TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Lauri Leiten

Thonny arenduskeskkonna pistikprogramm skriptide käivitamiseks üle SSH

Bakalaureusetöö (9 EAP)

Juhendaja: Aivar Annamaa

Tartu 2019

Thonny arenduskeskkonna pistikprogramm skriptide käivitamiseks üle SSH

Lühikokkuvõte:

Bakalaureusetöös antakse ülevaade integreeritud arenduskeskkonnast Thonny ja turvakestast. Käesoleva töö tulemusena luuakse Thonny jaoks pistikprogramm. See pistikprogramm vahendab Thonny ja kaugarvuti vahelist suhtlust läbi SSH.

Võtmesõnad:

programmeerimine, Python, Thonny, SSH, SFTP

CERCS: P175 Informaatika, süsteemiteooria

Thonny Development Environment Plugin for Executing Scripts via SSH

Abstract:

This bachelor thesis gives an overview of the integrated development environment Thonny and secure shell. As a result of this thesis a plugin for Thonny will be built. The plugin will be able to communicate with remote machines using SSH.

Keywords:

programming, Python, Thonny, SSH, SFTP

CERCS: P175 Informatics, systems theory

Sisukord

1. Sissejuhatus

Programmeerimiskeel Python on hea keel, milles programmeerimisega algust teha – see keel disainiti võimalikult algajasõbralikuks ja selle tutvustamiseks on loodud hulganisti õppematerjale [1]. Pythoni õppimise lihtsustamiseks on hea kasutada abivahendeid, näiteks Thonny, Lego Mindstorms EV3 ja Raspberry Pi. Thonny [2] on algajatele mõeldud integreeritud arendus-keskkond, Lego Mindstorms EV3 [3] on robootikakomplekt ja Raspberry Pi [4] on monoplaatarvuti, millel on universaalsed ühenduspesad [5]. Nende ühenduspesade kaudu on võimalik tarkvara abil kontrollida LED tulede või muude seadmete tööd [6].

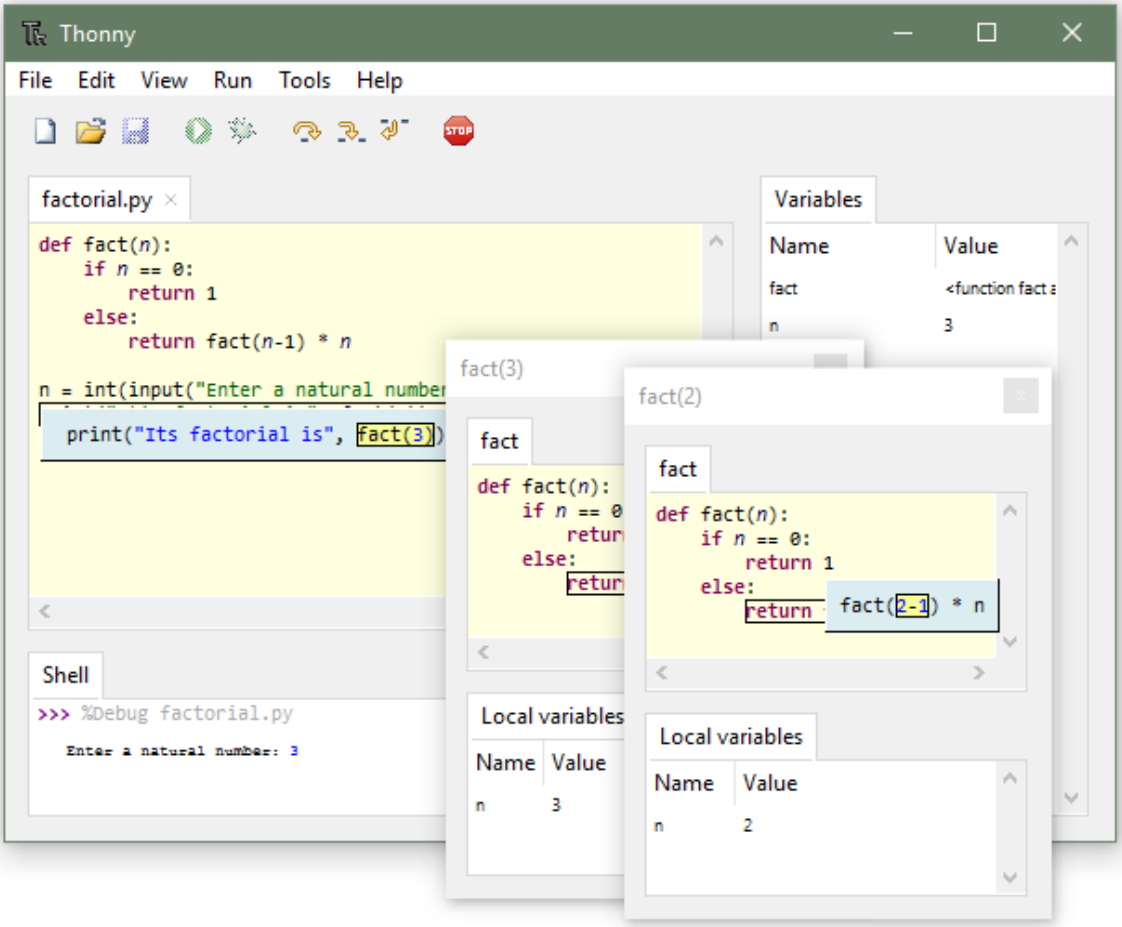
EV3's on Pythoni skriptide käivitamine kasutaja enda arvutist raskendatud. Selle mure lahendamiseks on võimalik masinale installeerida operatsioonisüsteem ev3dev, mis toetab masina kontrollimist turvakesta abil [7]. Raspberry Pi kasutatavasse operatsioonisüsteemi on juba Thonny ja turvakest sisse ehitatud [8-9], aga Thonnyt on võimalik kasutada ainult läbi graafilise kasutajaliidese [4]. See tähendab seda, et masinale on vaja külge ühendada välisseadmed (hiir, klaviatuur, kuvar). Kui igapäevaselt kasutab inimene teist arvutit, siis on tal kindlasti mugavam programmeerida enda arvutis ja käivitada kaugarvutis programme turvakesta abil. Hetkel puudub Thonnys võimalus programme läbi turvakesta käivitada.

Bakalaureusetöö eesmärk on luua Thonny jaoks pistikprogramm, mis oleks võimeline kaug-arvutiga turvakesta abil suhtlema.

Peatükis 2 tutvustatakse integreeritud programmeerimiskeskkonda Thonny. Peatükis 3 antakse ülevaade turvakestast, seda kasutavast failiedastusprotokollist ning turvakesta implementeerivatest teekidest keeles Python. Peatükk 4 tutvustab valminud programmi.

2. Thonny

Thonny (vt joonis 1) on algajatele mõeldud avatud lähtekoodiga integreeritud programmeerimiskeskkond, mis loodi Tartu Ülikooli arvutiteaduste instituudi juures [2]. Thonny on mõeldud arendamiseks keeles Python [2] ning seda levitatakse MIT litsentsi all [10].



Joonis 1. Ekraanitõmmis Thonny kasutajaliidesest [2].

