for line in run_process(command):
  File "C:\Users\X5\Documents\aa ATI\Villem\L-put--master\unpacker.py", line 25, in run_process
    line = p.stdout.readline().decode("utf8")
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x84 in position 47: invalid start byte
```

Joonis 9. Vales kodeeringus teksti lugemisel UTF-8 kodeeringuga lugemisel tekkiv veateade.

Probleeme oli ka programmi graafilise kasutajaliidesega. Graafiline kasutajaliides on fikseeritud suurusega ning osadel masinatel ei mahtunud kõik täidetavad väljad akna sisse. Selle parandamiseks liigutati kõiki väljasid akna keskmele lähemale.



Joonis 10. Programmi graafiline kasutajaliides, kus üks lisavaliku väli on aknast väljas.

6.2 Logide eraldamise testimine

Kuna logifailid võivad olla väga erinevad, tuli eraldi testida logide eraldamise funktsionaalsust. Selle jaoks, et kontrollida, kuidas programm saab hakkama erinevate juhtudega, tuli luua logifailid, milles käidi läbi erijuhud. Selle bakalaureusetöö raames loodi seitse logifaili, millega sai testida logide eraldamist (Lisa 1). Samuti on lisatud tulemusfailid, mis tekivad, kui need logid programmile ette anda.

Programmi esialgses versioonis ei suutnud programm logidest üles leida failinimesid ja seega lõi programm ühe uue logifaili iga "text_widget_id" kohta algses logis. Rekursiivse funktsiooni silumisel tekitas Thonny iga rekursiooni sammu kohta uue akna, millel oli ka oma "text_widget_id". Selle tõttu võis ühe logifaili eraldamisel tekkida sadu faile. Selle testimiseks loodi logifailid *for.txt* ja *debug.txt*, kus mängiti läbi rekursiivse funktsiooni silumine.

Nagu mainitud peatükis 3.2.2, siis iga kord, kui avatakse uus alamaken, antakse talle uus "text_widget_id". See tähendab, et võib tekkida olukord, kus ühe Pythoni failiga tegeletakse mitmes erinevas alamaknas. See võib juhtuda, kui ühe Thonny käivituse ajal avatakse, suletakse ja avatakse uuesti sama faili. Selle jaoks, et testida, kuidas programm hakkama saab, kui faile avatakse, suletakse ja salvestatakse, loodi logifailid *save.txt*, *open.txt* ja *closeopen.txt*. Failis *save.txt* loodi uus fail ja salvestati see. Failis *open.txt* avati juba olemasolev fail. Failis *closeopen.txt* avati, sulgeti ja avati sama faili.

Kuna ühes süsteemis võib korraga olla mitu sama nimega Pythoni faili, on ka võimalik, et ühe Thonny käivituse ajal avatakse mitu faili, millel on sama nimi, kuid mis ei ole samad failid. Selle pärast pidi ka testima, kuidas programm saab hakkama siis, kui ühes logis on infot mitme sama nimega failiga tegelemise kohta. Selle jaoks loodi failid *samanimi.txt* ja *samanimi2.txt*, kus avatakse mitut erinevat Pythoni faili, mille nimi on *test.py*.

Eelnimetatud failidega testimine osutus edukaks, kuna programm sai peale mõningasi täiustusi kõigi eelnimetatud failide eraldamisega hakkama. Tulemusfailid olid oodatud kujul ja viimase versiooniga probleeme ei tekkinud.

6.3 Küsimustik

Lisaks juhendajatele testisid programmi ka mõned aine “Programmeerimise alused” praktikumijuhendajad. Selles aines oli äsja olnud kontrolltöö, mille raames pidid juhendajad logisid kontrollima. Juhendajatele anti programmi GitHubi link (Lisa II) ja küsimustiku link.

Loodud küsimustikus oli 13 küsimust, kasutati nii valikvastusega küsimusi kui ka selliseid, kus kasutaja pidi ise teksti kirjutama. Loodud küsimused ja võimalikud vastused on järgnevas loetlelus. Kui vastusevariante pole mainitud, oli oodatud tekstilist vastust.

1) Kuidas hindaksite programmi kasulikkust? 1-5
2) Kas kasutaksite programmi sellisel kujul ka tulevikus? Jah/Ei
3) Kui eelmise küsimuse vastus oli "Ei", siis kirjutage, miks?
4) Kas juhend oli piisavalt põhjalik? Jah/Ei
5) Kui eelmise küsimuse vastus oli "Ei", siis kirjutage, mis puudu oli.
6) Kas programmi käivitamisel ilmnes probleeme? Jah/Ei
7) Kui eelmise küsimuse vastus oli "Jah", siis palun kirjeldage neid probleeme.
8) Kas enne selle programmi kasutamist oli teie arvutis juba 7-Zip olemas? Jah/Ei
9) Kuidas hindaksite programmi kiirust? 1-5
10) Kas programmi kasutamisel ilmnes probleeme? Jah/Ei
11) Kui eelmise küsimuse vastus oli "Jah", siis palun kirjeldage neid probleeme.
12) Kui on mõni funktsionaalsus programmist puudu, siis palun kirjutage sellest.
13) Kui sooviksite programmi juures midagi muuta, siis palun kirjutage sellest.

