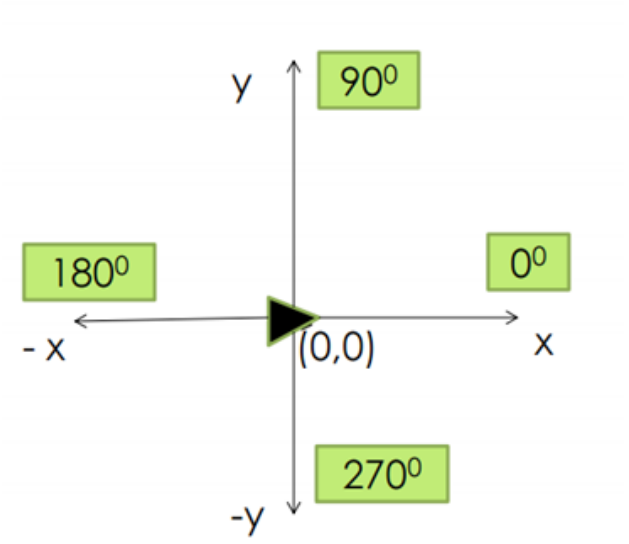
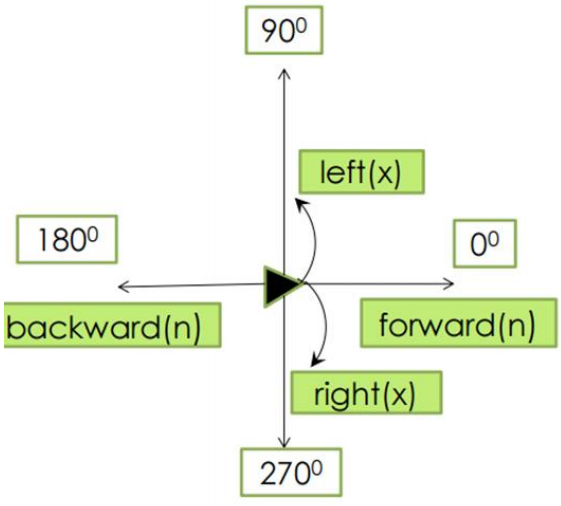
KORNJAČINA GRAFIKA

Za razumijevanje python kornjačine grafike, potrebno si je predočiti prostor u obliku koordinatnog sustava, raspodijeljenog u četiri kvadranta. Stupnjevi**: 0°, 90°, 180°, 270°** predočavaju zakretanje olovke (crnog trokuta) koji kreće iz ishodišta (0,0).



Kretanje bismo si predočili na ovakav način:



Pomicanje po koordinatnom sustavu se provodi ovisno na položaj crnog trokuta. Neki od položaja su prikazani u tablici ispod.

Funkcija	Opis djelovanja
forward(n), fd(n)	Pomiče olovku ravno za n koraka u smjeru u kojem je trenutno okrenuta
backward(n), bk(n), back(n)	Vraća olovku natrag za n koraka
right(x), rt(x)	„Zakreće“ olovku u smjeru smjera kazaljke na satu za x stupnjeva (x može biti realan broj)
left(x), lt(x)	„Zakreće“ olovku u smjeru obrnutom od smjera kazaljke na satu za x stupnjeva

Python grafiku u obliku kornjačine grafike, radi se pretežito s funkcijama i njezinim parametrima.

Funkcija	Opis djelovanja
goto(x, y), setpos(x, y), setposition(x, y)	Postavlja olovku na točku s koordinatama (x, y)
setx(x)	Postavlja prvu koordinatu olovke na x, a y ostaje nepromijenjena
sety(y)	Postavlja drugu koordinatu olovke na y, a x ostaje nepromijenjena
setheading(x), seth(x)	Usmjerava olovku tako da olovka pokazuje u smjer kuta x

Također u Pythonu razlikujemo funkcije za definiranje početnih pozicija olovke odnosno kornjače ili crnog trokuta.

