Töölehed Pythoni materjali juurde

**Sisu**

[Python. Avaldised 3](#_Toc358218332)

[Pythoni laused. Moodulid 4](#_Toc358218333)

[Python. Ideaal 5](#_Toc358218334)

[Python. Funktsioonid 6](#_Toc358218335)

[Python. Kordused 7](#_Toc358218336)

[Python. Rehmaatika 8](#_Toc358218337)

[Python. Tekstiavaldised 9](#_Toc358218338)

[Python. Algoritme loenditega 10](#_Toc358218339)

[Python. Sõnastik 11](#_Toc358218340)

[Pyhton. Edetabel 12](#_Toc358218341)

[Ristkülik ja ring Pythonis 13](#_Toc358218342)

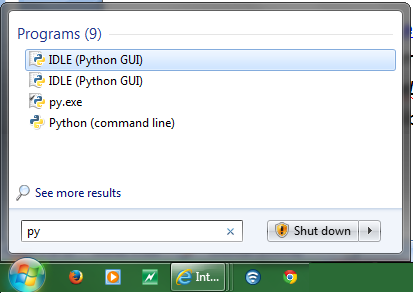
[Python. Funktsiooni parameetrid 14](#_Toc358218343)

[Python. Funktsioonide uurimine 15](#_Toc358218344)

[Pythoni funktsioonid (meetodid) moodulis turtle.py 16](#_Toc358218345)

# Python. Avaldised

Vt. ka <http://rlpa.ttu.ee/python/Python_sl.pdf> ja õpik (<http://rlpa.ttu.ee/python/Python.pdf>) lk 9.

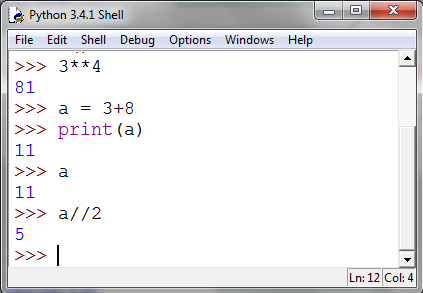
Programmide sisestamiseks, redigeerimiseks ja täitmiseks kasutame siin interaktiivset kasutajaliidest IDLE (*Interactive DeveLopment Environment*), mille haldur kuulub Pythoni põhikomplekti.

IDLE käivitamiseks võib kasutada Windowsi Start-menüüst korraldust

Start > Programs > Python 3.x > IDLE (Python GUI)

või tippida Start-menüü otsivälja py... ja valida IDLE (Python GUI).

Kuvatakse interaktiivse kasutajaliidese aken **Python Shell**.

 Shelli aknasse (käsuaknasse) väljastatakse tulemused ja teated.

Akent võib kasutada interaktiivseteks arvutusteks nn kalkulaatori režiimis. Käsureale, mille alguses on sümbolid >>>, saab sisestada väärtusi, avaldisi, lauseid ja programmi fragmente.

Selline töörežiim võimaldab mitmesuguste operatiivsete arvutuste tegemist ning korralduste andmist näiteks programmide silumiseks, aga ka keele­vahenditega tutvumiseks.

Märkandmete põhitüüpideks ehk -klassideks on:

stringid (tekstid) – str, täisarvud – int, ujupunktarvud (reaalarvud) – float, tõeväärtused – bool.

Arvkonstandid ja loogikakonstandid (True, False) esitatakse programmis ilma piirajateta.

Koma asemel on arvu murdosa eraldajaks punkt.

Tekst- ehk stringkonstandid paigutakse nn piirajate jutumärkide (“tekst“) või ülakomade (’tekst’) vahele. Piirajad konstandi väärtuse hulka ei kuulu. Jutumärgid ja ülakomad on piirajatena samaväärsed ja peavad ühe konstandi jaoks olema samad; neid võib kasutada üksteise sees, kui üks neist peaks sisalduma tekstikonstandis.

Kasutatakse ka kolmekordsete piirajate """ või ''' vahel asuvaid stringe. Väärtus võib paikneda mitmel real ja täita näiteks ka pikema kommentaari rolli.

Erisümbolid tekstis: **\n** (reavahetus), **\t** (tabulaator).

Tehetes ehk operatsioonides kasutatakse järgmisi operaatoreid (tehtemärke):

* aritmeetikatehted:

\*\* (astendamine)

\* (korrutamine)

/ (jagamine)

// (täisarvuline jagamine)

% (jääk)

+ (liitmine)

- (lahutamine)

* stringitehted:

**+** tekstide ühendamine (sidurdamine)

**\*** teksti kordamine (operandideks tekst ja täisarv)

* võrdlustehted: **==** , **!=** , **<** , **<=** , **>** , **>=**
* loogikatehted: **not**, **and**, **or**

Aritmeetika- ja loogikatehted on siin prioriteetide kahanemise järjekorras.

Ülesanne: Proovida käsuaknas (Shelli aknas) avaldiste koostamist ja omistamislauseid.

Avaldiste näiteid proovimiseks:

2 + 3 \* 6

(2 + 3) \* 6 / 3

3 \*\* 268 / 5

68 // 5

68 % 5 #jääk"auto" + "123"

"123" \* 5

2 \* "midagi"

3 == 4

3 == "3"

4 < 6

2<4 < 6

2<4 and 4<6

**Muutuja** on mäluväli ühe väärtuse säilitamiseks.

Muutuja omadused on: **nimi**, **väärtus**, **tüüp**, **skoop** (mõjupiirkond)

**Nimi** on tähtede, numbrite ja allkriipsude (\_) jada, mis peab algama tähega (või allkriipsuga).

Python on tõstutundlik – nimedes eristatakse suur- ja väiketahti.

Muutujaid ei kirjeldata, tüüp määratakse omistatud väärtuse järgi.

**Omistamismärgiks** (operaatoriks) on Pythonis **=**

Omistamislausete näiteid proovimiseks

a = 2

b = 4 \*\* 3

c = a + b

t = "abc"

u = "123"

koos = u + t

L = "auto"< "maja"

K = 'päike'>="kuu"

koos = L and K

**Täiendatud omistamine**, operaator omistamismärgi ees näitab muutust sama muutuja väärtuses:

**n += 1** (sama, mis n = n+1)

**a \*= 3** (sama, mis a = a\*3)

**Mitmene omistamine**:

mitu = summa = nr = 0 (sobib algväärtustamiseks)

a, b, c = 2, 2.5, len(tekst)\*2 (muutujate arv omistamislause vasakus pooles ja väärtuste arv lause paremas pooles peavad olema vastavuses)

**Mõned Pythoni funktsioonid:**

Nurksulud [xxxx] tähendavad, et vastav element ei ole kohustuslik, punktiir – element võib korduda.

**type(**muutuja**)** – näitab väärtuse või muutuja tüüpi.

Teisendusfunktsioonid **int()**, **float()**, **str()** muudavad tüüpi.

**input**([selgitav tekst]) – kuvab teate ja loeb väärtuse (tagastatava väärtuse tüübiks tekst).

**print**([*objekt*, *...*], *\**, *sep=' '*, *end='\n'*, *file=sys.stdout*) – väärtuste (tekstide) väljastamine ekraanile/faili.

dir() – kasutusel olevatest muutujate, funktsioonide jm nimed

dir(objekt) – objekti omadused ja meetodid.

Proovida omistamislauseid (täisarv, reaalarv, tekst), vaadata tüüpi.

Muuta tekst arvuks ja vastupidi: float(tekst), int(tekst), str(arv).

# Pythoni laused. Moodulid

Suurem osa funktsioone paikneb moodulites (eraldi failides), mis tuleb põhimooduliga siduda:

**import mooduli\_nimi** - mooduli funktsioone ja konstante saab kasutada kujul mooduli\_nimi.fn()

**from mooduli\_nimi import \*** - saab kasutada kõiki mooduli funktsioone ilma mooduli nime lisamata

**from mooduli\_nimi import fn\_nimi** - moodulist saab kasutada ainult funktsiooni fn\_nimi

Vt. ka õpik (<http://rlpa.ttu.ee/python/Python.pdf>) lk. 11-13.

Põhiosa matemaatikafunktsioone asub moodulis **math**.

**Ülesanne: Kopeerida programm Pythoni programmiredaktorisse ja proovida käivitada.**

Ristküliku külgede järgi leitakse sama suure pindalaga ringi läbimõõt ning kujundite ümbermõõtude suhe. Pythonis väljastatakse tulemused print-käsu (print-funktsiooni) abil käsuaknasse (*shelli* aknasse).

