TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Alexis Alliksaar

Programmeerimise ainele PyCharmi toe loomine

Bakalaureusetöö (9 EAP)

Juhendaja: Reimo Palm, PhD

Tartu 2024

Programmeerimise ainele PyCharmi toe loomine

Lühikokkuvõte:

PyCharmi kasutamine programmeerimise õppimiseks võimaldab algajatel programmeerijatel saada kasu tänapäevase professionaalse integreeritud programmeerimiskeskkonna võimalustest ja annab neile kogemuse professionaalse arenduskeskkonnaga töötamisel. PyCharmi toe loomiseks programmeerimise baaskursusele Programmeerimine loodi IntelliJ platvormi arenduskeskkondasid Lahenduse keskkonnaga ühendav pistikprogramm ja eestikeelne juhend arenduskeskkonnaga tutvumiseks. PyCharmi kasutuselevõtmine Programmeerimise aine raames annab võimaluse edasi uurida professionaalsete arenduskeskkondade kasutamise mõju programmeerimise õpetamisel.

Võtmesõnad:

Integreeritud programmeerimiskeskkond, IDE, PyCharm, Thonny, programmeerimise baaskursus

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

Integrating PyCharm with the Programming course

Abstract:

Using PyCharm for learning programming gives novice programmers the ability to benefit from the features of the modern, professional, integrated development environment and provides them with experience of working with professional IDEs. To support PyCharm usage in the introductory programming course, Programming, a plugin was developed, which integrates the IntelliJ Platform IDEs with the Lahendus environment, and a user guide in Estonian was created for the IDE. Introducing PyCharm within the Programming course provides the opportunity to further investigate the impact of using professional development environments in programming education.

Keywords:

Integrated development environment, IDE, PyCharm, Thonny, introductory programming course

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

Sisukord

1. Sissejuhatus

Integreeritud programmeerimiskeskkond (ingl *integrated development environment, IDE*) on programmeerimiskeskkond, mis on rakendustarkvarasse sisse ehitatud. Harilikult koosneb integreeritud programmeerimiskeskkond lähtekoodi redaktorist, kompilaatorist või interpretaatorist või mõlemast ning programmeerimise automatiseerimise abivahenditest, millele enamasti lisandub ka silur[1]. Programmeerimiskeskkonna eesmärk on suurendada programmeerija efektiivsust ja lihtsustada ta tööd arendusprotsessi jooksul. Seetõttu on arenduskeskkond üks olulisemaid tööriistu programmeerijale.

Küsimus, millist arenduskeskkonda kasutada programmeerimise õpetamiseks, on siiani akadeemilise arutelu all. Mõned arvavad, et professionaalsete arenduskeskkondade kasutamine sellel eesmärgil ei ole sobilik, aga samas leidub näiteid, kus professionaalseid arenduskeskkondi on edukalt õpetamiseks rakendatud. Kuna programmeerima õppimist peetakse keeruliseks, siis on oluline uurida võimalusi, kuidas toetada algajate programmeerijate arengut läbi integreeritud programmeerimiskeskkondade.

Käesoleva töö eesmärk on muuta Tartu Ülikooli aine Programmeerimine LTAT.03.001 PyCharmi arenduskeskkonnaga läbitavaks ja luua PyCharmile vähemalt sama hea tugi kui on hetkel aine raames kasutataval arenduskeskkonnal Thonny. PyCharm pakub Thonnyst rohkem funktsionaalsuseid efektiivseks programmeerimiseks ja õppetööks. PyCharmi toe loomine laiendab aine läbimiseks kasutatavate arenduskeskkondade valikut ning annab tudengitele võimaluse saada kogemust professionaalse arenduskeskkonnaga töötamisega.

PyCharmi toe loomiseks oli tarvis arendada PyCharmile Lahenduse keskkonnaga ühilduv pistikprogramm. Pistikprogramm või plugin (ingl *plugin/plug-in*) on tarkvara komponent või moodul, mida saab integreerida olemasolevasse süsteemi või rakendusse, et laiendada selle funktsionaalsust või lisada uusi omadusi. Pistikprogrammid pakuvad lihtsat ja modulaarset viisi tarkvara kohandamiseks või täiendamiseks, ilma et peaksid muutma rakenduse põhiteeki või programmikoodi[2]. Loodud pistikprogrammi on võimalik kasutada ka teistes Lahenduse keskkonnaga seotud programmeerimise ainetes. Samuti lõin algajatele programmeerijatele PyCharmi tutvustava materjali.

[1] http://www.vallaste.ee/

[2] https://chat.openai.com/

Toon töös välja Thonny ja PyCharmi erinevused nende olulisemate funktsionaalsuste osas ning kirjeldan loodud pistikprogrammi ja selle arendusprotsessi. Viimaseks toon välja võimalikud suunad edasisteks uuringuteks ja teadustööks.

2. Taustainfo

Programmeerimine on Tartu Ülikooli arvutiteaduse instituudi programmeerimise baaskursus, mille eesmärk on anda alusteadmised programmeerimise põhikonstruktsioonidest ja esmased oskused algoritmide ja programmide koostamiseks. Kursus hõlmab süstemaatilist õpet programmeerimiskeeles Python. Senimaani on kursuse raames kasutatud programmeerimisülesannete lahendamiseks integreeritud programmeerimiskeskkonda Thonny, sest sellesse on sisse ehitatud kasutaja tegevuste logimise tugi ning see on läbi pistikprogrammi integreeritud kursusel kasutatava lahenduste esitamise keskkonnaga Lahendus[3]. Tudengite poolt esitatud kasutaja tegevuste logisid analüüsitakse kursusel plagiaadituvastuse ning programmeerimise õpetamise alase uurimistöö eesmärkidel.

2.1 Thonny

Thonny on avatud lähtekoodiga Pythoni arenduskeskkond. Aivar Annamaa, Thonny autor, on kirjutanud [1], et Thonny arendamisel peeti silmas kasutajasõbralikkust algajate programmeerijate suhtes. Ta mainib, et Thonny tööriistade hulka kuuluvad võimalused graafiliselt kujutada programmeerimise põhilisi mõisteid nagu muutujad, käskude täitmisjärg, avaldiste väärtustamine, funktsioonide väljakutsumine, rekursioon, viidatüüpi muutujad, objektid, andmestruktuurid ja sisend-väljundvood. Lisaks on Thonny kodulehel [2] välja toodud keskkonna poolt pakutavad funktsioonid:

1. Kaasapandud Pythoni versioon, et kasutaja ei peaks ise Pythonit alla laadima.
2. Lihtsasti kasutatav koodisilur, millega on võimalik iga avaldise väärtuse muutumist uurida.
3. Süntaksivigade ja muutujate skoopide märgistamine koodiredaktoris.
4. Automaatne koodilõpetamissoovituste pakkumine.
5. Algajasõbralik käsuviip.
6. Graafiline kasutajaliides Pythoni paketihalduri jaoks.

Thonny üheks peamiseks eeliseks teiste arenduskeskkondade ees on selle madal arvutiressursside kasutus. Selle tulemusel on seda võimalik efektiivselt kasutada ka süsteemides, kus arvuti jõudlusele esinevad riistvaralised piirangud. Näiteks on Thonny

[3] https://lahendus.ut.ee/

eelinstallitud operatsioonisüsteemi Raspberry Pi OS [3] peal, mis on mõeldud Raspberry Pi monoplaatarvutite jaoks.

2.2 PyCharm

PyCharm [4] on JetBrainsi poolt arendatud IntelliJ platvormil põhinev avatud lähtekoodiga integreeritud programmeerimiskeskkond, mis on peamiselt mõeldud professionaalsetele Pythoni tarkvaraarendajatele. Jetbrains pakub PyCharmist kahte versiooni.

- PyCharm Community Edition, mis on avatud lähtekoodiga ja sisaldab kõiki peamisi programmeerimiskeskkonna võimalusi. On sobilik rakenduste arendamiseks, mis on kirjutatud ainult Pythonis ja seega piisav Programmeerimise aine läbimiseks.
- PyCharm Professional Edition, mis sisaldab lisaks tugimooduleid erinevate spetsiifilisemate rakenduste arendamiseks, nagu näiteks erinevate veebirakenduste raamistike või SQL keele tugi. On tudengitele saadaval tasuta õppe-eesmärgil, kuid muidu rakendatakse tellimusel põhinevat mudelit.

2023. aasta Stack Overflow tarkvaraarendajate uuringus [5] tehtud küsitluses märkis veidi üle 10% vastanutest, et nad on viimase aasta jooksul regulaarselt PyCharmi kasutanud, paigutades selle üldises IDE-de edetabelis populaarsuselt kaheksandale kohale. PyCharm oli uuringus populaarseim Pythonile spetsialiseerunud arenduskeskkond. PyCharmi kodulehel [4] on välja toodud peamised PyCharm Community Editioni keskkonna funktsioonid.

1. Intelligentne koodiredaktor, mis pakub intelligentseid koodilõpetamissoovitusi, analüüsib koodi ja märkab võimalikke vigu, nagu näiteks süntaksivigu, võimalikke loogikavigu või koodi mittevastavust Pythoni stiilistandarditele ja pakub vigadele automaatselt lahendusi ning tõstab esile koodi süntaksi, mis teeb koodilugemise kergemaks.
2. Koodis navigeerimine, mis võimaldab mugavalt üles leida soovitud elemente, nagu näiteks väljakutsutava funktsiooni defineerimise. Samuti on Pythoni standardmoodulite dokumentatsiooni võimalik lugeda otse arenduskeskkonnas.
3. Koodi ümberkorraldamise (ingl *refactoring*) võimalus, mida kasutades saab lihtsustada või muuta programmi struktuuri, näiteks muutuja või funktsiooni nime muutmine terves projektis.

