Ping pong

NAPOMENA PRIJE ČITANJA: Sav kôd koji sam upisao u ovaj dokument i koji sam ujedno upisao u svoj program, bit će *nakošen*. Sav kôd pisan je izvan petlje za igru ako drukčije nije naznačeno. Kod očitavanja koordinata pojedinih objekata, uvijek se koordinata, tj. pozicija tog objekta, mjeri od središta (to sam na nekoliko mjesta napomenuo, ali to nije tako samo gdje je napomenuto već uvijek pa sam to samo ovdje želio istaknuti).

Kada smo počeli upisivati za naš projekt u tablicu, svi su upisivali igrice pa sam zaključio da bih i ja mogao napraviti jednu. Nakon nekog vremena, odlučio sam se za ping pong. Na tu ideju sam došao na jako zanimljiv način. Naime, odradio sam sve školske obaveze za taj dan i odlučio sam se malo odmoriti gledajući televiziju, ali na televiziji nije bilo ništa pretjerano zanimljivo pa sam samo prebacivao programe kako bih možda uspio pronaći nešto zanimljivo. Prebacujući tako programe, naišao sam na emisiju o životu u sedamdesetim godinama prošlog stoljeća. Nisam pogledao cijelu emisiju već sam samo pročitao sadržaj emisije, saznao njen sadržaj, a obzirom da mi on nije bio pretjerano zanimljiv, odlučio sam tražiti dalje što ću gledati. No baš u trenutku kada sam želio prebaciti na sljedeći program, počeli su govoriti o prvim igricama. To su naravno jednostavne igrice za koje mi znamo jer su prije bile jako popularne. Rekli su nešto više o igrici ping pong. Ta mi se igrica nekako uvijek sviđala, valjda zato što je retro i poprilično jednostavna. Pogledao sam prilog o igricama do kraja, a kada su počeli govoriti o modi tih vremena, prebacio sam program. Na ideju da baš ova igrica bude moj projekt nisam došao odmah već u trenutku kada sam morao donijeti konačnu odluku o tome koja će biti tema mog projekta. Želio sam upisati temu, ali jednostavno nisam imao ideje. Tada mi je kroz glavu prošla misao da bih mogao napraviti igricu ping pong. Nije djelovala presloženo i kao nešto što ne bih mogao napraviti jer su neki imali i složenije igrice. Tako je konačna odluka donesena i počeo sam raditi na projektu.

