2.1 Muutujad

SISSEJUHATUS

Programmide koostamisel on võimalik kasutada väga erinevaid komponente näiteks muutujaid, valikulauseid, tsükleid, alamprogramme jne. Selles osas vaatame lähemalt muutujaid.

MUUTUJAD

Tinglikult võiks muutujaid võrrelda maitseainete topsidega. Selleks, et maitseaineid lihtsamini üles leida, on kaval moodus topsid vastavalt sildistada. Sildi järgi leitakse vastav maitseaine paremini üles.

Muutujaid võiks võrrelda maitseainete topsidega

Kui köögis on olulisel kohal maitseained, siis programmeerimises on tähtsad andmed. Kui köögis hoitakse maitseaineid topsides, siis programmeerimises hoitakse andmeid mälupiirkondades. Kui köögis pannakse maitseained vastava sildiga topsi, siis programmeerijate maailmas pannakse andmed vastava nimega mälupiirkonda. Sellist nimega mälupiirkonda kutsutaksegi **muutujaks**. Muutujal on nimi ja muutujale väärtuse andmine käib Pythonis võrdusmärgi abil.

Järgnevalt annamegi ühele muutujale väärtuse.

MUUTUJALE VÄÄRTUSE ANDMINE

Anda muutujale mingi väärtus. Mida see tähendab? See tähendab, et muutujale antakse mingid andmed, mida ta peab endas hoidma. Umbes nii, et topsis, mille peale on kirjutatud kaneel, on hoiul kaneel.

Pythonis muutuja nime valimisel on omad reeglid ja tavad nagu ka näiteks tavaelus lapsele nime panemisel. (Reeglid ise on küll lapsele nimepanemise omadest mõnevõrra erinevad.) Pythonis

muutujale nime panemisel tuleb silmas pidada, et nimi ei tohi sisaldada tühikuid. Tühiku asemel kasutatakse vajadusel alakriipsu (nt `vanima_lapse_vanus`) või jäetakse vahe üldse ära (nt `vanimaLapseVanus`). Lisaks on tavaks, et muutuja nimi on kirjutatud väikeste tähtedega (va sõnade algustähed alates teisest).

Soovi korral saab lugeda erinevate vormindamistavade kohta Wikipedia leheküljelt.

Nüüd on teie enda valida, kas loete teksti edasi või vaatate videot, kus samuti sarnase näitega tegeletakse.

https://www.youtube.com/watch?v=Vz8EfZRSMB0

Vaatame nüüd näidet, kus väärtustame erinevaid muutujaid. Anname muutujale `poisi_vanus` väärtuseks poisi vanuse 12. Selleks käivitame Thonny ning trükime uude programmiaknasse (File -> New) esimesele reale:

```
poisi_vanus = 12
```

Toome mängu vanaema, andes teisel real muutujale `vanaema_vanus` väärtuseks 59:

```
vanaema_vanus = 59
```

Talletatud infot saame kasutada samade muutujate abil. Võime näiteks leida poisi ja vanaema vanuste summa, trükkides programmi kolmandale reale:

```
vanuste_summa = poisi_vanus + vanaema_vanus
```

Ekraanile väljastamiseks tuleks lisada käsk `print`.