Funktsiooni **input** (võimaldab programmil küsida andmeid kasutajalt) väljund on tekst, seepärast tuleb arvutuste tegemiseks teisendada väärtus arvuks.

NB! Arvus on täis- ja murdosa eraldajaks punkt. Koma kasutatakse eraldajana loetelus.

|  |  |
| --- | --- |
| **import** math  **print** ("Võrdse pindalaga ristkülik ja ring")  a = **float** (**input** ("Anna laius => ")) *# algandmete* *sisestamine*  b = **float** (**input** ("Anna kõrgus => ")) *#*  S = a \* b  P = 2 \* (a + b) *# pindala ja ümbermõõt*  d = **math.sqrt**(4 \* S / **math.pi**) *# läbimõõt*  suhe = P / (**math.pi** \* d)  **print** ("pindala=", S) *# tulemuste*  **print** ("läbimõõt=", **round**(d, 3)) *# väljastamine*  **print** ("suhe=", **round**(suhe, 2)) *# Shelli aknasse* | rist_ring_arvuta |

**Ülesanne:**

**Eelmises näites asendada *import math* käsuga *from math import \** ja teha vastavad muudatused tekstis.**

Süntaksivigade puhul näitab programmiredaktor probleemset kohta.

Programmi täitmisel ilmnevate vigade kohta antakse info (veateade, rea nr, lause tekst) *shelli* aknas.

Ülesanne:

Koostada Pythoni programm kõrvaloleva kujundi ümbermõõdu ja pindala arvutamiseks. Mõõtude väärtused küsida kasutajalt.

Tulemusi saab kontrollida alloleva tabeli abil.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **jrk** | **h1** | **h2** | **a** | **b** | **c** | **r** | **P** | **S** |
| *1* | 3 | 5 | 3 | 1 | 2,8 | 1 | 22,582 | 10,558 |
| *2* | 5 | 3 | 4 | 2 | 4 | 1,5 | 26,923 | 10,931 |
| *3* | 8 | 4 | 5 | 1,8 | 3 | 2 | 36,317 | 32,034 |
| *4* | 10 | 7 | 6 | 0,7 | 2 | 2,8 | 45,828 | 54,970 |
| *5* | 12 | 5 | 7 | 2,5 | 5 | 4,2 | 53,001 | 33,582 |

# Ristkülik ja ring Pythonis

Koostada rakendus, mis leiab etteantud ristküliku külgede abil sama suure pindalaga ringi raadiuse ja joonestab mõlemad kujundid. Leida ka kujundite ümbermõõtude suhe. Joonistamisel tuleks kasutada sobivat mastaabitegurit.

Antud on ristküliku küljed **a** ja **b** (küsida kasutajalt, vt fn turtle.numinput()).

Leida tuleb ristküliku pindala **S**, mis on ühtlasi ka ringi pindala, ja ringi raadius **r**.

Tuletada meelde valemid ristküliku ja ringi pindalade leidmiseks. Kuidas leida ringi pindala järgi ringi raadius?

Pythoni kooseisu kuulub moodul turtle, mis sisaldab nn „kilpkonnagraafika“ vahendeid. Joonestamise operatsioone täidab spetsiaalne graafikaobjekt – **turtle** (kilpkonn) – Scratchi spraidi analoog. Kilpkonni (objekte) võib olla mitu ja neil võivad olla erinevad kujud. Vaikimisi on kilpkonni üks ja sellel on nooletaoline kuju. Kilpkonnaga on seotud pliiats ja objektil on suur hulk meetodeid, mille abil saab seda liigutada, pöörata, muuta pliiatsi suurust, värvust jms.

Mõned funktsioonid(meetodid) Pythoni moodulist turtle.py:

**Objekti liikumine:**

[**forward()**](http://docs.python.org/py3k/library/turtle.html#turtle.forward) | [**fd()**](http://docs.python.org/py3k/library/turtle.html#turtle.fd) liigu edasi

[**backward()**](http://docs.python.org/py3k/library/turtle.html#turtle.backward) | [**bk()**](http://docs.python.org/py3k/library/turtle.html#turtle.bk) | [**back()**](http://docs.python.org/py3k/library/turtle.html#turtle.back) liigu tagasi

[**right()**](http://docs.python.org/py3k/library/turtle.html#turtle.right) | [**rt()**](http://docs.python.org/py3k/library/turtle.html#turtle.rt) pööra paremale

[**left()**](http://docs.python.org/py3k/library/turtle.html#turtle.left) | [**lt()**](http://docs.python.org/py3k/library/turtle.html#turtle.lt) pööra vasakule

[**goto()**](http://docs.python.org/py3k/library/turtle.html#turtle.goto) | [**setpos()**](http://docs.python.org/py3k/library/turtle.html#turtle.setpos) mine(x,y)

[**setx()**](http://docs.python.org/py3k/library/turtle.html#turtle.setx), [**sety()**](http://docs.python.org/py3k/library/turtle.html#turtle.sety) määra x, määra y

[**setheading()**](http://docs.python.org/py3k/library/turtle.html#turtle.setheading) | [**seth()**](http://docs.python.org/py3k/library/turtle.html#turtle.seth) määra nurk

[**home()**](http://docs.python.org/py3k/library/turtle.html#turtle.home) mine(0, 0), nurk 0

[**circle()**](http://docs.python.org/py3k/library/turtle.html#turtle.circle) ring(raadius, kaar=360, hnurk)

[**dot()**](http://docs.python.org/py3k/library/turtle.html#turtle.dot) punkt; ette anda diameeter, värv

[**undo()**](http://docs.python.org/py3k/library/turtle.html#turtle.undo) eelmise tegevuse tühistamine

[**speed()**](http://docs.python.org/py3k/library/turtle.html#turtle.speed) animatsiooni kiirus 1..10 või 0

**Objekti omadused:**

[**position()**](http://docs.python.org/py3k/library/turtle.html#turtle.position) | [**pos()**](http://docs.python.org/py3k/library/turtle.html#turtle.pos) asukoht

[**towards()**](http://docs.python.org/py3k/library/turtle.html#turtle.towards) suund (etteantud punkti poole)

[**xcor()**](http://docs.python.org/py3k/library/turtle.html#turtle.xcor), [**ycor()**](http://docs.python.org/py3k/library/turtle.html#turtle.ycor) x-asukoht, y-asukoht

[**heading()**](http://docs.python.org/py3k/library/turtle.html#turtle.heading) nurk

[**distance()**](http://docs.python.org/py3k/library/turtle.html#turtle.distance) kaugus punktist või teisest obj.st

[**textinput()**](http://docs.python.org/py3k/library/turtle.html#turtle.textinput) väärtuse lugemine, tulemi tüüp str

[**numinput()**](http://docs.python.org/py3k/library/turtle.html#turtle.numinput) väärtuse lugemine, tulemi tüüp float

**Pliiatsi seaded:**

[**pendown()**](http://docs.python.org/py3k/library/turtle.html#turtle.pendown) | [**pd()**](http://docs.python.org/py3k/library/turtle.html#turtle.pd) | [**down()**](http://docs.python.org/py3k/library/turtle.html#turtle.down) pliiats alla

[**penup()**](http://docs.python.org/py3k/library/turtle.html#turtle.penup) | [**pu()**](http://docs.python.org/py3k/library/turtle.html#turtle.pu) | [**up()**](http://docs.python.org/py3k/library/turtle.html#turtle.up) pliiats üles

[**pensize()**](http://docs.python.org/py3k/library/turtle.html#turtle.pensize) | [**width()**](http://docs.python.org/py3k/library/turtle.html#turtle.width) pliiatsi suurus

[**isdown()**](http://docs.python.org/py3k/library/turtle.html#turtle.isdown) kas pliiats all

[**color()**](http://docs.python.org/py3k/library/turtle.html#turtle.color) värv (joon, taust)

[**pencolor()**](http://docs.python.org/py3k/library/turtle.html#turtle.pencolor) joone värv

[**fillcolor()**](http://docs.python.org/py3k/library/turtle.html#turtle.fillcolor) tausta värv

[**filling()**](http://docs.python.org/py3k/library/turtle.html#turtle.filling) kas toimub kujundi täitmine

[**begin\_fill()**](http://docs.python.org/py3k/library/turtle.html#turtle.begin_fill) täitmise algus

[**end\_fill()**](http://docs.python.org/py3k/library/turtle.html#turtle.end_fill) täitmise lõpp

[**reset()**](http://docs.python.org/py3k/library/turtle.html#turtle.reset) joonise kustutamine, algseaded

[**clear()**](http://docs.python.org/py3k/library/turtle.html#turtle.clear) joonise kustutamine

[**write()**](http://docs.python.org/py3k/library/turtle.html#turtle.write) objekt kirjutab teksti

**Graafikaakna seaded**

[**bye()**](http://docs.python.org/py3k/library/turtle.html#turtle.bye) graafikaakna sulgemine

[**exitonclick()**](http://docs.python.org/py3k/library/turtle.html#turtle.exitonclick) akna sulgemine hiireklõpsuga

[**setup()**](http://docs.python.org/py3k/library/turtle.html#turtle.setup) akna suurus (laius, kõrgus) ja asukoht

[**title()**](http://docs.python.org/py3k/library/turtle.html#turtle.title) tekst akna päises

Kui programmis on joonistamiskäske, kuvatakse automaatselt graafikaaken Python Turtle Graphics.