2.1 Ülevaade

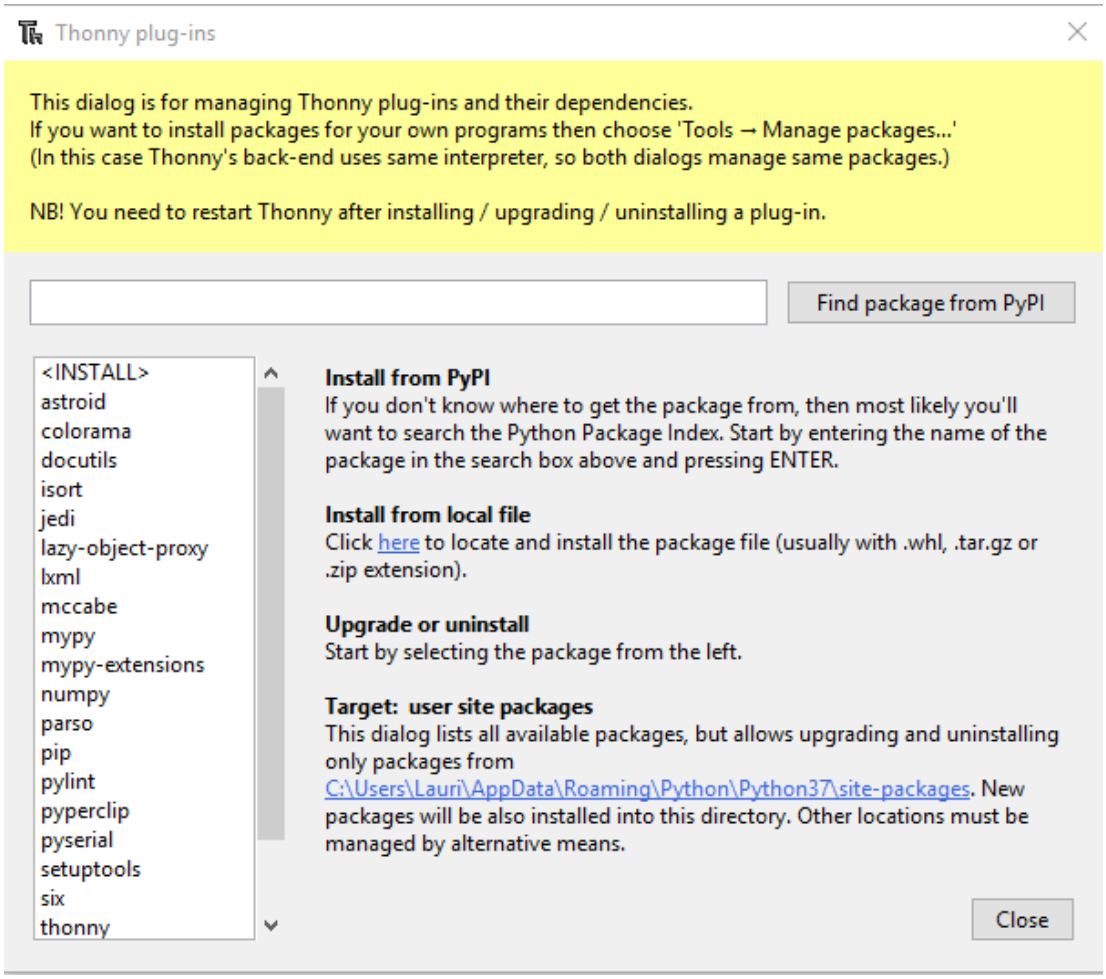
Thonny on kirjutatud keeles Python [10] ja kasutab moodulit Tkinter graafilise kasutajaliidese loomiseks [11].

Thonny kodulehe [2] andmetel on Thonny mõeldud selleks, et teha programmeerimiskeele Python õppimine algajatele programmeerijatele võimalikult lihtsaks. Selle tõttu on kasutajaliides võimalikult minimaalne – sealt on eemaldatud kõik kasutajat segada võivad tööriistad. Samuti on Thonny kasutajate töö hõlbustamiseks olemas abivahendid programmide töö vaatlemiseks – programmi

töötamise ajal muutujate kuvamine, lihtne silur ja funktsiooni väljakutsete eraldi kuvamine. Lihtsaks on tehtud ka Thonny installeerimine – installeerida on vaja ainult Thonny, Python on programmiga kaasas.

2.2 Pistikprogrammid

Thonnys on võimalik installeerida kolmandate isikute poolt loodud Pythoni teeke. Jooniselt 2 on näha kahte viisi nende teekide lisamiseks. Esimesel juhul laetakse tarkvara alla Pythoni tarkvarahoidlast PyPI ja see nõuab internetiühenduse olemasolu. Teine variant on installeerida teek kasutaja arvutist.



Joonis 2. Ekraanitõmmis pistikprogrammide lisamise kasutajaliidesest.

Pistikprogrammidega on võimalik muuta nii Thonny välimust, lisada programmi funktsionaalsusi kui ka muuta Thonny põhifunktsionaalsusi [11][12].

3. Turvakest

Käesolev peatükk annab ülevaate turvakestast, seda kasutavast turvalisest failiedastusprotokollist ja neid protokolle implementeerivatest Pythoni teekidest.

3.1 Ülevaade

Turvakest (inglise k. *Secure Shell*, lühend *SSH)* [13-16] on võrguprotokoll, mille eesmärgiks on luua võimalus turvaliseks autentimiseks ja ühenduse loomiseks turvamata võrgus. Turvakestast on kaks suuremat väljalaset – versioon 1 ja versioon 2 [16]. Tänapäeval on kasutusel ainult versioon 2 ja seetõttu tuleb juttu ainult sellest väljalaskest.

Ühenduse usaldusväärsuse ja turvalisuse tagamiseks kasutatakse krüptograafilisi algoritme [13]. Protokoll on loodud võttes arvesse klient-server arhitektuuri põhimõtteid [15]. Protokolli kasutatakse enamasti turvaliseks kauglogimiseks ning see protokoll loodi osaliselt selletõttu, et asendada sarnase ülesandega ebaturvalise protokolle näiteks *telnet* [13][14].

Turvakest koosneb kolmest peamisest kihist [15]:

- Transpordikiht [15,17]

  Seab ülesse krüpteeritud ühenduse kliendi ja serveri vahel. Kui transpordikihi ühendus luuakse üle TCP/IP, kuulab server vaikimisi ühendusi pordil 22. Selles kihis toimub ainult hostipõhine autentimine, kasutajapõhise autentimise jaoks on eraldi autentimiskiht.

- Autentimiskiht [15,18]

  Tegeleb kasutajapõhise autentimisega. Serveri poolt toetatud autentimismeetodeid saab klient serverilt küsida – enimkasutatavad viisid on võtmepõhine ja paroolipõhine autentimine. See kiht ei tegele ise krüpteeritud ühenduse loomisega, vaid eeldatakse, et transpordikihis on varasemalt turvaline ühendus üles seatud.

- Ühenduskiht [15,19]

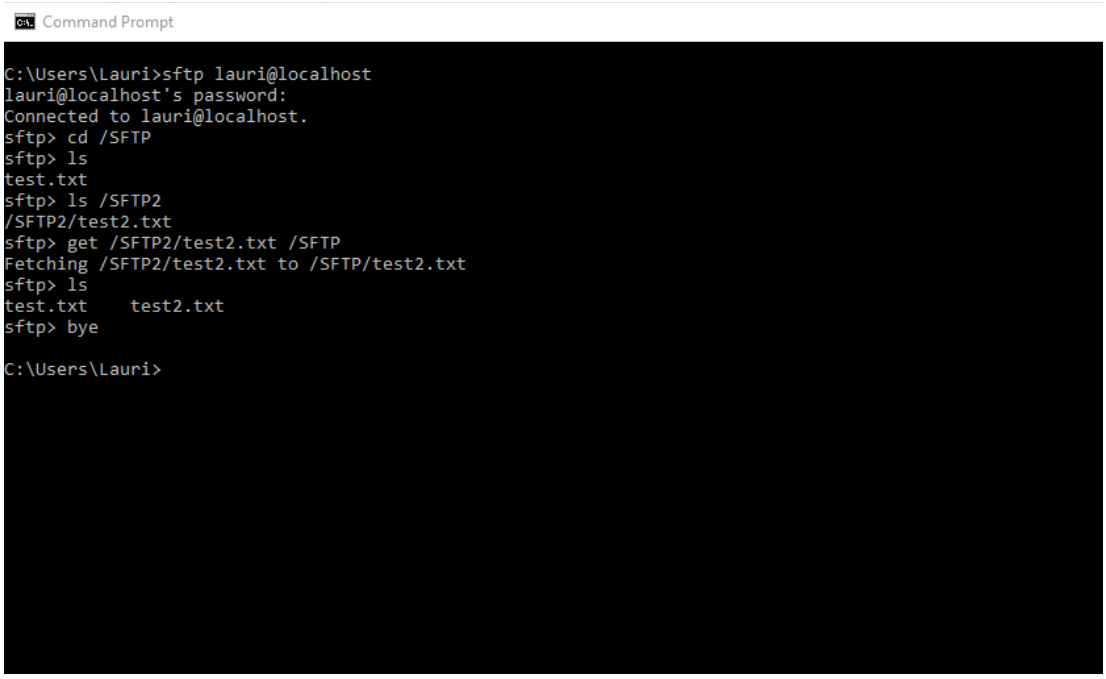
  Tegeleb SSH kanalite loomisega - üle ühe SSH ühenduse võib luua mitu kanalit. See kiht eeldab, et varasemalt on loodud ühendus transpordikihis ja kasutaja on autentimiskihis edukalt autenditud. Kanaleid kasutatakse kaugarvutis olevate failide haldamiseks ja kaugarvuti kesta käskude saatmiseks. Samuti on kanalitel olemas tugi pseudo-terminalide jaoks.