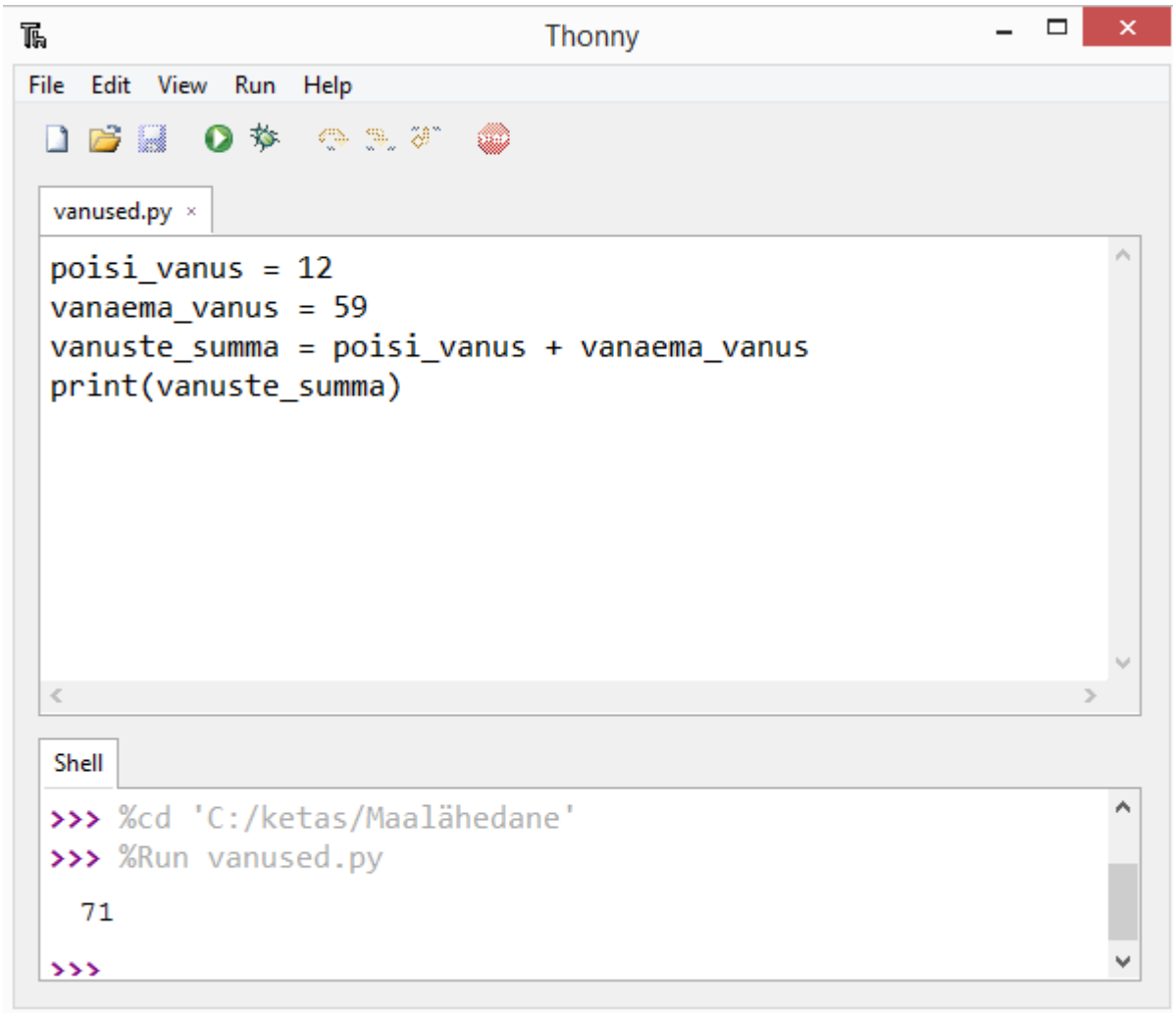
```
print(vanuste_summa)
```

Kokku saime järgneva programmikoodi:

```
poisi_vanus = 12
vanaema_vanus = 59
vanuste_summa = poisi_vanus + vanaema_vanus

print(vanuste_summa)
```

Programmi käivitades (Run -> Run current script) peaks tulemus olema selline:



Nuputa! Kuidas leida poisi ja vanaema keskmine vanus?

Muutujatega tegutsemisel tuleb meeles pidada, et muutuja peab enne selle kasutamist olema väärtustatud. Proovige käivitada juba varasemalt vaadeldud näidet sellisena:

```
poisi_vanus = 12
vanuste_summa = poisi_vanus + vanaema_vanus
vanaema_vanus = 59
print(vanuste_summa)
```

Millise teate väljastas Python? Miks selline veateade ilmnes ja kuidas viga parandada saab?

ARVULISTE VÄÄRTUSTE OMISTAMINE MUUTUJALE

Muutujale arvulise väärtuse omistamist juba vaatasime. Käib see siis näiteks nii:

```
lemmik_arv = 7
print(lemmik_arv)
```

Lisaks sellele on võimalik arvudega sooritada erinevaid tehteid. Vaadake tähelepanelikult, sest mõned märgid tähendavad teisi asju, kui me harjunud oleme:

- + liitmine
- - lahutamine
- / jagamine
- * korrutamine
- ** astendamine
- % jäägi leidmine
- // täisosa leidmine
- Klõpsake, et näha veelgi kasutatavaid matemaatilisi tehteid Wikipediast

Ülesanne

Mida väljastatakse ekraanile?

```
arv = 3**2
print(arv)
```

Vali 1.5

Vali 3

Vali 6

Vali 9

Vali Veateade

Lisaks täisarvudele võib muutujale anda väärtuseks ka koma sisaldava arvu. Koma rollis kasutatakse Pythonis punkti (näiteks 3,2 kirjutatakse Pythonis 3.2). Selliseid arve nimetatakse ujukomaarvudeks.

