TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Kelian Kaio

Thonny arenduskeskkonna pistikprogramm automaatse tagasisidega ülesannete lahendamiseks

Bakalaureusetöö (9 EAP)

Juhendaja: Aivar Annamaa

Tartu 2016

Thonny arenduskeskkonna pistikprogramm automaatse tagasisidega ülesannete lahendamiseks

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärgiks on luua pistikprogramm automaatse tagasisidega ülesannete lahendamiseks Pythoni programmeerimiskeele algõppeks loodud arenduskeskkonnale Thonny'le. JetBrainsi poolt loodud programmis PyCharm Edu saab koostada ja lahendada automaatselt testitavaid ülesannete komplekte ning antud pistikprogramm võimaldab neid Thonny's käivitada ja lahendada.

Võtmesõnad:

Python, programmeerimine, PyCharm Edu, Thonny, arenduskeskkond, automaatne tagasiside

CERCS: S281 - Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

Thonny Integrated Development Environment plugin for solving tasks with automatic feedback

Abstract:

The goal of this thesis is to create a plugin which allows for individuals to solve tasks with automatic feedback by using Thonny, an integrated Python development environment which was designed for novice programmers. PyCharm Edu was developed by JetBrains which allows to create courses with automatic feedback. The plugin crafted for this thesis allows for these PyCharm Edu courses to be opened and solved in an easy and efficient manner.

Keywords:

Python, programming, PyCharm Edu, Thonny, IDE, automatic feedback

CERCS: S281 - Computer-assisted education

Sisukord

Sissejuhatus

Programmeerimise õppimist alustades, olgu see iseseisvalt või ülikooli kursusel, on oluline, et ühe teema juurest teise liikudes oleks eelnev täielikult selgeks saadud. Juhul kui õpilane ei saa teemast või sellest, miks kood ei tööta, täielikult aru, vajab ta abi. Ülikooli kursustel on üldjuhul osalejaid nii palju, et õppejõud ei jõua kõikidele tudengitele piisavalt kiiresti tagasisidet anda ning iseseisvalt õppides puudub tagasiside täielikult. Tihtipeale tuleb ka ette, et õpilane peab tegelema järgmise kodutöö teemaga enne, kui ta on saanud tagasisidet eelmise teema kohta.

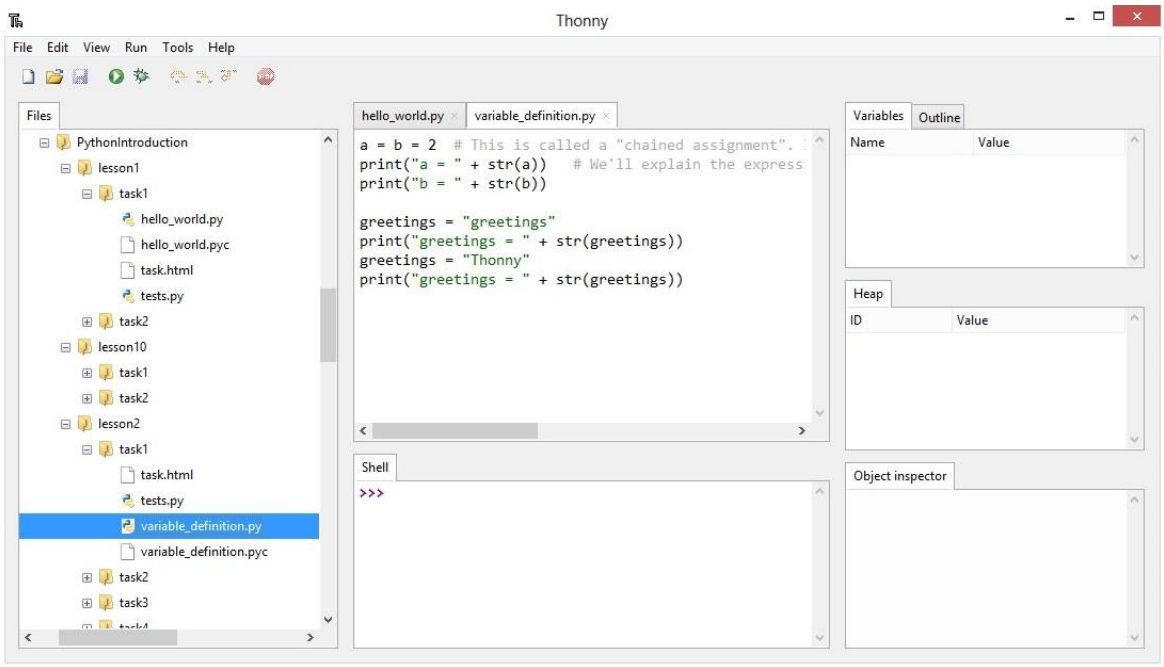
Selle jaoks on JetBrains loonud eraldi programmi nimega PyCharm Edu, kus saab lahendada ülesannete komplekte ja saada kohest tagasisidet [1]. Tartu Ülikoolis aga kasutatakse Pythoni programmeerimise algkursustel Thonny arenduskeskkonda. Õpilasele on oluline, et kõik vajalikud töövahendid oleksid kättesaadavad talle tuntud keskkonnas.

Käesoleva bakalaureusetöö eesmärk on luua arenduskeskkonnale Thonny pistikprogramm automaatse tagasisidega ülesannete lahendamiseks, millega on võimalik avada PyCharm Edu’s loodud ülesannete komplekte.

Töö esimeses peatükis annab autor ülevaate Thonny arenduskeskkonnast. Teises peatükis tutvustatakse PyCharm Edu ning nende loodud pistikprogrammi arenduskeskkonnale PyCharm 5.0. Lisaks arutletakse pistikprogrammi kasutajaliidese üle ning tuuakse välja, millistest failidest ülesannete komplektid ehk kursused koosnevad. Kolmandas peatükis antakse põhjalik ülevaade pistikprogrammi kasutajaliidesest, tööst ja puudustest. Neljandas peatükis seletatakse bakalaureusetöö protsessi ja mis vigu tehti. Viiendas peatükis antakse soovitusi edasisteks arendamisvõimalusteks. Töö lisades on olemas viide pistikprogrammi seadistamise juhendile ja kasutusjuhendile.

1. Thonny ülevaade

Thonny (vt joonist 1) on Aivar Annamaa poolt loodud vabavaraline arenduskeskkond, mis on mõeldud Pythoni programmeerimiskeele algõppeks. Thonny on algajasõbralik keskkond, kus on võimalik visualiseerida programmi töökäiku (nt samm-sammult koodi läbivaatamine) ja see aitab õppijal oma kirjutatud koodist paremini aru saada ning vigu leida [2].



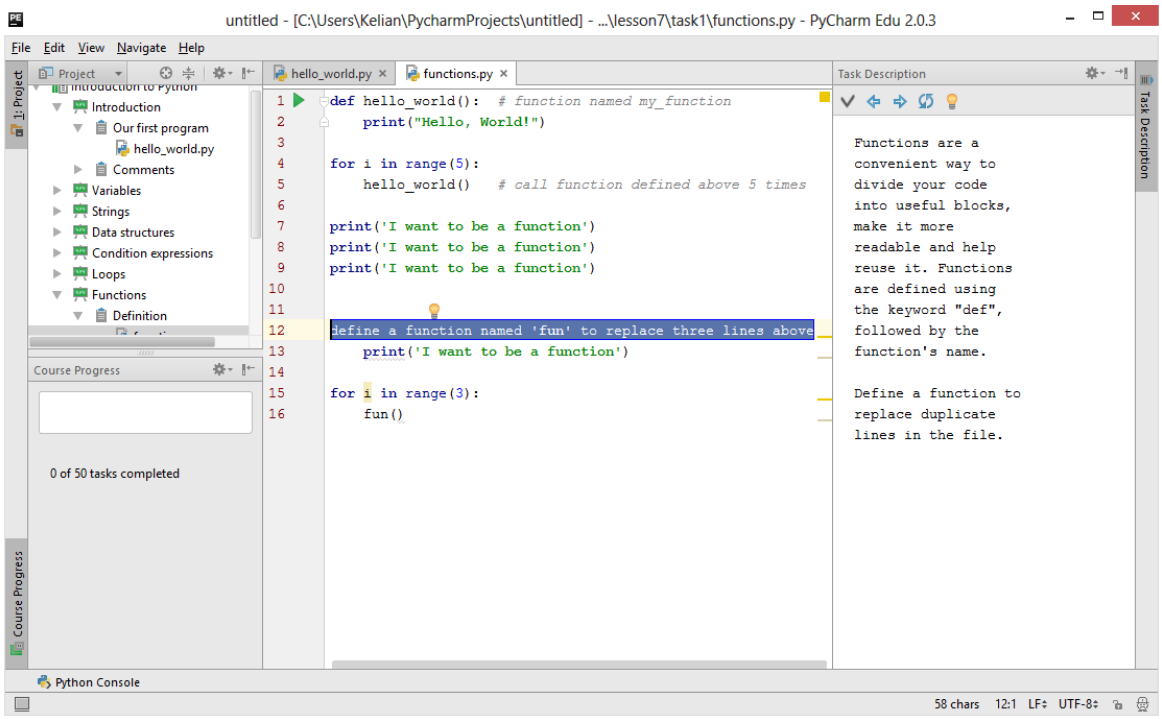
Joonis 1. Thonny kasutajaliides.