Tavaliselt kasutab ühendus kliendi ja server vahel kõiki kolme kihti.

3.2 Turvaline failiedastusprotokoll

Turvaline failiedastusprotokoll (inglise k. *Secure Shell File Transfer Protocol*, lühend *SFTP*) on protokoll, mis annab kasutajale võimaluse hallata kaugarvutis asuvaid faile [20]. SFTP on disainitud ühendusena kasutama turvakesta versioon kahte ning toetab kõiki turvakesta turvalisuse ja autentimise funktsionaalsusi [21].

STFP tegeleb ainult failide haldamisega – ühenduse loomine, kasutaja autentimine ja kanali loomine on jäetud turvakesta teha [20]. Protokoll toimib päring-vastus mudeli alusel [20]. Iga päring tähistab ühte operatsiooni failisüsteemis ja klient saab serverisse korraga teha mitu päringut [20].



Joonis 3. Näide serveris failioperatsioonide tegemisest kasutades SFTP'd Windowsi käsuviibas.

Failioperatsioonide tegemist SFTP protokolli kasutades näeb joonisel 3.

3.3 Implementatsioonid Pythonis

SSH ja SFTP protokolle on Pythonis implementeeritud mitmes teegis [22]. Järgnevalt tuuakse välja valik teekidest koos lühiülevaatega [22]:

- Paramiko [23]

  Teek on kirjutatud ainult programmeerimiskeeles Python ja selles on olemas SSH2 ja SFTP tugi. Paramiko pakett ja API on ka väga hästi dokumenteeritud. Teegi API abstraktsioonitase on kõrge.

- ssh2-python [24]

  See teek kasutab C-keeles programmeeritud teeki libssh2. Seetõttu on teegi suhtlus SSH serveriga kiire. Olemas on SSH2 ja SFTP tugi. Dokumenteeritus on puudulik ja teegi API abstraktsioonitase on madal.

- Fabric ja Spur

  Paramiko peale ehitatud teegid, lihtsustavad tunduvalt Paramiko kasutamist [22]. Nende teekide kasutamine on tehtud kasutajatele võimalikult lihtsaks [22].

- PySSH [25] ja Conch [26]

  PySSH ja Conch ei tööta uusimate Pythoni versioonidega, on halvasti dokumenteeritud ning nende arendamine on pooleli jäetud.

4. Pistikprogramm

Käesolevas peatükis antakse ülevaade töö käigus loodud pistikprogrammist – selle funktsionaalsustest ja programmi toimimisest. Välja on ka toodud põhjalik installeerimis- ja konfigureerimisjuhend käesoleva pistikprogrammi jaoks.

4.1 Ülevaade

Programm on mõeldud kasutamiseks Thonny pistikprogrammina. Thonny on programmeeritud keeles Python [10] ja kasutab moodulit Tkinter graafilise kasutajaliidese loomiseks [11]. Seetõttu on samad valikud tehtud ka töö käigus loodud pistikprogrammi luues.

Rakendus annab kasutajale võimaluse käivitada Pythoni koodi ning piiratud arvu süsteemikesta käsklusi kaugarvutis. Võimalik on käivitada interaktiivsel käsureal sisestatud Pythoni käsklusi ja koodiredaktoris avatud Pythoni faili. Fail ja Pythoni käsklused käivitatakse kaugmasinas olevas Pythoni keskkonas, mis seatakse üles programmi töö alustades.

Suhtlus kaugarvutiga toimub kasutades SSH2 protokolli ning seda protokolli teostavad teegid on juba Pythonile olemas. Seetõttu otsustati kasutada kaugarvutiga suhtluseks teeki ssh2-python. Algselt oli programmis kasutusel sarnaste funktsioonidega teek Paramiko.

Põhjuseid teise teegi peale üleminekuks oli mitmeid:

- teek on tunduvalt kiirem [24];
- teek nõuab vähem ressursse [24].

Täpsemalt selgitatakse rakenduse tööd alampeatükis „Programmi töö“.

4.2 Installeerimine ja konfigureerimine

Pistikprogrammi installeerimise eelduseks on arenduskeskkonna Thonny, soovitatavalt versioon 3.1.2, olemasolu. Teiste Thonny versioonidega ei pruugi pistikprogramm korrektselt töötada. Konfigureerimiseks on vajalik kaugarvuti operatsioonisüsteemiga Linux. Selles kaugarvutis peab olema ka töötav SSH server. Pistikprogrammi repositoorium asub veebilehel https://bitbucket.org/Starrimus/thonny-ssh/.

Järgnevalt on toodud installeerimisjuhend:

1) Lae pistikprogrammi repositooriumist alla *ssh* nimeline ZIP-fail.
2) Ava Thonny.
3) Liigu menüüs *Tools -> Manage plug-ins.*
4) *Install from local file* pealkirja all vajuta *here* peale.

5) Taaskäivita Thonny.

Peale nende juhiste täitmist peaks nähtavale ilmuma *SSH* nimeline menüü.

Kõigepealt tuleb programmi sisestada kaugarvuti andmed.

1) Liigu menüüs *Tools -> Options* ja vali alamleht *Interpreter*.
2) Vali rippmenüüst *Python in a remote machine (through SSH)*.
3) *Remote server IP* lahtrisse kirjuta kaugarvuti *IPv4* aadress.
4) *Remote server Port* lahtrisse kirjuta kaugarvuti SSH serveri port.
5) *Remote server Python PATH* lahtrisse võib kirjutada kaugarvutis asuva Pythoni kataloogitee. Selle lahtri võib jätta ka tühjaks.
6) Vajuta OK.

Peale OK vajutamist tuleb kasutaja ekraanile viit, milles küsitakse kasutaja andmeid. Kui viita ilmub punases tekstis kiri *Status: connection failed,* siis ühendus antud andmetega ebaõnnestus. Uuesti proovimiseks tuleb viit sulgeda ja siis teha uuesti läbi konfigureerimise sammud 1-5.

Kui kasutaja ekraanile viita ei ilmu, siis on järelikult arvutiga ühendus loodud ja saab konfigureerimisega edasi minna:

1) *Username* lahtrisse tuleb sisestada kasutajanimi, millega soovitakse sisse logida.
2) Viida paremal pool tuleb valida sisselogimisviis. Valida on võimalik kahe viisi vahel – võtmepõhine ja paroolipõhine.
    a. Valides *publickey* tuleb *Select file…* klikates valida kasutajaga seotud privaatvõti. *Private key password* lahtrisse tuleb kirjutada selle võtme parool. Välja võib jätta ka tühjaks, kui parool puudub.
    b. Valides *password* tuleb *Password* lahtrisse kirjutada kasutaja parool.
3) Vajuta nupu *Login* peale.