Ostale funkcije za upravljanje olovkom (crnim trokutom):

Funkcija	Opis djelovanja
penup(), pu(), up()	Podiže olovku iznad papira tako da ne ostavlja trag prilikom kretanja
pendown(), pd(), down()	Vraća olovku na papir tako da sljedeće crtanje ostavlja trag
isdown()	Vraća True ako je olovka na papiru, inače vraća False
position(), pos()	Vraća trenutne koordinate olovke
heading()	Vraća smjer olovke
home()	Vraća olovku na sredinu grafičkog ekrana (0, 0) te postavlja smjer olovke na 0 stupnjeva – ukoliko je olovka na papiru ova naredba ostavlja trag
clear()	Briše sadržaj grafičkog prozora
undo()	Briše posljednji napravljeni korak
reset()	Briše sve crteže i postavlja olovku u početni položaj. (Radi isto što i kombinacija naredbi clear() i home().)
hideturtle(), ht()	Skriva olovku za crtanje
showturtle(), st()	Prikazuje olovku za crtanje
isvisible()	Vraća True ako je olovka za crtanje vidljiva, inače vraća False

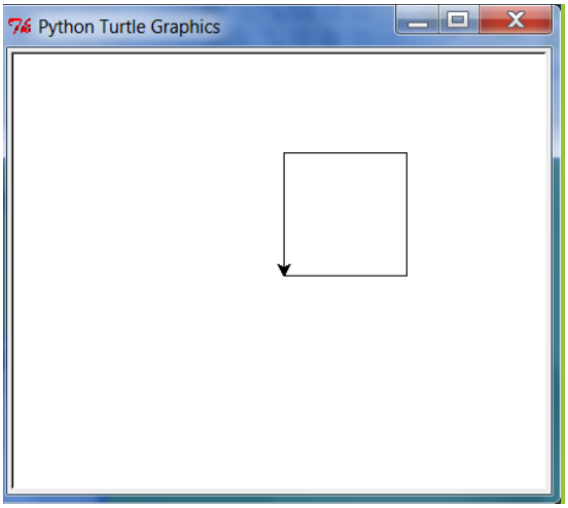
Uključivanje grafičkog sučelja kornjačine grafike radi se pozivom biblioteke **from turtle import*** . Na taj način smo omogućili pythonu da uz dodatna dva editora otvori i treći editor za iscrtavanje geometrijskih crteža koji se zove **Python Turtle Graphic.**

PRIMJERI:

Primjer 1: Potrebno je nacrtati kvadrat veličine 100 piksela.

```
from turtle import*

def main():
    for i in range(4):
        fd(100)
        lt(90)

main()
```

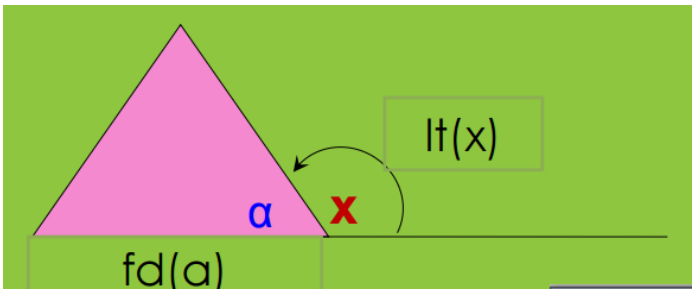
Pomoću funkcije smo četiri puta pozvali **fd** kao idi naprijed 100 piksela i **lt** odnosno zakreći olovku u obrnutom smjeru od smjera kazaljke na satu. Zakretanje se obavilo za 90°stupnjeva u lijevu stranu. Tako smo svaki puta nakon iscrtavanja 100 piksela dobili ravnu liniju koja se s ostale 3 linije pod kutem od 90°spojila u kvadrat.