4. Koodisilur, mis peatab programmi töö soovitud koodireal või erindi viskamisel ning võimaldab uurida programmi seisu.
5. Integreeritus versioonihaldussüsteemidega võimaldab oma projekti hallata läbi graafilise kasutajaliidese.

IntelliJ platvormi arenduskeskkondade modulaarse ülesehituse tõttu on võimalik neile lisada enda arendatud või olemasolevaid pistikprogramme lisafunktsionaalsuste võimaldamiseks.

2.3 Thonny ja PyCharmi võrdlus

2021. aastal JetBrainsi ja Python Software Foundationi poolt tehtud Pythoni tarkvaraarendajate uuringus [6] toodi välja, et 89% arendajatest kasutavad koodiredaktoris automaatseid koodilõpetamise soovitusi, 87% koodi ümberkorraldamise võimalusi ja versioonihaldussüsteeme, 79% kasutavad koodi lintimist ehk programmikoodist automaatset programmi- ja stiilivigade leidmist[4] ja 74% koodisilurit.

Thonny kasutab automaatsete koodilõpetamiste soovitamiseks teeki Jedi[5], mis on Pythoni koodi staatilise analüüsi tööriist, tänu millele pakub Thonny kasutajale seni kirjutatud lauseosa põhjal tähestikulises järjekorras sama algusega loogilisi võimalusi koodi lõpetamiseks. PyCharmi lähenemine automaatsetele soovitustele on sarnane, aga lisaks on välja toodud, et soovituste järjestamisel arvestatakse ka lause konteksti (see funktsionaalsus töötab masinõppe põhjal) [4]. Samuti proovitakse seni kirjutatud lauseosa sobitada võimalike variantidega ka mujal kui sõne alguses [4]. See annab eelise Thonny ees, sest kui programmeerija soovib välja kutsuda mõnda nimelist objekti, siis ta ei pea mäletama selle nime algust, vaid piisab sellest, kui tal on ükskõik milline osa sellest nimest meeles.

Thonnyl puuduvad koodi ümberkorraldamise tööriistad, samas kui PyCharm lubab sooritada koodi peal operatsioone nagu funktsioonide või meetodite signatuuride muutmine, elementide ümbernimetamine ja kopeerimine või liigutamine, koodilõigust funktsiooni või meetodi tuletamine jt [4].

[4] https://akit.cyber.ee/

[5] https://jedi.readthedocs.io/en/latest/

Samuti on PyCharm, erinevalt Thonnyst, integreeritud erinevate versioonihaldussüsteemidega, mis võimaldab Programmeerimise kursuse tudengitel mugavalt arendada aines sooritatavat grupiprojekti.

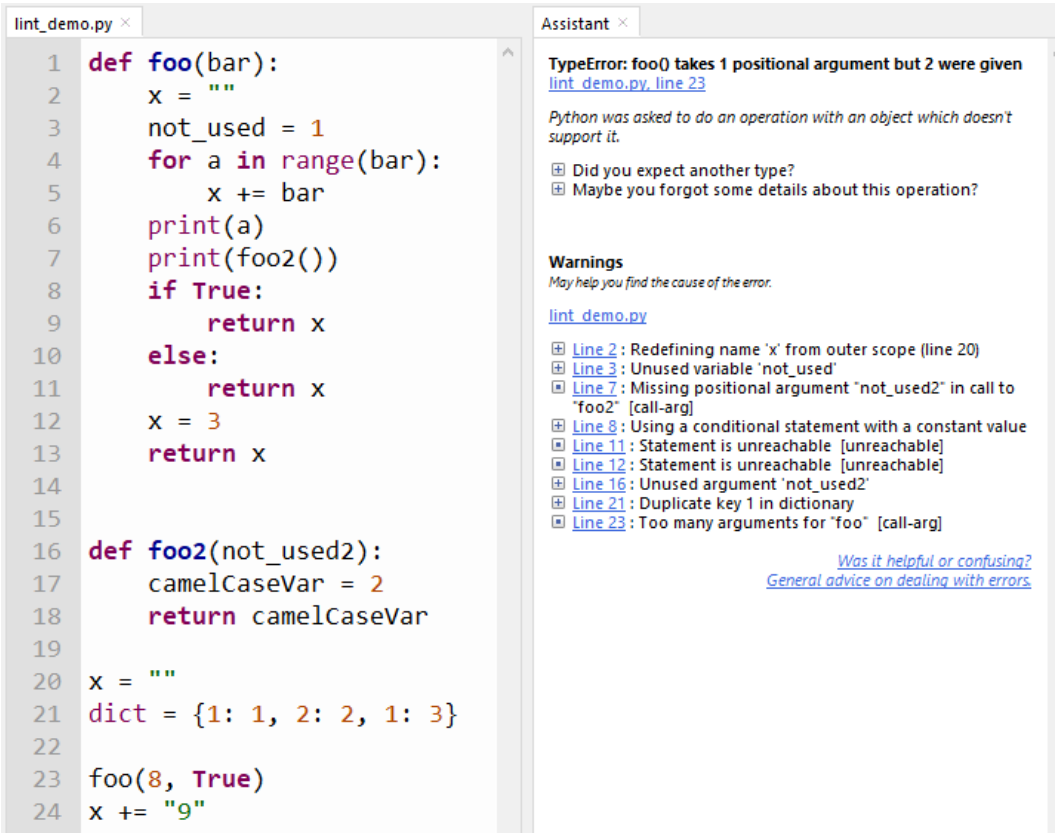
Thonny kasutab koodi lintimiseks staatilisi koodianalüsaatoreid Mypy[6] ja Pylint[7]. Lintijate tulemused on kasutajale esitatud tekstilisel kujul programmi jooksutamisel ja süntaksivigade puudumisel eraldi aknas nimega „Assistant“ (eestikeelses kasutajaliideses „Juhendaja“) pealkirja „Warnings“ all. PyCharm kasutab lintimiseks peamiselt samuti Pylinti [7], kuid erinevalt Thonnyst kujutatakse avastatud vead koodi kirjutamise jooksul tekstiredaktoris graafiliselt, mis koheselt juhib kasutaja tähelepanu probleemile. PyCharmis on veateated enamasti paigutatud kolme peamisse kategooriasse vastavalt vea raskusastmele. Lisaks pakutakse kasutajale osade vigade paranduseks automaatselt soovitusi. Erinevalt Thonnyst märgib PyCharm ära ka Pythoni stiilistandardile PEP-8 (Python Enhancement Proposal #8)[8] mittevastavad koodilõigud. Järgmistel lehekülgedel asuvatel joonistel 1 ja 2 on esitatud Thonny ja PyCharmi lintijate võrdlus süntaksivigadeta koodi korral ning joonistel 3 ja 4 süntaksivigadega juhul.

Ioannis Karvelas [8] on kirjutanud, et üks murekoht, mis aeglustab algajate programmeerijate efektiivsust, on põhjustatud arenduskeskkondade mitteselgetest veateadetest. Tao Dong ja Kandarp Khandwala [9] on kirjutanud, et programmeerimise veateadete kujutamisel aitab visuaalse disaini tehnikate kasutamine vigadest aru saada ning neid kiiremini parandada. PyCharmi lintija graafiline veateadete esitamine, vigade paranduste soovitamine, selged veateated ning lintimisprotsessi toimimine ka süntaksivigadega koodi puhul annavad sellele siinkohal eelise teiste arenduskeskkondade ees.