Seda arenduskeskkonda kasutatakse Tartu Ülikoolis esimesel programmeerimiskursusel õpetamiseks ja õppimiseks. Lisaks kasutati seda esimest korda 2015/2016 sügissemestril Tartu Ülikooli vaba juurdepääsuga e-kursusel ehk MOOCis (ingl k Massive Open Online Course) "Programmeerimisest maalähedaselt" Pythoni õpetamiseks [3].

Thonny on avatud lähtekoodiga ning toetab kolmandate osapoolte poolt loodud pistikprogramme [2]. Arenduskeskkond on kirjutatud Pythonis ning kasutab standardset Pythoni moodulit Tkinter [4], et luua graafiline kasutajaliides. Thonny't on võimalik alla laadida avalikust BitBucketi repositooriumist [5].

2. PyCharm Educational Edition

PyCharm Edu (vt joonist 2) on vabavaraline ja avatud lähtekoodiga arenduskeskkond õpilastele ja õpetajatele, mille on loonud JetBrains. Esimene versioon tuli välja 30. oktoobril 2014. Õppimiskeskkond on loodud põhimõttel, et algaja programmeerija saab õppida sarnases keskkonnas, mida kasutavad professionaalid [1]. Arenduskeskkonnas lahendatakse ülesandeid interaktiivselt ehk kasutajal on ees ülesande kirjeldus ja koodis kohatäited, kuhu tuleb sisestada õige kood, ning võimalus küsida vihjeid. Ülesande kontrollimisel antakse kasutajale automaatset tagasisidet.



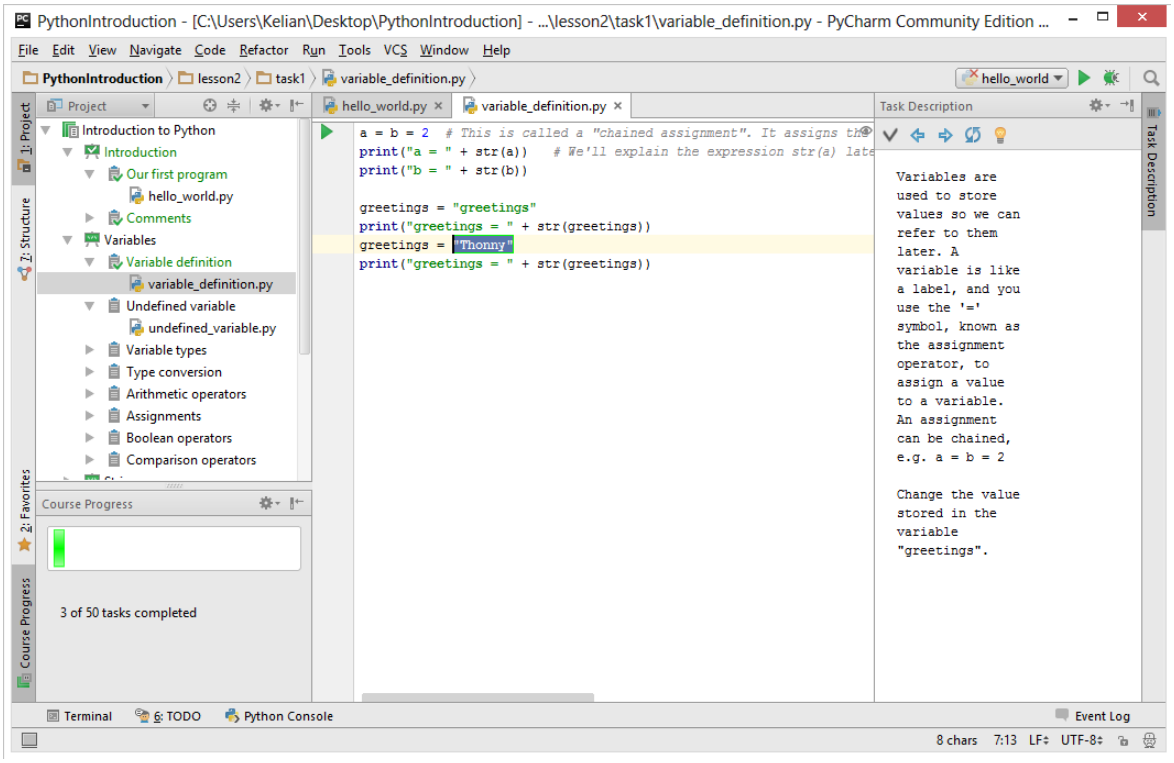
Joonis 2. PyCharmi kasutajaliides.

Lisaks on võimalik arenduskeskkonnas luua ja muuta enda tehtud ülesannete komplekte ehk kursusi. Neid on võimalik teiste inimestega lihtsasti jagada [6].

2.1 PyCharm Educational Edition pistikprogramm

JetBrains tuli 3. novembril 2015 välja oma pistikprogrammiga, mida saab kasutada alates PyCharm 5.0 Professional ja Community versioonidega. Pistikprogrammi hariduslik funktsionaalsus on sama, mis eraldiseisval PyCharm Edu keskkonnal. Lisaks on võimalik pistikprogrammiga luua ülesandeid, mis toetavad PyCharm Professional Editioni funktsionaalsusi (nt interaktiivne Django kursus) [7]. Nagu on näha joonistel 2 ja 3, on nii PyCharm

Edu pistikprogramm kui ka eraldiseisev PyCharm Edu arenduskeskkond samasuguse kasutajaliidesega.



Joonis 3. PyCharm Edu pistikprogrammi kasutajaliides.

PyCharm Edu eraldiseisvat arenduskeskkonda soovitatakse kasutada, kui alustatakse programmeerimise õppimisega. PyCharm Edu pistikprogramm on mõeldud edasijõudnud programmeerijatele, kes soovivad õppida keerulisemaid asju (nt Django) [7].

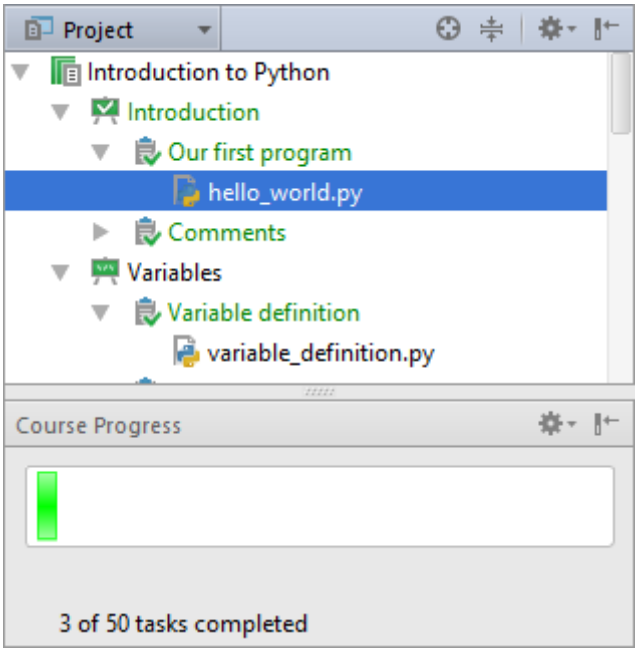
Thonny arenduskeskkond on loodud algajatele ning seepärast on interaktiivse lahendamise pistikprogramm väga hea lahendus. Õpilased saavad lahendada ülesandeid, kasutades Thonny funktsionaalsusi (näiteks samm-sammult koodi läbivaatamine), mis aitavad neid programmi töökäigu arusaamisel. Seepärast on Thonny interaktiivse ülesannete lahendamise pistikprogrammi eeskujuks just PyCharm Edu pistikprogramm.

2.2 PyCharm Educational Edition pistikprogrammi kasutajaliides

Järgmistes alampeatükkides antakse põhjalikum ülevaade PyCharm Edu pistikprogrammi põhifunktsionaalsustest.

Projekti ja progressi vaade

Ülesannete komplekti ehk kursuse avamisel PyCharm Edu pistikprogrammis kuvatakse projekti vaade. Kursust näeb projekti vaates teemade ja ülesannete kaupa (vt joonist 4).

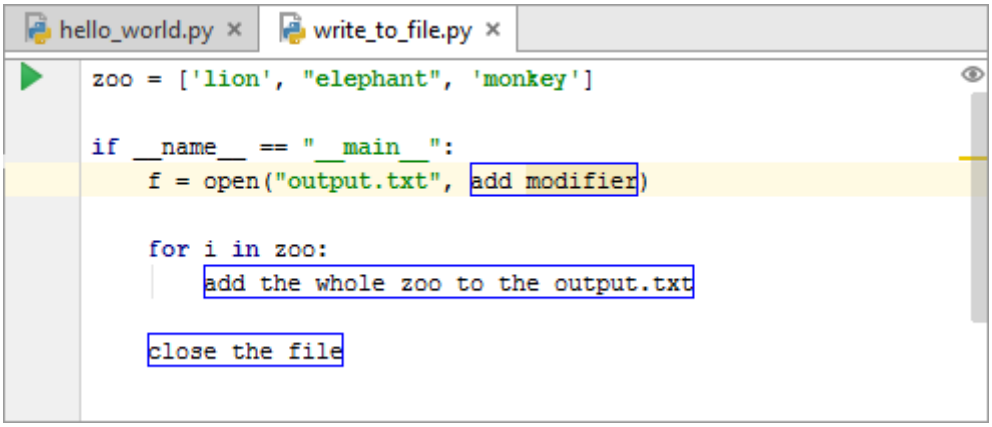


Joonis 4. Projekti ja kursuse progressi vaade.