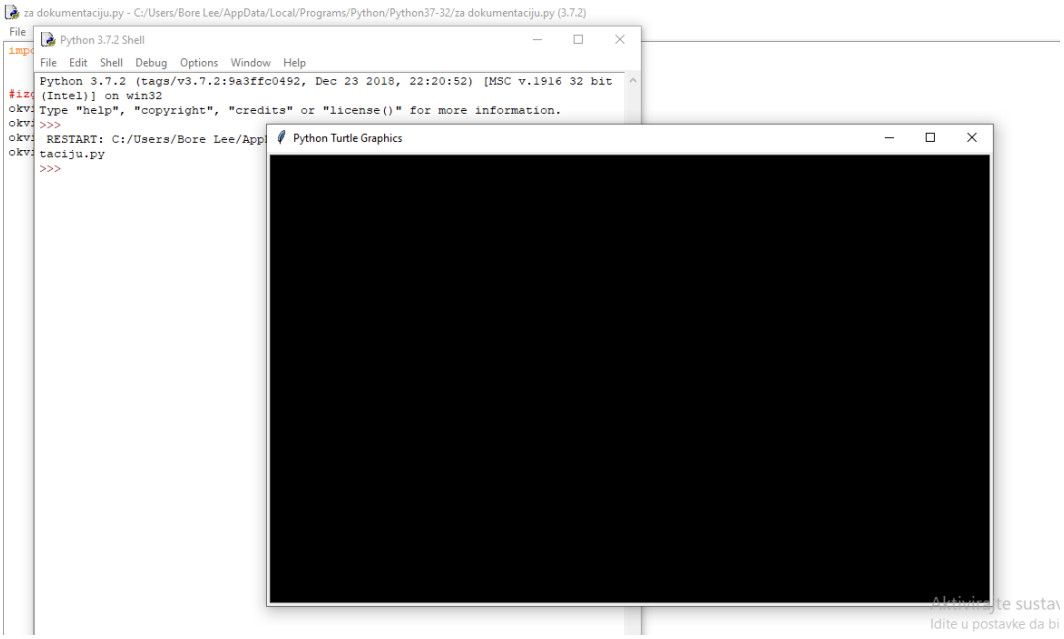
Sada kada sam znao što je moja tema, mogao sam početi. Obzirom da nisam znao ništa o programiranju igrica u Pythonu, ali ni o programiranju igrica općenito, morao sam pronaći način kako uopće započeti s programiranjem igrica. Pronašao sam dva načina koja se najčešće spominju: korištenjem („importanjem“) biblioteke „turtle“ ili biblioteke „pygame“. Gledanjem videozapisa na internetu, ali i čitanjem raznih foruma, zaključio sam da je „turtle“ puno jednostavniji za početnike kao ja. U ovome trenutku sam morao razmisliti što sve moram imati kako bi moja igrica funkcionirala. Obzirom da u „Shellu“ u Pythonu ne mogu napraviti pokretne oblike (ploču, loptice), bio mi je potreban novi prozor u koji ću smjestiti svoju igricu. Kao što sam već rekao, „importao“ sam „turtle“, a zatim pomoću klase (*okvir=turtle.Screen()*) stvorio okvir kojemu sam odredio boju pozadine (*okvir.bgcolor()*) i dimenzije (*okvir.setup(width=800, height=500)*) u pikselima. Važno je napomenuti jednu stvar, a to je da je ovaj okvir kao koordinatni sustav u matematici, tj. svaki pisel ovog okvira je određen svojom koordinatom. Na središtu okvira se nalazi točka s koordinatama (0,0), horizontalno se pruža os x, a vertikalno os y. Te osi naravno imaju svoje pozitivne i negativne dijelove. Zanimljivo je što radi *okvir.tracer(0)*. To omogućava da se animacije u prozoru zbivaju jako brzo, točnije najbrže moguće zato što ova nula označava brzinu kojom će se zbivati animacije, a maksimalna brzina se označava nulom. Bez *okvir.tracer(0)* bismo vidjeli kako „turtle“ stvara svaki pojedini dio igre koji ćemo kasnije dodati (npr. vidjeli bismo kako „turtle“ stvara lopticu i lijevu i desnu ploču te bi se loptica jako sporo gibala). Kôd za to izgleda ovako:

Untitled

File Edit Format Run Options Window Help

```
import turtle


#izgled igraćeg okvira (boja, dimenzije)
okvir=turtle.Screen()
okvir.bgcolor('black')
okvir.setup(width=800, height=500)
okvir.tracer(0)
```

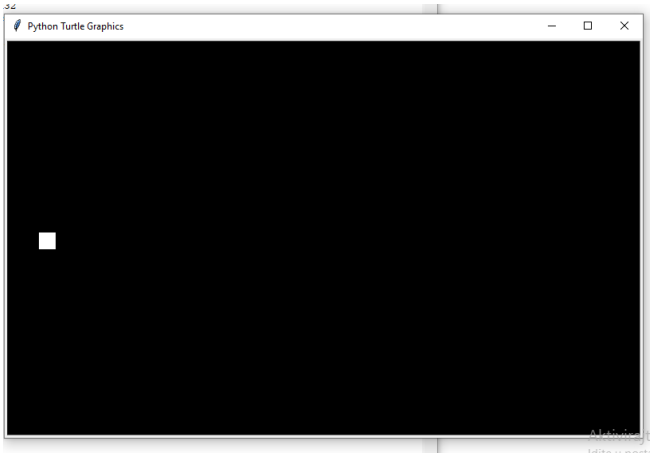
Ovdje je i fotografija kako izgleda taj prazni okvir.