Programmis saab määrata akna suuruse. Kui seda ei tehta, valib süsteem mõõtmed ise, arvestades arvuti

ekraani suurust.

Mõõtühikuks on piksel. Koordinaatsüsteemi nullpunkt on akna keskel.

# Python. Valikulaused

Valikulaused (If-laused) võimaldavad tegevuste täitmist või vahelejätmist sõltuvalt etteantud tingimustest.

Tingimused esitatakse võrdluste või loogikaavaldiste abil.

|  |  |
| --- | --- |
| if (*tingimus)*:  *if\_laused*  [ elif (*tingimus):*  *elif\_laused* ] …  [ else :  ***else\_laused* ]** | * Lause täitmisel kontrollitakse kõigepealt tingimust if‑osalauses, kui see on tõene, täidetakse if-ploki laused, kõik ülejäänu kuni valikulause lõpuni jääb vahele * Vastupidisel juhul kontrollitakse järjest tingimusi elif-lausetes (kui neid on) ning kui leitakse esimene tõene, täidetakse selle ploki laused ning jätkatakse if-lausele järgnevast lausest * Kui ükski tingimus pole tõene, täidetakse else-ploki laused (kui need on)   Pythonis ei ole spetsiaalseid piirajaid! Võrdse taandega laused moodustavad ploki. |

|  |  |
| --- | --- |
| arv = **int**(**input** ("sisesta arv: "))  **if** arv > 0**:**  **print** ("see on positiivne arv")  **elif** arv < 0**:**  **print** ("see on positiivne arv")  **else:**  **print** ("see arv on ju null!") | * Koostada või kopeerida kõrvalolev skript Pythoni tekstieditori aknasse (ctrl+n – uus tekstiaken) * salvestada fail (laiend .py) * käivitada (F5 või Run/Run Module)   NB! kopeerimisel võivad tekkida probleemid taanetega |

**Proovida seda ja ka järgnevat skripti käivitada Pythoni programmide visualiseerimisvahendiga** [**http://pythontutor.com/**](http://pythontutor.com/)

|  |  |
| --- | --- |
| print ("Tere! Olen Python!") # teate kuvamine  nimi = input("Mis on Sinu nimi => ") # väärtuse lugemine  print (nimi, '? Oi kui ilus nimi!!!')  teade = "! Kas leian Sinu keha indeksi? 0 – ei, 1 – jah => "  vastus = input (nimi + teade)  **if** int (vastus) == 0 : # koolon peab kindlasti olema  print ("Kahju! See on jube kasulik info!")  **else** :  pikkus = int(input(nimi + "! Sinu pikkus (cm) => "))  mass = float(input("Anna kaal ka (kg) => "))  indeks = mass /(0.01 \* pikkus)\*\*2  print (nimi,"! Sinu keha indeks on:", round(indeks, 1))  print () # tühi rida (reavahetus)  print ('Kohtumiseni, ' + nimi + '! Igavesti Sinu, Python!') |  |

# Python. Ideaal

Koostada Pythonis rakendus inimese keha omaduste leidmiseks.

Inimese soo (mees/naine), vanuse *t* (aastates), pikkuse *l* (cm) ja kaalu *m* (kg) alusel tuleb alltoodud valemite abil arvutada ideaalne mass *mid* (kg), rasvaprotsent *r*, kehamassi indeks *kind*, tihedus *ρ* (kg/m3), ruumala *V* (dm3), pindala *S* (m2) ning anda kehamassi indeksi alusel sõnaline *hinnang*.

Algandmed küsida kasutajalt (sisestada klaviatuurilt), tulemus näidata ekraanil (shelli aknas).

Saadud tulemust võib võrrelda alloleva tabeli andmetega.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sugu** | N | M | M | N |  |
| **Vanus** | 22 | 20 | 60 | 37 |  |
| **Pikkus** | 165 | 180 | 175 | 160 | cm |
| **Kaal** | 48 | 70 | 90 | 78 | kg |
| **Ideaalne mass** | 55,6 | 72,5 | 78,8 | 55,6 | kg |
| **Rasvaprotsent** | 6,2 | 11,4 | 27,5 | 50,8 | % |
| **Kehamassiindeks** | 17,6 | 21,6 | 29,4 | 30,5 |  |
| **Tihedus** | 1086,9 | 1076,0 | 1042,3 | 993,4 | kg/m3 |
| **Ruumala** | 44,2 | 65,1 | 86,4 | 78,5 | dm3 |
| **Pindala** | 1,5 | 1,9 | 2,1 | 1,9 | m2 |
| **Hinnang** | kõhn | normaalne | ülekaalus | ei või olla! |  |

# Python. Funktsioonid

Osa rakendusest on otstarbekas vormistada eraldi protseduuri(de)na, mida on võimalik kasutada erinevate algandmetega.

Pythonis algab funktsiooni päis võtmesõnaga **def**, järgneb **nimi** ja sulgudes parameetrite loetelu (tühjad sulud peavad olema ka siis, kui parameetrid puuduvad), lõpus **koolon**. Parameetrid on muutujad, mis saavad väärtuse funktsiooni poole pöördumisel.

Kõik funktsiooni laused paiknevad päise suhtes taandega.

Lause **return *avaldis(ed)*** lõpetab funktsiooni töö ja tagastab väärtuse(d).

|  |  |  |
| --- | --- | --- |
| **def nimi** ([***parameetrid***])**:**  ***laused***  [ **return** *avaldis* ]  … | funktsiooni näide  **def** summa(a, b)**:**  **return** a+b | kasutamine  s = **summa**(4, 6) |
|  | **def** summa\_korrutis(a, b)**:**  S = a+b  K = a\*b  **return** S, K | arv1 = 5  arv2 = int(input(“Sisesta arv: “))  sum, kor = **summa\_korrutis**(arv1, arv2) |

Pöördumine funktsiooni poole: [***muutuja*** = ] **nimi** ([***argumendid***]) , ***muutuja*** on vajalik, kui funktsioon tagastab väärtuse. Argumentide arv, tüüp ja järjekord peavad vastama parameetritele.

Koostada näidete järgi paar funktsiooni ja proovida neid kasutada nii käsuaknas kui skriptis.

Koostada alltoodud valemite järgi kaks funktsiooni:

* rasvaprotsendi leidmine etteantud soo (mees/naine), vanuse *t* (aastates), pikkuse *l* (cm) ja kaalu *m* (kg) alusel
* hinnangu andmine pikkuse *l* (cm) ja kaalu *m* (kg) alusel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sugu** | N | M | M | N |  |
| **Vanus** | 22 | 20 | 60 | 37 |  |
| **Pikkus** | 165 | 180 | 175 | 160 | cm |
| **Kaal** | 48 | 70 | 90 | 78 | kg |
| **Rasvaprotsent** | 6,2 | 11,4 | 27,5 | 50,8 | % |
| **Hinnang** | kõhn | normaalne | ülekaalus | ei või olla! |  |

# Python. Kordused

Korduste kirjeldamiseks on Pythonis kaks liitlauset: **while**-lause ja **for**-lause.   
Korduse täitmise katkestab käsk **break**, käsk **continue** jätkab korduse täitmist järgmisest korduse sammust.