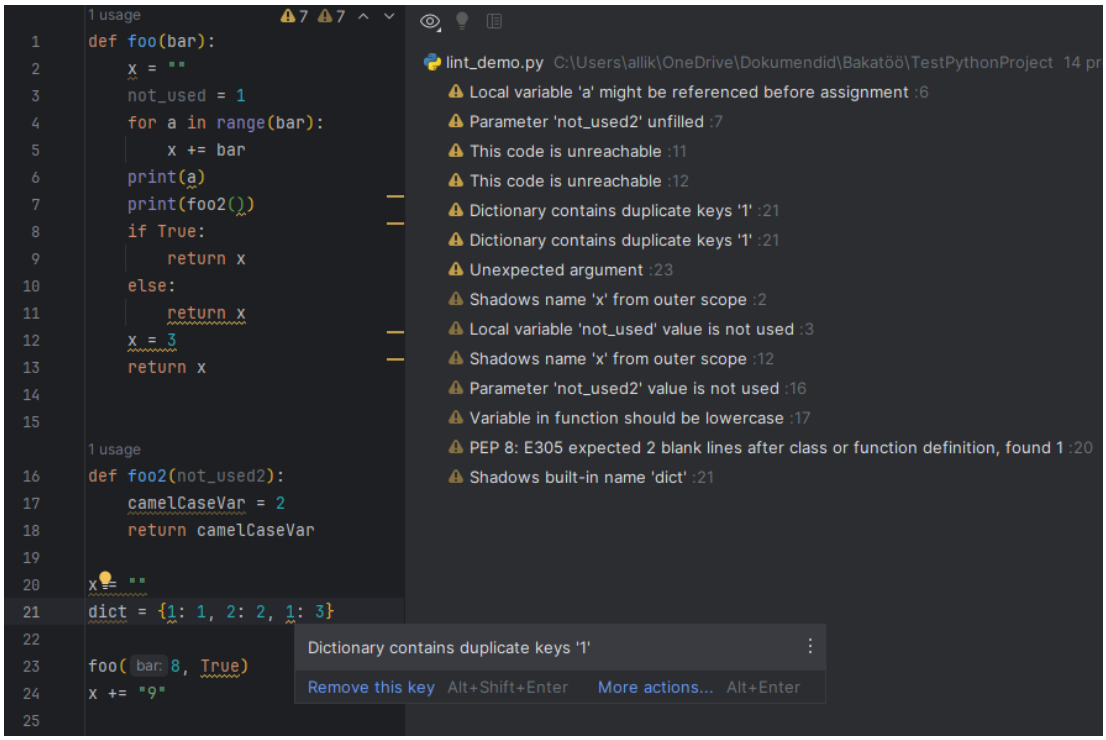
[6] https://mypy-lang.org/

[7] https://pylint.readthedocs.io/en/stable/

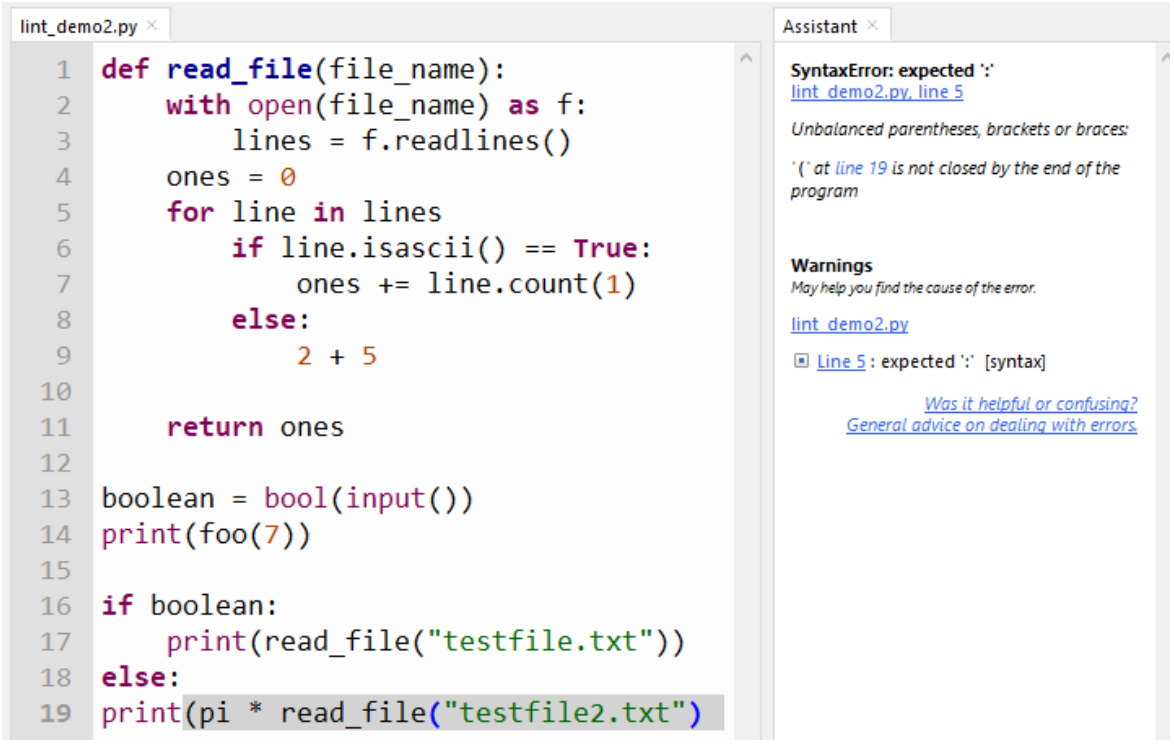
[8] https://pep8.org/



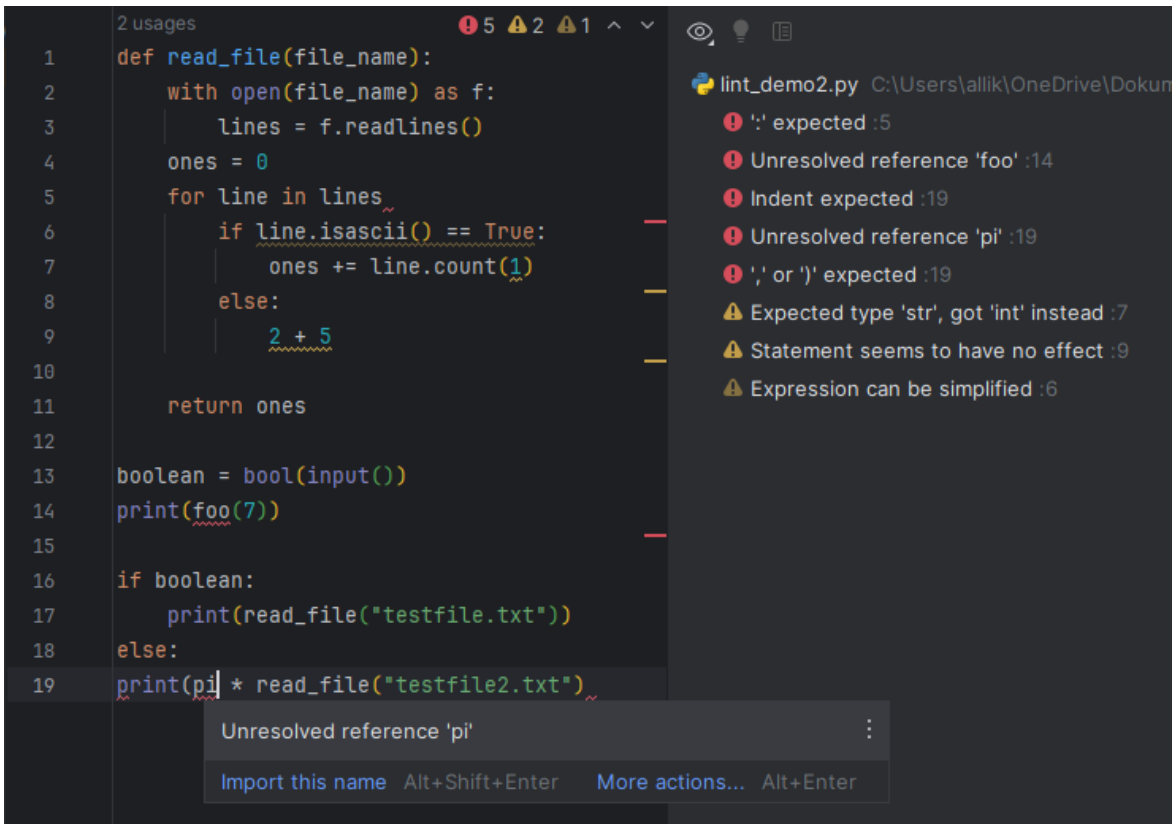
Joonis 1. Thonny lintija



Joonis 2. PyCharmi lintija



Joonis 3. Thonny lintija (süntaksivigadega kood)



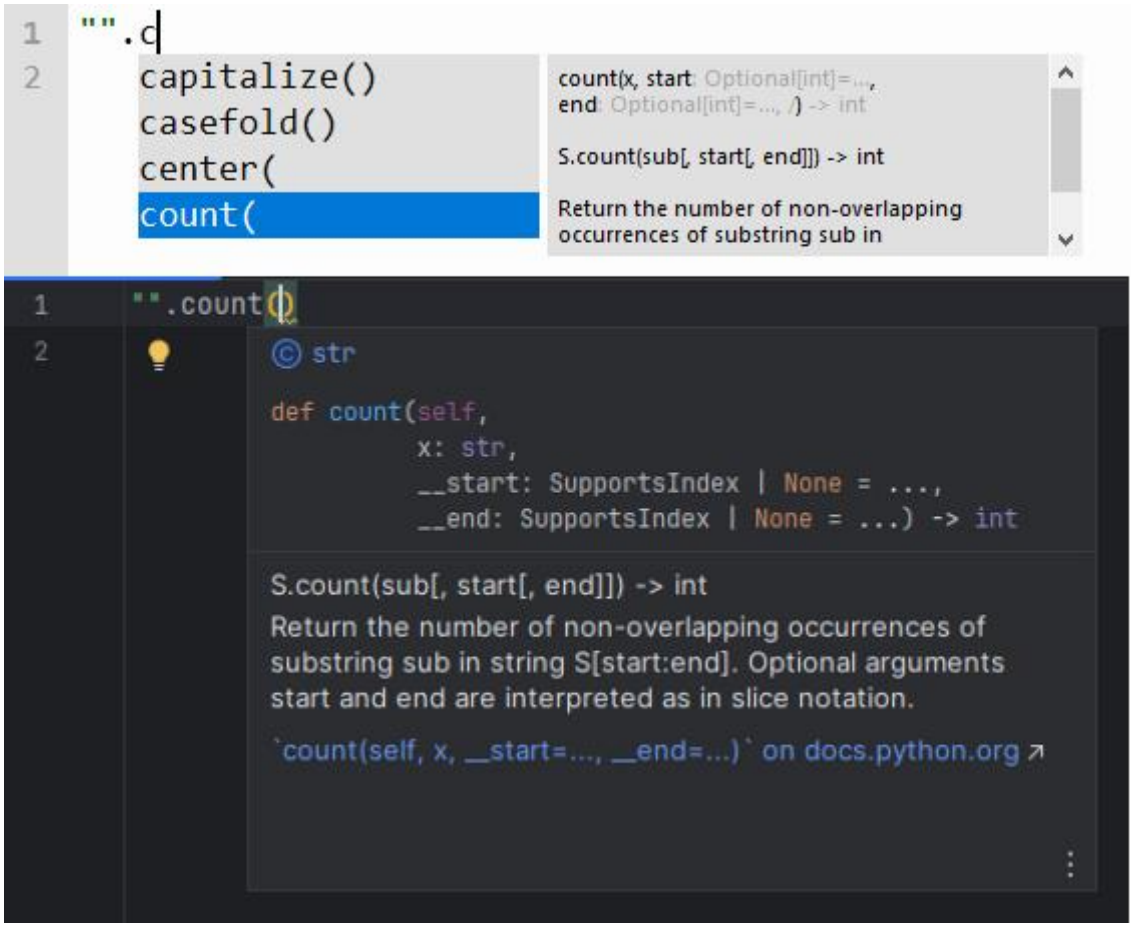
Joonis 4. PyCharmi lintija (süntaksivigadega kood)