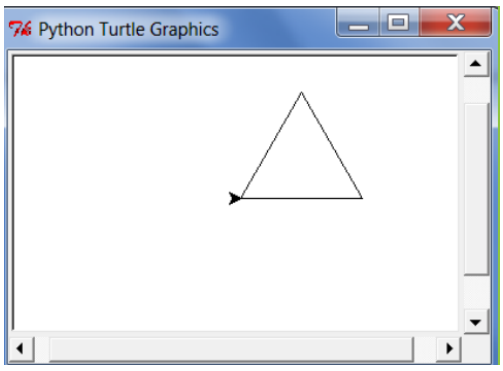
Primjer 2: Potrebnoje nacrtati jednakostraničan trokut stranica duljine 100 piksela. Stranice moraju biti pod kutem od 120°.

```
from turtle import*

def main():
    for i in range(3):
        fd(100)
        lt(120)

main()
```

Dakle najprije treba porazmisliti kojim smjerom krenuti u koordinatnom sustavu. Ako bi krenuli crtati iz ishodišta, mogli bi recimo krenuti u lijevu stranu pod odgovarajućim kutem (120°) kreč.ući se sa 100 piksela unatrag.



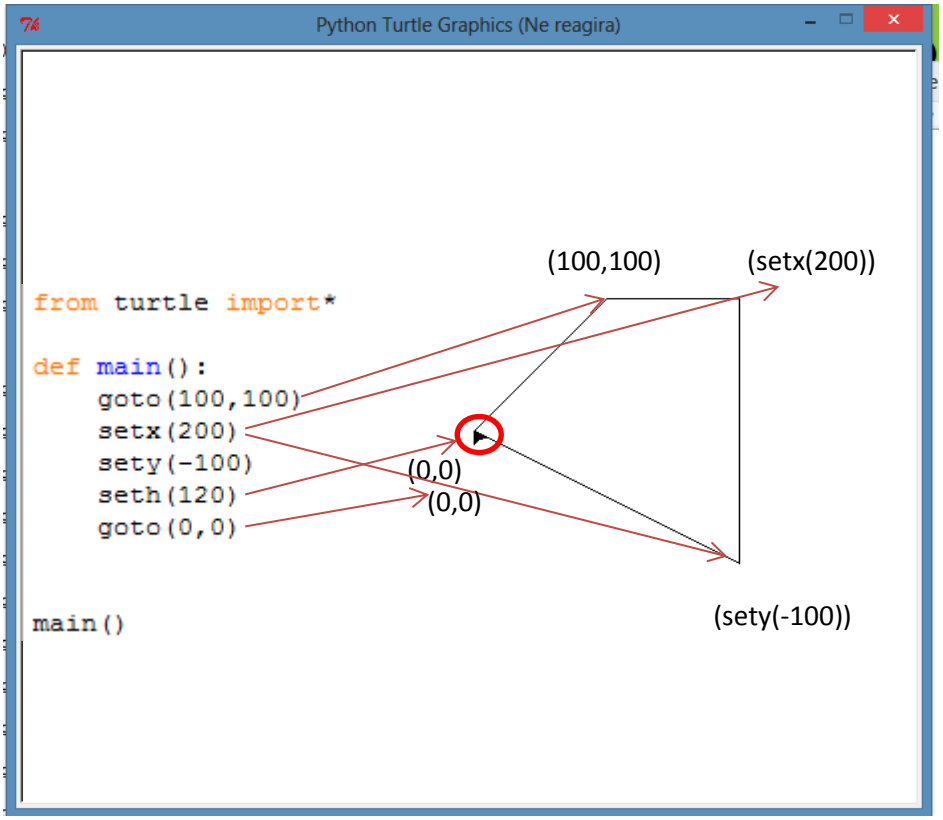
Krečući se u lijevu stranu, napravili bismo pravac točno 100 piksela, zatim bi u for petlji tri puta (jer troktu ima tri stranice) napravili isto: otklon kuta od 120° i pravac u 100 piksela. Tako bismo si omogućili spajanje pravaca i stvaranje vrhova trokuta.



Primjer 3: Potrebno je nacrtati n-terokut. Najprije se x i y os pozicionirati na 100 piksela. Nakon toga x postaviti na 200 piksela, zatim y postaviti na -100 piksela. Zaokrenuti olovku za 120°i na kraju olovku vratiti na početnu poziciju iz koje je krenula (iz ishodišta).