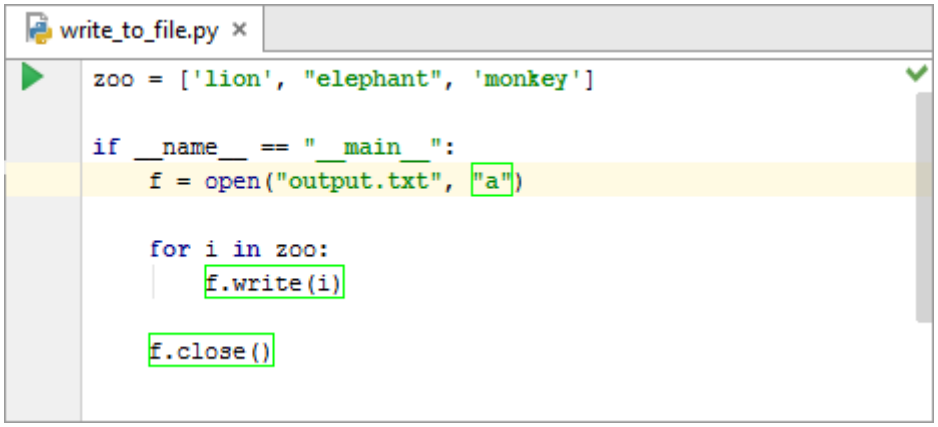
Rohelise teksti ja linnukesega näidatakse, mis ülesanded on kasutaja juba lahendanud. Kasutaja näeb progressi vaates, mitu ülesannet on kogu kursusest tal tehtud ja kui suure protsendi see moodustab kursusest.

Koodi vaade

Kasutaja näeb avatud ülesandeid koodi vaates. Pistikprogramm lisab koodi värviliste kastide kujul kohatäited, mida peab kasutaja muutma või täitma, et ülesannet sooritada.



Joonis 5. Koodi vaade tegemata ülesande puhul.

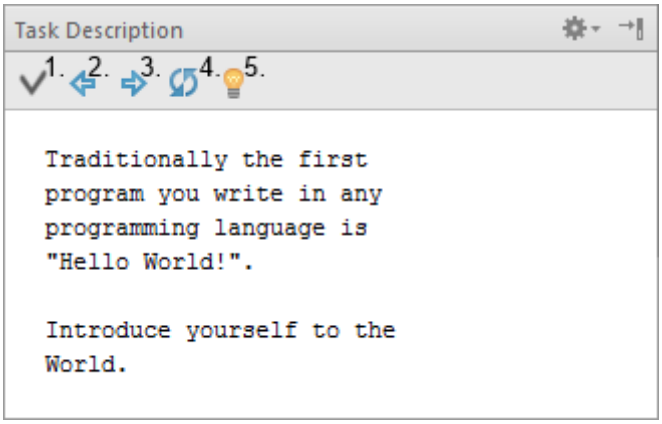


Joonis 6. Koodi vaade tehtud ülesande puhul.

Kohatäited on piiratud tumesinise kastiga (vt joonist 5), kui kasutaja ei ole sinna midagi asemele kirjutanud. Täidetud kohatäidete puhul on nad aga piiratud neoonrohelise kastiga, (vt joonist 6). Üleval nurgas on väike roheline linnuke, kui ülesanne on õigesti lahendatud.

Ülesande kirjelduse vaade

Ülesannete kirjelduste jaoks on pistikprogrammil eraldi vaade (vt joonist 7). Seda kuvatakse siis, kui ülesanne on avatud koodi vaates.

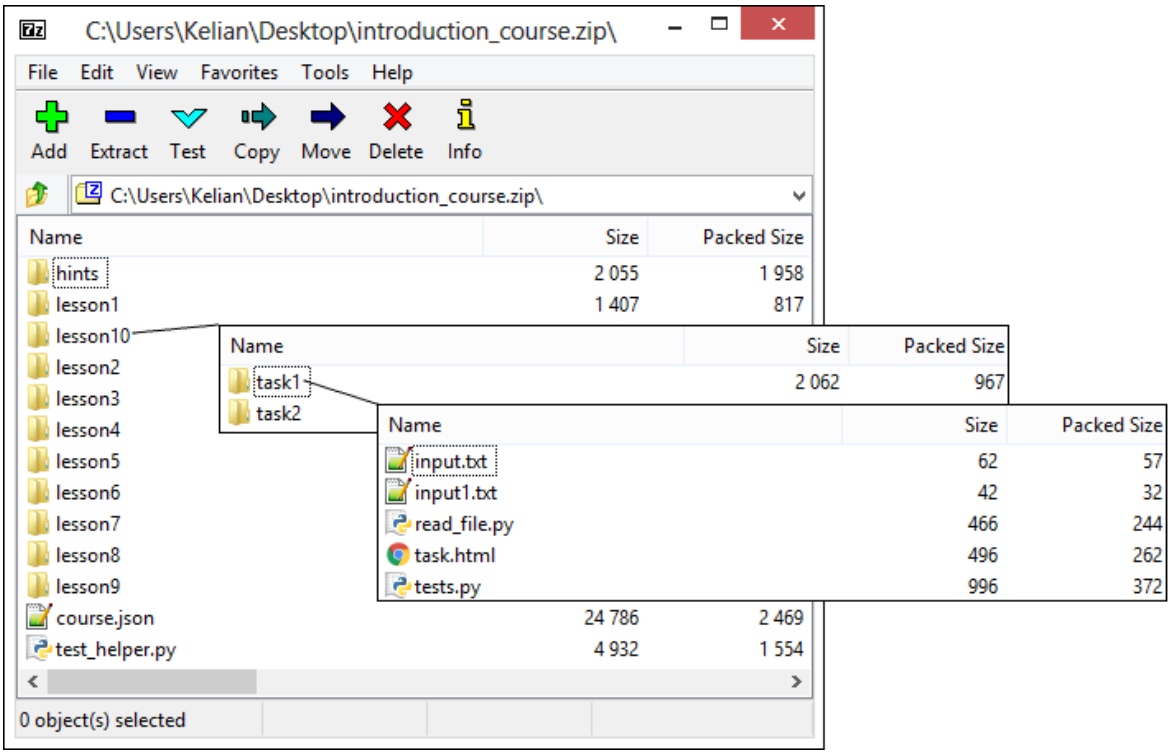


Joonis 7. Ülesande kirjelduse vaade.

Vaatel on viis erinevat nuppu, mis on vastavalt nummerdatud joonisel 7. Esimene nupp on hetkel avatud ülesande kontrollimiseks. Kui ülesanne on õigesti lahendatud, ilmub nupu kohale väike mullike, mis teavitab õnnestumisest. Vastasel juhul teavitab mullike, mis on koodis valesti. Teine ja kolmas nupp on ülesannete vahel navigeerimiseks. Neljas nupp on hetkel avatud ülesande algse seisundi taastamiseks, et kasutaja saaks soovi korral alustada ülesannet uuesti. Viies nupp on vihje näitamiseks. Kui kasutaja ei suuda ülesannet ära lahendada, saab ta sellele nupule vajutades hüpikaknas vihjeid.

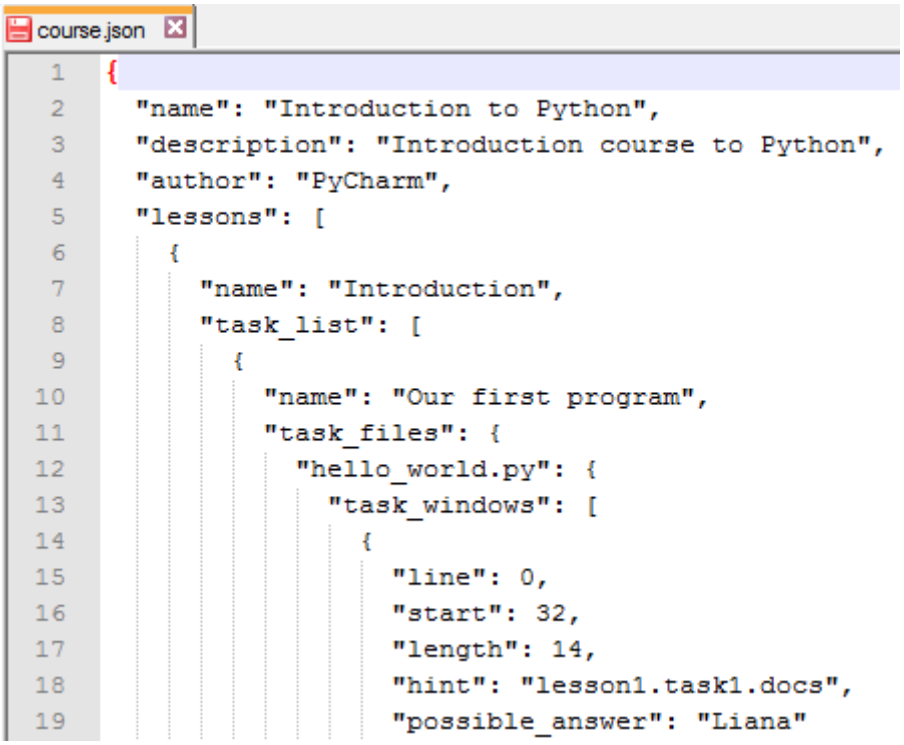
2.3 Kursuse kataloog

Selle bakalaureusetöö raames loodud Thonny pistikprogramm avab PyCharm Edu'ga loodud kursusi ja seega antud peatükis antakse ülevaade, mis failidest kursus koosneb.



