Pythoni kilpkonnagraafika (moodul Turtle)

Võid uurida lisaks Logo ajalugu (https://en.wikipedia.org/wiki/Logo_(programming_language)), mida on peetud ka algselt lastele programmeerimise tutvustamise abivahendiks.

Pythoni Turtle mooduli joonistamine töötab Logole sarnaselt.

Järgnevas tekstis mainitud näiteprogrammid leiad kursuse veebilehelt lingi „Turtle näited" alt.

Juhuslikud arvud

Mängude ja muidu lõbusate asjade tegemiseks on tihti abi juhuslikest valikutest. Juhuslike valikute tegemiseks aitavad kaasa juhuslikud arvud. Nende arvude väljaarvutamiseks on matemaatikas mitmeid teooriaid. Seega pole nad midagi nii väga juhuslikud. Õigem ongi neid kutsuda pseudojuhuslikeks arvudeks.

Funktsioonid juhuslike arvude tegemiseks tuleb eelnevalt vastavast moodulist importida:

```
from random import *
```

Moodulis `random` on palju funktsioone, kuid kasulikumad võiksid olla:

`randint(algus,lõpp)` - täisarv antud vahemikust

`choice(list)` - `list` on väärtuste jada (arvud, stringid jms), valitakse suvaline nendest

`shuffle(list)` - vahetab etteantud listi elemendid suvalisse järjekorda

Näiteks:

Tekitame listi (järjendi, massiivi) nimega v2rvid:
```
v2rvid = ['red','blue','green','magenta']
```
Laseme arvutil valida loetelust juhuslikult ühe värvi:
```
v2rv = choice(v2rvid)
```
Laseme arvutil valida juhusliku arvu vahemikus 1-st 100-ni:
```
juhuslik_arv = randint(1,100)
```

Turtle moodul

Ka terve "kilpkonnagraafika" funktsioonikogu on eraldi moodulis. Funktsioonid tuleb importida:

```
from turtle import *
```
Joonistatakse pliiatsiga (*pen*, mis on nagu kilpkonna saba). Kui saba on maas (`down()`), jääb jälg, kui saba üleval (`up()`), siis jälge ei jää.

Joonistamine: variant 1. Kilpkonn liigub soovitud suunas

Kilpkonnal on nö suund. Kui kilpkonnale öelda liigu edasi 50 pikslit, siis ta liigub selles suunas, kuhu pea näitab. Kilpkonn saab ka liikuda tagasi (`forward()`, `backward()`, ka `fd()`, `bk()`).

Kilpkonna saab pöörata praeguse suuna suhtes (`right()`, `left()`, ka `rt()`, `lt()`), andes ette pöördenurga kraadides (vaikimisi) või radiaanides.

Näiteks joonistame täisnurga (kopeeri arvutisse):

```
forward(100) # Liigu edasi 100 pikslit
left(90)     # Pööra vasakule 90°.
forward(100) # Liigu edasi 100 pikslit
```

Ülesanne 1 Täisnurkne kolmnurk

Kuidas saada soovitud suuruses võrdhaarset täisnurkset kolmnurka? Täiendame programmi selliselt, et kasutajalt küsitakse kolmnurga kaatetite pikkused (võrdsed) ja edasi joonistatakse kolmnurk.

Ülesannet saab ajada matemaatilisemaks suvaliste kaatetite pikkustega. Mida siis veel lisaks teha tuleks?

Pythoni abi leiab siit: https://docs.python.org/3/library/math.html

Saab ka öelda, et keera pea sellesse või teise suunda sõltumata sellest, kuhu poole ta hetkel vaatab (`setheading()`).

Näiteks:

`setheading(0)` või `setheading("east")` keerab kilpkonna näoga paremale

Peale kilpkonna pööramist hakkab ta liikuma uude suunda.

Ülesanne 2 Lihtne liikumine

Teeme koos numbri 2. Proovi ise joonistada tähti või numbreid, kujundades nad kandilistena. Nagu me näeme neid näiteks bensiinijaamade tabloodel. Kas õnnestub oma nimi kirja panna? See võib osutuda üsna tülikaks. Aga paar tähte võiks ikka proovida. Et tähtedele vahed tekiksid, tõsta kilpkonna saba üles (`up()`), liigu uue tähe algusesse ja langeta saba (`down()`).

Sellise meetodi abil saab joonistada kujundeid ka polaarkoordinaate kasutades – loodan, et sõnastasin matemaatiliselt õigesti (nt `arhimedese_spiraal.py`)

Joonistamine: variant 2. Kilpkonn liigub etteantud koordinaatidele

Joonistusaknas on x-y koordinaatteljestik. Punkt 0,0 jääb akna keskele, nagu koolis matemaatikatunnis.

Kilpkonna käest saab küsida, millistel koordinaatidel ta paikneb (`position()`, `xcor()`, `ycor()`). Ja talle saab ka ette anda koordinaadid, kuhu ta minema peab (`setpos()`, `setx()`, `sety()`). Etteantud koordinaatidele liikumiseks ei ole teda vaja peaga nõutud suunda pöörata. Ta liigub edukalt ka nö külgsuunas. Iga liikumine jätab maha jälje (kui just saba üles pole tõstetud).

Näiteks:

```
x,y = position()     # Saime teada praeguse asukoha
```

```
setpos(x+50, y+50)  # Liigutame diagonaalis kirdesse
```
Sarnaselt saab küsida ja muuta ka ainult ühte koordinaati:

```
x = xcor()
setx(x+50)
```

Selles režiimis saab näiteks joonistada edukalt igasuguseid funktsioonide graafikuid, kui suudame leida parameetrilised võrrandid x ja y väljaarvutamiseks.