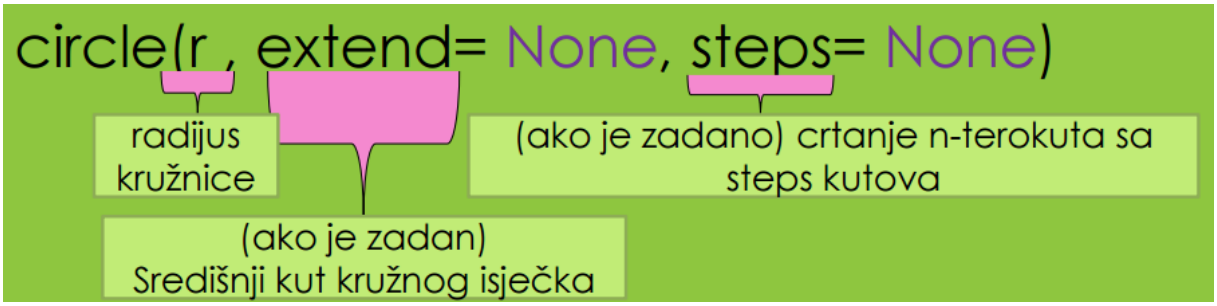
```
from turtle import*

def main():
    goto(100,100)
    setx(200)
    sety(-100)
    seth(120)
    goto(0,0)


main()
```

Naredbom goto smo odredili poziciju pravca koji izlazi iz ishodišta i završava u sjecištu x i y osi (100,100). Nakon toga smo za pojedinačnu os zadavali pravce u odgovarajućoj duljini u pikselima. Nakon toga smo pomoću **seth** zaokrenuli olovku za 120°i vratili je u početnu poziciju to jest u ishodište.



Primjer 4: Crtanje kruga i romboida.

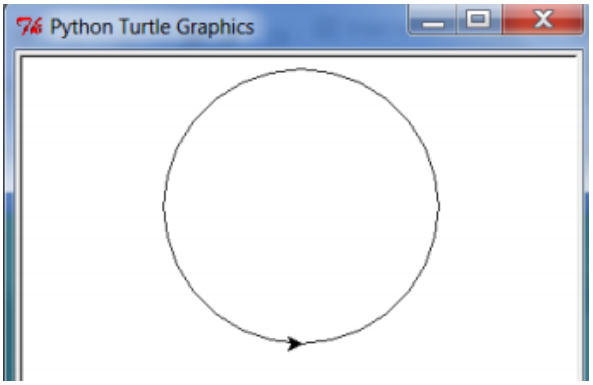
Da bi mogli nacrtati krug, mormao prvo vidjeti deklaraciju naradbe za crtanej kruga:



Dakle prvi parametar uvijek podrazumijeva određivanje radijusa kruga. Nakon toga određujemo kut kružnog isječka to jest graničnu vrijednost ispisivanja kruga. Ako je zadano **steps** onda određujemo broj kutova kako bismo recimo nacrtali romboid.

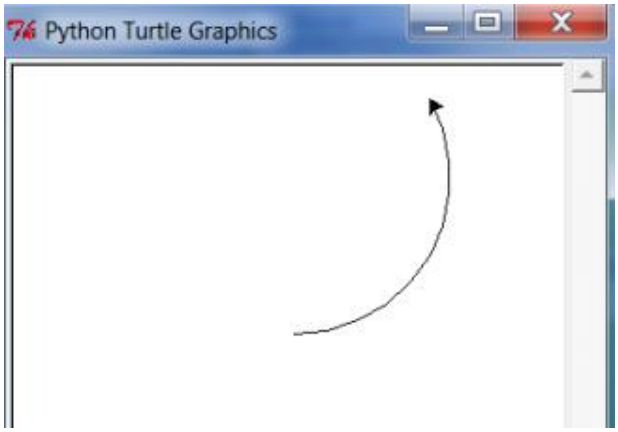
Primjer 4a: Nacrtati krug radijusa 100.

```
from turtle import*

def main():
    circle(100)


main()
```