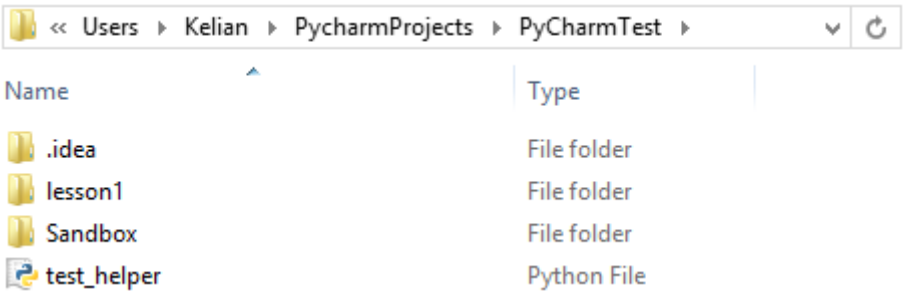
Joonis 8. Kursuse kokkupakitud failide kataloog ehk zip-fail.

Kursus laetakse alla zip-failina, mida on võimalik avada PyCharm Edu's. Kursuse zip-fail (vt joonist 8) koosneb *hints* kaustast (mis võib ka puududa), teemade (ingl k *lesson*) kaustadest, *course.json* ja *test_helper.py* failist. Teemade kaust koosneb ülesannete (ingl k *task*) kaustadest, mis omakorda sisaldavad ülesande jaoks vajaminevaid faile ning *task.html* ja *tests.py* faili. Failis *task.html* on ülesande kirjeldus HTML formaadis ning *tests.py* failis on antud ülesande jaoks vajaminevad testid, et kontrollida, kas kasutaja on ülesande õigesti lahendanud. Failis *course.json* on kogu informatsioon kursuse kohta, millest osa on näidatud joonisel 9. Thonny pistikprogrammi jaoks on kõige tähtsam informatsioon seal kursuse, teemade ja ülesannete nimed ning kohatäidete positsioonid. Vihjed võivad asuda oma *hints* kausta docs-failides või *course.json* failis. Failis *test_helper.py* on kõik põhilised testid, mida iga ülesande faili puhul jooksutatakse, et kontrollida, kas õpilane on ülesande õigesti teinud.



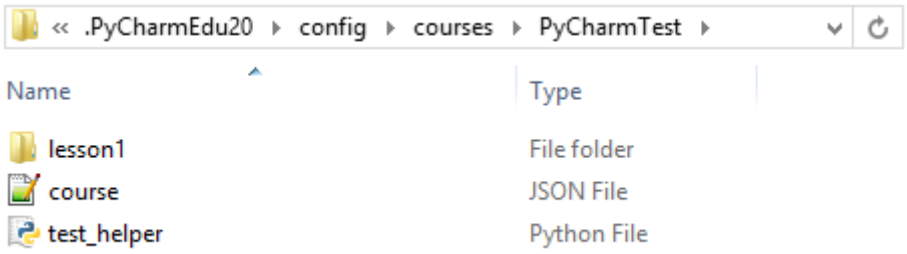
Joonis 9. Course.json fail.

Pärast kursuse kokkupakitud failide kataloogi avamist PyCharm Edu's paigutatakse kursuste failid kahte asukohta: ~*PycharmProjects* ja ~\.*PyCharmEdu20\config\courses*.



Joonis 10. Kursuse kaust PyCharmProjects kaustas.

Esimeses asukohas (vt joonist 10) hoitakse teemade kaustu, *Sandbox* ja *.idea* kausta ning *test_helper.py* faili. Teemade kaustades hoitakse ülesannete kaustu ning nendes ülesannete faile. Kui kasutaja lahendab programmis ülesandeid ehk muudab ülesande faile, salvestatakse uus muudetud fail ülesande kaustas vana asemele. *Sandbox* kaustas hoitakse faile, mida õpilane ise on loonud, ning *.idea* kaustas on kõige tähtsam Thonny pistik-programmi loomisel *study_project.xml* fail. Sinna salvestatakse kohatäidete uued positsioonid, kui kasutaja on neid ülesande failis muutnud, ja kas ülesanne on tehtud või mitte.



Joonis 11. Kursuse kaust PyCharmEdu kaustas.

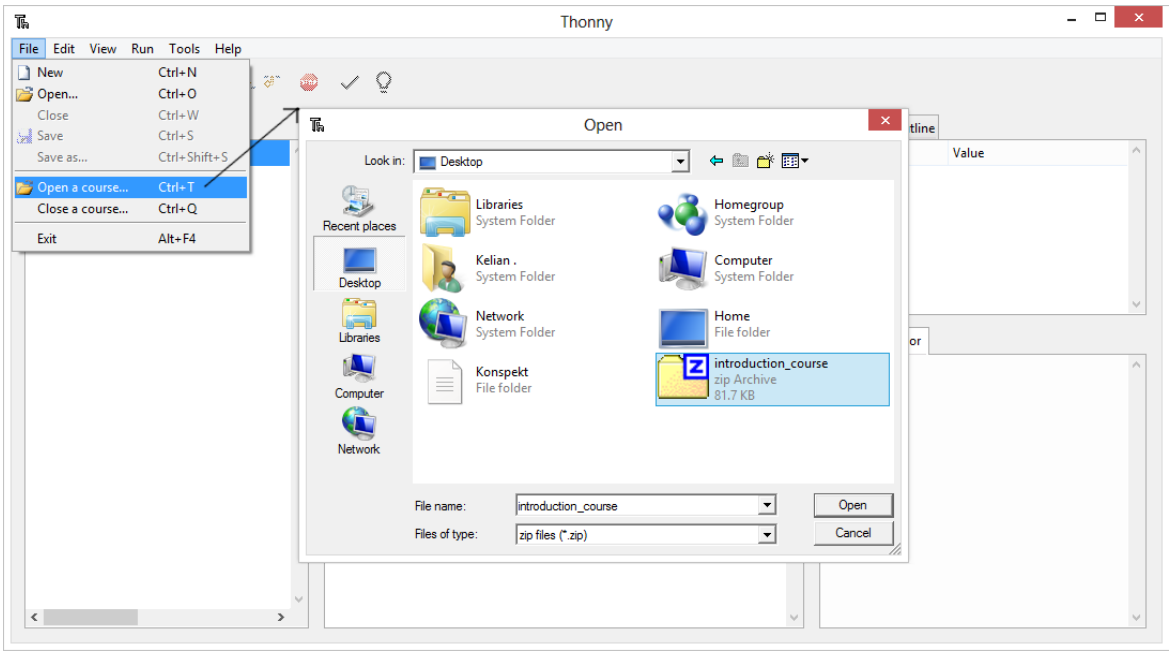
Teises loodud kaustas (vt joonist 11) hoitakse teemade kaustu ning *course.json* faili. Teemade kaustades on täpselt samad failid, mis eelmises asukohas, kuid neid faile ei muudeta. See on vajalik, et programm saaks taastada failide algset seisu ning kontrollida, kas kasutaja on ülesande lahendamisel algset faili muutnud.

3. Pistikprogrammi ülevaade

Thonny arenduskeskkond on üles ehitatud nii, et kolmandate osapoolte pistikprogramme on lihtne juurde lisada. Programm peab olema kirjutatud ühte Pythoni faili, sisaldama meetodit *load_plugin()* ning asuma Thonny kaustas *Plugins*. Kasutades Thonny's implementeeritud meetodeid on võimalik arenduskeskkonnale lisada juurde vaateid ja käsklusi ning siduda sündmusi uute funktsioonidega.

3.1 Kursuse avamine ja sulgemine

Kursuse avamiseks (vt joonist 12) tuleb Thonny's valida *File -> Open a course…* või kasutada kiirklahve Ctrl + T ning valida kursuse zip-faili asukoht.