Järgmine näide on ujukomaarvude omistamisest kahele muutujale, nende summa arvutamisest ja ekraanile väljastamiseks:

```
esimene_arv = 3.2
teine_arv = 10.15
arvude_summa = esimene_arv + teine_arv
print(arvude_summa)
```

Muutuja väärtust saab programmis ka muuta. Proovige järgi, mis tulemus tuleb sellisel juhul:

```
esimene_arv = 3.2
teine_arv = 10.15

esimene_arv = 5

arvude_summa = esimene_arv + teine_arv
print(arvude_summa)
```

Ülesanne

Mida väljastatakse ekraanile?

```
a = 1
b = a
a = 2
print(b)
```

Vali 1

Vali 2

Vali 3

Vali b

Vali Veateade

ARVUDE KÜSIMINE KASUTAJALT KÄSUGA *INPUT()*

Sageli ei kirjutata andmeid programmi sisse, vaid neid küsitakse kasutajalt käsuga `input()`.

Võite kohe edasi lugeda või vaadata hoopis videot, mis ka sama teemat tutvustab.

https://www.youtube.com/watch?v=5hsnPZn0oYo

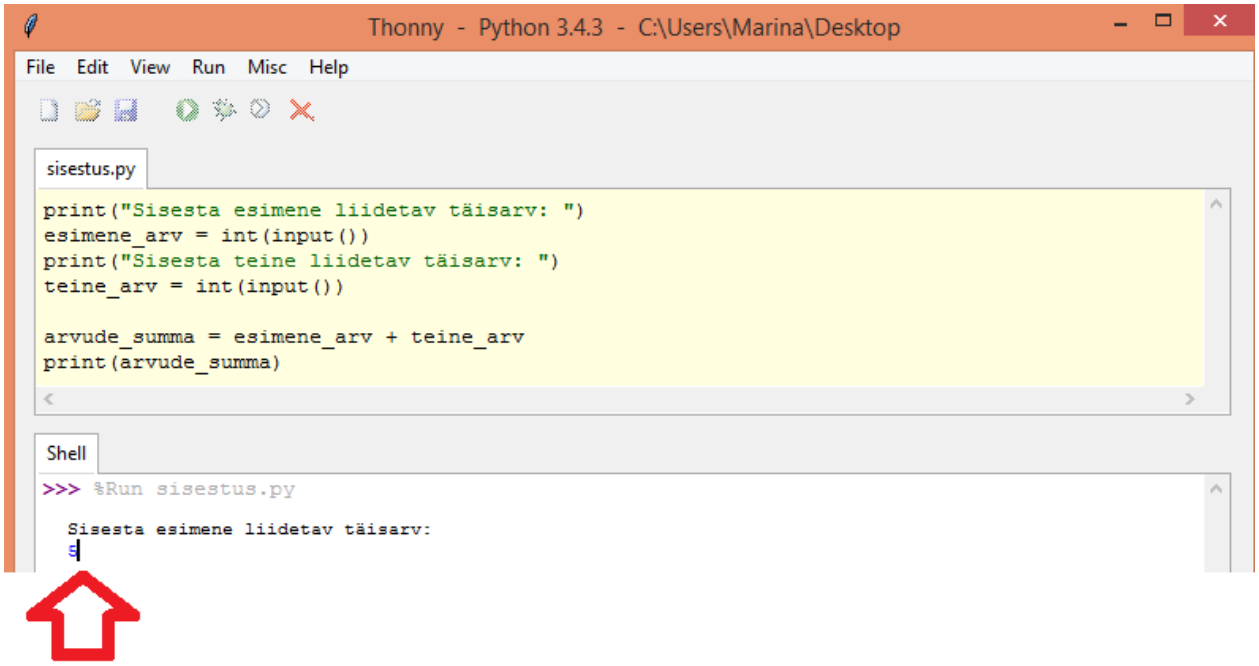
Näiteks võib programmi kirjutada selliselt, kus liidetavad arvud küsitakse kasutajalt.

```
print("Sisesta esimene liidetav täisarv: ")
esimene_arv = int(input())
print("Sisesta teine liidetav täisarv: ")
teine_arv = int(input())

arvude_summa = esimene_arv + teine_arv
print(arvude_summa)
```

Paneme tähele, et sisendi küsimisel tuleb sisend panna käsu `int()` sulgude vahele. Seda tuleb teha põhjusel, et Python saaks aru, et kasutaja sisestatut käsitletaks täisarvudena ja neid oleks võimalik hiljem kokku liita. Erinevatest **andmetüüpidest** räägime juba järgmisel leheküljel!