**while-lause:** lauseid täidetakse, kui *tingimus* on tõene

Lõpmatu korduse tekitamiseks kirjutada alati tõene tingimus – **while** True:

|  |  |  |
| --- | --- | --- |
| **while** *tingimus* :  [laused\_1]  [**if** *tingimus1***:** **break**]  [laused\_2] | **kordus kui** *tingimus*  *laused*  [**kui** *tingimus1* **siis** **katkesta**]  *laused* |  |

Tutvuda allolevate korduslausetega. Proovida Pythonis.

|  |  |
| --- | --- |
| k = int(input(“Stardiloendus algab arvuga: “))  print ("Valmis olla!")  **while** k > 0 **:**  print (k)  k = k - 1  print ("Start!") | **def** *Fy*(x) :  **return** 3\*math.sin(2\*x + 1)  **def** *tab\_2*(a, b, h) :  x = a  **while** x <= b + h/2 **:**  **print** (x,"\t", **round**(*Fy*(x), 5))  x = x + h  *tab\_2*(-5, 5, 1) |

**for-lause:**

|  |  |  |
| --- | --- | --- |
| **for** element **in**  **jada** **:**  [laused]  [**if** **tingimus** : **break**]  [laused] | **kordus** iga *element*  *jada*-s  *tegevused*  **kui** *tingimus* **siis** **katkesta**  *tegevused* |  |

***Jada*** võib esitada väärtuste või objektide loeteluna, **loendina**, **range** funktsiooni abil jm.

Lauseid täidetakse jada iga elemendi korral.

Funktsiooniga **range** saab määrata täisarvude jada: **range**([*algus*,] *lõpp* [, *samm*])

vaikimisi: *algus* = 0, *samm* = 1. **NB!** jada viimane väärtus on alati väiksem kui *lõpp.*

Tutvuda järgmiste korduslausetega. Proovida Pythonis.

Shelli aknas peaks print-lausete tulemustena tekkima allolevad arvujadad. "\t" - tabulaator

|  |  |  |
| --- | --- | --- |
| **for** x **in** 7, -3, 2.1, 13, -5 **:**  **print** (x, "**\**t", 2 \* x + 3) | **for** x **in** **range** (5) **:**  **print** (x, "\t", 2 \* x + 3) | **for** x **in range** (1, 11, 2) **:**  **print** (x, "\t", 2 \* x + 3 |
| 7 17  -3 -3  2.1 7.2  13 29  -5 -7 | 0 3  1 5  2 7  3 9  4 11 | 1 5  3 9  5 13  7 17  9 21 |

|  |  |
| --- | --- |
| **for** x **in** (7, -3, 2.1, 13, -5):  **print** (x, "\t", 2 \* x + 3)  7 17  -3 -3  2.1 7.2  13 29  -5 -7 | **def** *Fy*(x) :  **return** 3\*math.sin(2\*x + 1)  **def** *tab\_1*(a, b, n) :  h = (b - a) / n  **for** i **in** **range**(n + 1) :  x = a + i \* h  **print** (x,"\t", round(*Fy*(x), 5))  ***tab\_1***(-5, 5, 10) |

# Lihtsaid ülesandeid kordus- ja valikulausetega

|  |  |
| --- | --- |
| Siin on toodud skript kasutajalt küsitud arvude töötlemiseks. Eraldi leitakse positiivsete ja negatiivsete arvude keskmised, selleks arvutatakse jooksvalt sobivate arvude summad ja nende arv.  Ülesanne:  kopeerida skript | # positiivsete ja negatiivsete keskmine  n = **input**("Mitu arvu ")  SP = np = SN = nn = 0  **for** k **in** **range**(n)**:** # for-lause päis  a = **float** (**input**("anna arv "))  **if** a > 0 **:** # if-lause päis  SP = SP + a  np += 1 # np = np + 1  **else :** # else-lause päis  SN = SN + a  nn += 1  **if** np != 0 **:**  **print** ("Positiivsete keskmine = ", SP / np)  **if** nn != 0 **: print** ("Negatiivsete keskmine = ", SN / nn)  # võib ka nii (samal real) |

# Python. Rehmaatika

Koostada etteantud algoritmi järgi rakendus, mis võimaldab harjutada liitmist ja lahutamist.

Programm kuvab etteantud arvu tehteid, annab punkte õigete vastuse eest ja mõõdab aega. Kasutaja peab võimalikult kiiresti sisestama vastused.

Arvud ja tehe tekitakse juhuarvude abil. NB! Peavad olema ainult positiivsed arvud (ka vastus).

Kasutaja valib taseme, millest sõltub tehetes kasutatavate arvude suurus ja punktide arv õige vastuse eest.

Etteantud tehete arvu täitumisel kuvatakse testile kulunud aeg.

***tase*** – tehete raskusaste

***mini*** ja ***maxi*** - arvude väikseim ja suurim väärtus, näiteks: **1.** tase: **1...10**, **2**. tase: **10...20**, **3.** tase: **20...50**

***a*** ja ***b*** - juhuarvud vahemikus *mini...maxi,* ***c***= a + b

***tehe*** - juhuarv: **1** - liitmine, **2** - lahutamine

***tulem*** - ***c*** kui liitmine, ***a*** - kui lahutamine

***vastus*** - kasutaja vastus

***punkte*** - punktide jooksev ja lõplik summa

***p*** - punkte ühe tehte eest, sõltub tasemest näiteks: **1.** tase - p = **2**, **2.** tase p = **5**; **3.** tase p = **10**

***algaeg*** - alguse aeg, ***aeg*** - testi aeg

Moodulis **time** on funktsioonid **clock**() – jooksev aeg (taimer) ja **sleep**(pp) – paus pp sekundit.

Moodulis **random** on funktsioon **randint**(a,b) - juhuslik täisarv etteantud vahemikus.

Muutujate **min**, **max** ja **p** väärtuse määramine olenevalt tasemest võiks olla eraldi funktsioon (tagastab kolm väärtust).

|  |  |
| --- | --- |
| Rehmaatika_N.gif | loe *n*  punkte = 0  kutsu Tee\_Tase => mini, maxi, p  algaeg = timer() # time.clock()  kordus *n* korda  a = juhuarv (mini, maxi)  b = juhuarv (mini, maxi)  c = a + b  tehe = juhuarv(1, 2)  kui tehe = 1 siis  kuva a & " + " & b  tulem = c  muidu  kuva c & " - " & a  tulem = b  loe vastus  kui vastus = tulem siis  punkte = punkte + p  muidu  kuva "Vale!"  lõpp kordus  kuva punkte, aeg |

# Python. Tekstiavaldised

Koostada Pythonis funktsioonid etteantud teksti esitamiseks tagurpidi.

* sümbolid vastupidises järjestuses
* sõnad vastupidises järjestuses.

Tekstikonstant paigutatakse jutumärkide või ülakomade vahele (“teksti“, ’konstant’);

Erisümbolid: \n (reavahetus), \t (tabulaator);

**+** tekstide ühendamine (sidurdamine); **\*** teksti kordamine (operandideks tekst ja täisarv)

**len(**tekstiavaldis**)** – teksti pikkus (sümbolite arv tekstis)

tekstist sümbolite eraldamiseks kasutatakse järjenumbreid (algavad 0-st) nurksulgudes:

tekst[i], tekst[a:b], tekst[2:4], tekst[:i], tekst[i:], tekst[-1], tekst[:-3]

**Mõned stringide meetodid:**

str.**count** (*a\_str*) – loendab alamstringi esinemise arvu stringis

str.**find** (*a\_str*) – leiab alamstringi alguse stringis (järjenumbri e indeksi)

str.**lower** (),str.**upper** () – teisendab teksti koopia väike- või suurtähtedeks

str.**lstrip**(), str.**rstrip**(), str.**strip**() – eemaldab juhtsümbolid vasakult, paremalt või kõik

str.**split**() – jagab lause sõnadeks, moodustades loendi

|  |  |
| --- | --- |
| tagurpidi1 | **funktsioon** **tagurpidi**(tekst1)  mitu = pikkus(tekst1)  tekst2 = ’’  nr = mitu  **Kordus** mitu **korda**  s = tekst1 nr-s täht  tekst2 = ühenda tekst2 ja s  nr = nr – 1  **Lõpp kordus**  **tagasta** tekst2 |
| tagurpidi_sond | **funktsioon** **tagurpidi\_sõnad**(tekst1)  mitu = pikkus(tekst1)  tekst2 = ’’  tekst3 = ’’  nr = 1  **kordus** mitu **korda**  s = tekst1 nr-s täht  **kui** pole s = ’ ’  tekst3 = ühenda tekst3 ja s  **muidu**  **kui** tekst2 = ’’  tekst2 = tekst3  **muidu**  tekst2 = ühenda tekst3 ja tühik ja tekst2  **lõpp kui**  Tekst3 = ’’  **lõpp kui**  nr = nr + 1  **lõpp kordus**  **kui** tekst2 = ’’  tekst2 = tekst3  **muidu**  tekst2 = ühenda tekst3 ja tühik ja tekst2  tekst3 = ’’  **lõpp kui**  **tagasta** tekst2 |

**NB!** Selle harjutuse (nagu ka paljude teiste harjutuste puhul) ei kasuta me kõiki Pythoni häid võimalusi.   
Programm võiks järgida etteantud algoritmi.