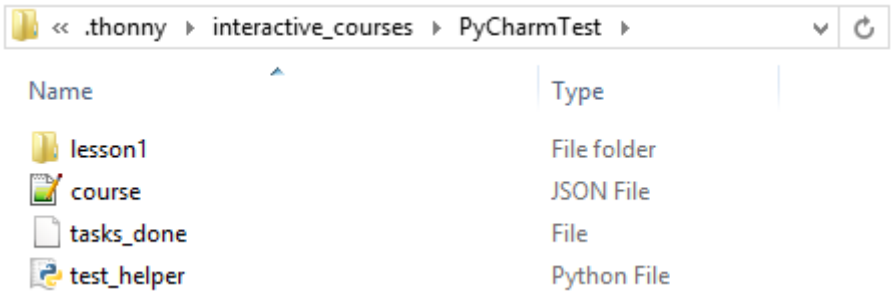


Joonis 12. Kursuse avamine.

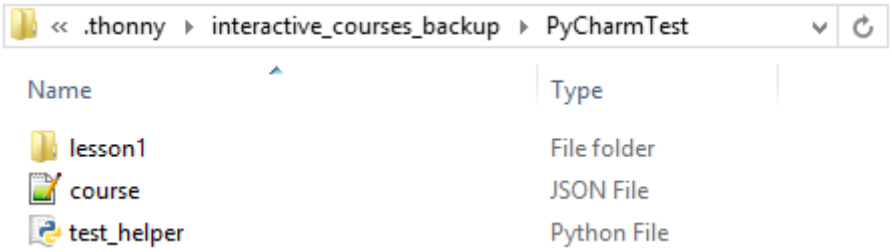
Nagu eelnevas peatükis mainitud, peab programmi töötamiseks kursuse failid asuma kahes kohas. Seega zip-faili avamisel kopeeritakse kõik kokkupakitud kataloogis olevad failid kahte kohta: *~\.thonny\interactive_courses* ja *~\.thonny\interactive_courses_backup*. Kursuse kaust on mõlemas asukohas sama nimega, mis oli kursuse zip-fail.

Esimeses asukohas (vt joonist 13) hoitakse muudetud faile. Uued kohatäidete positsioonid salvestatakse *course.json* faili. Kursuse zip-failis puudub *.idea* kaust ning seega selles kaustas olev *study_projext.xml* fail, kuhu on salvestatud, kas ülesanne on tehtud või mitte. Kuna viimast faili on väga raske ja mahukas ise luua, salvestatakse informatsioon tehtud ja tegemata ülesannetest faili *tasks_done*, mille pistikprogramm ise loob. Iga ülesande jaoks

on failis rida selle kohta, kas ülesanne on lahendatud või mitte vastavalt *<teema nimi>_<ülesande nimi>_true* ja *<teema nimi>_<ülesande nimi>_false*.



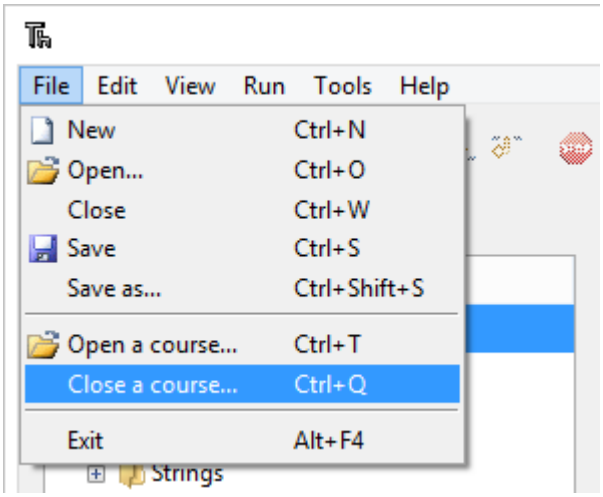
Joonis 13. Kursuse failid.



Joonis 14. Esialgsed kursuse failid.

Teises kaustas (vt joonist 14) hoitakse esialgsed failid. Selle järgi saab programm kontrollida, kas kasutaja on ülesande lahendamisel faile muutnud. Hiljem on võimalik edasise arendusvõimalusena implementeerida ülesande taastamise, sest on olemas muutmata failid ning *course.json* failis esialgsete kohatäidete positsioonid.

Pärast zip-faili avamist ning kaustade kopeerimist, lisab pistikprogramm kasutajaliidesele juurde vajalikud vaated: kursuse, ülesannete ja progressi vaate. Avatud kursuse kausta asukoht salvestatakse *.thonny* kausta *configuration* faili, et järgmine kord ei peaks kasutaja uuesti sama kursust avama, vaid programm teeb selle kasutaja eest ära.



Joonis 15.

Kursust saab sulgeda (vt joonist 15) kasutades käsklust: *File -> Close a course...*. Sellele vajutades sulgeb Thonny interaktiivseks lahendamiseks mõeldud vaated ning kustutab failist *configuration* kursuse kausta asukoha. Kursust saab ka sulgeda kasutades kiirklahve Ctrl + Q.

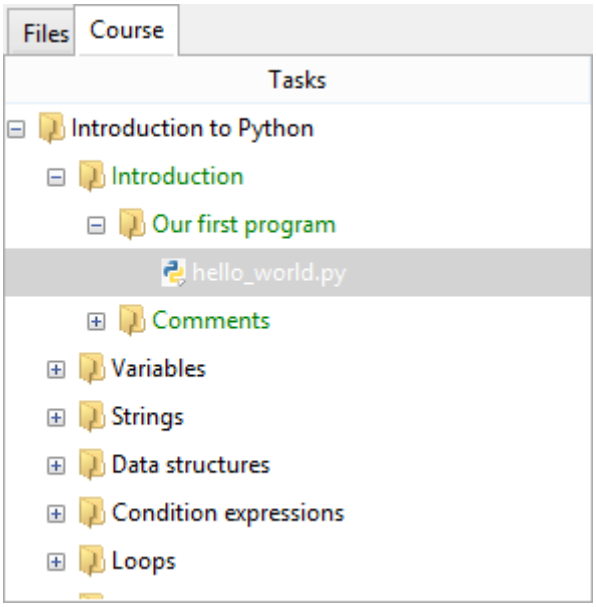
3.2 Pistikprogrammi vaated

Selles peatükis on põhjalik ülevaade kõikidest vaadetest, mis on vajalikud ülesannete interaktiivseks lahendamiseks.

Kursuse vaade

Kursuse kausta sisu kuvamiseks on eraldi vaade, sest *interactive_courses* kaustas (vt joonist 14) olev kursuse kaust sisaldab faili *test_helper.py*, *courses.json*, *task_done* ning igas ülesande kaustas on fail *tests.py*, mida kasutaja ei tohiks näha. Uus vaade sisaldab faili puud (vt joonist 16), mis ehitatakse faili *course.json* parsimise käigus. Failis sisaldub järgnev informatsioon, mis salvestatakse puusse:

a) kursuse nimi;
b) teema nimi,
c) teemade arv,
d) ülesande nimi,
e) ülesannete arv.

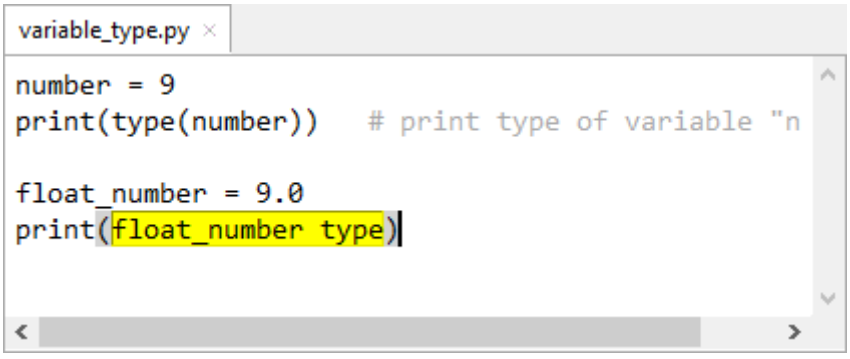


Joonis 16. Kursuse vaade.

Kursuse vaates ülesannete failide peale vajutades avatakse need koodi vaates. Kui kasutaja lahendab ülesande õigesti, muudetakse ülesande kausta nimi roheliseks. Kui on lahendatud kõik ülesanded, muudetakse vastava teema nimi roheliseks. Kursuse nimi muudetakse roheliseks, kui kõik teemade ülesanded on lahendatud.

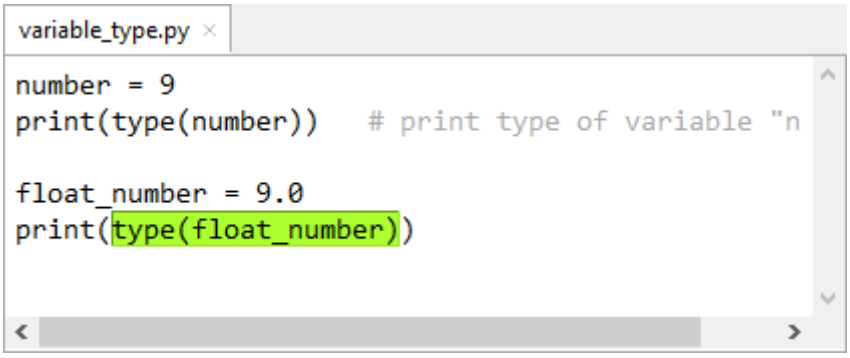
Koodi vaade

Ülesande avamisel kuvatakse fail koodi vaates ning lisatakse juurde kohatäited. Kohatäited saadakse kätte *courses.json* faili parsides (fail asub *interactive_courses* kausta kursuse kaustas).