Kui kasutada programmeerimiskeskkonda Thonny, siis ootab *input* kasutaja sisestust alumises aknas (*Shell*).



Kasutaja sisestas 5 ja pärast vajutab Enter-it.

Kui kasutajalt ei oodata arvu, vaid lihtsalt teksti, mis salvestatakse muutujasse, siis pole `int()` käsku vaja. Näiteks lemmikvärvi küsimine kasutajalt ja otse vastava muutuja väärtustamine näeks välja nii:

```
print("Mis on sinu lemmikvärv?")
lemmik = input()
print("Ah, et sinu lemmikvärv on " + lemmik + ". Ilus värv!")
```

Tegelikult saab kaks esimest rida ka kokku võtta:

```
lemmik = input("Mis on sinu lemmikvärv?")
print("Ah, et sinu lemmikvärv on " + lemmik + ". Ilus värv!")
```

Proovige nüüd ise!

2.2 Andmetüübid

SISSEJUHATUS

Erinevat tüüpi andmete jaoks on ettenähtud erinev hulk mälu ning nendega saab teha erinevaid tehteid.

ANDMETÜÜBID

Muutujale antav väärtus on alati mingit teatud tüüpi. Sageli on programmeerimiskeeltes olemas järgmised tüübid:

- **Täisarvud** (näiteks 10, -4, 0);
- **Ujukomaarvud** (näiteks 2.5, 3.14, -7.23, 2.0);
- **Sõne / tekst** (näiteks "kass", "koer", "inimene");
- **Tõeväärtused** (*True* või *False*).

NB! Konkreetse väärtuse tüübi saab teada, kasutades käsku `type()`. Tulemuseks on näiteks täisarv (*int*), ujukomaarv (*float*), sõne (*str*) või tõeväärtus (*bool*). Proovige näiteks:

```
type(4)
type(4.5)
type("Maria")
```

või

```
type(True)
```

TEKSTILISE VÄÄRTUSE ANDMINE MUUTUJALE. SÕNE

Eelmisel leheküljel õppisime, kuidas anda muutujale väärtuseks arv (täisarv või ujukomaarv). Nüüd vaatame, kuidas muutujale anda väärtuseks tekst sõnena. Tuleb tähele panna, et see pole trükiviga, vaid tõepoolest on tegemist terminiga "sõne". Sõne ingliskeelne vaste on *string* ja programmeerimises võib sageli kohata ka lühendatud varianti **str**.

Tekstiline väärtus omistatakse muutujale jutumärkide vahel. Mitu teksti saame kokku panna, kasutades `+` märke.

Proovige näiteks, kuidas töötab järgnev programm:

```
nimi1 = "Mihkel"
nimi2 = "Mari"
tervitus = "Tere, " + nimi1 + " ja " + nimi2 + "!"
print(tervitus)
```

Liitmismärgil on programmeerimisel mitu tähendust:

- Arvude puhul tähendas `+` märk arvude kokkuliitmist (`5 + 3` annab tulemuseks `8`).
- Sõnede puhul tähendab `+` märk sõnede üksteise otsa kokkupanemist (`"tere" + "tore"` annab tulemuseks `"teretore"`)

Sõne võib koosneda ka näiteks ühest tühikust. Järgmises näites pannakse muutujate väärtuste vahele tühik.

```
linn = "Tartu"
hoone = "raekoda"
kokku = linn + " " + hoone
print(kokku)
```

Nagu eelmistest programminäidetest näha, siis sõned on jutumärkidega, muutujanimed aga ilma.

NB! Sellisel viisil saab omavahel kokku panna **ainult** sõnesid. Arvu lisamiseks antud lausesse peaks kasutama funktsiooni `str()`, et see esmalt sõneks teisendada. Näiteks kui muutuja `arv` sisaldaks väärtust `5`, siis `str(arv)` annaks meile väärtuse `"5"` (sõnena).

Ülesanne

Mida väljastatakse ekraanile?

```
a = 4
b = 5.5
s = a + b
print("Summa on " + s)
```

Vali Summa on 9.5

Vali Summa on9.5

Vali Summa on 9,5

Vali Summa on9,5

Vali Veateade

SÕNEDE LISAVÕIMALUSED

Sõnedega saab veel erinevaid toredaid asju teha. Näiteks saab sõnesid "korrutada".

```
lause = "mull " * 20
print(lause)
```

Tulemuseks saame:

```
>>>
mull mull mull mull mull mull mull mull mull mull
mull mull mull mull mull mull mull mull mull mull
>>>
```

Ülesanne

Mida väljastatakse ekraanile?

```
a = "1"
b = 1
print("Arvud on " + 5 * a + " ja " + str(5 * b))
```