# Python. Algoritme loenditega

|  |  |  |
| --- | --- | --- |
| Moodustada kaks loendit — puud ja hinnad. Hinnad võiks genereerida kas juhuslikult või küsida igal käivitamisel uuesti kasutajalt.  Koostada funktsioonid, mis leiavad:   * hindade aritmeetilise keskmise * maksimaalse hinna ja sellele vastava puuliigi nimetuse | **Puud** | **Hinnad** |
| haab | 120 |
| kask | 170 |
| kuusk | 230 |
| lepp | 160 |
| mänd | 210 |
| saar | 310 |
| tamm | 350 |
| vaher | 330 |

Lisaülesanne:

Luua uus loend puuliikidest, mille hind on suurem etteantud väärtusest (funktsioon tagastab loendi(d))

|  |  |
| --- | --- |
| Hindade **aritmeetilise keskmise** leidmisel tuleb kõigepealt leida hindade summa, liidetavate arvuks on loendi pikkus (väärtuste arv).  Sisend: loend **Hinnad**, n (väärtuste arv loendis)  Väljund: keskmine  Abi: S (summa), i (järjenumber) | …  S = 0  i = 0  n = Hinnad.pikkus  **kordus** n **korda**  S = S + Hinnad(i)  i = i +1  **lõpp kordus**  keskmine = S / n  … |
| **Suurima väärtuse** leidmisel peab muutuja **max** väärtuseks saama suurim loendi väärtustest. Algul eeldame, et esimene väärtustest on suurim, seejärel võrdleme **max** kõigi ülejäänud väärtustega. Kui leidub mõni suurem, omistatakse see muutujale **max**. Meelde tuleb jätta ka suurima asukoht. Muutuja **nr** saab uue väärtuse iga kord, kui muutub **max**. | …  max = Hinnad(0)  nr = 1  k = 1  **kordus** n - 1 **korda**  k = k +1  **kui**  Hinnad(k) > max **siis**  max = Hinnad(k)  nr = k  **lõpp kui**  **lõpp kordus**  puu = Puud(nr)  … |

len(loend) – loendi pikkus (elementide arv loendis)

loend[nr] – element järjenumbriga nr **NB! järjenumbrid algavad 0-st**

del(loend[2]) – järjenumbriga määratud väärtuse kustutamine loendist

**Mõned loendite meetodid:**

loend.append(x) – Lisab x-i (väärtus või objekt) loendi lõppu

loend.extend(*L*) – lisab antud loendi teise loendi lõppu.

loend.insert(*i*, *x*) – lisab x loendisse järjenumbriga määratud elemendi ette

loend.remove(*x*) – eemaldab loendist esimese x väärtuse

loend.pop([*i*]) – loeb antud järjenumbriga väärtuse, ühtlasi eemaldab selle loendist

loend.index(*x*) – tagastab x-i asukoha (indeksi) loendis

loend.sort() – sorteerib loendi elemendid kasvavas järjestuses

loend.reverse() – loendi elemendid vastupidises järjestuses

NB! Selle harjutuse (nagu ka paljude teiste harjutuste puhul) ei kasuta me kõiki Pythoni häid võimalusi (näiteks funktsiooni max või loendi meetodit index). Programm võiks järgida etteantud algoritme.

# Python. Sõnastik

## Test

|  |  |
| --- | --- |
| Antud on sõnastik – kaks loendit.  Koostada rakendus kasutaja teadmiste kontrollimiseks: küsida kasutajalt juhuslik inglisekeelne sõna, võrrelda kasutaja antud vastust sama järjenumbriga väärtusega eestikeelsete sõnade loendis ja öelda, kas vastus on õige või vale. Loendada tuleb ka õigete vastuste arv. | õigeid = 0  **kordus**  i = juhuarv (1 kuni eesti.pikkus)  **küsi** … **loe** vastus  **kui**  vastus = inglise(i) **siis**  õigeid = õigeid + 1  ütle „Õige vastus!“  **muidu**  ütle „Vale vastus!“  **lõpp kui**  **Lõpp kordus** |

Lisaülesandeid:

* vale vastuse korral küsitakse sama sõna uuesti
* loendada ka valede vastuste arv
* muuta skripti nii, et sõnu küsitaks järjest – alates esimesest kuni viimaseni.

## Tõlkimine

Antud on sõnastik – kaks loendit. Koostada rakendus kasutaja antud sõna tõlkimiseks ühest keelest teise.

|  |  |
| --- | --- |
| **Tegevuse kirjeldus:**  Kasutajalt küsitakse inglise­keelne sõna.  Seejärel tuleb ükshaaval järjest vaadata iga eestikeelset sõna (kordus ’loendi pikkus’ korda) ja võrrelda antud sõnaga.  Kui leitakse, tuleb inglise­keelsete sõnade hulgast võtta sama järje­numbriga sõna.  Kui kogu loend on üle vaadatud, aga sõna ei leitud, tuleb ka seda teatada kasutajale. | **loe** sõna  j = 1  **kordus** eesti.pikkus **korda**  **kui**  sõna = eesti(j) **siis**  **kuva** inglise(j)  **stopp**  **lõpp kui**  j = j + 1  **lõpp kordus**  **kuva** "Sellist sõna minu sõnastikus ei ole” |

**NB!** Selle harjutuse (nagu ka paljude teiste harjutuste puhul) ei kasuta me esialgu kõiki Pythoni häid võimalusi (näiteks loendi meetodit index). Programm võiks järgida etteantud algoritme.

loend = [väärtus1, väärtus2, ...]

loend[nr] – element järjenumbriga nr **NB! järjenumbrid algavad 0-st**

len(loend) – loendi pikkus (elementide arv loendis)

list(...) – loendi loomine järjendist, hulgast jm

loend.append(x) – lisab x (väärtus või objekt) loendi lõppu

Väärtuste lugemine failist loendisse (sõnade kaupa):

f=open("eesti.txt", 'r') – faili avamine lugemiseks

eesti = f.read().split() – faili sisu lugemine ja osadeks jagamine, tulemuseks on loend

f.close() – faili sulgemine

# Pyhton. Andmed failides

Mahukamaid andmeid on sageli otstarbekas hoida failides. Algandmed loetakse failist muutuja(te)sse (ühesuguse struktuuriga andmeid on mugav hoida näiteks loendites), programmi töö tulemused salvestatakse faili.

Fail avatakse funktsiooniga open. Tulemuseks on failiobjekt, mille kaudu saab infot faili kohta ja andmeid lugeda/kirjutada.

*f* = **open**(*fnimi*, *töötlusviis*)

*f* – objektimuutuja , f*nimi* – faili täisnimi: [***tee***]nimi.laiend

’**r**’ – lugemine (vaikimisi), ’**w**’ – kirjutamine, ’a’ – lisamine, ’t’ – tekstirežiim (vaikimisi), ’b’ – kahendkood jm

Pärast kasutamist tuleks fail sulgeda meetodiga close()

f.close()

**Failiobjekti omadusi ja meetodeid:**

*f*.**name** – faili nimi

*f*.**close**() – faili sulgemine

*f*.**closed** – kas fail on suletud

*f*.**readable**(), *f*.**writable**() – kas fail on lugemiseks / kirjutamiseks

*f*.**read**() – kõigi andmete lugemine, tulemuseks tekst

*f*.**readline**() – ühe kirje (rea) lugemine

*f*.**readlines**() – kogu teksti lugemine kirjete kaupa loendisse

*f*.**seek**() – lugemisviit nihutatakse etteantud kohta

*f*.**seekable**() – kas lugemisviita on võimalik nihutada

*f*.**tell**() – lugemisviit – mitu sümbolit/baiti on loetud

*f*.**truncate**() – faili tühjendamine

*f*.**write**() – kirje (stringi) kirjutamine tekstifaili

*f*.**writelines**() – loendi kirjutamine faili – ainult tekstid, elemendi lõpus soovitavalt ’\n’

**print**(väärtus(ed), sep=’ ’, end=’\n’, file=*f\_obj*) – tekstifaili kirjutamine (mugavam kui meetod write)

Failis võib olla esitatud tabel: andmestik mis moodustub ridadest (kirjetest). Kõigil kirjetel peaks olema ühesugune struktuur. Kirje koosneb ühest või mitmest **väljast** (väärtusest), mis eraldatakse üksteisest mingi eraldajaga (tühik, tabulaator **\t** vm). Väljade väärtuste sees seda eraldajat ei ole. Reavahetuse sümbol kirje lõpus on **\n**.