Thonny koodisiluriga on võimalik väga detailselt uurida, kuidas programmi järjest täidetakse ja avaldisi väärtustatakse. Siluril on kaks erinevat töörežiimi: programm peatatakse esimesel koodireal ning programm peatatakse esimesel kohatud katkepunktil, mille asukohti on võimalik märkida koodiredaktoris. Seejärel on kasutajal võimalus liikuda järgmiste koodiplokkide juurde või sügavamalt analüüsida käsil olevat koodilõiku. Kui programmi töö on peatatud, on võimalik aknas „Variables“ („Muutujad“) uurida defineeritud muutujate väärtuseid. PyCharmi koodisilur töötab väga sarnastel põhimõtetel, kuid selle puhul peatatakse programmi töö vaid katkepunktide kohtamisel või siis, kui programmi täitmisel visatakse erind (seda funktsionaalsust Thonny siluril ei ole). Lisaks on silumise ajal programmi peatudes peale muutujate väärtuste vaatamise võimalik kasutada ka Pythoni konsooli, et täpsemini programmi seisu mõista või seda isegi muuta silumise ajal. PyCharmis saab ka katkepunktidele määrata tingimusi, mille täitumisel tuleks programm peatada, mis muudab oluliselt lihtsamaks tsüklite silumise.

Nii Thonnys kui ka PyCharmis on võimalik vaadata Pythoni standardteegi dokumentatsiooni otse arenduskeskkonnas. Thonnys on dokumentatsiooni vaatamiseks tarvis sisse lülitada automaatsete koodilõpetamissoovituste funktsionaalsus ja sel juhul dokumentatsiooni näitamine. Seejärel näidatakse dokumentatsiooni vaid koodilõpetamissoovitusi valides. PyCharmis on dokumentatsiooni vaatamine mugavam, sest piisab kui hõljuda hiirega soovitud funktsiooni peal või vajutada kiirklahve „Ctrl + Q“, kui tekstikursor on funktsiooni nime peal. Samuti on välja toodud link ametlikule Pythoni dokumentatsioonile[9]. Joonisel 5 on esitatud Thonny ja PyCharmi dokumentatsiooni kujutamisest võrdlus.

PyCharm kasutab oluliselt rohkem süsteemi ressursse kui Thonny, mis on peamiselt põhjustatud koodi indekseerimisest PyCharmi käivitamisel. Indekseerimise tulemusena [7] loob PyCharm olemasolevast programmikoodist mälus hoitava andmebaasi, mida kasutatakse arenduskeskkonna peamiste funktsioonide jaoks, nagu intelligentsete koodilõpetamissoovituste pakkumine. Samuti töötab indekseerimise põhjal projektist koodilõikude otsimise funktsionaalsus, mille abil on võimalik lihtsasti üles leida varem kirjutatud koodijuppe.

[9] https://docs.python.org/3/



Joonis 5. Thonny (üleval) ja PyCharmi (all) dokumentatsiooni kujutamine

PyCharm annab kasutajale oluliselt rohkem võimalusi arenduskeskkonna seadete muutmiseks, et rakendust personaliseerida, kuid ka vaikeseaded toetavad programmeerijat hästi, näiteks on sisse lülitatud automaatsete koodilõpetamissoovituste pakkumine ja dokumentatsiooni kuvamine.

2.4 PyCharmi kasutamine programmeerimise baaskursusel

PyCharmi kasutamine Programmeerimise aine raames annaks tudengitele kogemuse IntelliJ platvormi arenduskeskkonnaga töötamisel, mida kasutatakse laialdaselt nii teistes arvutiteaduse instituudi ainetes, kui ka üldiselt tarkvaraarendusega tegelevates ettevõtetes. PyCharmi populaarsuse tõttu on sellele arendatud mitmeid pistikprogramme, sealhulgas programmeerimise õppimist toetavaid pistikprogramme, nagu Live Coding in Python[10], mis

[10] https://plugins.jetbrains.com/plugin/9742-live-coding-in-python

kujutab kasutajale programmikoodis esinevate muutujate väärtused programmi kirjutamise ajal.

Matias Salinas jt [10] on kirjutanud, et algajatel programmeerijatel esineb raskusi professionaalsete arenduskeskkondadega töötamisel, nagu Code::Blocks[11], mis on mõeldud programmeerimiseks keeltes C, C++ ja Fortran. Ühe probleemkohana on välja toodud arenduskeskkonna kasutajaliidese mõistmine ja sellega töötamine. Samas kirjutavad Jialiang Tan jt [11], et nad suutsid edukalt võtta kasutusele professionaalse arenduskeskkonna Visual Studio Code'[12] programmeerimise baaskursusel Pythonis, tänu arenduskeskkonda algajatele programmeerijatele tutvustavate materjalide loomisele. Töö raames lõin PyCharmi koduleheküIje [4], Bruce M. Van Horn II PyCharmi juhendi [7] ja enda kogemuste põhjal eestikeelse juhendi, mis aitab tudengitel tutvuda arenduskeskkonna olulisemate funktsioonidega.

2.5 Olemasolevad lahendused

Rene Kütt [12] on loonud kasutaja tegevuste logide genereerimise pistikprogrammi PALG (Programming Activity Log Generator)[13], mis on ühilduv erinevate JetBrainsi arenduskeskkondadega, mis baseeruvad IntelliJ platvormil, sealhulgas PyCharmiga. Kuna pistikprogrammi poolt genereeritavad logid on sama ülesehitusega mis Thonny poolt genereeritud kasutaja tegevuste logid, siis ei ole tarvis arendada lisafunktsionaalsust, vaid nii Thonny kui ka PALG poolt genereeritud logifaile saab korraga analüüsida, kasutades rakendust PALA (Programming Activity Log Analyzer)[14].

Lahendus on Tartu ülikooli arvutiteaduse instituudi poolt pakutav teenus, mis põhineb vabavaralisel tarkvaraplatvormil easy [13]. Lahenduse keskkonda kasutakse Programmeerimise aines tudengite poolt programmeerimisülesannete lahenduste esitamiseks. Õppejõud saavad läbi keskkonna hallata erinevaid kursuseid, kursuse ülesandeid ja osalejaid. Lahenduse keskkond võimaldab läbi TSL-i [14] (Test Specific Language) luua ülesannetele automaatkontrolle, mis vähendab õppejõudude koormust