Joonis 17. Koodi vaade tegemata ülesande puhul.

Kui antud ülesanne on tegemata, on kohatäited kollase värvusega (vt joonist 17). Tehtud ülesande puhul on nad helerohelised (vt joonist 18).

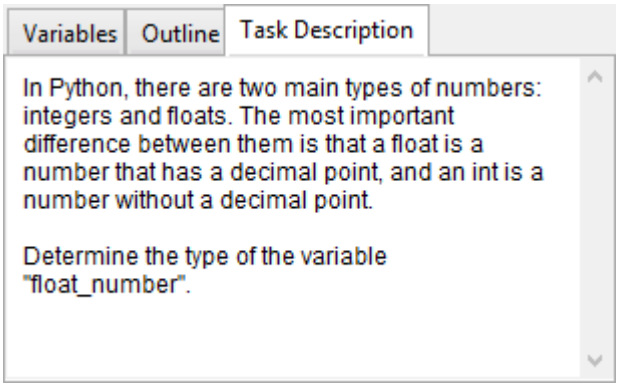


Joonis 18. Koodi vaade tehtud ülesande puhul.

Nii ülesande salvestamisel kui ka ülesande kontrollimisel salvestatakse kohatäidete uued positsioonid *courses.json* faili. Ülesande kontrollimisel salvestab pistikprogramm kohatäidete sees oleva teksti ülesande kausta, et programm saaks kontrollida, kas ülesanne on õigesti lahendatud. Faili nimi peab olema kujul *<ülesande faili nimi>_windows*.

Ülesande kirjelduse vaade

Avatud ülesande faili puhul kuvatakse ülesande kirjeldus eraldi vaatesse (vt joonist 19).

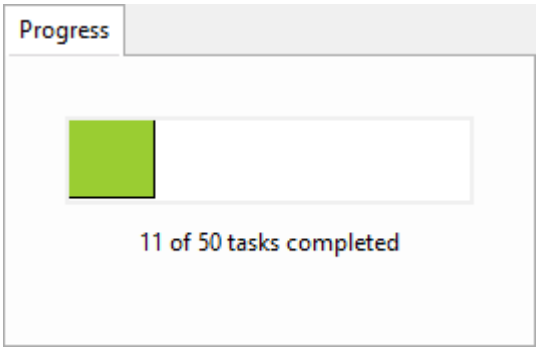


Joonis 19. Ülesande kirjelduse vaade.

Ülesande kirjeldus asub ülesande kaustas failis *task.html* ja on seega HTML formaadis. Kirjelduse vaate jaoks on kasutatud spetsiaalset teeki *tkinterhtml*, mis on loodud Aivar Annamaa poolt [8]. Teek võimaldab kuvada teksti võttes arvesse HTML-märgendeid.

Progressi vaade

Kogu kursuse progressi näidatakse eraldi vaates (vt joonist 20). Vaade sisaldab progressiriba, mis näitab graafiliselt, kui suur osa kursusest on tehtud. Selle all on kuvatud täpsemalt mitu ülesannet on kõikidest ülesannetest tehtud.

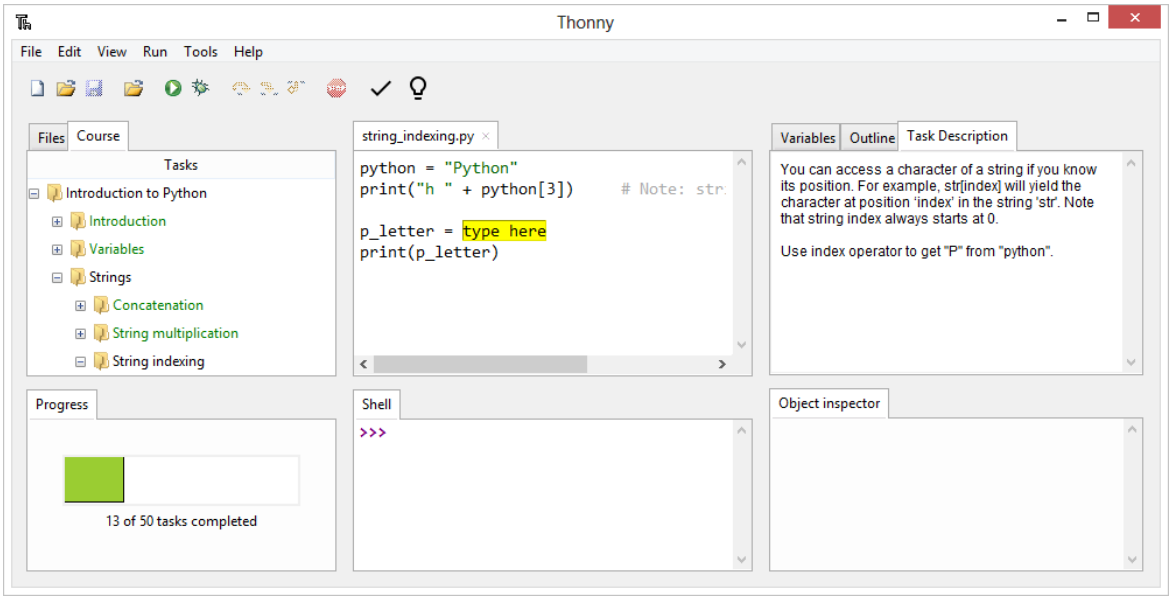


Joonis 20. Kursuse progressi vaade.

Kursuse avamisel arvutatake mitu ülesannet on kogu kursusel. Failist *tasks_done* (asub *interactive_courses* kaustas olevas kursuse kaustas) loetakse sisse, mis ülesanded on tehtud ning jäetakse see listi kujul meelde. Iga kord, kui kasutaja lahendab ülesande ära, lisatakse see listi ning kirjutatakse ka faili.

3.3 Ülesannete kontrollimine

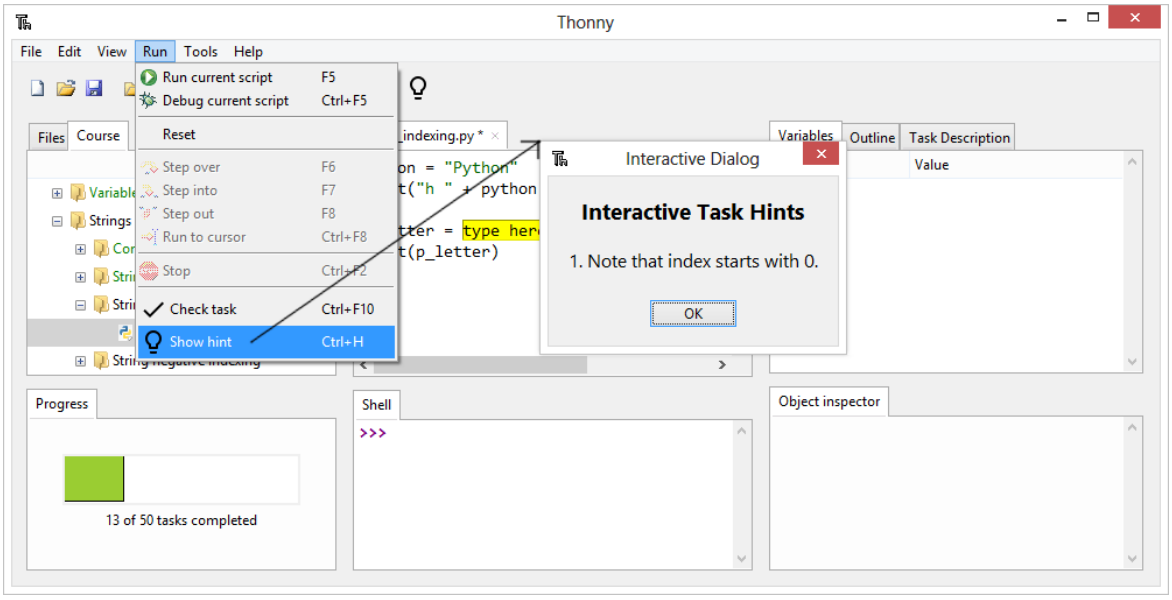
Ülesande lahendamiseks tuleb kursuse vaatest valida ülesanne ning selle peale vajutada, et ülesande kood ilmuks koodi vaatesse, kuhu pistikprogramm lisab vajalikud kohatäited. Kasutaja saab kirjelduse vaatest lugeda, mida peab koodis muutma (vt joonist 21).