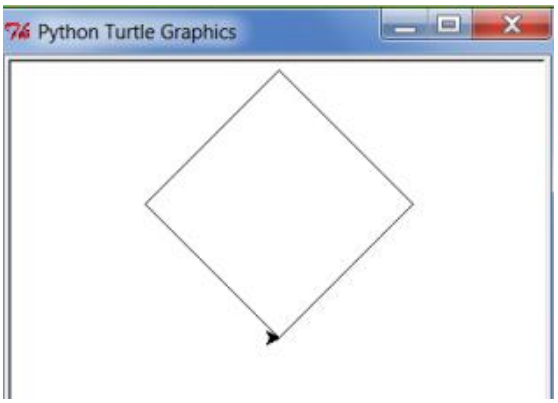
Primjer 4b: Nacrtaj krug radijusa 100 i kuta zakreta 120°

```
from turtle import*

def main():
    circle(100,120)


main()
```



Primjer 4c: Nacrtati romboid radijusa 100.

```
from turtle import*

def main():
    circle(100,360,4)


main()
```

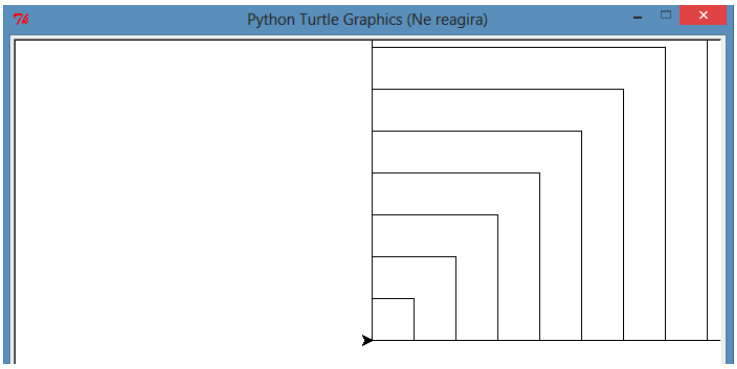


Najprije smo postavili radijus na 100 piksela Taj radijus kod romboida označava udaljenost kuteva. Točnije to je dijagonala koja dodiruje nasuprotne kutove romboida. Kružni isječaj od 360° znači da će se iscrtati cijeli to jest puni krug. U slučaju crtanaj romboida to nam je sigurnost da će sve stranice i kutoevi biti iscrtani. Broj kuteva smo odredili na 4 što znači da imamo izoktrenuti kvadrat odnosno romboid koji se sastoji od 4 vrha.

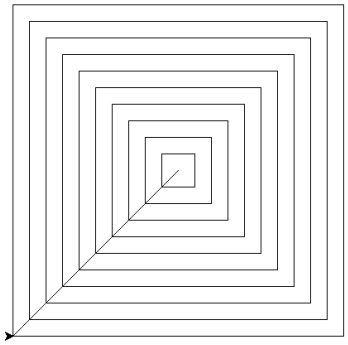
Primjer 5: Nacrtati 10 kvadrata ako da najmanji kvadrat ima stranicu dugu 40 piksela a najveći 400 piksela.Pozicionirati kvadrate u ishodište koordinatnog sustava.

```
from turtle import*

def kvadrat(stranica):
    goto(0,0)
    for i in range(4):
        fd(stranica)
        lt(90)


def main():
    for i in range(10):
        stranica=(i+1)*40
        kvadrat(stranica)

main()
```