- Vali Arvud on "5" ja 5
- Vali Arvud on 5 ja "5"
- Vali Arvud on 5 ja 11111
- Vali Arvud on 11111 ja 5
- Vali Veateade

TÕEVÄÄRTUSE OMISTAMINE MUUTUJALE

Kui arvud ja sõned on võib-olla tuttavamana tunduvad andmetüübid (oleme ju arvude ja tekstiga ikka tegutsenud), siis tõeväärtused on võib-olla mõnevõrra võõramad. Nimelt saab muutujale väärtuseks anda ka väärtusi `True` (tõene) ja `False` (väär).

```
liigaasta = False
print(liigaasta)
```

Tõeväärtuste tüüpi nimetatakse algebralise loogika looja George Boole'i järgi inglise keeles *boolean* tüübiks. Kohata võib ka sõna *boolean* lühendatud varianti *bool*. Põhjalikumalt vaatame tõeväärtuste rakendusi tingimuslausete juures.

2.3 Silmaring. Kodumasinad

Tänapäevases maailmas pole mingisugune ime vajutada nuppu ja selle tulemusena saada oma tahtmine, olgu selleks siis mis iganes - kiiresti keedetud vesi, puhtaks pestud pesu või ahjus valminud hea õhtusöök. Kuid kui palju me teame sellest, mis on peidetud antud tegevuste taha? Mis tegelikult juhtub, kui me vajutame nuppu? Kas me ehk isegi oleme kaasa aidanud sellele maagilisele tegevusele?

Üheks nutikaimaks seadmeks, mis paljudel enesega sisuliselt igal pool kaasas käib, on nutitelefon. Nutitelefon on muutunud meie ühiskonnaks täiesti tavaliseks nähtuseks ning kõik, mis me sellega teeme, on harjumuspärane. Kuid kas me tegelikkuses kasutame ära maksimaalselt võimalusi, mida see nutikas seade meile pakub?

Kui me mõtleme, et mida me oskame oma nutitelefonidega teha, siis meenuvad meile sellised asjad nagu helistamine, sõnumite saatmine, internetiavaruste kasutamine, pildistamine, e-mailide kirjutamine ja lugemine. Kuid kas me oleme kunagi mõelnud näiteks oma kodulooma toitmisele läbi oma seadme või hoopis pesumasina käivitamist? Kohati tunduvad need asjad ulmelised, kuid need on siiski võimalikud selles samas tavalises infotehnoloogia nutimaailmas, kus me elame.

Selle kõige põneva taga on programmeerimine. Programmeerimine võimaldab meil panna seadmeid tegema seda, mida me ise tahame, tuleb anda ainult vastav korraldus. Kui mõelda koduste seadmete ja masinate peale, siis selliseid, mis ilma programmeerijate kirjutatud programmideta õigesti töötaksid, on üpris vähe. Programmidest võime mõelda ka laiemalt - näiteks käikudega jalgrattale kontekstis. Ratta käikudest võib mõelda kui programmijuppidest, mis täidavad oma ülesande, kui sa need välja kutsud (käiku vahetad). Käike vahetades valid sa programmi, millega ratas edasipidi sõitma hakkab. Seega oled sa põhimõtteliselt tegelenud programmeerimisega.

Kui aga mõelda programmeerimise peale veidike keerulisemast ja tõetruumast vaatenurgast, võime vaadelda näiteks pesumasinat. Tänapäevased pesumasinad on väga võimsad ning viimaste tehnoloogiate järgi loodud pesumasinad on programmeeritud nii, et nad saavad ise aru, mis materjalist ja kui mustad meie riided on. Selle põhjal arvutab masin programmipõhiselt vajalikud temperatuurid ja pesuainete kogused ning oma pesu näeme uuesti alles siis, kui see on puhas ja triigitud. Enamikes kodudes meil kahjuks või õnneks selliseid masinaid ilmselt ei leidu, kuid kui mõelda harjumuspärase pesu pesemise viisi peale pesumasinas, oleme me kõik

programmeerijad. Me valime vastava programmi, millega pesu pesta, määrame temperatuuri, loputuskordade arvu ning tsentrifuugi. Iga selle valiku tegemine on programmeerimine, kuna valikutest valime välja meile sobilikud ning välistame mittevajaliku.