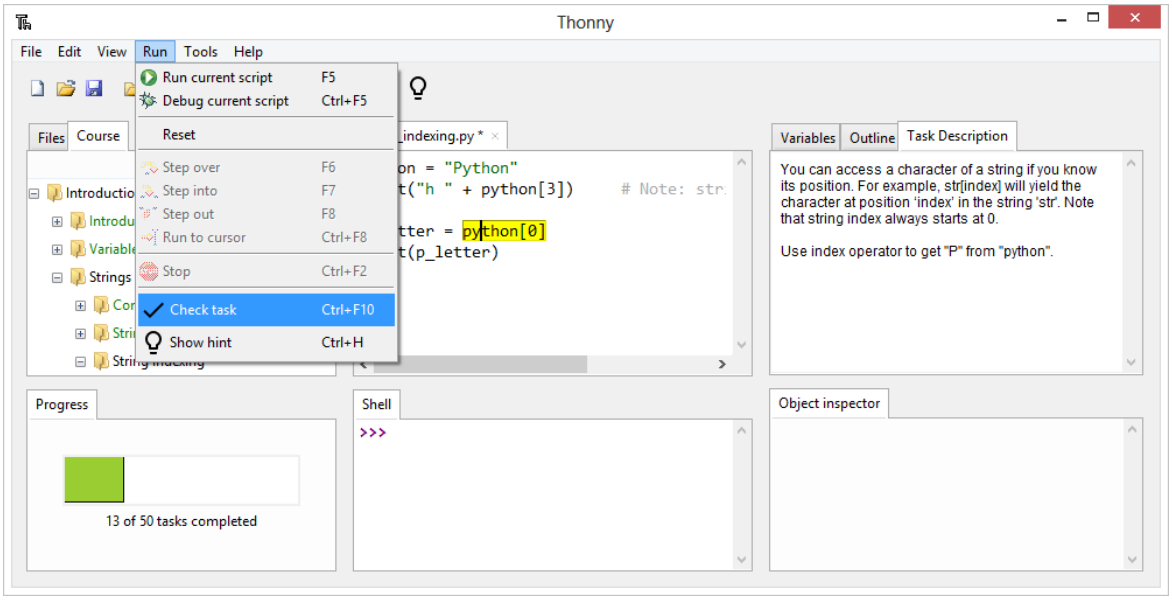
Joonis 21. Ülesande lahendamine.

Ülesande lahendamiseks peab täitma ülesande kohatäited õige koodiga. Hetkel kohatäide perfektselt ei tööta ehk teksti ei saa sisestada kasti selle algusest ega lõpust ning kohatäidet ei tohi täielikult ära kustutada, sest siis ei saa seda enam tagasi.

Ülesande lahendamisel on võimalik ka abi küsida. Selleks on kolm võimalust: valida käsklus *Run -> Show hint*, vajutada tööriistaribal lambipirni pildiga nuppu või kasutada kiirklahve Ctrl + H (vt joonist 22). Vihjed asuvad kursuse kausta *hints* kaustas või *course.json* failis. Avatud ülesande kõik vihjed kuvatakse korraga ühes hüpikaknas.



Joonis 22. Vihje näitamine.



Joonis 23. Ülesande kontrollimine.

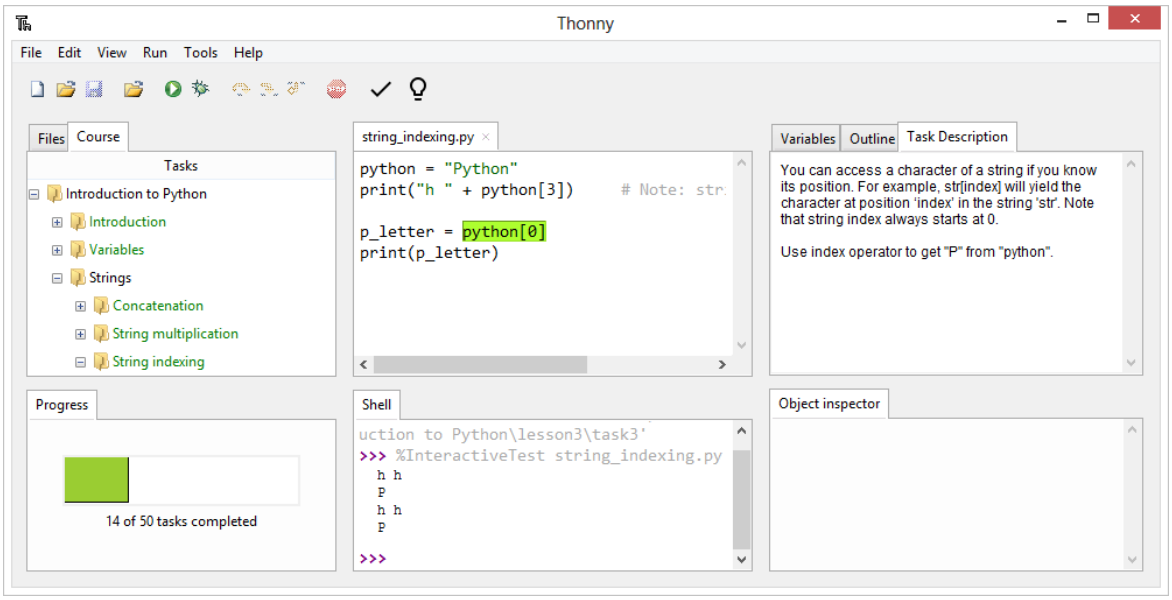
Ülesande kontrollimiseks on ka kolm võimalust: valida käsklus *Run -> Check task*, vajutada linnukese (ingl k *checkmark*) pildiga nuppu tööriistaribal või kasutada kiirklahve Ctrl + F10 (vt joonist 23). Pistikprogramm võtab kohatäidete sees olevad tekstid ning salvestab need

ülesande kausta eraldi faili, mille nimi on kujul *<ülesande faili nimi>_windows* nagu eelnevalt mainitud. Failis on iga ülesande kohatäide kujul *#educational_plugin_window = <kood kohatäites>*.

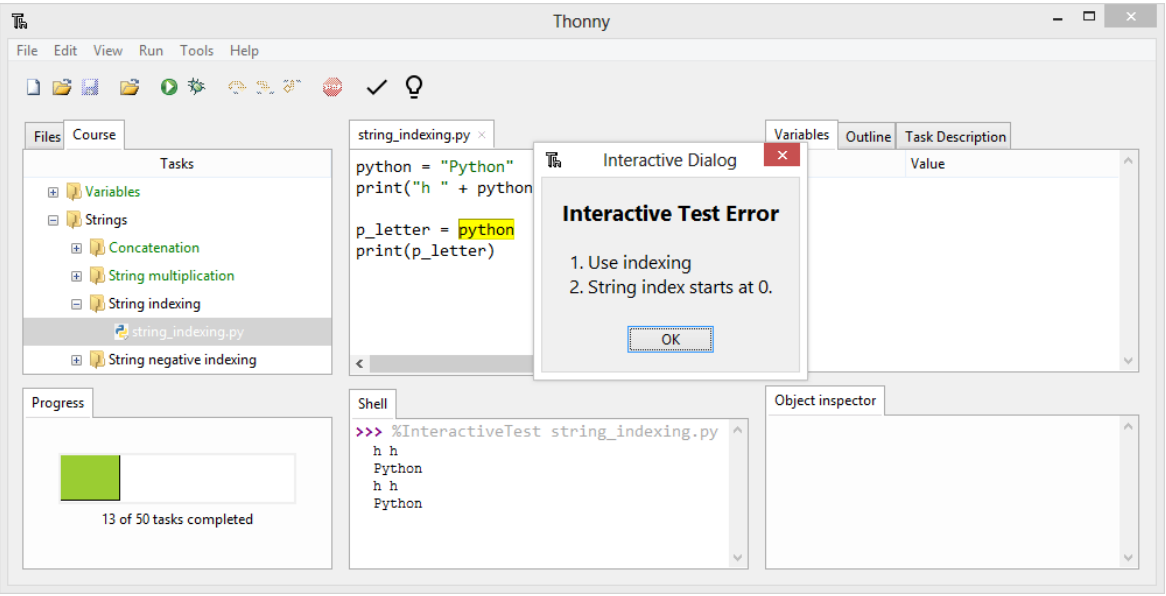
Ülesandeid kontrollitakse kursuse kaustas olevate failidega: *test_helper.py* ja *tests.py*. Nende jooksutamiseks on vaja pistikprogrammil lisada koodi argumentidesse kursuse kausta, ülesande faili ja *test_helper.py* faili asukoht. Kontrollimiseks käivitatakse fail *tests.py*, mis kutsub enda koodis *test_helper.py* meetodeid. Ülesande faili puhul kontrollitakse järgmist:

a) kas ülesande fail on tühi;
b) kas ülesande faili on muudetud;
c) kas kohatäite kaste on kustutatud;
d) kas failis on süntaksi vigu;
e) kas kohatäites on õige kood.

Testide väljund kuvatakse koos koodi väljundiga *Shell* vaatesse. Iga testi kohta on väljundis rida kujul *#educational_plugin <testi nimi> test OK* või *#educational_plugin <testi nimi> test FAILED + <tagasiside>*, mille pistikprogramm võtab vaatest kiiresti välja ning salvestab selle eraldi listi. Peale seda kontrollib pistkprogramm, kas kõik testid läksid läbi. Kõikide testide õnnestumisel salvestatakse selle ülesande kohta märge faili *tasks_done*. Lisaks salvestatakse märge pistikprogrammis olevasse listi, kuhu on salvestatud kõik tehtud ülesanded. Graafilises kasutajaliideses (vt joonist 24) muudetakse kohatäited roheliseks, värskendatakse progressi vaadet ning muudetakse kursuse vaates ülesande kaust roheliseks.