Kui kõik andmed on õiged, siis peaks ühendus serveriga loodud olema . Peale eduka ühenduse loomist saab läbi Thonny programme käivitada kaugarvutis asuvas Pythoni kestas ja läbi interaktiivse käsurea saata kesta ka käsklusi.

4.3 Programmi töö

Selles alampeatükis selgitatakse rakenduse arhitektuuri ja tööpõhimõtteid. Rakenduse töö on jagatud kolmeks:

- kaugarvutiga ühendumine

  Kirjeldab põhjalikult kaugserverisse SSH protokolli abil ühendumise protsessi.

- faili ja koodi käivitamine kaugmasinas;

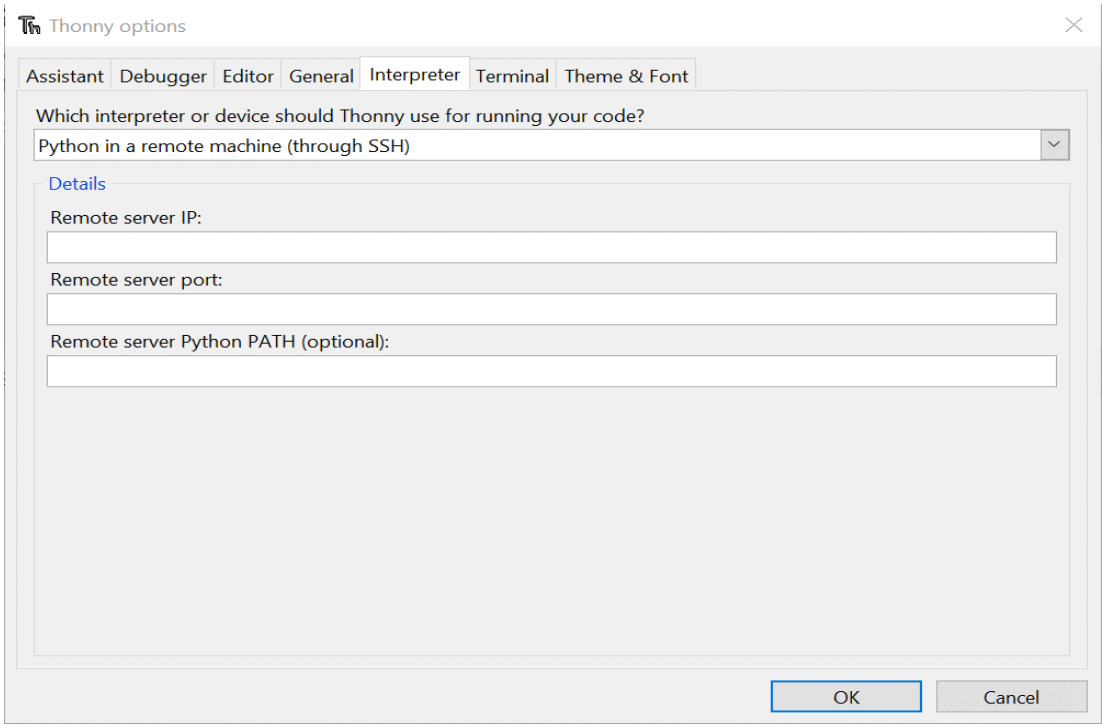
  Kirjeldab, kuidas pistikprogramm käivitab kaugserveris koodiredaktoris avatud faili, interaktiivsele käsureale kirjutatud Pythoni ja süsteemikesta käske.

- faili üleslaadimine kaugmasinasse ilma varasema ühenduseta

  Kirjeldab faili üles laadimise protsessi ilma varasemat ühendust kaugmasinasse loomata.

Kaugarvutiga ühendumine

Kui kasutaja on avanud Thonny sätted ja valinud interpretaatoriks rippmenüüs oleva valiku *Python in a remote machine (through SSH)* (lisatud pistikprogrammi poolt), siis avaneb kasutajaliides, mis on näha joonisel 4. Rakenduse edasiseks tööks on vajalikud kaugarvuti IPv4 ja SSH serveri port.



Joonis 4. Kaugarvuti informatsiooni sisestamise vaade.

Väli *Remote server IP* ootab sisendiks kaugarvuti õigesti vormindatud IPv4 aadress. Õigesti vormindatud aadress koosneb neljast punktiga eraldatud täisarvust ja iga täisarv peab olema vahemikus 0 ja 255 [27]. Korrektsed aadressid on näiteks 103.13.31.251 ja 1.1.1.153. Välja vaikeväärtus on 127.0.0.1.

Väli *Remote server port* ootab sisendiks SSH serveri pordinumbrit. Korrektsed pordinumbrid on 16-bitilised märgita täisarvud (arv vahemikus 0 ja 65535) [28]. Välja vaikeväärtus on 22.

Väli *Remote server Python PATH* ootab sisendiks kaugserveri Pythoni interpretaatori kataloogiteed. Kui see väli täidetakse, siis üritab programm kaugserveris kasutada Pythoni käivitamiseks seda kataloogiteed. Selle ebaõnnestumisel seatakse serveris üles teine keskkond. Kataloogitee peab olema absoluutne, seda ka valideeritakse.

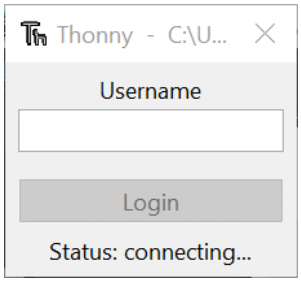
Väljade väärtused valideeritakse, kui väli kaotab fookuse.

Kui kasutaja vajutab OK nupule, siis valideeritakse väljade väärtused uuesti. Kui välja väärtus vastab nõuetele, siis salvestatakse väärtus Thonny konfiguratsioonifaili. Vastasel juhul konfiguratsioonifailis väärtus ei muutu. Kui varem konfiguratsioonifailis seda välja ei olnud, salvestatakse sinna vaikeväärtus.

Kui kasutaja vajutab *Cancel* nupule, siis jääb Thonny vana interpretaatorivaliku juurde ja mitte midagi konfiguratsioonifailis ei muudeta.

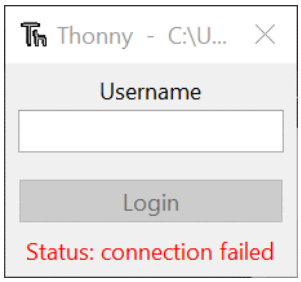
Peale OK nupu vajutamist alustab tööd Thonny mõistes tagarakendus. Thonnys on tagarakendus programmi osa, mille ülesanne on suhelda kasutajaliidesega – see tähendab, et tagarakenduse kaudu saadetakse kasutajaliidesele kõik Thonny kestas näidatavad sõnumid. Samuti on tagarakenduse ülesanne käivitada koodiredakoris avatud faili ja interaktiivsesse käsuritta kirjutatud koodi. Tagarakendus, mida programm kasutama hakkab, valitakse joonisel 4 näha olevas rippmenüüs.

Tagarakenduse lähtestamisel loetakse Thonny konfiguratsioonifailist kaugarvuti IP, serveriprogrammi port ja interpretaatori kataloogitee. Samuti luuakse kaks järjekorda – ühte neist sisestatakse programmi töö käigus eesrakenduses näitamist vajavad sõnumid ja teise järjekorda töödeldud käsud. Töödeldud käskude järjekorrast loeb käske teine, SSH ühendusega tegelev programmi osa.