[11] https://www.codeblocks.org/

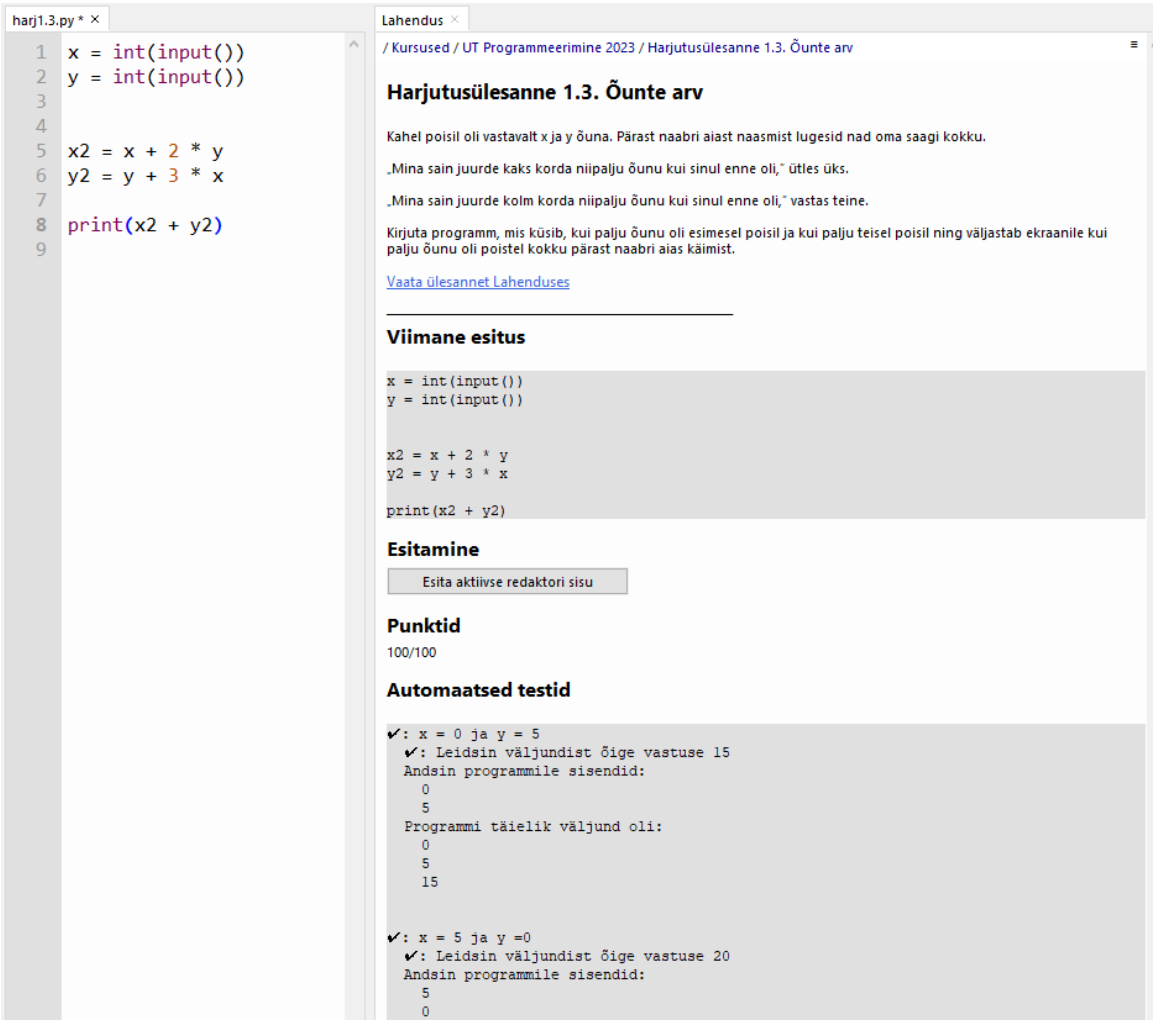
[12] https://code.visualstudio.com/

[13] https://plugins.jetbrains.com/plugin/21567-palg-programming-activity-log-generator

[14] https://kodu.ut.ee/~kuttrene/PALA/

kodutööde hindamisel ning võimaldab tudengitele anda kohest tagasisidet nende programmi töökindluse kohta.

Thonny on integreeritud Lahendusega läbi pistikprogrammi[15]. See võimaldab Lahenduse kursuse õpilasel Lahenduse keskkonda sisse logida läbi veebilehitseja ning seejärel teha pistikprogrammi abil keskkonnas API päringuid. Selle tulemusel kuvatakse arenduskeskkonnas ülesannete kirjeldused ja tagasiside kasutajale. Kasutajal on ka võimalus esitada aktiivse tekstiredaktori sisu ülesande lahenduseks, et automaatkontrolli olemasolul oleks võimalik saada kohest tagasisidet. Joonisel 6 on esitatud Thonny Lahenduse pistikprogrammi ülesande vaade.



Joonis 6. Thonny Lahenduse pistikprogrammi ülesande vaade

Käesoleva töö raames arendasin vastava funktsionaalsuse ka PyCharmile.

[15] https://pypi.org/project/thonny-lahendus/

3. IntelliJ platvormi Lahenduse pistikprogramm

Selles peatükis kirjeldan loodud vabavaralisele IntelliJ platvormile mõeldud Lahenduse pistikprogrammi arendamist, keskendudes süsteemi nõuetele, ülesehitusele, realiseerimise protsessile ja testimisele.

3.1 Nõuete kirjeldus

Enne arendusprotsessi algust ja arendusprotsessi jooksul panin paika järgmised pistikprogrammi funktsionaalsed ja mittefunktsionaalsed nõuded. Nõuded on kirjutatud lähtudes rakenduse ainsast kasutajarollist, kelleks on suvalisele Lahenduse kursusele registreeritud õpilane.

Funktsionaalsed nõuded.

1. Kasutaja autentimine peab toimuma läbi veebilehitseja.
2. Autenditud kasutajale kuvatakse kursuste loend, kuhu ta on registreeritud.
3. Autenditud kasutajale kuvatakse kursuse valimisel vastava kursuse ülesannete loend.
4. Autenditud kasutajale kuvatakse ülesande valimisel vastava ülesande kirjeldus ja kasutaja viimase lahenduse esituse tagasiside vastavale ülesandele, kui see leidub.
5. Autenditud kasutaja peab saama ülesande valimisel esitada vastava ülesande lahenduseks aktiivse tekstiredaktori sisu, misjärel kuvatakse kasutajale ta esituse tagasiside.
6. Autenditud kasutaja peab saama ülesande valimisel avada vastava ülesande veebilehitsejas.
7. Autenditud kasutaja peab saama värskendada talle kuvatavat sisu, sest kasutaja võib teha tegevusi läbi pistikprogrammi ja veebilehitseja paralleelselt ning seejärel on tarvis viia pistikprogrammi kasutajaliides uute muudatustega kurssi.
8. Autenditud kasutaja peab saama Lahenduse keskkonnast välja logida.

Mittefunktsionaalsed nõuded.

1. Pistikprogramm peab olema ühilduv kõikide IntelliJ platvormi viimase versiooni arenduskeskkondadega, milleks töö kirjutamise ajal on 2023.3.
2. Pistikprogrammi kasutajaliides peab olema lihtsasti mõistetav.
3. Pistikprogramm peab järgima IntelliJ platvormi SDK [15] parimaid tavasid, näiteks ressursside vabastamise ja graafilise kasutajaliidese koha pealt.

Lähtusin nõuetest kogu arendusprotsessi jooksul ja testisin seejuures pidevalt pistikprogrammi tööd, et see vastaks nõuetele.

3.2 Süsteemi ülesehitus

Pistikprogramm on realiseeritud programmeerimiskeeles Kotlin[16], kuna mul on kogemust tarkvara arendamisega Javas ning Kotlini Javaga ühilduvuse tõttu on pistikprogrammi arendamine mõlemas keeles sarnane, kuid Kotlin pakub mitmeid mugavaid võimalusi, nagu näiteks koodis tagasikutsete kirjutamine lambdafunktsioonidena, mida IntelliJ platvormi programmikoodis tihti kasutatakse [15].

Pistikprogrammis vastutavad autentimise ja Lahenduse API-ga[17] suhtluse eest vastavalt autentimise ja Lahenduse API kergteenused (ingl *light service*).

Lahendus [13] kasutab kasutajate autentimiseks tarkvara Keycloak[18], mis võimaldab läbi autentimistõendite kasutada ainulogimise tehnikat (ingl *single-sign on, SSO*). Pistikprogrammi autentimisteenus loob sisselogimist alustades klientarvutis lokaalse veebiserveri, mis pakub veebilehte relatiivsel aadressil "/login". Vastav veebileht avatakse kasutaja veebilehitsejaga ning seal saab ta sooritada sisselogimise, misjärel Javascripti skripti tulemusena edastatakse kasutaja autentimistõendid lokaalsele veebiserverile. Autentimisteenus vastutab ka tõendite värskendamise eest. Kui autentimistõendid on süsteemis olemas, siis on võimalik Lahenduse API kaudu HTTP-päringuid saata, et pärida informatsiooni kursuste ja ülesannete kohta või esitada ülesande lahendus.

Pistikprogrammi kasutajaliides põhineb Java graafilise kasutajaliidese teegil Swing[19] ja sellest loodud JetBrainsi edasiarendusel, mis tagab pistikprogrammi kasutajaliidese stiili ühtivuse IDE omaga. Pistikprogrammi kasutajaliides on realiseeritud tööriistaaknana (ingl *tool window*), mis on seotud kasutaja poolt avatud projektiga. IntelliJ platvormi arenduskeskkondade kasutajaliidese väljanägemine on hiljuti oluliselt muutunud (nn *New UI*) ja 2024. aasta lõpus plaanitakse lõpetada vana kasutajaliidese (nn *Classic UI*) tugi [7]. Pistikprogramm toetab mõlemat kasutajaliidese versiooni. Samuti on toetatud nii hele kui ka tume kasutajaliidese stiil.

---

[16] https://kotlinlang.org/

[17] https://github.com/kspar/easy/blob/master/doc/core/api.yaml.outdated

[18] https://www.keycloak.org/

[19] https://en.wikipedia.org/wiki/Swing_(Java)

IntelliJ platvormil jooksutatakse üldiselt koodi kaht tüüpi lõimedel, milleks on sündmuste edastamise lõim (ingl *Event Dispatch Thread, EDT*) ja taustaprotsesside lõimed [15]. Sündmuste edastamise lõimel toimub ka kasutajaliidese uuendamine, mistõttu peavad sellel lõimel tehtavad tegevused toimuma kiiresti, et vältida kasutajaliidese hangumist [15]. Seetõttu sooritatakse blokeeruvad operatsioonid, nagu HTTP-päringute tegemine, taustaprotsesside lõimedel. Taustaprotsessid edastavad andmeid teistele lõimedele läbi sõnumite siini (ingl *message bus*), mis on osa IntelliJ platvormi edasta-kuula (ingl *publish-subscribe*) mudelil põhinevast sõnumite edastamise infrastruktuurist [15]. IntelliJ platvormi arenduskeskkonnad lubavad kasutajal avada mitu erinevat projekti korraga erinevates akendes ja seega tekib ka mitu akent pistikprogrammi kasutajaliidesest. Sõnumite siini kasutamine võimaldab rakendada autentimisloogikat tervele arenduskeskkonnale, kuid suhtlust Lahenduse API-ga projektipõhiselt. Näiteks, kui kasutaja on avanud mitu projekti, peab ta sisselogimise sooritama vaid ühes pistikprogrammi aknas, misjärel uuendatakse ka teised aknad automaatselt, aga muidu võivad kasutajal erinevates projektides olla avatud erinevad Lahenduse ülesanded, kasvõi erinevatelt kursustelt.