Joonis 24. Õigesti lahendatud ülesanne.



Joonis 25. Valesti lahendatud ülesanne.

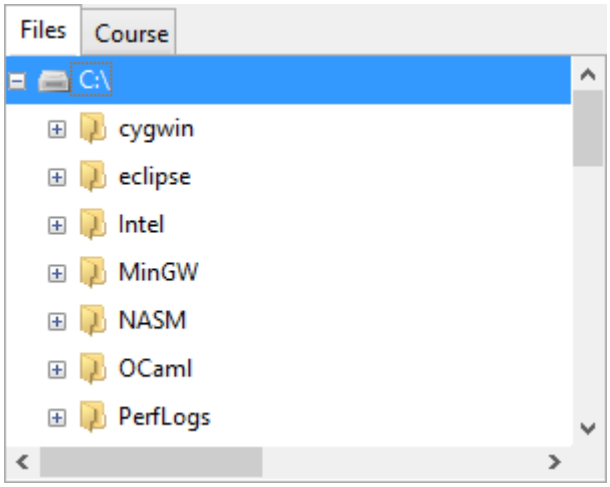
Vähemalt ühe testi ebaõnnestumisel testi õigesti lahendatuks ei loeta. Hüpikaknas kuvatakse tagasiside (vt joonist 25), mis aitab kasutajal ülesannet õigesti lahendada.

4. Töö protsess

Antud peatükis antakse ülevaade pistikprogrammi töö protsessist, mis järjekorras funktsionaalsusi implementeeriti ja mis probleemid tekkisid.

Tööd alustati PyCharm Edu programmi tutvumisega. Sellele järgnes Thonny arenduskeskkonna ja selle koodi üle vaatamine. Suur viga protsessis oli põhjalikku analüüsi ära jätmine. Töö autor oleks pidanud põhjalikult tutvuma, mis failidest koosnevad PyCharm Edu's avatud kursused, kus nad on ning millistest failidest koosnevad allalaetavad kursused. Selle asemel leiti esimesena PyCharm Edu *PyCharmProjects* kaustas kursuse kaust, mis võeti töö kirjutamisel aluseks. Informatsioon teemade ja ülesannete nimede ning kohatäidete positsioonide kohta saadi faili *study_project.xml* parsimise käigus. Lisaks oli failis märge, kas ülesanne on lahendatud või mitte. Kiirelt alustati pistikprogrammi koodi kirjutamisega, sest see oli autori jaoks põnev.

Esimesena implementeeriti koodi kirjelduse vaade. Teiseks võeti ette kursuse vaade, kus kasutaja saaks navigeeruda ülesannete vahel. Alguses oli mõte kasutada failide vaadet (vt joonist 26) ja kui kasutaja avab interaktiivseks lahendamiseks mõeldud kursuse, siis pistikprogramm jätab vaatest välja mittevajalikud failid. See aga osutus liiga keeruliseks ja Thonny'le tuli lisada kursuse vaade.



Joonis 26. Failide vaade.

Kursuse vaade koosnes alguses kahest veerust. Esimeses veerus olid kursuse failid ning teises märge "+" või "-" olenevalt, kas ülesanne oli tehtud või mitte. Hiljem muudeti vaade PyCharm Edu's oleva vaate sarnaseks (vt jooniseid 4 ja 16), sest nii on kasutajal kergem aru saada, mis ülesanded on tal tehtud ja mis mitte.

Peale seda implementeeriti koodi vaatesse kohatäited ja liiguti edasi ülesannete testimise juurde. Testifaile läbi vaadates ning püüdes nendest aru saada, avastati, et PyCharm Edu's asub kursuse kaust kahes kohas. Bakalaureusetöö autor mõistis, et kursuse kaustad tuleb salvestada kahte kohta Thonny enda failide alla.

Järgmiseks võeti ette progressi vaade ning seal probleeme ei tekkinud. Peale põhifunktsionaalsuste implementeerimist proovis töö autor PyCharm Edu's ise luua oma kursuse, et testida seda Thonny pistikprogrammis. Ilmnes, et teistele jagatav kursuse kaust on zip-failina ning seal puudub *.idea* kaust ja seal olev *study_projext.xml* fail. Seega pidi suure osa koodist ümber kirjutama, sest nüüd saadi kursuse kohta informatsioon hoopis *course.json* faili parsides. Lisaks puudub *course.json* failis märge selle kohta, kas ülesanne on tehtud või mitte, ja selle jaoks loodi ise faili *tasks_done*.

5. Edasised arendamisvõimalused

Kõige tähtsam pistikprogrammi seisukohalt on hetkel vigade parandus. Thonny sulgemisel visatakse vahepeal veateateid. Bakalaureusetöö autor pole märganud kindlat mustrit, kuna need veateated tekivad ning pole leidnud, mis seda põhjustab.

Parandamist vajab probleem kohatäidetega, mida mainiti ka eelmises peatükis. Kirjutades kohatäite kasti algusesse või lõppu, ei liigu kast sellega kaasa. Kohatäidet ei tohi hetkel ära kustutada, et asendada seda uue koodiga, sest siis kaob kast täielikult ära ja tagasi seda enam ei saa.

Kasutajale oleks ka abiks, kui on võimalus avada ja kustutada kursuseid, mis on juba salvestatud *.thonny* kursuste kausta.

Lisaks on võimalus implementeerida funktsionaalsusi, mis on PyCharm Edu'l ja antud pistikprogrammil veel puuduvad. Eelmises peatükis mainiti, et edasise arendusvõimalusena võiks lisada programmile võimaluse taastada ülesande esialgne kuju. Kasutajale oleks ka abiks ülesannete vaheline navigeerimine edasi-tagasi nooltega nagu näidatud joonisel 7. Kasutajale tuleks kasuks võimalus lisada kursusele kaust *Sandbox*, kuhu saab kasutaja luua testimiseks enda faile.

6. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua arenduskeskkonnale Thonny pistikprogramm automaatse tagasisidega ülesannete lahendamiseks, millega on võimalik avada PyCharm Edu's loodud ülesannete komplekte. Pistikprogrammi jaoks implementeeriti põhifunktsionaalsused, milleks on ülesannete komplekti avamine, lahendamine, vihjete küsimine ning tagasiside saamine. Edasise arendamisvõimalusena on võimalik lisada veel funktsionaalsusi.

Autori arvates oli väga põnev kirjutada pistikprogrammi, mis aitaks tudengitel õppida lihtsamini programmeerimist. Töö kirjutamise käigus õppis autor, et programmi alguses tuleks teha põhjalik analüüs sellest, mis failidega programm suhtlema hakkab, et hiljem vältida koodi ümber kirjutamist.

Valminud programm on kindlasti tudengitele kasulik Pythoni programmeerimiskeele õppimisel. Ülesannete lahendamisel pakub programm palju abi vihjete ning kohese tagasisidega.

7. Kasutatud materjalid

[1] Filippov, Dmitry. “JetBrains Debuts PyCharm Educational Edition” https://blog.jetbrains.com/pycharm/2014/10/jetbrains-debuts-pycharm-educational-edition/ (15.04.2016)

[2] Annamaa, Aivar.”Introducing Thonny, a Python IDE for Learning Programming”, Proceedings of the 15th Koli Calling Conference on Computing Education Research, 117-121 (2015).

[3] Tartu Ülikooli kursus “Programeerimisest maalähedaselt” https://courses.cs.ut.ee/2015/progmaa/fall/Main/Thonny (15.04.2016)

[4] Tkinteri dokumentatsioon. https://docs.python.org/2/library/tkinter.html (11.05.2016)

[5] Thonny. https://bitbucket.org/plas/thonny/downloads (15.04.2016)

[6] PyCharm Edu “How It Works” https://www.jetbrains.com/pycharm-edu/concepts/ (10.05.2016)

[7] Filippov, Dmitry. “Announcing Educational Plugin for PyCharm 5” https://blog.jetbrains.com/pycharm/2015/11/announcing-educational-plugin-for-pycharm-5/ (17.04.2016)

[8] Annamaa, Aivar. Pythoni teek tkinterhtml. https://pypi.python.org/pypi/tkinterhtml (10.05.2016)

Lisad

Lisa 1. Pistikprogrammi seadistamise juhend

Pistikprogrammi seadistamise juhend asub programmi BitBucketi repositooriumi Viki lehel, mis on kättesaadav järgnevalt aadressilt: https://bitbucket.org/kelian/thonny_interactive/wiki/Pistikprogrammi%20seadistamise%20juhend

Lisa 2. Pistikprogrammi kasutusjuhend

Pisikprogrammi kasutusjuhend asub programmi BitBucketi repositooriumi Viki lehel, mis on kättesaadav järgnevalt aadressilt: https://bitbucket.org/kelian/thonny_interactive/wiki/Pistikprogrammi%20kasutusjuhend

Lisa 3. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Kelian Kaio**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   Thonny arenduskeskkonna pistikprogramm automaatse tagasisidega ülesannete lahendamiseks,
   (*lõputöö pealkiri*)

   mille juhendaja on Aivar Annamaa,
   (*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, **12.05.2016**