Küsimustiku täitis kaks juhendajat ainest “Programmeerimise alused”. Programmi kasulikkuse ja edasise kasutamise kohta olid küsimustiku täitjad erinevatel arvamustel. Üks hindas programmi kasulikkust 5-ga kui teine vaid 2-ga. See, kes hindas 5-ga, vastas ka seda, et ta kasutaks programmi sellisel kujul edasi. Juhendiga ei olnud kummalgi probleeme, küll aga oli probleeme ühel juhendajal Linuxi peal programmi häälestamisega. Mõlemal vastajal oli arvutis juba olemas 7-Zip, mis tegi programmi paigaldamise lihtsamaks. Kuna testimiseks anti üsnagi algne versioon, oli peamine probleem see, et Pythoni failide järgi logidest eraldamine ei pannud tekkivatele logifailidele korrektseid nimesid. Nüüd on see viga parandatud ja eraldatud logide nimedes sisaldub Pythoni faili nimi, millega tegeleti.

7 Kokkuvõte

Selle bakalaureusetöö eesmärk oli luua programm, mis aitaks Thonny loodud logifailide korrastamisega. Töös kirjeldati Thonnyt ja logifaile, mida Thonny loob. Täpsemalt oli kirjeldatud logifailide osasid, mida valminud programm kasutas.

Valmis programm, mis suudab lahti pakkida populaarsemaid arhiivifaili formaate. Lisaks tavalisele lahtipakkimisele on võimalik lahti pakkida vaid logifaile, mille esimene kirje on etteantud ajavahemikus. Samuti saab programm hakkama ühest logist mitme logi eraldamisega, nii et igas uues logis on vaid info ühe Pythoni failiga tegelemise kohta.

Valminud programmi saab kasutada ainetes, kus on kasutusel Thonny, või uurimistöödes. Programmi on võimalik edasi arendada, et selle kasutamine oleks mugavam ja et funktsionaalsust lisada. Mõned võimalikud edasiarendused on välja toodud peatükis 5.3.

8 Viidatud kirjandus

[1] H. Meier, “Õppijate käitumismustrid programmeerimisülesande lahendamisel: logifailide analüüs”. TÜ arvutiteaduse instituudi magistritöö. 2015.

http://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=62175&year=2018 (Vaadatud 14.01.2019)

[2] R. Kodasmaa, “Programmeerimiskeele Python veateated programmeerimise algõppes”. TÜ arvutiteaduse instituudi magistritöö. 2017.

http://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=56952&year=2017 (Vaadatud 14.01.2019)

[3] K. Pedel, “E-kursuse "Programmeerimise alused" logifailide analüüs”. TÜ arvutiteaduse instituudi bakalaureusetöö. 2016.

http://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=53665&year=2016 (Vaadatud 14.01.2019)

[4] A. Aramaa, “Pythoni koodi muudatuste analüsaator”. TÜ arvutiteaduse instituudi bakalaureusetöö. 2014.

http://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=40741&year=2014 (Vaadatud 14.01.2019)

[5] Thonny kodulehekülg

https://thonny.org/ (Vaadatud 08.03.2019)

[6] JSON’i kodulehekülg

https://www.json.org/ (Vaadatud 08.03.2019)

[7] 7-Zipi kodulehekülg

https://www.7-zip.org/ (Vaadatud 18.04.2019)

[8] López M.M. TKINTER DATEPICKER (LIKE THE JQUERY UI DATEPICKER) (PYTHON RECIPE). 2016.

http://code.activestate.com/recipes/580725-tkinter-datepicker-like-the-jquery-ui-datepicker/ (Vaadatud 26.04)

[9] 7-Zipi käsurea kasutusjuhend

https://sevenzip.osdn.jp/chm/cmdline/index.htm (Vaadatud 03.05)

[10] 7-Zipi käsurea kasutusjuhend (arhiivi sisu loetlemise alamleht)
https://sevenzip.osdn.jp/chm/cmdline/commands/list.htm (Vaadatud 03.05)

[11] 7-Zipi käsurea kasutusjuhend (arhiivist eraldamise alamleht)
https://sevenzip.osdn.jp/chm/cmdline/commands/extract.htm (Vaadatud 03.05)

[12] 7-Zipi käsurea kasutusjuhend (failide ülekirjutamise alamleht)
https://sevenzip.osdn.jp/chm/cmdline/switches/overwrite.htm (Vaadatud 03.05)

[13] Stackoverflow küsimus Windowsi käsurea kodeeringu kohta (Vaadatud 08.05)
https://stackoverflow.com/questions/1259084/what-encoding-code-page-is-cmd-exe-using

Lisad

I. Testimiseks loodud logifailid

Pythoni failide järgi logifailide eraldamise testimiseks mõeldud logid (asuvad Google Drive'is)

II. Thonny Log File Editor

Selle bakalaureusetöö raames valminud programmi lähtekood on leitav aadressil https://github.com/villemtonisson/Thonny-Log-File-Editor.

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Villem Tõnisson,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose, **Programmeerimisülesannete lahendamisel tekkivate Thonny logifailide korrastamine**, mille juhendajad on Eno Tõnisson ja Heidi Meier, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

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

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

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

Villem Tõnisson

10.05.2019