Mälulekete vältimiseks on kuulavad ühendused sõnumite siiniga seotud liidest Disposable implementeerivate isenditega, et ühendused vabastataks Disposable isendi eluea lõpus. Mitmed pistikprogrammi kasutajaliidese komponendid implementeerivad liidest DumbAware, et märkida, et vastav komponent ei sõltu projekti indekseerimisest ning selle võib aktiivseks teha enne indekseerimise lõppu.

3.3 Realiseerimise protsess

Pistikprogrammi programmikoodi kirjutasin arenduskeskkonnas IntelliJ Idea, kasutades pistikprogrammide loomiseks mõeldud pistikprogrammi Plugin DevKit[20]. Pistikprogrammi jooksutamiseks ja testimiseks arenduskeskkonna peal kasutasin ehitustööriista Gradle[21] koos selle laiendusega Gradle IntelliJ Plugin[22], mis toetab pistikprogrammi arendamist Gradle'iga.

[20] https://plugins.jetbrains.com/plugin/22851-plugin-devkit

[21] https://gradle.org/

[22] https://github.com/JetBrains/intellij-platform-gradle-plugin

Versioonihalduse jaoks kasutasin Git'i versioonihaldussüsteemi ja lõin avaliku lähtekoodihoidla platvormil Github[23], mis on saadaval veebiaadressil https://github.com/alexisalliksaar/lahendus-intellij-plugin.

Arendusprotsessi lõppfaasis avaldasin pistikprogrammi IntelliJ platvormi arenduskeskkondade pistikprogrammide hoidlas JetBrains Marketplace[24], mille tulemusena on võimalik pistikprogramm alla laadida otse arenduskeskkonnast.

3.4 Testimine

Testisin pistikprogrammi vastavust funktsionaalsetele nõuetele ja selle ühilduvust IntelliJ platvormi arenduskeskkondadega PyCharm Community Edition (versioon 2023.3.5) ja PyCharm Professional Edition (versioon 2023.3.3). Rakendus töötas sujuvalt mõlema arenduskeskkonna peal. Lisaks testisin kasutajaliidese toimimist ja väljanägemist IntelliJ platvormi vana ja uue kasutajaliidese versiooni peal ning kasutajaliidese heleda ja tumeda stiili peal. Testimine toimus Windows 11 virtuaalmasina peal.

Pistikprogrammi kasutatavuse testimiseks koostasin tagasisideküsitluse, mille saatsin Programmeerimise ainega seotud õppejõududele, kellest viis katsetasid programmi tööd ja vastasid küsitlusele, mis on välja toodud Lisas III. Testgrupist olid kolm vastanut varem kasutanud JetBrainsi arenduskeskkondasid ja kaks ei olnud. Kaks vastanutest kasutasid PyCharmi Professional Editionit ja kolm Community Editionit. Testgrupi poolt kasutatud PyCharmi versioonid olid 2023.2.5, 2023.3.5 ja 2024.1. Kaks vastanutest kasutasid arenduskeskkonna *New UI*-d ja kolm *Classic UI*-d.

Küsitlusest ilmnes, et pistikprogrammi kasutajaliidese väljanägemine sobib väga hästi kokku ülejäänud arenduskeskkonna omaga, mida hinnati skaalal ühest viieni keskmiselt hindega 4.6. Pistikprogrammi kasutajaliidese kasutajasõbralikkust ning kasutajate üldist kogemust pistikprogrammi kasutamisel hinnati keskmisest kõrgemaks, vastavalt keskmistele hinnetele 3.6 ja 3.8 viie punkti skaalal.

Kaks vastajat puutus kokku programmivigadega ning PyCharmi 2023.2 versiooni peal puututi kokku kasutajaliidese stabiilsusega seotud probleemidega. Tagasiside põhjal arendasin programmi edasi, et kõrvaldada esiletoodud probleeme. Samuti täiustasin

[23] https://github.com/

[24] https://plugins.jetbrains.com/

pistikprogrammi vigade logimise süsteemi, et tulevikus oleks lihtsam vigade allikaid tuvastada. Tagasisidest ilmnes ka vajadus eestikeelse pistikprogrammi kasutajaliidese järele, misjärel lisasin vastava toe vaikimisi sätitud inglisekeelsele kasutajaliidesele lisaks.

4. Tulemused

Töö peamiseks tulemuseks on IntelliJ platvormi arenduskeskkondasid Lahenduse keskkonnaga ühildav pistikprogramm. Pistikprogrammi paigaldamisjuhised ning kasutusjuhendi leiab Lisas I. Kuna pistikprogramm on ühilduv kõigi IntelliJ platvormi arenduskeskkondadega (IntelliJ IDEA, PyCharm, CLion, DataSpell, jt) ja ei sõltu programmeerimiseks kasutatavast programmeerimiskeelest, siis on seda võimalik kasutada ka muude Lahenduse keskkonda kasutavate õppeainete raames. Pistikprogramm on ühilduv arenduskeskkondade versioonidega 2023.2 – 2024.1. Pistikprogrammi koodibaasi suuruseks on ligikaudu 2300 Kotlini koodirida.

Võimalikud edasiarendused pistikprogrammile:

- Luua spetsiaalne HTML-teksti renderdav komponent, mis saaks paremini hakkama ülesannete lähteteksti kujutamisega, mis on kirjutatud keeles HTML. Seni kasutusel oleva IntelliJ platvormi poolt pakutava SimpleHtmlPane komponendi ning veebilehitsejate kujutustes on veel mõningased erisused. Alternatiivina on võimalik kasutada HTML-teksti kuvamiseks arenduskeskkonda sisseehitatud JCEF veebilehitsejat.
- Rakendada ülesande lahenduse tagasisides esitatud ülesande lahenduse programmikoodile süntaksi esiletõstmist.

Töö raames parandati ka programmiviga IntelliJ platvormi kasutaja tegevuste logimise pistikprogrammis PALG, kus logifailid genereeriti kasutades süsteemi vaikekodeeringut, mitte kodeeringut UTF-8, mistõttu ei suutnud logifailide analüsaator PALA tuvastada täpitähti.

Töö raames loodi ka Programmeerimise aine tudengitele mõeldud eestikeelne PyCharmi juhend. Mõningate tekstimuudatustega on juhendit võimalik rakendada üldiselt algajatele programmeerijatele PyCharmi tutvustamiseks. Juhend käsitleb teemasid:

1) PyCharmi allalaadimine;
2) projekti loomine ja organiseerimine;
3) pistikprogrammide paigaldamine;
4) koodi jooksutamine;
5) koodi silumine;
6) vead koodis / süntaksimärgendus;

7) koodi ümberkorraldamine;
8) Pythoni konsooli kasutamine;
9) üldised nipid arenduskeskkonna efektiivseks kasutuseks;
10) Pythoni pakettide installimine;
11) versioonihaldussüsteemid;

Loodud juhend on kättesaadav aadressil https://kodu.ut.ee/~alexisa/PyCharmi_juhend.pdf, nagu on välja toodud Lisas II.

Lahenduse pistikprogrammi ja PyCharmi juhendi loomise tulemusena on PyCharmil olemas sama hea tugi Programmeerimise aine jaoks, kui on Thonnyl.

5. Arutelu

Mitmed teadusallikad, nagu näiteks teadustööd Matias Salinase jt [10], Michael Köllingi jt [16] ja Colin Depradine'i ja Glenda Gay' [17] poolt, käsitlevad integreeritud arenduskeskkondade kasutamist objektorienteeritud programmeerimise õpetamiseks peamiselt keeltes Java või C++. Aharon Yadin [18] on kirjutanud, et programmeerimise baaskursusel Pythoni kasutamine Java asemel oli üks kolmest põhjusest, mis aitas vähendada kursuse läbikukkunute arvu 77.4% võrra. Sellegipoolest pole minu teadmiste kohaselt tehtud palju uurimistöid, mis käsitleksid Pythoni jaoks mõeldud integreeritud arenduskeskkondasid programmeerimise õppimiseks. Samuti pole minu teadmiste kohaselt palju uuritud JetBrainsi arenduskeskkondade kasutust programmeerimise õpetamiseks, kuigi need on viimase kümne aasta jooksul palju populaarsust kogunud.

Programmeerimise ainele PyCharmi toe loomine loob võimaluse kasutada PyCharmi alternatiivselt Thonnyle. Igal aastal leidub Programmeerimise aine tudengeid, kes väljendavad soovi kasutada aines PyCharmi, seega on selle võimaluse järele vajadus, mida töö rahuldab. PyCharmi kasutamine võimaldab programmeerimise õppijatel saada kasu tänapäevase arenduskeskkonna poolt pakutavatest võimalustest, et efektiivsemalt töötada.