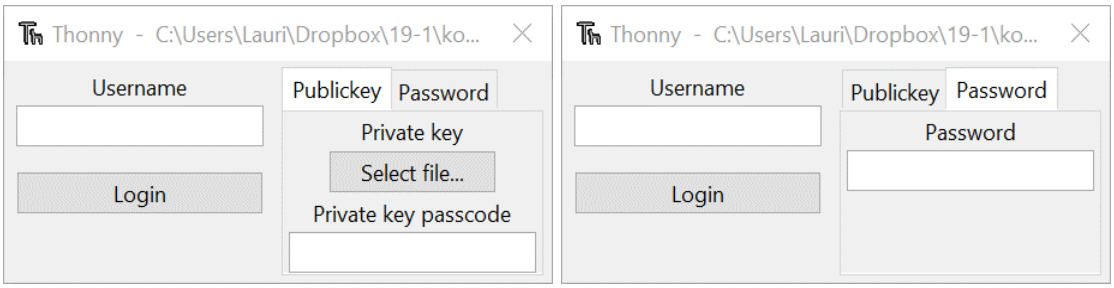
Joonis 5. Kasutaja andmete küsimise esimene vaade.

Tagarakenduse lähtestamise käigus luuakse joonisel 5 näha olev aken. Akna loomisel loob programm eraldi lõime, milles ühendutakse läbi SSH kaugarvutisse. Kaugarvutisse ühendumine on vajalik selleks, et teada saada serveri poolt toetatud autentimisviisid. Serverist saadud võimalikke autentimisviise võrreldakse pistikprogrammi toetatud viisidega.



Joonis 6. Serveriga ühendumise ebaõnnestumise vaade.

Kui kaugarvuti SSH serveriga ei õnnestu ühendust luua, siis kuvatakse veateade, mida on näha joonisel 6 olevast kasutajaliidesest.



Joonis 7. Serveriga ühendumise õnnestumise vaade.

Kui ühendus on edukas, siis kuvatakse joonisel 7 näha olev kasutajaliides. Joonisel 7 on aken jagatud kaheks – vasakul on kasutajanime väli ja sisse logimise nupp, paremal on võimalik valida võimalikke autentimisviiside vahel. Käesoleva töö kirjutamise ajal on pistikprogrammi poolt toetatud võtme- ja paroolipõhised autentimised.

Akna vasakul pool asuva välja *Username* väärtus ei tohi olla tühi. Paremal pool olevas akna osas valideeritakse ainult valitud vahelehe sisu.

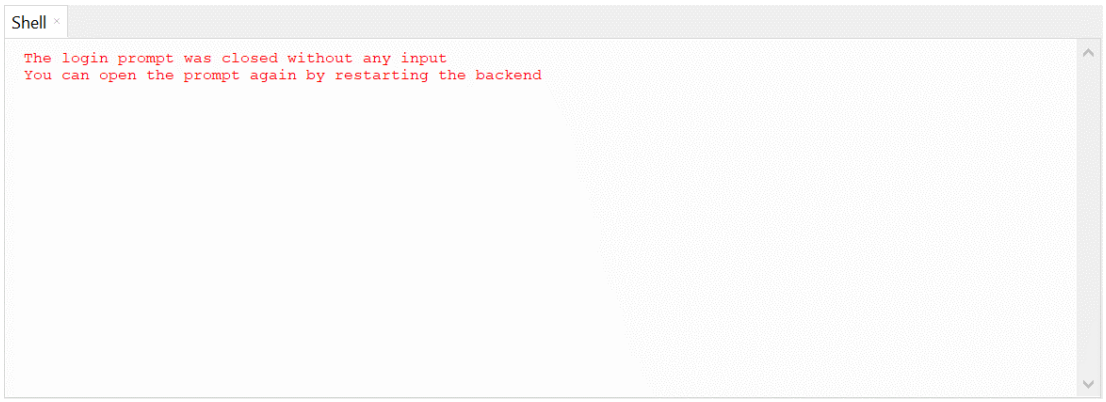
1) Võtmepõhine autentimine (vaheleht nimega *Publickey*)

   Nupu *Select file…* peale vajutades tuleb kasutajale ette failibrauser, kus saab valida privaatvõtme faili. Valitud failinimi salvestatakse muutujasse. Selle muutuja väärtus ei tohi olla tühi. Väljal *Private key passcode* ei ole valideerimist.

2) Paroolipõhine autentimine (vaheleht nimega *Password*)

   Välja *Password* väärtus ei tohi olla tühi.

Kui mõne valideeritava välja väärtus ei ole korrektne, programm tööga edasi ei lähe ja kasutaja peab andmed reeglitele vastavaks muutma. Väli, mille väärtus ei ole korrektne, värvub punaseks. Kui *Select file…* taustal oleva muutuja väärtus ei ole korrektne, värvuvad selle nupu servad punaseks.



Joonis 8. Veateada akna kinnipanekul nupust X.

Akna külge on registreeritud kaks sündmust, mis saadetakse tagarakendusele.

Järgnevalt on toodud need kaks sündmust:

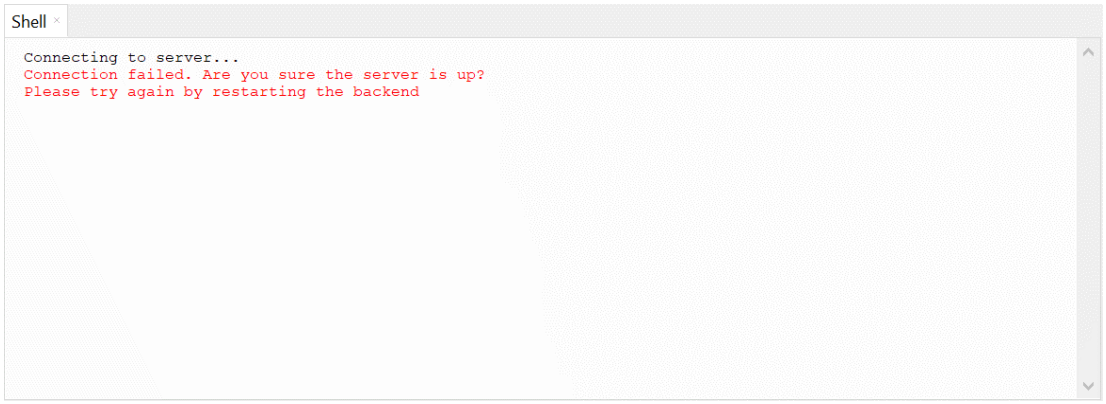
1) Akna kinnipanek

   Kui aken sulgub peale *Login* nupu vajutamist, siis ei toimu midagi ja programm läheb tööga edasi.

   Kui aken suletakse nupust X, siis ilmub Thonny kesta joonisel 8 nähtav tekst ja programm lõpetab töö. Tagarakendus tuleb Thonny nupu *Stop/Restart backend* peale klikkamisega taaskäivitada.

2) Kasutaja andmete aknast saatmine
   Kui kasutaja klikkab *Login* nupule ja kõikide valideeritavate väljade väärtused on korrektsed, siis antakse tagarakendusele selle sündmusega teada, et kasutajaandmed on valmis SSH'ga tegelevale programmi osale edastamiseks.

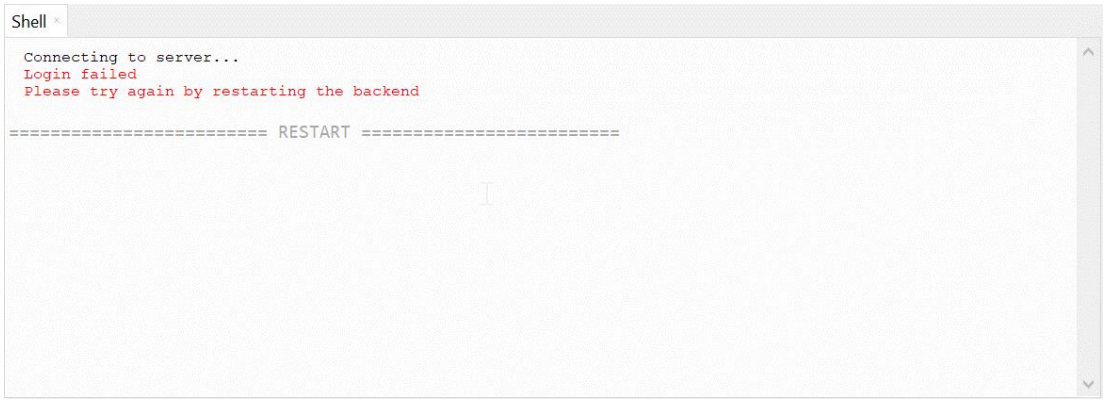
Kasutaja andmete kättesaamisel edastab tagarakendus need andmed eraldi lõimes olevale SSH ühendusega tegelevale klassiobjektile. Selle klassi ülesseadmisel genereeritakse sessiooni ID, mida kasutatakse failide üleslaadimisel ja koodiredaktoris avatud faili käivitamisel, ja mille abil kontrollitakse klassile edastatud andmete korrektsust.