Üks kindlasti tüütumaid tegevusi kodus on koristamine. See on tegevus, mida võiks lõputult teha. Kuid ka sellele on meie kaasaeg mõelnud ning siinkohal mitte koduabilise seisukohast, keda me võiksime ka programmeerida, andes sisendiks raha, vaid jutt on kodumasinatest, mis üpris keerukate programmijuppide abiga suudavad teha ära suure osa meie üksluisemaist tegevusest. Kui me kujutame ette tavalist koristuspäeva, meenub ikka tolmu võtmine, asjade korrastamine ja õigetesse kohtadesse panemine, tolmuimejaga põrandate imemine, põrandapesu ... Aga kui meil oleks võimalik umbes 70% koristustööst korraldada nii, et me ei peaks seepärast otseselt ise muretsema?

Tänapäevaseid tolmuimejaid leidub ka selliseid, mis töötavad aku pealt ja teevad kogu töö ise. Vastavalt sellele, kuidas kasutaja on masina programmeerinud, käib ta intervalliti kõik põrandad üle. Imepäraseks teeb selle masina aga programm, mis on õpetanud teda aru saama, kus on astakud, millest ta võiks alla kukkuda, asjad, mille otsa ta võiks põrgata. Nendest ta hoiab "teadlikult" eemale. Kui ta aga mõistab, et tema aku hakkab tühjaks saama, on ta piisavalt mõistlik, et ise laadima minna. Tolmuimejate programmeerimisest ja sellest, mida on võimalik programmidega teha, on näha järgmiselt videolt (minutil 8:30).

https://www.youtube.com/watch?v=OdjPSxWxWKg&feature=youtu.be&t=8m30s

Samasuguse põhimõttega on olemas ka aknapesumasinad, põrandapesumasinad ning muruniidukid.

Kui tulla tagasi nutitelefonide juurde või ka tahvelarvutite juurde, siis võimaluse korral on võimalik muuta suur osa oma elust programmeerituks. Võimalused, mida tehnika viimane sõna meile täna pakub, on pea uskumatud. Päeva sellises elus kujutab hästi järgmine video.

https://www.youtube.com/watch?v=6239lNmz2hY

Uskumatu, kas pole?

Teisest küljest on igal heal omad vead ehk tuleb olla ikkagi igati ettevaatlik. Lubades tehnoloogial kontrollida nii suurt osa meie elust võib osutuda ka ohtlikuks. Pidevalt tuleb meeles pidada, et turvariskid on suured. Ikka leidub geniaalseid inimesi, kes oskavad läbi murda meie keerukatest süsteemidest ja muuta meie nõrkused enda tugevusteks.

Kokkuvõtvalt võime öelda, et meie elu sisaldab programmeerimist väga palju suuremal osal kui me seda esialgu ise ette kujutasime. Paljud meie tegevused ja meid ümbritsevad masinad on seotud programmeerimisega, mis on lihtsustanud meie elu väga suurel määral ja teevad seda ka edaspidi. Ehk innustab see meid suuremal määral tähelepanu pöörama programmeerimisele ja sellealastele oskustele. Olles ise teadlikumad, on kindlasti ka meie maailm põnevam.

Järgmine ülesanne jääb ootama seda aega, kui me oskame Pythonis valikuid programmeerida. See aeg tuleb varsti!

Luua väike programm pesumasina kasutamiseks. Programm küsib kasutaja käest, millist pesu soovitakse pesta ning programm vastab vastavalt, kui kaua on masin töös ning mis kraadide juures tuleks pesu pesta. Villane pesu 1 tund alla 30˚ juures, õrn pesu 1,5 tundi 40˚ juures ning tavaline

pesu 1,5 tundi 40˚ - 80˚ juures. Andmeid võib ka muuta või ise režiime juurde mõelda (loputuskorrad, tsentrifuugimine/väänamine).

Materjali koostasid Airika Veinjärv ja Merili Raudmäe. Kohendatud kursuse korraldajate poolt.

2.4 Kontrollülesanne II

Käpalisi (e orhideelisi) on maailmas üle 20 000 liigi, Eestis on neist esindatud 35. Mõned on nähtaval ainult lühikest aega - nii ka selle ülesande kangelane. Ülesanne on pühendatud lehitule pisikäpale - 2019. aasta orhideele.

Koostada programm, mille

- 1. real luuakse muutuja nimega aasta ning antakse sellele väärtuseks 2019 (arvuna);
- 2. real luuakse muutuja nimega orhidee ning antakse sellele väärtuseks "lehitu pisikäpp" (sõnena);
- 3. real luuakse muutuja nimega lause_keskosa ning antakse sellele väärtuseks ". aasta orhidee on " (sõnena);
- 4. real luuakse muutuja nimega lause, mille väärtuse saamiseks ühendatakse üheks sõnaks muutujad aasta, lause_keskosa ja orhidee (vajadusel tuleb kasutada funktsiooni, mis teisendab arvu sõneks);
- 5. real väljastatakse muutuja lause väärtus ekraanile.

