1. LOGO ja Pythoni kilpkonnagraafika (moodul Turtle)

Programmeerimise tõdede õpetamine lastele ei ole mingi viimase aja leitutis. Üle 50 aasta tagasi pühendusid sellele Cynthia Solomon ja Seymour Papert koos meeskonnaga. Tulemuseks oli keel LOGO ning artklid õpetamise kohta (kaks viidet 1971 ja 1976 a artiklitele leiad ka kursuse veebist), samuti matemaatika õpetamisest programmeerimise toel (samuti artikli viide kursuse veebis).

Võid uurida lisaks LOGO ajalugu vikipeediast (https://en.wikipedia.org/wiki/Logo_(programming_language)),

Paraja hulga piltidega LOGO kilpkonna ajaloo leiad siit:

(http://cyberneticzoo.com/cyberneticanimals/1969-the-logo-turtle-seymour-papert-marvin-minsky-et-al-american/)

LOGO kilpkonna põhimõtte alusel töötavad ka tänapäevased lasterobotid - nt Bee Botid jms. S. Papert osales Lego Mindstorm komplekti väljatöötamisel, mis omakorda sai nime S. Paperti raamatu järgi (https://dl.acm.org/doi/pdf/10.5555/1095592).

1.1. Elementaarsed liikumised LOGO keeles

Katsetame kõigepealt LOGO veebirakenduses: https://www.calormen.com/jslogo/

Põhimõtted on lihtsad: kilpkonn liigub sinna, kuhu pea näitab. Saab määrata sammude arvu. Kilpkonn liigub ka tagurpidi tagasi. Kilpkonn pöörab nii vasakule kui paremale. Peale pööramist liigub ta edasi uues suunas. Pöördenurk antakse kraadides.

Vt ka 1. joonistamise varianti Turtle mooduli peatükis 1.3.1.

Valik käske:

`forward 100` ehk `fd 100` - kilpkonn liigub edasi 100 sammu

`back 100` ehk `bk 100` - kilpkonn liigub tagasi 100 sammu

`clearscreen` ehk `cs` - kustutab kõik ja kilpkonn läheb lähteasendisse

`right 90` ehk `rt 90` - kilpkonn pöörab 90° päripäeva (paremale) pööramisel saab kasutada kraade 0 kuni 360

`right 180` ehk `rt 180` - kilpkonn pöörab 180° vastupäeva (vasakule)

Teeme mõned katsed, ülesanded materjalist "Programming in LOGO", link kursuse veebis.

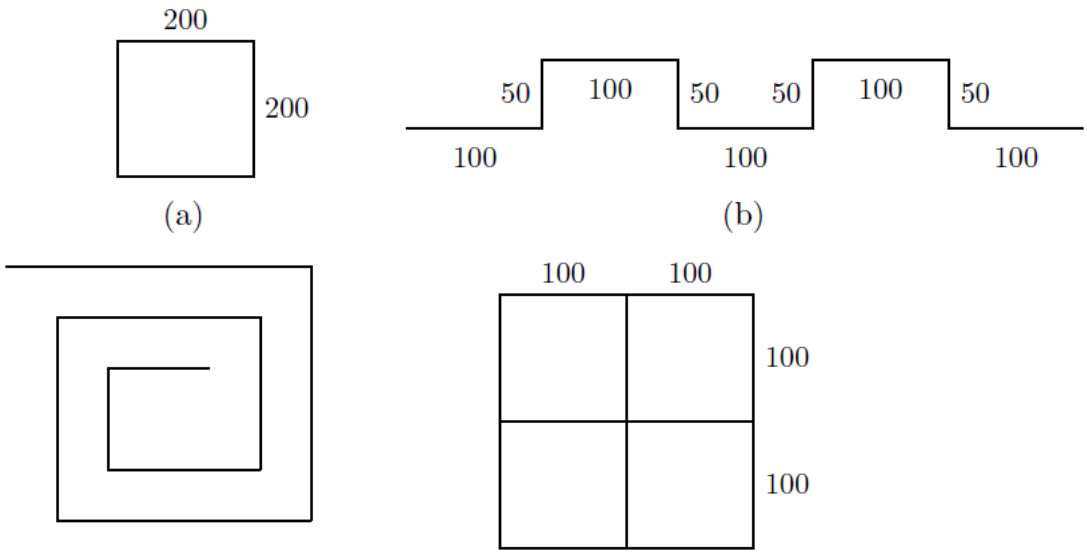
Ülesanne 1 Kilpkonn liigub

Sisesta järgmised käsud (võivad olla nii üksteise all kui kõrval) ja mõtle **täpselt** läbi, mida mingi käsk tegi:

```
fd 100 rt 90 fd 150 rt 90 fd 50 lt 90 fd 150 rt 90 fd 50
```

Ülesanne 2 Kujundid

Proovi samu käske kasutades joonistada järgmiseid kujundeid



1.2. Juhuslikud arvud ja muu juhuslikkus Pythonis

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

Funktsioonid juhuslike arvude tegemiseks tuleb eelnevalt vastavast moodulist importida:

```
import random
```

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

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

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

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

Näiteks:

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

1.3. Turtle moodul

Katsetame edasi Pythoni Turtle mooduliga - kilpkonnagraafika funktsioonikoguga. Mooduli funktsioonid tuleb importida:

```
from turtle import *
```

Seekord näeb importimise käsk välja pisut teistmoodi, et mooduli nime iga funktsiooni (käsu) ees mitte korrata.

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

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

NB! Pane tähele, et ehkki kilpkonna tegevus Pythoni Turtle mooduli käskudega (funktsioonidega) ning päris LOGO-keeles on sarnane, erineb käskude kasutamine siiski mingil määral - pööra tähelepanu sulgudele!