Joonis 9. Veateade ühenduse ebaõnnestumisel.

Peale nende toimingute läbiviimist seatakse vajalik keskkond üles kaugarvutis. Selleks ühendub rakendus kaugarvutisse. Kui ühendus ei õnnestu, ilmub Thonny kesta joonisel 9 nähtav veateade.

Ühenduse õnnestumisel üritab rakendus autentida saadud andmetega. Kui autentimine ebaõnnestub, siis ilmub Thonny kesta joonisel 10 nähtav veateade ja kasutajalt küsitakse uuesti sisselogimise andmeid.



Joonis 10. Veateade sisselogimise ebaõnnestumisel.

Peale ühenduse loomist ja autentimist luuakse seanss SFTP jaoks ning kaks kanalit. Esimene kanal luuakse käskude saatmiseks süsteemikesta ja teine kanal Pythoni käskude käivitamiseks.

Rakendus üritab tuvastada kaugarvuti operatsioonisüsteemis. Operatsioonisüsteem on vaja tuvastada Pythoni keskkonna püsti panekuks serveris.

Kui operatsioonisüsteemi on tuvastatud, siis pannakse kaugarvutis püsti Pythoni keskkond. Selleks kontrollitakse kõigepealt, kas operatsioonisüsteem on pistikprogrammi poolt toetatud. Hetkel on rakenduse poolt toetatud ainult Linux kernelit kasutavad operatsioonisüsteemid. Kui operatsioonsüsteem on selline, mida ei toetata, siis programm lõpetab töö ja saadab Thonny kesta veateate.

Kui interpretaator on konfiguratsioonifailist loetav ja tegemist on korrektse kataloogiteega, üritatakse seda faili käivitada. Selle õnnestumisel on Python üles seatud ja töötab. Vastasel juhul otsitakse kaugarvutist programmile tuntud asukohtadest pistikprogrammiga ühilduvat Pythonit. Kui see leitakse, siis käivitatakse ja kasutatakse seda. Vastasel juhul üritatakse Thonny ja sellega kaasas olev Python alla laadida. Selle ebaõnnestumisel lõpetab programm töö ja saadab Thonny kesta veateate.
Kui Python on kaugarvutis püsti pandud, siis kontrollitakse, kas Python ka reaalselt masinas töötab. Kui Python ei tööta, siis lõpetab programm töö ja saadab Thonny kesta veateate.

Faili ja koodi käivitamine kaugmasinas

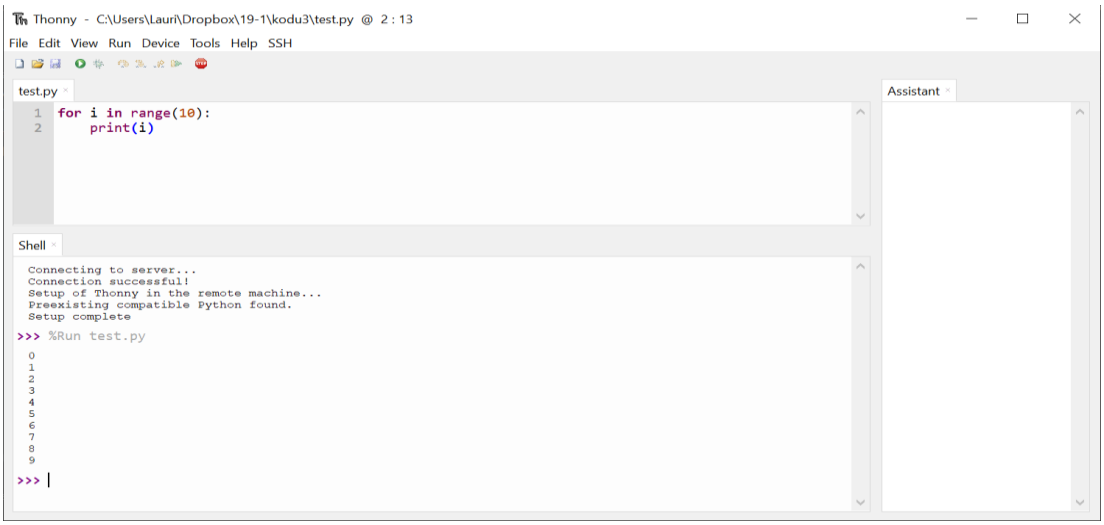
Selles peatükis kirjeldatud funktsionaalsused nõuavad, et interpretaatoriks on Thonny sätetest valitud antud pistikprogrammi poolt loodud võimalus. Täpsemalt on selle valiku tähtsusest räägitud alampeatükis „Kaugarvutiga ühendumine“.

Pistikprogrammi kasutades on kaugarvutis võimalik käivitada koodiredaktoris avatud faili, interaktiivsele käsureale kirjutatud Pythoni ja süsteemikesta käske.

Järgnevalt kirjeldatakse neid kolme protsessi:

1) Faili käivitamine kaugmasinas

   Faili käivitamiseks tuleb kasutajal vajutada Thonny kasutajaliideses vastavat nuppu. Peale nupu vajutamist saadetakse käsklus ja tagarakendus laeb avatud faili kaugarvutisse kataloogiteele /<*sisselogitud kasutaja kodukaust*>/*.thonny-ssh/*<*sessiooni id*>*.py*. Enne programmi käivitamist luuakse samasse kausta lukkfail, kuhu laeti eelnev fail, nimega <*sessiooni id*>*_file.py*. Seda faili kasutatakse kontrollimaks, kas programm on oma töö lõpetanud.

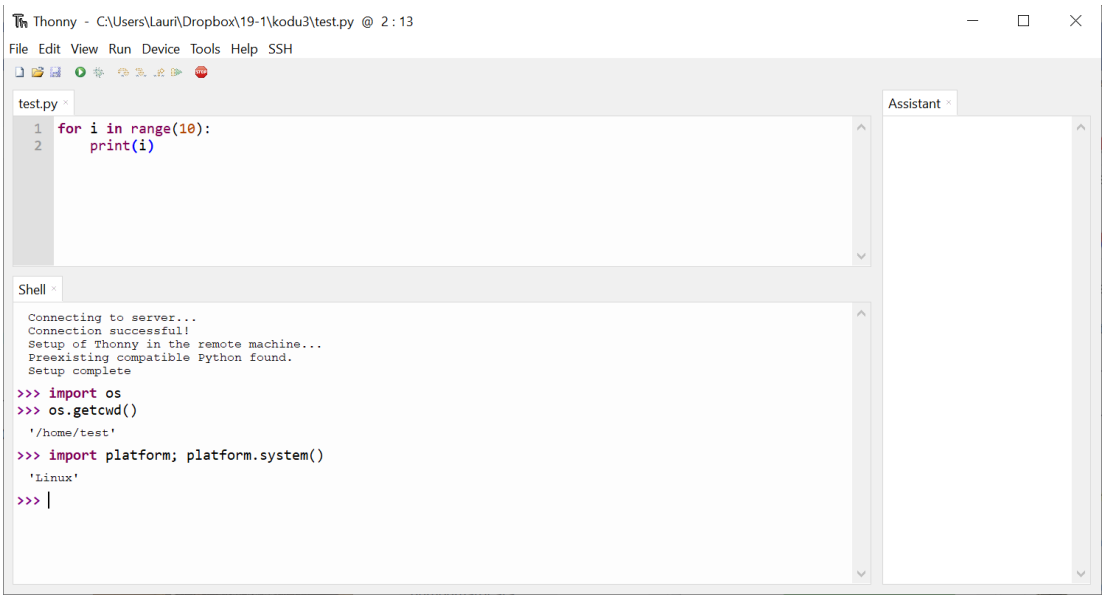


Joonis 11. Koodiredaktoris avatud faili käivitamine kaugarvutis.