Kuigi tegelikult pannakse lause lõppu punkt, siis siin ärge pange. (Automaatkontroll isegi annab punkti või mõne muu üleliigse osa eest veateate.)

Näide programmi tööst:

```
>>> %Run yl2.py
  2019. aasta orhidee on lehitu pisikäpp
>>>
```

Aasta orhidee kohta võib lugeda siin.

Kontrollülesannete lahendused esitatakse Moodle'is.

Kui olete juba hulk aega proovinud ülesannet iseseisvalt lahendada ja see ikka ei õnnestu, siis võib-olla saate abi murelahendajalt. Püütud on tüüpilisemaid probleemseid kohti selgitada ja anda vihjeid.

Eks seda tuleb ette ka edaspidi, et programm teeb põhimõtteliselt nõutud asja, aga väljastab midagi rohkem või vähem või kuidagi teisiti ei vasta täpselt ülesandele. (Näiteks on lahendus hoopis vingem kui ülesandes nõutud.) Sellisel juhul võib automaatkontroll teie lahenduse valeks lugeda. Kui teile tundub, et automaatkontroll töötab ebakorrektselt, siis palun kirjutage aadressil prog@ut.ee.

2.5 Maalähedane lugu II

“Rasmus, ole hää ja too keldrist neli kolmest ja mõned poolesed ka!”. Mõnel aastal polnud peaaegu ühelgi puul õunu ja teisel aastal jälle “uputas”. Sel aastal oli õunu palju ja mitmel moel püüti neid ka säilitada. Praegu oligi Adal mahla ja moosi tegemine käsil. Mahla pandi tavaliselt ikka kolmeliitristesse purkidesse ja moosi pooleliitristesse. Vanasti oli mahla isegi kümneliitristesse purkidesse pandud, aga need olid rohkem suurte pidude jaoks head. Aga kes see tänapäeval pulmi peab! Ada vargsi ikka lootis, et äkki Rasmus ja Liisu ...

Rasmus hõikas Liisu ka kaasa ja mindigi keldrisse. Keldri uks oli maja otsas. Tegelikult ei olnudki keldrit kohe maja ehitamise ajal kaevatud, vaid see tehti 1968. aasta paiku. Kelder polnud kogu maja all, vaid täitis ainult pool majaalust. Põhimõtteliselt sai keldriukse tabalukuga lukku panna, aga seda ei tehtud kunagi. Küll aga pandi mingitel aegadel lukku väikse aida uks, kus hoiti näiteks tööriistu.

Keldris oli mitu ruumi. Kohe sisse minnes oli pisem ruum, milles olid riiulid, kus hoiti näiteks kõrvitsaid ja suvikõrvitsaid. Üks kord oli Ada isegi arbuusi proovinud kasvatada, aga see ikka meie kliimas ei tulnud välja. Hea oli panna ka korv kurkidega või midagi muud, millega kohe varsti midagi edasi tehti. Edasi oli keldri kõige suurem ruum, kus põrandal hoiti kartuleid, porgandeid, peete ja muid juurvilju. Nende jaoks kindlaid salvesid ei olnud, eraldamiseks pandi lauad vahele. Ühel aastal oli suurem kogus ühte sorti, siis jälle teist. Viimasel ajal oli see punase koorega kartul päris hästi saaki andnud!

Kolmandas ruumis olid riiulid, kus hoiti igasuguseid purke ja pudeleid. Oli siin nii täidetud kui tühje. Eriti viimastel aastatel, kui lihtsam oli poest kõike osta, kippus tootmine ületama tarbimist. Mõnedki hoidised olid siin juba hulk aastaid olnud. Siltide pealt oli näha, millega tegemist ja mis aastal tehtud. Siit ruumist pididki Rasmus ja Liisu purgid võtma. Aga ei läinud see üldse kiiresti, sest keldris oli palju põnevaid asju. Oli siis vanaaegseid pudeleid, mille kork oli traatidega küljes. Oli erinevate aegade limonaadipudeleid, sildid juba tuhmunud, aga ikkagi loetavad.

Aega läks isegi nii palju, et Ada neile keldrisse järele tuli. Nüüd said Liisu ja Rasmus küsida, aga kuna köögis olid potid tulel, siis võeti huvitavamad asjad üles kaasa. Näiteks said noored teada, miks ühel

“Kellukese” limonaadipudelil hoopis leikoplaastril silt “Äädikas” oli. See oli tegelikult õnnelikult lõppenud lugu!