Ülesanne 3 Funktsiooni graafik

Proovime koos joonistada siinusfunktsiooni graafikut.

Lisaks saab vaadata näidet epitsükloidi joonistamisest (`epitsykloid.py`). Kui tead veel mõnda põnevat funktsiooni, siis proovi sarnaselt joonistada.

Joonistamine: variant 3. Terved kujundid

On mõned käsud, millega saab joonistada terve kujundi. Näiteks saab joonistada ringi funktsiooniga `circle(raadius, ulatus, samme)`

```
circle(100)       #joonistab ringi raadiusega 100
circle(100,180)   #joonistab poolkaare, sest ulatus on 180
```
Parameetri `samme` abil saab määrata kui "korralik" ring tuleb, sest ring joonistatakse kui hulknurk ja mida rohkem külgi, seda ilusam. Tasub ka tähele panna, et kilpkonna asukoht ringi joonistamise alguses ei ole mitte keskpunkt, vaid hoopis ringjoone all servas.

Joonistada saab ka täppe (`dot(raadius, värv)`). Kusjuures raadius võib olla üsna suvalise suurusega. Kilpkonna hetke asukoht jääb täpi keskpunktiks.

Näiteks:

```
dot(100, "red")   # joonistab punase ringi (täpi) raadiusega 100
```
Ja teisi kujundite funktsioone otseselt ei olegi. Näiteks ristküliku jaoks oleks vaja tõmmata neli joont ja vahepeal kilpkonna vajalikus suunas pöörata. Kui seda vaja teha palju, on kasulik koostada funktsioon. Vaata näidet `maja_aiaga.py`. Aga see võib natuke liiga keeruliseks osutuda.

Selles ülesandes on aga kasutatud veel midagi, millest seni juttu ei ole olnud – nimelt erinevaid värve.

Värvid

Kaks peamist värvi määramise võimalust on:

- nimede kasutamine;

- RGB värvimudeli kasutamine.

Värvide nimesid on päris palju (näide `t2pid_v2rvilised.py`);

Näites kasutatud nimed (nn *color string*) on olemas Pythoni värvispetsifikatsioonis. Kui õigeid nimesid teada, on nende abil hea värve määrata, sest nimi on kirjeldav ja koodi lugedes arusaadav. Kuidas

saame näiteprogrammi kasutades teada, kui palju nimelisi värve meie kasutuses on?

Veel üks värvide loetelu: http://www.science.smith.edu/dftwiki/index.php/Tk_Color_Names

Mis on RGB värvimudel? Millised on põhivärvused ja millised täiendvärvused? Kuidas neid RGB abil määrata saab?

RGB mudeli kasutamiseks on mõistlik määrata värvirežiim funktsiooniga `colormode(255)` - sel juhul saab määrata kõik mudeli värvid vahemikus `0 .. 255`.

Nimede seos RGB-ga: https://www.tcl.tk/man/tcl8.4/TkCmd/colors.htm

Vaatame peamiste värvide jaoks näidet `ringid_ja_v2rvid.py` ning `ringid_ja_v2rvid_viisakam.py`

Värve saab kasutada nii joonte jaoks kui ka mingi kujundi sisu täitmiseks ehk olemas on joone värv ja täidisvärv, nagu tavaliselt graafikas. Saab joonistada kontuuri (jooni) `pencolor()` abil seatud värviga ja lasta kontuur seest värvida värviga, mis on eelnevalt seatud `fillcolor()` abil.

Funktsioonidega `begin_fill()` ja `end_fill()` määratakse värviga täidetav kujund, värvitakse see osa, mis joonistati nende kahe käsu vahel. Näiteks selleks, et saada seest täidetud ristkülik sobivad järgmised koodiread:

```
setpos(200, 200)     # üks nurk paika
pencolor("red")      # punane joon
fillcolor("green")   # roheline sisu
begin_fill()         # siit alates algab kujund, mis seest värvitakse
setpos(300, 200)     # joon piki x-telge
setpos(300, 300)     # joon piki y-telge
setpos(200, 300)     # joon piki x-telge
setpos(200, 200)     # joon piki y-telge
end_fill()           # peale seda täidetakse ruut rohelise värviga
```

Saab määrata ka joonistamisel kasutatava pliiatsi (kilpkonna saba) jämedust (ehk tekkiva joone paksust) funktsiooniga `pensize(joone_paksus)`. Joone paksus on täisarv.

Ülesanne 4 Täpid rivisse

Joonista üksteise kõrvale 5 punast täppi raadiusega 10 punkti.

Kuidas joonistada üksteise kõrvale kasutaja poolt soovitud arv täppe, mis on kasutaja soovitud raadiusega? Vihje - kasutajalt küsitakse täppide arv ja ühe täpi raadius. Kasutada tuleb tsüklit soovitud arvu täppide saamiseks. Välja saab arvutada, kus on järgmise täpi keskpunkt.

Lisaks ja lõpetuseks

Funktsioone on loomulikult veel. Saab tekitada mitu kilpkonna, kes eraldi oma asju ajavad. Muuta joonistamise kiirust, küsida andmeid kilpkonna omaduste, asukoha jms kohta, kustutada jälge, kirjutada joonistamise aknasse teksti ja küsida kasutajalt sisendit (print ja input töötavad käsuaknast) jne. Aga seda kõike võib lisaks uurida Turtle dokumentatsioonist (link kursuse veebis).

Turtle moodulit on käsitletud ka valikkursuse "Rakenduste loomise ja programmeerimise alused" õppematerjalis ning Tartu Ülikooli programmeerimise õpikus (lingid veebilehel).

Loomulikult on internetist leitav ka lihtsalt otsides arvukalt näiteid, videoõpetusi ja muud sarnast.