1.3.1. Joonistamine: variant 1. Kilpkonn liigub sinna, kuhu pea (või saba) näitab

Kilpkonnal on suund. Kui kilpkonnale öelda liigu edasi 50 sammu (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 3. Täisnurkne kolmnurk

Kuidas saada soovitud suuruses võrdhaarset täisnurkset kolmnurka? Täiendame programmi selliselt, et kasutajalt küsitakse kolmnurga kaatetite pikkused (võrdsed) ja edasi joonistatakse kolmnurk. Ehkki on ka mugavam viis joonistamise alguspunkti liikumiseks, proovi natuke matemaatikat - hüpotenuusi pikkus? Nurk ehk kui palju ja kuhu poole peab kilpkonna pöörama.

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 4 Lihtne liikumine

Proovi ise joonistada tähti või numbreid, kujundades nad kandilistena. 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()`).

1.3.2. Tsüklilause ja joonistamine

Eespool toodud piltidest nii sakke (b) kui ka kandilist spiraali saab joonistada tsüklit kasutades. Tegelikult ka ruutu.

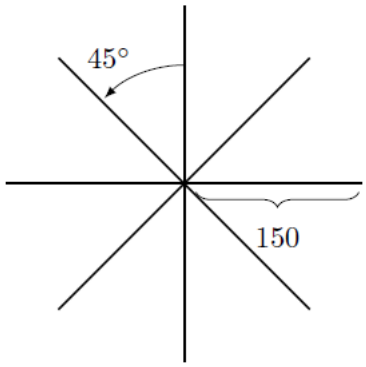
Näide **ruutu joonistamisest**:

```
for külg in range(4):
    fd(100)
    rt(90)
```

Üldistus: **hulknurga joonistamine**. Kuidas arvutada välja, mitu kraadi peab kilpkonn pöörama, et näiteks 8-nurka joonistada? 360 / 8. Kas see teooria sobib ka eelneva ruudu joonistamisega?

Ülesanne 5 Täheke

Kuidas joonistada täheke? (Vihje - kilpkonnaga sai ka tagurpidi liikuda.). Aga 16-haruline täheke? Kas oskame teha üldisema programmi, kus saame määrata tähekese harude arvu ning selle järgi muu vajaliku välja arvutada?



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

1.3.3. Isetehtud käsud ehk funktsioonid

Programmeerimiskeele käskudest saab moodustada uusi käske. Näiteks võime teha ise käsud tähekese või hulknurga joonistamiseks. Selliseid käske nimetatakse funktsioonideks.

Järgnev funktsioon `hulknurk` joonistab hulknurga sellise külgepikkuse ja nurkade arvuga, nagu soovitakse. Tähele tuleb panna, et funktsioon ei hakka ise vabatahtlikult tööle. Ta tuleb tööle panna ehk välja kutsuda ja selle käigus täpsustatakse hulknurga andmed.

```
def hulknurk(küljepikkus, nurkadearv):
    nurk = 360/nurkadearv
    for külg in range(nurkadearv):
        fd(küljepikkus)
        rt(nurk)
hulknurk(100, 3)   # Funktsiooni väljakutse, kus määratakse külje
pikkuseks 100 ja nurkade arvuks 3
```

Kas oskaksid teha funktsiooni tähekese joonistamiseks?

1.3.4. Joonistamine: variant 2. Kilpkonn liigub etteantud koordinaatidele

Joonistusaknas on x-y koordinaatteljestik. Punkt 0,0 jääb akna keskele (nagu koolis matematika tunnis).

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 külgsuunas. 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)
```

Laseme kilpkonna suvalist trajektoori mööda jalutama.

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

Ülesanne 6 Funktsiooni graafik

Proovime koos joonistada siinusfunktsiooni graafikut. Siin on vaja moodulist math kasutada funktsioone `sin()` ja `radians()`. Esimene leiab nurga siinuse, aga see nurk peab olema radiaanides. Ja selle jaoks on teine funktsioon.

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

1.3.5. 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(diameeter, värv)`). Kusjuures diameeter võib olla üsna suvalise suurusega. Kilpkonna hetke asukoht jääb täpi keskpunktiks.

Näiteks:

```
dot(100, "red")   # joonistab punase ringi (täpi) diameetriga 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.

1.3.6. Värvid

Kaks peamist värvi määramise võimalust on:

- nime 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: https://www.wikipython.com/tkinter-ttk-tix/summary-information/colors/

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

RGB mudelit saab Turtle moodulis kasutada kahte moodi:

Variant 1: tuleb määrata värvirežiim funktsiooniga `colormode(255)` - sel juhul saab määrata kõik mudeli värvid vahemikus `0 .. 255` ja kirjutada värvide väärtused ükshaaval sulgudesse. Näiteks `fillcolor(255, 0, 0)` annab värviks puhta punase.

Variant 2: Kolme värvi kombinatsioon esitatakse ühe kuueteistkümnendsüsteemi arvuna, kus kahe positsiooni kaupa on toodud kolme värvi väärtused. Näiteks `fillcolor("#FF00FF")` annab fuksiinpunase (*magenta*) tooni (punane ja sinine on maksimumis ning roheline puudub).

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
```

Loomulikult saab joonistada värvilist riskülikut (ja teisi hulknurki) ka varem vaadatud funktsiooniga.

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

Ülesanne 7 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.

1.4. Lisaks ja lõpetuseks

Kilpkonnafunktsioone 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).

Ja loomulikult internetist on leitav arvukalt näiteid, videoõpetusi ja muud sarnast.