Samuti avab käesolev töö võimaluse edasi uurida professionaalse arenduskeskkonna mõju programmeerimise õppimisele. Thonny ja PyCharmi kasutamine Programmeerimise aines paralleelselt võimaldab uurida tudengite kogemusi ja tulemusi kummagi arenduskeskkonnaga töötamisel, et paremini mõista arenduskeskkondade kasutamise eeliseid või puuduseid programmeerimise õppimisel. Saadud uurimistulemuste põhjal on võimalik Thonnyt ja PyCharmi edasi arendada, et muuta arenduskeskkonnad algajatele programmeerijatele veelgi paremaks.

6. Kokkuvõte

Töö eesmärk oli luua Programmeerimise ainele PyCharmi tugi. Tänu Lahenduse keskkonnaga ühildava pistikprogrammi loomise on Programmeerimise aine tudengitel võimalus Lahenduse keskkonnaga suhelda läbi PyCharmi, sarnaselt nagu see on võimalik läbi Thonny. Töös kirjeldati PyCharmi funktsionaalsuste eeliseid Thonny ees ning algajatele programmeerijatele koostatud eestikeelne PyCharmi juhend aitab neil vastavatest funktsionaalsustest kasu lõigata ja arenduskeskkonda tundma õppida. Kuna vastav tugi PyCharmile varem puudus, siis on töö uudne.

Töö avab võimaluse edasi uurida professionaalsete arenduskeskkondade kasutamise mõju programmeerimise õpetamisel, et paremini toetada algajate programmeerijate arengut.

7. Viidatud kirjandus

[1] Annamaa A. Thonny, a Python IDE for Learning Programming. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15)*. 2015. lk 343. https://dl.acm.org/doi/pdf/10.1145/2729094.2754849 (01.12.2023)

[2] Thonny koduleheküIg. https://thonny.org (01.12.2023)

[3] Raspberry Pi OS Vikipeedia leheküIg. https://en.wikipedia.org/wiki/Raspberry_Pi_OS (07.04.2024)

[4] PyCharmi koduleheküIg. https://www.jetbrains.com/pycharm (02.12.2023)

[5] Stack Overflow 2023 Developer Survey. https://survey.stackoverflow.co/2023/#section-admired-and-desired-integrated-development-environment (02.12.2023)

[6] Python Developer Survey 2021. https://lp.jetbrains.com/python-developers-survey-2021/#DevelopmentTools (04.12.2023)

[7] Van Horn II B.M., Nguyen Q. Hands-On Application Development with PyCharm: Build applications like a pro with the ultimate python development tool, Edition 2. Birmingham: Packt Publishing Ltd. 2023.

[8] Karvelas I. Investigating Novice Programmers' Interaction with Programming Environments. *Innovation and Technology in Computer Science Education (ITiCSE '19)*. 2019. https://dl.acm.org/doi/pdf/10.1145/3304221.3325596 (28.04.2024)

[9] Dong T. , Khandwala K. The Impact of "Cosmetic" Changes on the Usability of Error Messages. *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA '19)*. 2019. lk 1-6. https://dl.acm.org/doi/10.1145/3290607.3312978 (01.05.2024)

[10] Salinas M., Leger P., Fukuda H., Cardozo N., Duarte V., Figueroa I. Evaluations of Integrated Programming Environment for First-Year Students in Computer Engineering. *Journal of Universal Computer Science, vol. 29, no. 1*. 2023. lk 73-97. http://dx.doi.org/10.3897/jucs.81329 (28.04.2024)

[11] Tan J., Chen Y., Jiao S. Visual Studio Code in Introductory Computer Science Course: An Experience Report. *Proceedings of ACM Conference (Conference'17)*. 2023. https://doi.org/10.48550/arXiv.2303.10174 (04.05.2024)

[12] Kütt R. Plagiarism Detection Tool for Programming Activity Logs. TÜ arvutiteaduse instituudi magistritöö. 2023. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=77249 (04.12.2023)

[13] Papli K. easy lähtekoodi repositoorium. https://github.com/kspar/easy/ (07.04.2024)

[14] Muuli E., Palm R., Lepp M. Simplifying the creation and maintenance of automated assessments of programming tasks via Test Specific Language. *Proceedings of the 2022 6th International Conference on Education and E-Learning (ICEEL '22)*. 2023. lk 14-20. https://dl.acm.org/doi/abs/10.1145/3578837.3578840 (28.04.2024)

[15] IntelliJ platvormi SDK dokumentatsioon. https://plugins.jetbrains.com/docs/intellij/welcome.html (13.04.2024)

[16] Kölling M., Quig B., Patterson A., Rosenberg J. The BlueJ system and its pedagogy. *Journal of Computer Science Education, 13*. 2003. http://dx.doi.org/10.1076/csed.13.4.249.17496 (05.05.2024)

[17] Depradine C., Gay G. Active participation of integrated development environments in the teaching of object-oriented programming. *Computers & Education, Volume 43, Issue 3*. 2003. lk 291-298. https://doi.org/10.1016/j.compedu.2003.10.009 (05.05.2024)

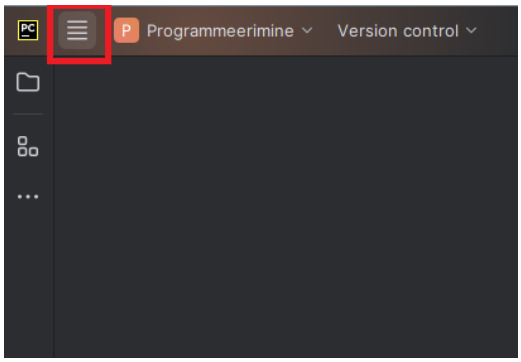
[18] Yadin A. Reducing the dropout rate in an introductory programming course. *ACM Inroads 2, 4*. 2011. lk 71-76. https://doi.org/10.1145/2038876.2038894 (05.05.2024)

Lisad

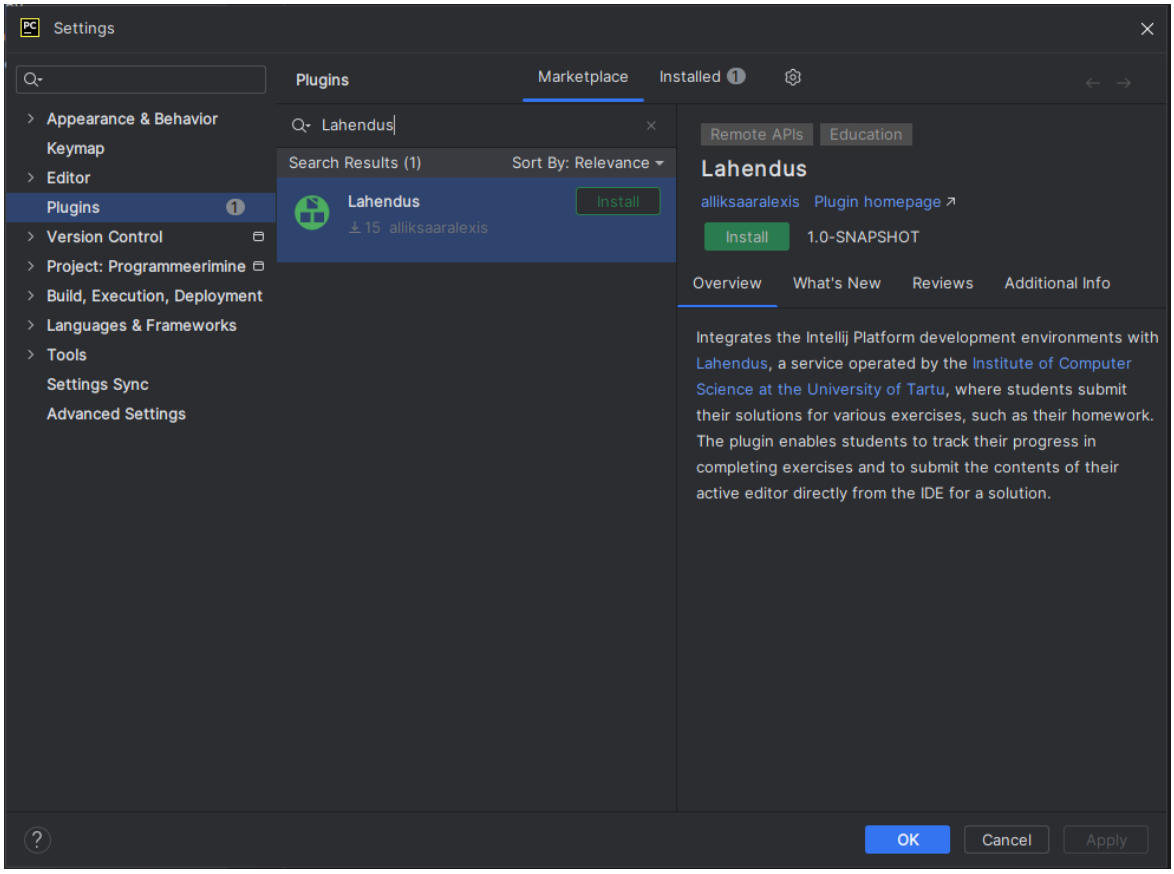
I. Pistikprogrammi kasutusjuhend

Pistikprogrammi paigaldusjuhised ning pistikprogrammi tööriistariba nuppude kirjeldused.