U programu smo koristili dvije funkcije od kojih je funkcija **kvadrat** koristila za iscrtavanej kvadrata s pozicijom u ishodištu koordinatnog sustava **(0,0).** Zbog toga jer nam je u zadatku određeno da duljina stranica mora varirati između 40 i 400 umjesto konkretne vrijednosti u naredbi **fd** napisali smo argument funkcije **stranica.** Taj argument je poslužio za automatsko upisivanje veličina redom 10 kvadrata koje je trebalo iscrtati. Kut je određen fiksno od 90°u smjeru suprotnom od kazaljke na satu. Dakle funkcija **kvadrat()** omogućila je crtanej jednog kvadrata. To iscrtavanej jednog kvadrata, umnoženo je 10 puta drugom funkcijom. Druga funkcija **main()** bez argumenta, omogućila je kreiranje 10 kvadrata različitih duljina. Bitna je linija koda **stranica=(i+1)*40.** Svaku vrijednost kvadrata, zapisali smo u **stranica**. Prva vrijednost s kojom je započeto crtanje je 1. U for petlji smo definirali ponavljanje 10 puta čime je petlja započela brojati od 1 prema 10. Tako da se svaki broj počevši od 1 množio s početnom duljinom stranica najmanjeg kvadrata **40 piksela.** Dio koda **i+1** označava da se svaka iduća vrijednost uvećava za jedan. Dakle prva vriejdnost u toj liniji koda bi bila **stranica=(0+1)*40**, zatim **stranica=(1+1)*40, stranica=(2+1)*40.... stranica=(9+1)*40.** Budući da je kvadrat započet s crtanjem u ishodištu, olovka će se uvijek pozicionirati na ishodište i iz njega crtati kvadrate sve većih duljina stranica počevši od donjeg lijevog kuta. Stoga rješenje izgleda ovako:



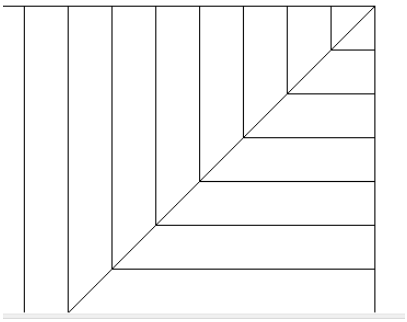
Primjer 6: Nacrtati 10 kvadrata ako da najmanji kvadrat ima stranicu dugu 40 piksela a najveći 400 piksela.Pozicionirati kvadrate na sljedeći način:



U odnosu na prethodni primjer, crtanje kvadrata pozicionirali smo u negativnu xy-os. Koristili smo argument funkciej kvadrat za pozicioniranje crtanej kvadrata. Budući da je zadano da python svaki puta započinje crtanje od donjeg lijevog kuta, potrebno ga je bilo pozicionirati u negativnu xy-os. Time je postignuto da nakon svakog nacrtanog kvadrata, python se vrati u sjecište dviju negativnih osi s pomakom. Pomak je omogućen u liniji koda **stranica=(i+1)*40.**

```
from turtle import*

def kvadrat(stranica):
    goto(-stranica/2,-stranica/2)
    for i in range(4):
        fd(stranica)
        lt(90)


def main():
    for i in range(10):
        stranica=(i+1)*40
        kvadrat(stranica)

main()
```

Svaki puta kada se broj kvadrata u for petlji uvećava za jedan i množi sa 40 u funkciji **kvadrat()** ta vrijednost se prebacuje u negativnu xy-os. Dakle kada je na primjer za kvadrat 2 u liniji koda **stranica=(i+1)*40** ispisan rezultat stranica=(2+1)*40 odnosno 120 tada u liniji koda **goto(-stranica,-stranica)** bude zapisana vrijednost **(-120,-120)** i tako uzastopce za svaki kvadrat. Naravno vrijednost (-120,-120) u negativnoj xy-osi predstavlja iduće sjecište iz kojeg python počinje crtati idući kvadrat. Kada pozicionirane vrijednosti stranica ne bi bile prepolovljene na dva dijela kao u naredbi **goto(-stranica/2,-stranica/2),** onda bi python iscrtavao nešto poput ovoga:



Uvijek bi kretao od ishodišta nazančenog u naredbi goto(,) samo što bi se kvadrati iscrtavali svaki puta u donjem lijevom kvadrantu koordinatnog sustava i povećanjem duljine stranica kvadrata, prema sjecištu negativnih vrijednosti povlačili sve dublje u donji lijevi kvadrant koordinatnog sustava. Ako ih podijelimo sa dva na primjer onda smo postigli da se sjecište negativnih vrijednosti prepolovi dio na pozitivni dio a dio na negativni dio koordinatnog sustava. I tek tada dobivamo točan rezultat: