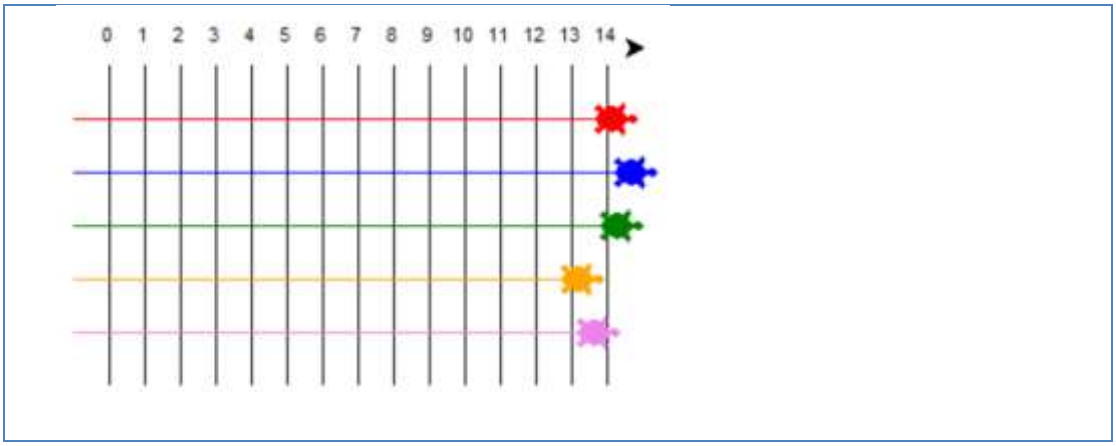
13. NODARBĪBA. UZDEVUMS "TURTLE RACE"

BRUŅRUPUČA GRAFIKA

Jūs esat izpildījāt uzdevumu, kas atrodas failā

13_Python_OOP_Turtle_turtle_race.pdf

Šajā uzdevumā Jūs izveidojāt bruņrupuču skriešanas sacensības ar vairākiem 4 – 5 bruņrupučiem.



Ja šis uzdevums nav izdarīts, tad izdariet to vispirms.

Lai izvairītos no koda atkārtošanās, veidojot katru bruņrupuci atsevišķi un piešķirot tam īpašības, Jūs izmantojāt ciklu un katru izveidoto bruņrupuci ievietojāt sarakstā.

```
# saraksts ar brunrupucu kraasaam
colors = ['red', 'blue', 'green', 'orange', 'violet']
# tukss saraksts, kuraa ievietos brunrupucus
turtles = []
for i in range(n):
    my_turtle = Turtle()
    my_turtle.color(colors[i])
    # iestata brunrupucim vajadziigaas iipasiibas
… … …
    turtles.append(my_turtle) # pievieno brunrupuci sarakstam
```

Arī bruņrupuča atrašanās vietu nosaka šajā ciklā, attiecīgi palielinot y koordināti.

Arī bruņrupuča skrējienā jāizmanto atkārtojuma konstrukcija.

Ja lekcijā uzdotais uzdevums nav pabeigts, pabeidziet to.

Svarīgi!

1) Bruņrupučiem jābūt izvietotiem sarakstā, tā, lai visas darbības ar bruņrupučiem notiktu, izmantojot ciklu.

2) Visām vērtībā, piemēram, bruņrupuču koordinātēm, skaitam un citiem lielumiem izmantojiet mainīgos, lai programma būtu viegli labojama un papildināma.

Tālāk jāpapildina iepriekš izveidotā programma.

1. uzdevums

Pievienojiet bruņrupuča pagriezienu.

Izmantojot ciklu: **for turn in range(x)**, lieciet katram bruņrupučiem veikt 360 grādu pagriezienu pēc tam, kad tas pirms skrējiena ir nostājies uz starta līnijas.

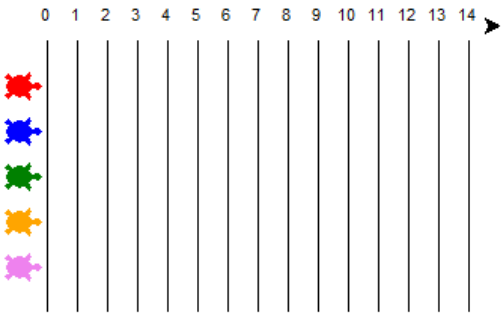
Svarīgi, lai pēc pagriešanās bruņrupucim būtu pareizs virziens, tā, lai tas varētu uzsākt skrējienu.

Piemēram, **ada.right**(36) liek bruņrupuci pagriezties par 36 grādiem.

Pilns pagrieziens ir 360 grādi. Bruņrupucis var griezties 10 grādus 36 reizes, 5 grādus 72 reizes vai jebkuru citu pagriezienu skaitu tā, lai kopā būtu 360 grādu.

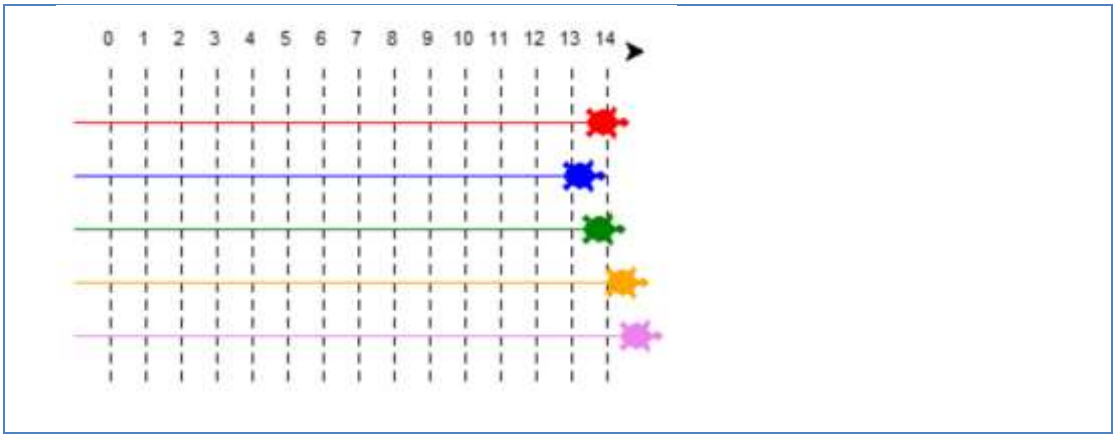
Šo pagriešanās ciklu jāievieto tajā koda daļā (ciklā), kur tiek izveidoti bruņrupuči un novietoti uz starta līnijas.

Šeit bruņrupuči pirms starta veic "deju" – pagriezienu par 360 grādiem.



2. uzdevums

Izveidojiet trasei raustītas līnijas.



Vienlaidus līniju vietā izveidojiet raustītas līnijas, kas iezīmē trasi.

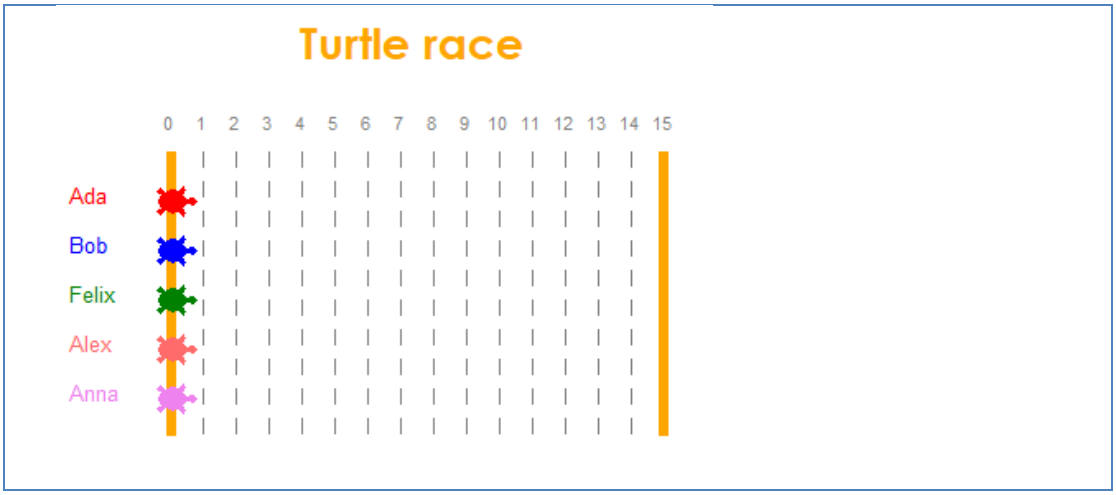
Lai to izdarītu, izteiksme, kas zīmē vienlaidus līniju, jāaizvieto ar **for** ciklu, kurā izsauc funkcijas **forward**(), **penup**() un **pendow**().

3. uzdevums

Noformējiet trasi atbilstoši paraugam:

- Pievienot virsrakstu
- Izveidojiet starta un finiša līnijas
- Pievienojiet uzrakstus ar visu bruņrupuču vārdiem

Pievērsiet uzmanību tam, ka bruņrupuči atrodas tieši uz starta līnijas.



Bruņrupuču vārdu uzglabāšanai izveidojiet atsevišķu sarakstu.

Katra bruņrupuča vārds tiek izvadīts tādā pašā krāsā kā bruņrupucis.

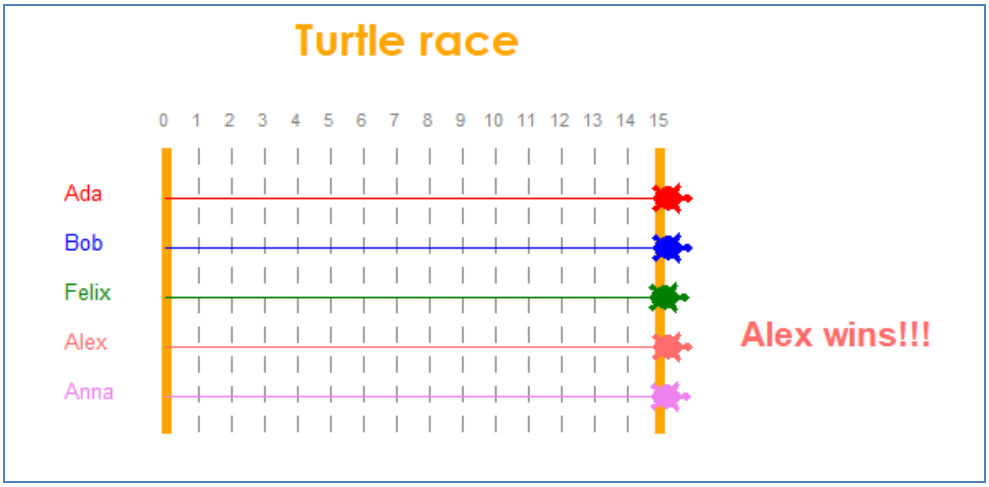
4. uzdevums

Uzvarētāja noskaidrošana.

Papildiniet programmu, lai pēc skrējiena tiktu parādīts uzvarētāja vārds.

Katrs bruņrupucis skrējienu turpina tik ilgi, kamēr sasniedz finiša līniju. Tiklīdz bruņrupucis ir sasniedzis finiša līniju, viņš skriešanu pārtrauc, bet pārējie turpina skriet.

Protams, pēdējā iterācijā bruņrupucis var nedaudz pārsniegt finiša līniju (katra soļa garums tiek noteikt ar gadījuma skaitļa ģeneratora palīdzību). Kad pēdējais bruņrupucis sasniedzis finišu, tiek izvadīts uzvarētāja vārds.



Šim nolūkam jāveic izmaiņas ciklā, kurā notiek bruņrupuča skriešana.

Sākumā bruņrupuča skriešana tika realizēta ar **for** cikla palīdzību, **for** cikla atkārtojumu skaits bija 100.

```
ada.goto(-160, 100)
ada.pendown()

for turn in range( 100 ):
    ada.forward(randint(1,5))
```

Tad tagad šis **for** cikls jāaizvieto ar **while** ciklu, kurš tiek atkārtots, līdz visi bruņrupuči sasnieguši finiša līniju.

Bruņrupuča objektam ir funkcija, ar kuras palīdzību var iegūt tā koordinātes. Šī funkcija ir jāizmanto programmā.

Katram bruņrupucim programmā jāuzglabā šāda informācija: cikla atkārtojumu skaits - to varam uzskatīt par laika vienībām, kurās bruņrupucis veicis skrējienu. Uzvarētājs ir tas bruņrupucis, kurš trasi veicis ar vismazāko iterāciju skaitu. Tomēr pastāv iespēja,

ka divi vai vairāki bruņrupuči trasi ir veikuši ar vienādu iterāciju skaitu, tādā gadījumā par uzvarētāju var pasludināt to bruņrupuci, kas ar pēdējo iterāciju tomēr ir aizskrējis tālāk.

Piemēram, izvadīta papildus informācija skrējienam:

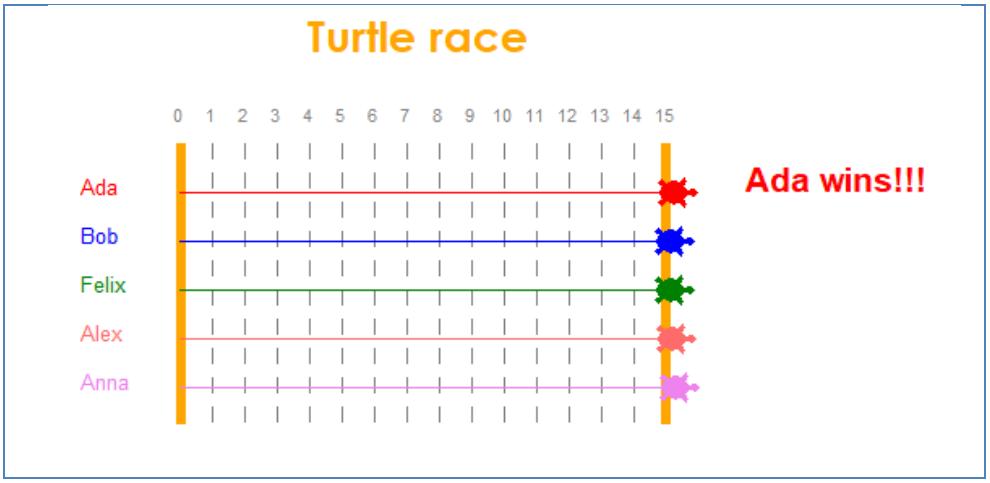
```
Finish: 160
Win:
Ada steps =  100 coord =  162.99999999999946
Win:
Alex steps =  100 coord =  162.99999999999952
Finished:
Felix steps =  101 coord =  160.99999999999946
Finished:
Bob steps =  105 coord =  162.99999999999952
Finished:
Anna steps =  108 coord =  161.9999999999995
-----------------
Winner is Alex
162.99999999999952
```

Jūs arī varat izvadīt šādu informāciju, tas palīdzēs kontrolēt programmas izpildi.

Pēc šiem datiem var redzēt, ka bruņrupuči Ada un Alex trasi ir veikuši ar vienādu iterāciju skaitu (`steps = 100`). Tomēr Alex koordināte (`coord = 162.99999999999952`) ir lielāka par Adas koordināti (`coord = 162.99999999999946`), tāpēc uzvarētājs ir Alex.

Apskatot šo papildus informāciju var redzēt, ka nebūtu grūti izvadīt arī visu skrējiena dalībnieku rezultātu tabulu, norādot katra bruņrupuča iegūto vietu.

Vēl viens programmas izpildes piemērs.



Izvadīta papildus informācija skrējienam:

```
Finish: 160
Win:
Ada steps =  94 coord =  162.9999999999995
Finished:
Bob steps =  97 coord =  160.9999999999995
Finished:
Felix steps =  106 coord =  160.99999999999957
Finished:
Alex steps =  106 coord =  161.99999999999952
Finished:
Anna steps =  107 coord =  163.99999999999952
-----------------
Winner is Ada
162.9999999999995
```

Šoreiz uzvarētājs ir bruņrupucis Ada, kas trasi veicis 94 iterācijās.