Faili loomise järel käivitatakse üleslaetud programm. Näidisprogrammi käivitamise tulemust on näha joonisel 11. Kui programm on töö lõpetanud, siis lukkfail kustutatakse.

2) Pythoni käskluste käivitamine

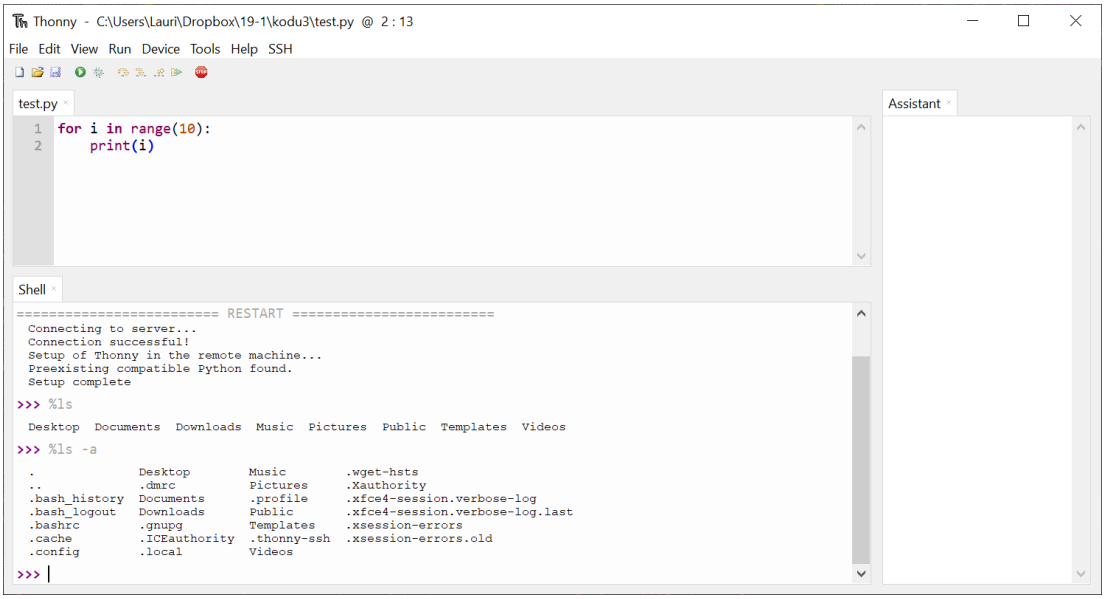
Käsu käivitamiseks tuleb kasutajal interaktiivsele käsureale kirjutada käsk ja vajutada reavahetusklahvile. Reavahetusklahvile vajutades saadetakse käsklus edasi tagarakendusele. Tagarakenduses kontrollitakse, kas see käsk on keelatud käskude nimistus. Töö kirjutamise ajal on keelatud käskude hulgas ainult käsud, millega on võimalik Pythoni interaktiivsest versioonist väljuda. Käsule lisatakse märgend, mille järgi saab programm hiljem otsustada, millal tuleb serverist lugemine lõpetada. Joonisel 12 on näha edukate käskude käivitamise tulemust.



Joonis 12. Pythoni koodi käivitamine kaugarvutis.

3) Süsteemikesta käskluste käivitamine

Käsk saadetakse reavahetusklahvi vajutusega tagarakendusele, mis lisab käsule märgendi tähistamaks programmi lõppu. Süsteemikestas on võimalik käivitada ainult piiratud arv käske.



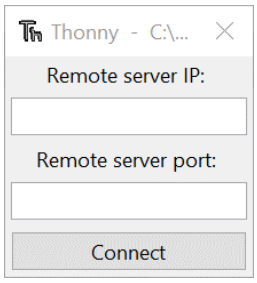
Joonis 13. Käsu käivitamine kaugarvuti süsteemikestas.

Töö kirjutamise ajal on pistikprogrammi poolt toetatud käsk *ls* koos parameetritega *-a* ja *-l*. Joonisel 13 on näha edukat süsteemikesta käsu käivitamist.

Faili üleslaadimine kaugmasinasse ilma varasema ühenduseta

Selles peatükis kirjeldatud funktsionaalsus ei nõua varasemat kaugarvutiga ühendumist ja on kasutatav ainult koodiredaktoris avatud faili mugavaks üles laadimiseks kaugmasinasse.

Faili saab üles laadida igasse kaugmasinasse, millel on olemas SSH server SFTP toega. Joonisel 14 on näha liidest, kus küsitakse kaugarvuti IP'd ja SSH serveri porti. Valideerimisreeglid on samad, mis joonisel 4 kujutatud liidese samanimelistel väljadel.

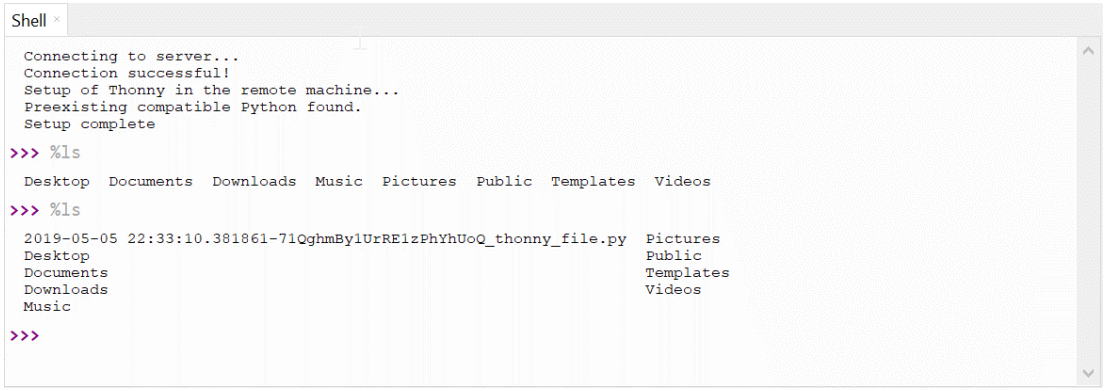


Joonis 14. Kasutajaliides kaugarvuti IP ja serveri pordi küsimiseks.

Kui kasutaja vajutab *Connect* nupule ja mõni väli ei valideeru, värvub väli, mille väärtus ei ole korrektne, punaseks.

Kui kõik väljad valideeruvad, tuleb kasutajale ette kasutajaliides näha joonistel 5-7. Arhitektuur on sama, kui nendel joonistel. Peale nende andmete sisestamist üritatakse koodiredaktoris avatud

fail kaugmasinasse üles laadida ja Tkinteri *messagebox* liidest kasutades kasutajale faili üleslaadimise staatust näidata.



Joonis 15. Fail sai üles laetud.

Joonisel 15 on näha eduka faili üleslaadimist. Masinasse laetud faili nimeks saab *<tänane kuupäev ja kellaaeg>-<sessiooni ID>_thonny_file.py*

4.4 Puudused ja edasiarendamise võimalused

Peamine puudus programmis on see, et pistikprogramm toetab kaugarvutina ainult Linuxi kernelil põhinevaid operatsioonisüsteeme – Raspberry Pi ja ev3dev kasutavad Linuxi kernelit ja seetõttu oli teiste Linuxi kerneliga operatsioonisüsteemide toe lisamine triviaalne. Seetõttu ei saa kaugarvutina kasutada kahte populaarset operatsioonisüsteemi Windows ja macOS. Nende operatsioonisüsteemide toe lisamine suurendaks võimalikke kaugarvutite hulka märgatavalt.