Ülesanne:

1. Lugeda failidest eesti.txt ja inglise.txt väärtused loenditesse. Kirjutada väärtused uude tekstifaili nii, et sama tähendusega sõnad on ühes reas (eraldajaks tühik).
2. Lugeda väärtused uuest failist. Moodustada:

* kaks loendit (eesti ja inglisekeelsed sõnad eraldi)
* üks loend, mille elemendid on sõnapaarid (loendid)

import os, os.path

**os**.getcwd() – töökataloog

**os.path**.exists(failinimi) – kas eksisteerib selline fail

*tekst***.split**() – sringi meetod, mis jagab teksti osadeks, moodustades loendi

# Pyhton. Edetabel

Koostada Pythonis funktsioon, mis lisab edetabelisse (faili, kus hoitakse võistlejate nimesid ja punktisummat punktide kahanemise järjestuses) sobivasse kohta uue kirje (nimi ja punktisumma).

|  |  |
| --- | --- |
| Edetabeli info loetakse failist loenditesse.  Lisatakse antud katse andmed: kasutaja nimi ja saadud punktid, arvestades vajalikku kahanevat järjestust  Loend kirjutatakse samasse faili, mis eelnevalt avatakse kirjutamiseks.  Tekstifail koosneb **kirjetest** ehk ridadest.  Kirje koosneb ühest või mitmest **väljast**, mis eraldatakse üksteisest mingi eraldajaga (tühik, tabulaator vm).  Välja sees tühikuid ei ole.  Kirje lõpus on sageli (mittenähtav) reavahetuse sümbol **\n**.  **Faili avamine:**  *f* = **open**(*f\_nimi*, *töötlusviis*)  *f* - objektimuutuja, viit failile  *f\_nimi* - faili täisnimi: [***tee***]nimi.txt  "**w**" - kirjutamine, "**r**" - lugemine (vaikimisi)  **Faili sulgemine:** *f*.**close**()  **Andmete lugemine:**  a) faili lugemine korraga: *loend* = *f.***read**()  b) kirjete kaupa, meetodiga *f***.read**()  c) korduslauses: **for** ***kirje*** **in** *f***:** [*tegevused* ***kirjega***]  **Faili** **kirjutamine:**  *a) f.***write**(*string*)  b) **print**([*objekt*, *...*], *\**, *sep=' '*, *end='\n'*, *file=f)* | prots lisa\_etab(nimi, punkte)  kui faili "edetabel.txt" pole, siis  kirjuta faili nimi ja punkte  tagasta 1  muidu  nimed = [ ]; punktid = [ ]  ava fail "edetabel.txt"  kordus iga kirje jaoks failis  elem = kirje väljad  lisa nimedele elem[0]  lisa punktidele elem[1]  lõpp kordus  sulge fail    mitu = pikkus(nimed)  kordus k = 0.. mitu-1  kui punkte < punktid[k] siis  sisesta nimed [k] (nimi)  sisesta punktid.[k] (punkte)  välju kordusest  lõpp kui  lõpp kordus  kui pikkus(nimed) = mitu siis  lisa nimed (nimi)  lisa punktid (punkte)  k = k + 1  lõpp kui  ava fail "edetabel,txt" kirjutamiseks  kordus i = 0...pikkus(nimed)  kirjuta faili (nimed[i]&’\t’&punktid[i]) & ’\n’)  lõpp kordus  sulge fail  tagasta k |

Lisaülesanded:

* Edetabelifaili nimi võiks olla ette antav (funktsiooni parameeter)
* Faili võiks lisada ka järjenumbri

import os, os.path

**os**.getcwd() – töökataloog

**os.path**.exists(failinimi) – kas eksisteerib selline fail

*tekst***.split**() – sringi meetod, mis jagab teksti sõnadeks, moodustades loendi

# Python. Sõnastikud

***Sõnastik (dictionary)***

Sõnastik on andmetüüp loetelu jaoks: elementide poole pöördutakse **nimi[võti]**, võti (ik *key*) võib olla peale täisarvu ka reaalarv, tekst, tõeväärtus. Sõnastiku moodustavad paarid võti:väärtus, mis eraldatakse komadega.

Piirajatena on kasutusel looksulud {}. Sõnastik ei ole järjestatud.

d = {võti1:väärtus1, võti2:väärtus2, ...}

d = {} – tühja sõnastiku loomine

**dict**(...) – sõnastiku loomine

**len**(d) – elementide arv sõnastikus

d[võti] = väärtus – uue elemendi loomine või vanale väärtuse omistamine

**del** d[võti] – elemendi eemaldamine

võti **in** d, võti **not in** d – kas selline võti on / ei ole sõnastikus

d.**clear**() – sõnastiku tühjendamine

d.**pop**(võti) – kui võti on sõnastikus, tagastab sellele vastava väärtuse ja eemaldab elemendi

d.**keys**() – kõik võtmed sõnastikust

d.**values**() – kõik väärtused sõnastikust

Ülesanne:

Lugeda failist tekst.

Loendada iga sümboli esinemise arv tekstis ja esitada see nimekiri tabeli kujul shelli aknas.

Lisaülesanne:

esitada tabel shelli aknas kolmeveerulisena.

# Python. Funktsiooni parameetrid

Sageli kasutatavad tegevused on otstarbekas vormistada funktsioonidena, mida saab kasutada erinevate algandmetega. Funktsiooni parameetrite hulk, tüübid ja järjekord tuleks valida nii, et funktsiooni oleks mugav kasutada.

Kõikidele parameetritele ei pea funktsiooni poole pöördumisel alati väärtusi andma. Sellised parameetrid asuvad funktsiooni kirjelduses parameetrite loetelu lõpus ja neile on määratud vaikeväärtus (väärtus, mis omistatakse parameetrile siis, kui sellele ei anta väärtust pöördumisel).

Näiteks: def kiir(x, y, suund, pikkus, joon=1, värv=’black’

Ülesanne:

Koostada Pythonis funktsioonid kujundite joonistamiseks. Valida sobiv parameetite hulk ja järjestus.

Funktsioonide näiteid

|  |  |
| --- | --- |
| **Ristkülik**  Parameetrid:  vasaku serva ja alumise serva asukohad (x, y),  laius, kõrgus,  joone ja tausta värvid ning joone laius (vaikeväärtustega). | **from** turtle **import** \*  **def** **ristkylik**(x,y, laius, korgus,t\_varv=0,j\_varv=0,joon=0):  penup()  **if** joon!=0**:** pensize(joon)  **if** type(j\_varv)==str **or** type(j\_varv)==tuple**:**  pencolor(j\_varv)  **if** type(t\_varv)==str **or** type(t\_varv)==tuple**:**  fillcolor(t\_varv)**;** begin\_fill()  setpos(x, y) *#võib ka goto()*  pendown()**;** goto(x+laius,y)**;** goto(x+laius, y+korgus)  goto(x,y+korgus)**;** goto(x,y)**;** penup()  **if** filling()**:** end\_fill() |
| **Ring**  Parameetrid:  keskpunkti asukoht (x, y)  raadius r  tunnus (kas täita) – vaikimisi väärtus True  Värvid, joone paksuse jm võib määrata varem, enne funktsiooni poole pöördumist. | **def** **ring**(x, y, r, tunnus=True):  penup()  seth(0); goto(x, y-r)  pendown()  **if** tunnus**:**  begin\_fill()  circle(r)  **if** filling()**:**  end\_fill()  penup() |

**Veel võiks olla:**

* **joon**, parameetriteks otspunktide koordinaadid, joone paksus ja värv
* **joon**, parameetriteks alguspunkti koordinaadid, pikkus ja suund
* **horisontaalne joon, vertikaalne joon**, parameetrid: algpunkti (või keskpunkti) koordinaadid, pikkus, värv jm
* **ellips** (keskpunkt, laius, kõrgus)
* **korrapärane hulknurk**

jne

Ülesanne.

Koostada rakendus, milles kasutatakse loodud funktsioone mingi kompositsiooni moodustamiseks.

Kasutada kordus- ja valikulauseid, küsida andmeid kasutajalt.

Funktsioonid võivad asuda eraldi failis. Oma rakendusele saab need lisada käsuga **import**.

# Python. Funktsioonide uurimine

Koostada kolme funktsiooni graafikud kasutaja poolt valitud lõigul.

Andmed: lõigu algus, lõigu lõpp, jaotiste arv







**a** - lõigu algus**, b** - lõigu lõpp**, n** - jaotiste arv: määrab kasutaja; **n** võiks olla 100...1000

**x** (0...n)- argumendi väärtuste loend

**y1**(0...n), **y2**(0...n), **y3**(0...n) - funktsioonide väärtuste loendid

Korraliku joonise saamiseks tuleks valida sobivad mastaabitegurid või kasutajakoordinaadid. Selleks on vaja teada kõikide funktsioonide väikseimaid ja suurimaid väärtusi. Argumendi ja funktsiooni väärtuste mitmekordse arvutamise asemel on otstarbekas kord leitud väärtused kirjutada loenditesse. Loendite moodustamine peaks olema eraldi funktsioon.

Suurima ja väikseima väärtuse järjenumbri leidmiseks kasutada vastavaid omaloodud funktsioone või standardfunktsioone min() ja max().

Joonise tegemiseks saab kasutada funktsioone Pythoni moodulist turtle.py:

[**pendown()**](http://docs.python.org/py3k/library/turtle.html#turtle.pendown) | [**pd()**](http://docs.python.org/py3k/library/turtle.html#turtle.pd) | [**down()**](http://docs.python.org/py3k/library/turtle.html#turtle.down) # pliiats alla

[**penup()**](http://docs.python.org/py3k/library/turtle.html#turtle.penup) | [**pu()**](http://docs.python.org/py3k/library/turtle.html#turtle.pu) | [**up()**](http://docs.python.org/py3k/library/turtle.html#turtle.up) # pliiats üles

[**pensize()**](http://docs.python.org/py3k/library/turtle.html#turtle.pensize) | [**width()**](http://docs.python.org/py3k/library/turtle.html#turtle.width) # pliiatsi suurus

[**pencolor()**](http://docs.python.org/py3k/library/turtle.html#turtle.pencolor) # joone värv

[**fillcolor()**](http://docs.python.org/py3k/library/turtle.html#turtle.fillcolor) # tausta värv

[**goto()**](http://docs.python.org/py3k/library/turtle.html#turtle.goto) | [**setpos()**](http://docs.python.org/py3k/library/turtle.html#turtle.setpos) # mine(x,y)

[**dot()**](http://docs.python.org/py3k/library/turtle.html#turtle.dot) # punkt (diameeter, värv)

[**clone()**](http://docs.python.org/py3k/library/turtle.html#turtle.clone) # objekti (kilpkonna) teisiku loomine

[**getturtle()**](http://docs.python.org/py3k/library/turtle.html#turtle.getturtle) # viit esmasele (anonüümsele) objektile

[**turtles()**](http://docs.python.org/py3k/library/turtle.html#turtle.turtles) # kõigi kilpkonnade (objektide) hulk

[**setworldcoordinates()**](http://docs.python.org/py3k/library/turtle.html#turtle.setworldcoordinates) # koordinaadid (x1,y1, x2, y2)

[**setup()**](http://docs.python.org/py3k/library/turtle.html#turtle.setup) # akna suurus (laius, kõrgus) ja asukoht (x, y)

[**title()**](http://docs.python.org/py3k/library/turtle.html#turtle.title) # tekst akna päises

**Lisaülesanded:**

* Leida ja märkida graafiku x-teljel iga funktsiooni miinimumi ja maksimumi asukohad ning kirjutada joonisele (või väljastada käsuaknasse) nende punktide koordinaadid.
* Leida iga funktsiooni keskmine väärtus antud lõigul.

# Pythoni funktsioonid (meetodid) moodulis turtle.py

#### Objekti (turtle) seaded

**Liikumine**

[**forward()**](http://docs.python.org/py3k/library/turtle.html#turtle.forward) | [**fd()**](http://docs.python.org/py3k/library/turtle.html#turtle.fd) # liigu edasi

[**backward()**](http://docs.python.org/py3k/library/turtle.html#turtle.backward) | [**bk()**](http://docs.python.org/py3k/library/turtle.html#turtle.bk) | [**back()**](http://docs.python.org/py3k/library/turtle.html#turtle.back) # liigu tagasi

[**right()**](http://docs.python.org/py3k/library/turtle.html#turtle.right) | [**rt()**](http://docs.python.org/py3k/library/turtle.html#turtle.rt) # pööra paremale

[**left()**](http://docs.python.org/py3k/library/turtle.html#turtle.left) | [**lt()**](http://docs.python.org/py3k/library/turtle.html#turtle.lt) # pööra vasakule

[**goto()**](http://docs.python.org/py3k/library/turtle.html#turtle.goto) | [**setpos()**](http://docs.python.org/py3k/library/turtle.html#turtle.setpos) # mine(x,y)

[**setx()**](http://docs.python.org/py3k/library/turtle.html#turtle.setx), [**sety()**](http://docs.python.org/py3k/library/turtle.html#turtle.sety) # määra x, määra y

[**setheading()**](http://docs.python.org/py3k/library/turtle.html#turtle.setheading) | [**seth()**](http://docs.python.org/py3k/library/turtle.html#turtle.seth) # määra nurk

[**home()**](http://docs.python.org/py3k/library/turtle.html#turtle.home) # mine(0, 0), nurk 0

[**circle()**](http://docs.python.org/py3k/library/turtle.html#turtle.circle) # ring(raadius, kaar=360, hnurk)

[**dot()**](http://docs.python.org/py3k/library/turtle.html#turtle.dot) # punkt (diameeter, värv)

[**stamp()**](http://docs.python.org/py3k/library/turtle.html#turtle.stamp) # jälg, väljastab id. numbri

[**clearstamp()**](http://docs.python.org/py3k/library/turtle.html#turtle.clearstamp) # kustutab nimetatud jälje

[**clearstamps()**](http://docs.python.org/py3k/library/turtle.html#turtle.clearstamps) # kustutab jälgi …

[**undo()**](http://docs.python.org/py3k/library/turtle.html#turtle.undo) # eelmise tegevuse tühistamine

[**speed()**](http://docs.python.org/py3k/library/turtle.html#turtle.speed) # animatsiooni kiirus 1..10 või 0

**Objekti info**

[**position()**](http://docs.python.org/py3k/library/turtle.html#turtle.position) | [**pos()**](http://docs.python.org/py3k/library/turtle.html#turtle.pos) # asukoht

[**towards()**](http://docs.python.org/py3k/library/turtle.html#turtle.towards) # suund (etteantud punkti poole)

[**xcor()**](http://docs.python.org/py3k/library/turtle.html#turtle.xcor), [**ycor()**](http://docs.python.org/py3k/library/turtle.html#turtle.ycor) # x-asukoht, y-asukoht

[**heading()**](http://docs.python.org/py3k/library/turtle.html#turtle.heading) # nurk

[**distance()**](http://docs.python.org/py3k/library/turtle.html#turtle.distance) # kaugus punktist või teisest obj.st

**Nurgaühikud**

[**degrees()**](http://docs.python.org/py3k/library/turtle.html#turtle.degrees) # nurga ühiku määramine, täisring 360°

[**radians()**](http://docs.python.org/py3k/library/turtle.html#turtle.radians) # nurga ühiku määramine - täisring 2\*math.pi rad

**Pliiatsi seaded**

[**pendown()**](http://docs.python.org/py3k/library/turtle.html#turtle.pendown) | [**pd()**](http://docs.python.org/py3k/library/turtle.html#turtle.pd) | [**down()**](http://docs.python.org/py3k/library/turtle.html#turtle.down) # pliiats alla

[**penup()**](http://docs.python.org/py3k/library/turtle.html#turtle.penup) | [**pu()**](http://docs.python.org/py3k/library/turtle.html#turtle.pu) | [**up()**](http://docs.python.org/py3k/library/turtle.html#turtle.up) # pliiats üles

[**pensize()**](http://docs.python.org/py3k/library/turtle.html#turtle.pensize) | [**width()**](http://docs.python.org/py3k/library/turtle.html#turtle.width) # pliiatsi suurus

[**pen()**](http://docs.python.org/py3k/library/turtle.html#turtle.pen) # pliiatsi omadused

[**isdown()**](http://docs.python.org/py3k/library/turtle.html#turtle.isdown) # kas pliiats all

[**color()**](http://docs.python.org/py3k/library/turtle.html#turtle.color) # värv (joon, taust)

[**pencolor()**](http://docs.python.org/py3k/library/turtle.html#turtle.pencolor) # joone värv

[**fillcolor()**](http://docs.python.org/py3k/library/turtle.html#turtle.fillcolor) # tausta värv

[**filling()**](http://docs.python.org/py3k/library/turtle.html#turtle.filling) # kas toimub kujundi täitmine

[**begin\_fill()**](http://docs.python.org/py3k/library/turtle.html#turtle.begin_fill) # täitmine algab

[**end\_fill()**](http://docs.python.org/py3k/library/turtle.html#turtle.end_fill) # täitmine lõppeb

[**reset()**](http://docs.python.org/py3k/library/turtle.html#turtle.reset) # joonise kustutamine, algseaded

[**clear()**](http://docs.python.org/py3k/library/turtle.html#turtle.clear) # joonise kustutamine

[**write()**](http://docs.python.org/py3k/library/turtle.html#turtle.write) # objekt kirjutab teksti

**Kuvandi seaded**

[**showturtle()**](http://docs.python.org/py3k/library/turtle.html#turtle.showturtle) | [**st()**](http://docs.python.org/py3k/library/turtle.html#turtle.st) # näita obj

[**hideturtle()**](http://docs.python.org/py3k/library/turtle.html#turtle.hideturtle) | [**ht()**](http://docs.python.org/py3k/library/turtle.html#turtle.ht) # peida obj

[**isvisible()**](http://docs.python.org/py3k/library/turtle.html#turtle.isvisible) # kas obj on nähtav

[**shape()**](http://docs.python.org/py3k/library/turtle.html#turtle.shape) # obj kuvand (määramine või nimi)

[**shapesize()**](http://docs.python.org/py3k/library/turtle.html#turtle.shapesize) | [**turtlesize()**](http://docs.python.org/py3k/library/turtle.html#turtle.turtlesize) #objsuurus(laius,kõrgus,joon)

[**settiltangle()**](http://docs.python.org/py3k/library/turtle.html#turtle.settiltangle) # obj pöördenurk (mitte suund)

[**tiltangle()**](http://docs.python.org/py3k/library/turtle.html#turtle.tiltangle) # obj pöördenurk (suund ei muutu)

[**tilt()**](http://docs.python.org/py3k/library/turtle.html#turtle.tilt) # obj pöördenurga muutus (suund ei muutu)

**Objektiga (turtle) seotud sündmuste kasutamine**

[**onclick(fun, klahv=1)**](http://docs.python.org/py3k/library/turtle.html#turtle.onclick) # hiireklõps objektil käivitab fn.i *fun* koordinaatidega x, y; klahv on hiire klahvi nr

[**onrelease()**](http://docs.python.org/py3k/library/turtle.html#turtle.onrelease)

[**ondrag()**](http://docs.python.org/py3k/library/turtle.html#turtle.ondrag)

**Veel turtle meetodeid**

[**begin\_poly()**](http://docs.python.org/py3k/library/turtle.html#turtle.begin_poly) # kompleksjoone alustamine

[**end\_poly()**](http://docs.python.org/py3k/library/turtle.html#turtle.end_poly) # kompleksjoone lõpetamine

[**get\_poly()**](http://docs.python.org/py3k/library/turtle.html#turtle.get_poly) # viit viimati loodud kompleksjoonele

[**clone()**](http://docs.python.org/py3k/library/turtle.html#turtle.clone) # objekti (kilpkonna) teisiku loomine

[**getturtle()**](http://docs.python.org/py3k/library/turtle.html#turtle.getturtle) # viit esmasele (anonüümsele) objektile

[**setundobuffer()**](http://docs.python.org/py3k/library/turtle.html#turtle.setundobuffer) # tegevuste pinu suurus

[**undobufferentries()**](http://docs.python.org/py3k/library/turtle.html#turtle.undobufferentries) # tegevused pinus

#### Graafikaakna seaded

[**bgcolor()**](http://docs.python.org/py3k/library/turtle.html#turtle.bgcolor) # taustavärv

[**bgpic(fail)**](http://docs.python.org/py3k/library/turtle.html#turtle.bgpic) # taustapilt (gif-fail)

[**clearscreen()**](http://docs.python.org/py3k/library/turtle.html#turtle.clearscreen) # kogu joonise kustutamine

[**resetscreen()**](http://docs.python.org/py3k/library/turtle.html#turtle.resetscreen) # kogu joonise kustutamine ja algseaded obj.dele

[**screensize()**](http://docs.python.org/py3k/library/turtle.html#turtle.screensize) # ekraani suurus (laius, kõrgus [, taust])

[**setworldcoordinates()**](http://docs.python.org/py3k/library/turtle.html#turtle.setworldcoordinates) # koordinaadid (x1,y1, x2, y2)

**Animatsiooni seaded**

[**delay()**](http://docs.python.org/py3k/library/turtle.html#turtle.delay) # viivitus (millisekundites)

[**tracer()**](http://docs.python.org/py3k/library/turtle.html#turtle.tracer) # animatsioon sisse/välja või kiirem/aeglasem

[**update()**](http://docs.python.org/py3k/library/turtle.html#turtle.update) # tehtud joonise/muudatuste näitamine ekraanil

**Aknaga seotud sündmuste kasutamine**

[**listen()**](http://docs.python.org/py3k/library/turtle.html#turtle.listen)

[**onkey()**](http://docs.python.org/py3k/library/turtle.html#turtle.onkey) | [**onkeyrelease()**](http://docs.python.org/py3k/library/turtle.html#turtle.onkeyrelease)

[**onkeypress()**](http://docs.python.org/py3k/library/turtle.html#turtle.onkeypress)

[**onclick()**](http://docs.python.org/py3k/library/turtle.html#turtle.onclick)| [**onscreenclick()**](http://docs.python.org/py3k/library/turtle.html#turtle.onscreenclick)

[**ontimer()**](http://docs.python.org/py3k/library/turtle.html#turtle.ontimer)

[**mainloop()**](http://docs.python.org/py3k/library/turtle.html#turtle.mainloop) | [**done()**](http://docs.python.org/py3k/library/turtle.html#turtle.done)

**Eriseaded ja -meetodid**

[**mode()**](http://docs.python.org/py3k/library/turtle.html#turtle.mode) # esitusviis (“standard“, “logo“, “world“)

[**colormode()**](http://docs.python.org/py3k/library/turtle.html#turtle.colormode) # värvi (RGB) esitusviis 1.0 või 255

[**getshapes()**](http://docs.python.org/py3k/library/turtle.html#turtle.getshapes) # kuvandite loend

[**register\_shape()**](http://docs.python.org/py3k/library/turtle.html#turtle.register_shape) # uue kuvandi registreerimine

[**turtles()**](http://docs.python.org/py3k/library/turtle.html#turtle.turtles) # kõigi kilpkonnade hulk

[**window\_height()**](http://docs.python.org/py3k/library/turtle.html#turtle.window_height) # akna kõrgus

[**window\_width()**](http://docs.python.org/py3k/library/turtle.html#turtle.window_width) # akna laius

**Veel aknaga seotud meetodeid**

[**bye()**](http://docs.python.org/py3k/library/turtle.html#turtle.bye) # graafikaakna sulgemine

[**exitonclick()**](http://docs.python.org/py3k/library/turtle.html#turtle.exitonclick) # akna sulgemine hiireklõpsuga

[**setup( )**](http://docs.python.org/py3k/library/turtle.html#turtle.setup) # akna suurus (laius, kõrgus) ja asukoht (x, y)

[**title()**](http://docs.python.org/py3k/library/turtle.html#turtle.title) # tekst akna päises

**Väärtuste sisestamine (input)**

[**textinput()**](http://docs.python.org/py3k/library/turtle.html#turtle.textinput) # väärtuse lugemine (päis, tekst), tulemi tüüp str

[**numinput()**](http://docs.python.org/py3k/library/turtle.html#turtle.numinput) # väärtuse lugemine (päis, tekst), tulemi tüüp float