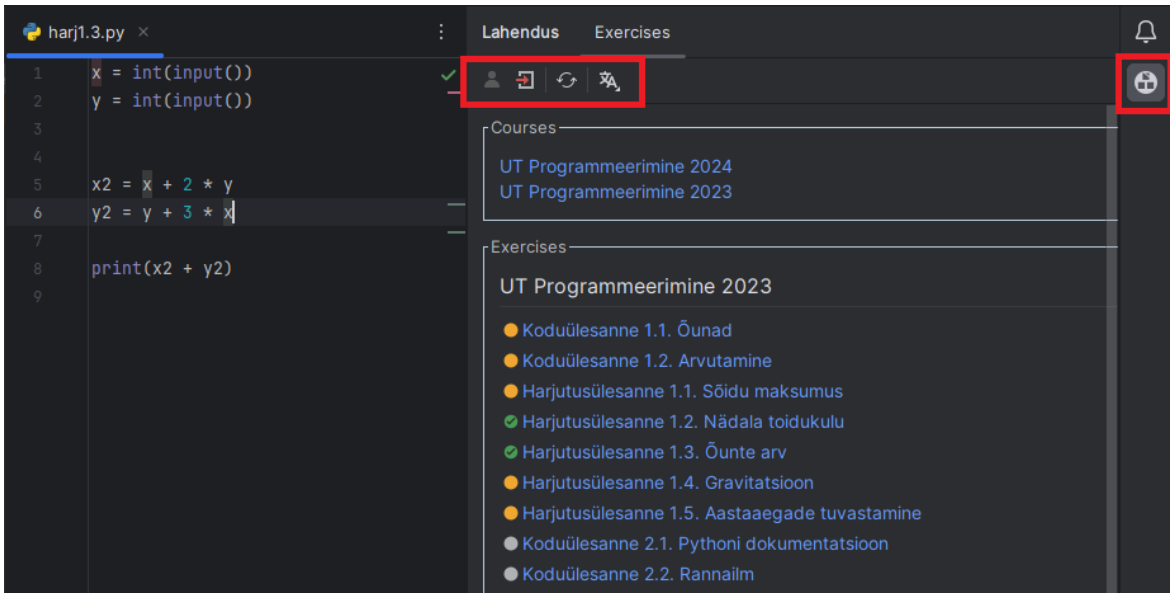
Pistikprogrammi paigaldamiseks valige PyCharmi menüü alt Settings → Plugins:



Otsige JetBrains Marketplace-ist pistikprogrammi Lahendus ja vajutage Install:



Avage Lahenduse pistikprogrammi aken, vajutades vaikimisi paremal üleval tööriistaribal asuvale Lahenduse logole. Välja on toodud pistikprogrammi ülesannete akna vaade, kui on sooritatud sisse logimine ja on valitud aktiivne kursus:

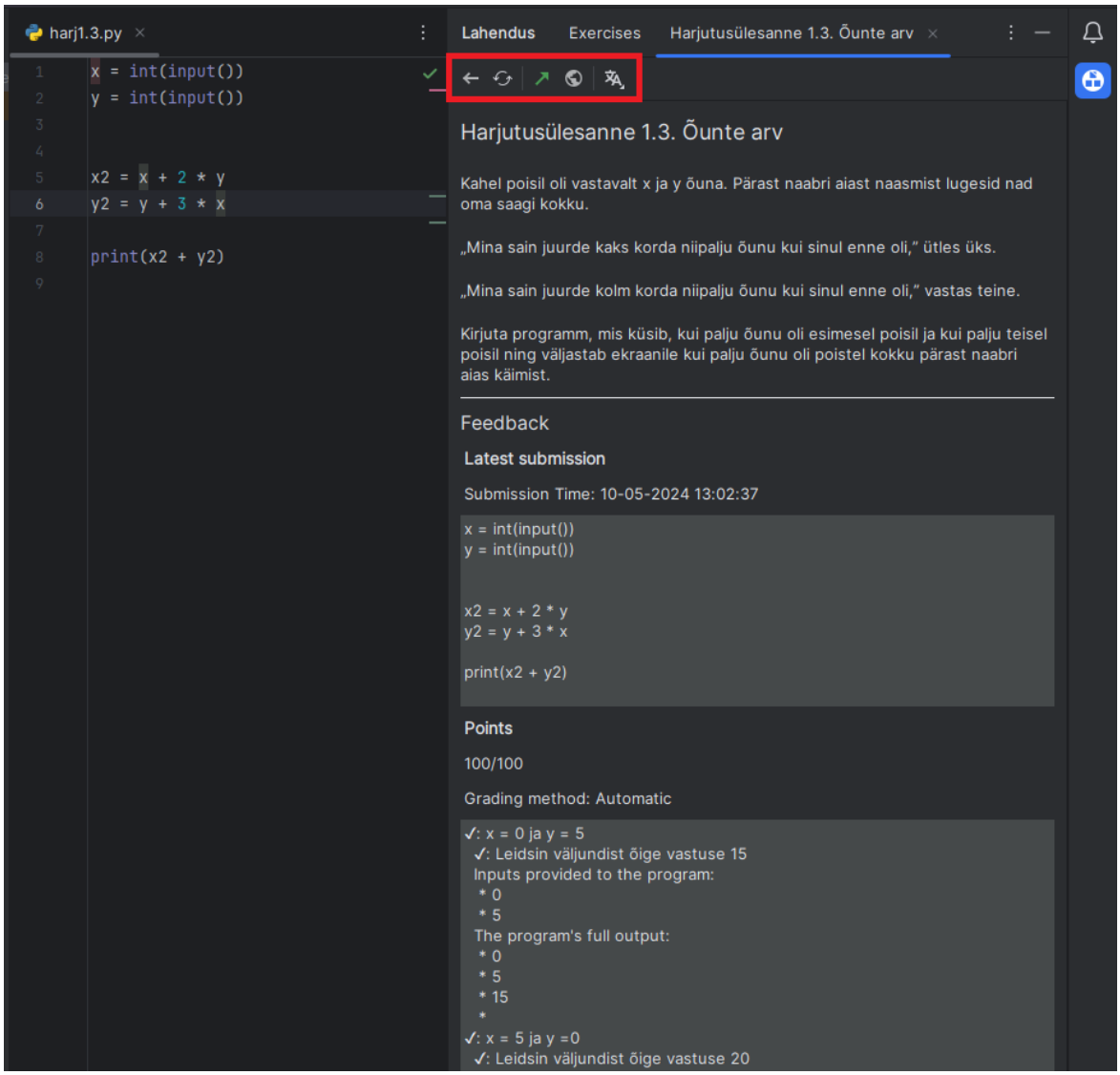


Ülesannete akna tööriistariba nuppude kirjeldused on välja toodud nende tööriistaribal esinemise järjekorras:

1) logi Lahendusse sisse;
2) logi Lahendusest välja;
3) värskenda kursuste ja ülesannete loendeid;
4) vaheta pistikprogrammi kasutajaliidese keelt (valikus inglise ja eesti keel);

Kõigi tööriistaribade nuppude tähenduste kohta ilmuvad ka vihjed nupu peal hiirega hõljudes.

Valides ülesannete loendist ülesande avaneb vastava ülesande aken:



Valitud ülesande akna tööriistariba nuppude kirjeldused on välja toodud nende tööriistaribal esinemise järjekorras:

1) navigeeri tagasi ülesannete aknale;
2) värskenda valitud ülesande akent;
3) saada aktiivse tekstiredaktori sisu Lahenduse keskkonda valitud ülesande lahenduseks;
4) ava valitud ülesanne veebilehitsejaga Lahenduse keskkonnas;
5) vaheta pistikprogrammi kasutajaliidese keelt (valikus inglise ja eesti keel);

II. PyCharmi juhend

Loodud PyCharmi juhendmaterjal on kättesaadav veebiaadressil https://kodu.ut.ee/~alexisa/PyCharmi_juhend.pdf.

III. Kasutatavuse tagasisideküsitlus

Küsimused:

1) Kas olete varem kasutanud JetBrainsi IDE-sid?
2) Millist PyCharmi versiooni te kasutasite?
3) Mis on teie PyCharmi versiooni number?
4) Kas te kasutasite IntelliJ platvormi "New UI"-d?
5) Kui hästi läheb pistikprogrammi kasutajaliidese väljanägemine kokku üldise IDE stiiliga? (skaalal ühest viieni)
6) Kuidas hindaksite pistikprogrammi kasutajaliidese kasutajasõbralikkust? (skaalal ühest viieni)
7) Mida annaks parandada, et teha pistikprogrammi kasutajaliides kasutajasõbralikumaks?
8) Kas teie arvates on pistikprogrammil puudu mõni oluline funktsioon? Kui jah, siis mida saaks parandada?
9) Kuidas hindaksite oma üldist kogemust pistikprogrammi kasutamisel? (skaalal ühest viieni)
10) Kas oli mõni konkreetne pistikprogrammi aspekt, mis teile positiivselt või negatiivselt silma jäi?
11) Kas te kohtasite pistikprogrammi kasutamisel vigu programmis?
    a) Palun kirjeldage tekkinud vigu ja võimalikke taasesitamise viise.
    b) Võimalusel lisage oma IDE logifail.
12) Kas teil on pistikprogrammi kasutamise kogemuse kohta veel märkusi või tagasisidet, mida sooviksite jagada?

IV. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Alexis Alliksaar,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose Programmeerimise ainele PyCharmi toe loomine, mille juhendaja on Reimo Palm, 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 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

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

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

Alexis Alliksaar

15.05.2024