Pistikprogramm ei toeta ka Thonny sisseehitatud siluri kasutamist. Selle lisamine võimaldaks kasutajatel paremini enda programmi tööd mõista.

4.5 Testimine

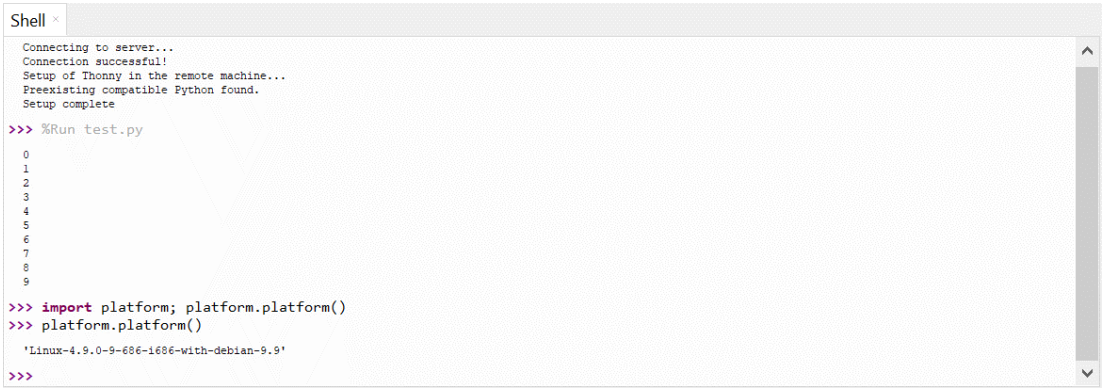
Testkeskkondadeks seati üles kaks virtuaalmasinat, virtuaalmasinate loomiseks kasutati programmi Virtualbox. Mõlematesse virtuaalmasinatesse installeeriti Linuxi kernelit kasutav operatsioonisüsteem ja SSH server OpenSSH. Windowsi ega macOSi kasutavate kaugmasinatega teste ei tehtud, sest need operatsioonisüsteemid pole pistikprgorammi poolt toetatud. Testiti, kas on võimalik serveritesse edukalt ühenduda, nendes püsti panna Pythoni keskkond ja käivitada serveris Pythoni koodi ja fail.

Virtuaalmasinate kirjeldused:

1) Masin 1

   Operatsioonisüsteem: 32-bitine Debian 6.3.0
   Linuxi kerneli versioon: 4.9.0
   OpenSSH versioon: 7.4

   Masina 1 abil testimist pistikprogrammiga on näha joonisel 16.



Joonis 16. Ühendumine ja testimine kasutades masinat 1.

2) Masin 2

   Operatsioonisüsteem: 64-bitine Ubuntu 18.04
   Linuxi kerneli versioon: 4.18.0
   OpenSSH versioon: 7.6

   Masina 2 abil testimist pistikprogrammiga on näha joonisel 17.

Joonis 17. Ühendumine ja testimine kasutades masinat 2.

Joonistelt 16 ja 17 on näha, et faili käivitamised õnnestusid mõlemas masinas ja samuti õnnestus ka Pythoni käsu *platform.platform()* käivitamine.

5. Kokkuvõte

Käesolevas bakalaureusetöös anti ülevaade arenduskeskkonnast Thonny ja turvakestast. Töö eesmärgiks oli luua pistikprogramm, millega saaks ühenduda turvakesta kasutades Raspberry Pi ja Lego Mindstorms EV3 masinatesse ja seal käivitada Pythoni programme. Selle idee implementatsioon võimaldab Thonny kasutajal kaugmasinas käivitada Pythoni skripte ja käske. Samuti saab kaugmasina kestas käivitada selle kesta jaoks mõeldud käske. Programm toetab kaugarvutina lisaks Raspberry Pile ja EV3le kõiki teisi Linuxi kernelit kasutavaid operatsioonisüsteeme. Rakenduse poolt lisatud kasutajaliides on tehtud võimalikult sarnaseks Thonny liidesega.

6. Viidatud kirjandus

[1] Why Python is Awesome https://pythonbasics.org/why-python-is-awesome/ (08.05.2019)

[2] Thonny koduleht. https://thonny.org/ (14.01.2019)

[3] About EV3 https://www.lego.com/en-us/mindstorms/about-ev3 (08.05.2019)

[4] Parents guide to Raspberry Pi. https://www.raspberrypi.org/learning/parents-guide/physical/ (14.01.2019)

[5] GPIO – Raspberry Pi Documentation https://www.raspberrypi.org/documentation/usage/gpio/README.md (08.05.2019)

[6] GPIO (General Purpose Input/Output) Definition https://techterms.com/definition/gpio (08.05.2019)

[7] Getting started with ev3dev. https://www.ev3dev.org/docs/getting-started// (14.01.2019)

[8] Silva A. 3 popular programming languages you can learn with Raspberry Pi. https://opensource.com/article/19/3/programming-languages-raspberry-pi (08.05.2019)

[9] SSH (Secure Shell) – Raspberry Pi Documentation https://www.raspberrypi.org/documentation/remote-access/ssh/ (08.05.2019)

[10] Thonny koodirepositoorium. https://github.com/thonny/thonny (14.01.2019)

[11] Theming, Thonny wiki. https://github.com/thonny/thonny/wiki/Theming (14.01.2019)

[12] Plugins, Thonny wiki. https://github.com/thonny/thonny/wiki/Plugins (14.01.2019)

[13] Ylönen T. SSH — Secure Login Connections over the Internet. *Sixth USENIX Security Symposium.* 1996. https://www.usenix.org/legacy/publications/library/proceedings/sec96/full_papers/ylonen/index.html (14.01.2019).

[14] e-Teatmik: IT ja sidetehnika seletav sõnaraamat. http://www.vallaste.ee.

[15] Ylönen T. RFC 4251 - The Secure Shell (SSH) Protocol Architecture https://tools.ietf.org/html/rfc4251 (06.05.2019)

[16] SSH Frequently Asked Questions. http://www.snailbook.com/faq/ssh-1-vs-2.auto.html (06.05.2019)

[17] Ylönen T. RFC 4253 - The Secure Shell (SSH) Transport Layer Protocol https://tools.ietf.org/html/rfc4251 (06.05.2019)

[18] Ylönen T. RFC 4252 - The Secure Shell (SSH) Authentication Protocol https://tools.ietf.org/html/rfc4251 (06.05.2019)

[19] Ylönen T. RFC 4254 - The Secure Shell (SSH) Connection Protocol https://tools.ietf.org/html/rfc4251 (06.05.2019)

[20] Ylönen T., Lehtinen S. https://www.ietf.org/proceedings/50/I-D/secsh-filexfer-00.txt (06.05.2019)

[21] SFTP File Transfer Protocol. https://www.ssh.com/ssh/sftp/ (14.01.2019)

[22] SecureShell – Python Wiki. https://wiki.python.org/moin/SecureShell (14.01.2019)

[23] Paramiko koodirepositoorium. https://github.com/paramiko/paramiko (14.01.2019)

[24] ssh2-python koodirepositoorium. https://github.com/ParallelSSH/ssh2-python (05.05.2019)

[25] PySSH. http://pyssh.sourceforge.net/ (14.01.2019)

[26] Conch. https://twistedmatrix.com/projects/conch/ (14.01.2019)

[27] IPv4 and IPv6 address formats. https://www.ibm.com/support/knowledgecenter/en/STCMML8/com.ibm.storage.ts3500.doc/opg_3584_IPv4_IPv6_addresses.html (05.05.2019)

[28] RFC 793 https://www.ietf.org/rfc/rfc793.txt (05.05.2019)

Lisad

I. Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Lauri Leiten,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose **Thonny arenduskeskkonna pistikprogramm skriptide käivitamiseks üle SSH**, mille juhendaja on Aivar Annamaa, 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.

Lauri Leiten

10.05.2019