> VAHELEPÕIGE Ada, Liisbeth ja Rasmus ei mõelnud muidugi sellest, et mingite asjade hoidmine, kättesaamine, sildistamine on äärmiselt olulisel kohal ka programmeerimisel. Kuidas otstarbekalt kasutada seda ruumi, millesse on võimalik asju panna? Silt peab võimalikult täpselt vastama sisule! Kuidas toimetada asjadega, mida enam vaja pole? Kuidas hoida asju vaenuliku käe või hammaste eest?

Üles sai viidud ka paar igivana pudelit, mille sisu ilmselt küll enam juua ei kõlvanud. Selleks oli viimane aeg, sest kompostihunniku juures tuli ühe pudeli kork peaaegu ise lahti ja kolmveerand pudeli sisu purskas välja.

Igatahes sai lõpuks ka mahl ja moos purkidesse. Liisu joonistas siltide peale veel õunad ka. Ühel õunal oli veel tükk ära hammustatud. Koos viidi kõik siis jälle keldrisse.

Kordavaid ja lisavaid materjale ja ülesandeid

Siia on kogutud selliste materjalide linke, mis aitavad eelnevaid teadmisi ja oskusi meelde tuletada või annavad uusi. Samuti on siin ülesanne, mida võiks ka proovida teha.

- Aivar Annamaa koostatud programmeerimise õpik (http://progeopik.cs.ut.ee/), mis on kasutusel Tartu Ülikooli aines Programmeerimine (6 EAP). Õpikus on väga palju erinevaid teemasid ja nende käsitlus on mõnevõrra teistsugune kui meie kursusel.
- Muutujate ja omistamise kordav materjal

Julgesti võite linke juurde pakkuda!

Lisaülesanne: Küpsisetort

Küpsisetordi tegemisel laotakse küpsised ristkülikukujulisele kandikule mitmes kihis nii, et igas kihis on sama palju küpsiseid. Küsida kasutajalt, mitu küpsist mahub kandikule laiuses ja mitu pikkuses ning kui mitme kihilist torti ta teha soovib. Seejärel küsida, kui mitu küpsist on ühes pakis. Küsige just sellises järjekorras, esimese kolme suurusena - laiust (küpsistes), pikkust (küpsistes) ja kihtide arvu ning viimase asjana küpsiste arvu pakis.

Lõpuks väljastada, mitu küpsisepakki tuleb sellise tordi tegemiseks osta. NB! Eeldame, et poolikut küpsisepakki osta ei saa.

Selle ülesande lahenduse võib esitada Moodle'is ja saada automaatset tagasisidet, aga kohustuslik see ei ole.

Kui kasutada programmeerimiskeskkonda Thonny, siis ootab *input* kasutaja sisestust alumises aknas (*Shell*).

Näide programmi tööst:

```
>>> %Run tort.py

  Mitu küpsist on tordi laius? 6
  Aga pikkus? 5
  Mitme korruselist torti soovid? 3
  Mitu küpsist on ühes pakis? 14
  Vaja läheb 90 küpsist
  seega tuleks osta 7 pakk(i) küpsiseid

>>>
```

Vihje

Küsimused, mida võiks endalt küsida:

1. Kuidas arvutada küpsiste arv, kui on teada tordi pikkus, laius ja kõrgus?
2. Kuidas leida vajalike küpsisepakkide arv? Mitu pakki on vaja osta, kui arvutused näitavad, et vaja on 7,2 paki jagu küpsiseid?

Teil võib vaja minna funktsiooni `ceil()`. Kuna seda funktsiooni väga tihti vaja ei lähe, siis ei ole see n-ö standardvarustuses. Selle funktsiooni kasutamiseks tuleb programmiteksti algusesse lisada rida: `from math import ceil`

Mida funktsioon `ceil` teeb? Uurige järgmist näidet selle kasutamisest.

```
from math import ceil

arv1 = 1.95059
arv2 = 3.2
arv3 = 8

arv1_ceil = ceil(arv1)
arv2_ceil = ceil(arv2)
arv3_ceil = ceil(arv3)

print("ceil(arv1) = " + str(arv1_ceil))
print("ceil(arv2) = " + str(arv2_ceil))
print("ceil(arv3) = " + str(arv3_ceil))
```

käivitamisel väljastatakse

```
ceil(arv1) = 2
ceil(arv2) = 4
ceil(arv3) = 8
```

Samuti võite `ceil`-funktsiooni idee leida Pythoni ametlikust ingliskeelsest dokumentatsioonist: https://docs.python.org/3.6/library/math.html.

Murelahendaja