Zatim sam morao dodati objekte koji su mi potrebni kako bih igrica funkcionirala (pod time mislim na ploče i lopticu). Objasnit ću postupak samo za jednu ploču i za lopticu jer je za drugu ploču postupak gotovo isti. Lijevu ploču sam isto tako stvorio pomoću klase (*ploča_l=turtle.Turtle()*). Ploča_l je lijeva ploča. Zatim sam odredio oblik ploče. Kada upišemo *ploča_l.shape()* možemo unutar ovih zagrada upisati neki oblik koji će program automatski generirati. Ja sam odabrao da to bude kvadrat pa to zapisujemo ovako: *ploča_l.shape('square')*. Međutim, kada sam pokrenuo program, u prozoru mi nije bilo ništa vidljivo. Kako bi objekti bili vidljivi, moramo u petlju koja se odnosi na igru (ja ću tu petlju zvati petlja za igru) upisati slijedeće:

```
#petlja za igru
while True:
    okvir.update()
```

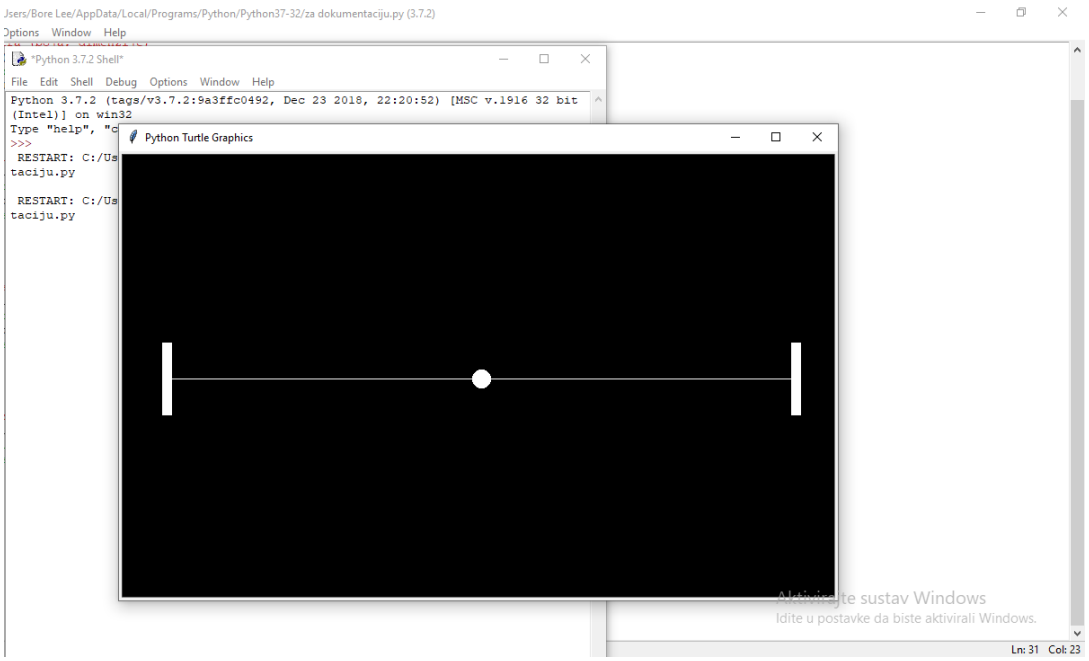
Ovo nam omogućava da se okvir stalno „updatea“ pa će svi objekti koje budemo dodavali u okvir, ali i njihovo kretanje, biti vidljivi. Kada sam to upisao, mogao sam vidjeti kako izgleda lijeva ploča. Izgledala je ovako:



Obzirom da ploče u igrici trebaju biti pravokutnog oblika, upisao sam *ploča_l.shapesize (stretch_wid=4, stretch_len=0.5)* kako bi kvadrat bio dulji četiri puta i duplo uži (oblik pravokutnika). Odredio sam da bude bijele boje. Pomoću *ploča_l.goto(0,-350)* sam oderdio da ta ploča, tj. njeno središte, bude na koordinatama y=-350 i x=0 (u sredini prozora po osi y) u okviru. Upisao sam I *ploča_l.speed(0)* kako bi se buduće animacije ,u ovome slučaju pomak ploče (ali kasnije pomak druge ploče, loptice i prikaz rezultata), odvijao najvećom mogućom brzinom. Isto sam napravio za desnu ploču (osim što sam je nazvao ploča_d i smjestio na koordinate y=350 i x=0). Za lopticu sam napravio slično (osim što sam je nazvao loptica, postavio da je loptica okruglog oblika *loptica.shape('circle')* te je stavio je na koordinate x=0, y=0). Loptici nisam mjenjao veličinu jer sam bio zadovoljan koliko je velika u odnosu na ploče koje sam napravio. Kôd za ove objekte je izgledao ovako:

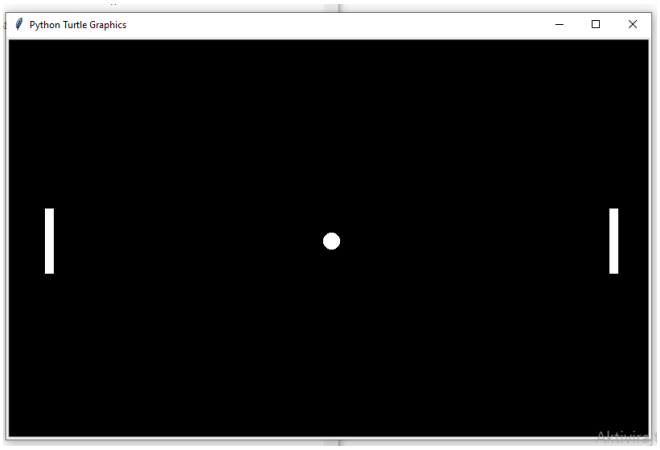
```
#izgled igraćeg okvira (boja, dimenzije)
okvir=turtle.Screen()
okvir.bgcolor('black')
okvir.setup(width=800, height=500)
okvir.tracer(0)



#lijeva ploča, tj. lijevi igrač (izgled,pozicija u okviru,...)
ploča_l=turtle.Turtle()
ploča_l.shape('square')
ploča_l.shapesize(stretch_wid=4, stretch_len=0.5)
ploča_l.color('white')
ploča_l.goto(-350,0)
ploča_l.speed(0)


#desna ploča, tj. desni igrač (izgled,pozicija u okviru,...)
ploča_d=turtle.Turtle()
ploča_d.shape('square')
ploča_d.shapesize(stretch_wid=4, stretch_len=0.5)
ploča_d.color('white')
ploča_d.goto(350,0)
ploča_d.speed(0)


#loptica (izgled,pozicija u okviru,...)
loptica=turtle.Turtle()
loptica.shape('circle')
loptica.color('white')
loptica.goto(0,0)
loptica.speed(0)
```

Kôd sam naravno dodavao na onaj koji sam već imao, tj. novi kôd sam pisao ispod i nisam ništa brisao, ali to nije vidljivo iz ovih fotografija i neće biti vidljivo iz razloga što bi onda ovdje bilo previše fotografija. Kada sam pokrenuo svoj kôd, okvir je izgledao ovako:



Jasno je da ova crta po sredini okvira ne smije postojati. Bio sam malo zbunjen zašto je ona ovdje, ali sam se onda sjetio da sve ovo zapravo „crta" kornjača, a ona za sobom ostavlja trag. Taj trag možemo maknuti ako kornjači „naredimo" da ne ostavlja trag za sobom dok se kreće te joj to obavezno moramo „narediti" prije nego što završi s crtanjem ovih objekata jer inače ta naredba neće imati utjecaja, tj. crta će i dalje postojati. Naredba glasi *ploča_l.penup()*, odnosno *ploča_d.penup()* i *loptica.penup()*. Kôd i okvir tada izgledaju ovako:

```
#lijeva ploča, tj. lijevi igrač (izgled,pozicija u okviru,...)
ploča_l=turtle.Turtle()
ploča_l.penup()
ploča_l.shape('square')
ploča_l.shapesize(stretch_wid=4, stretch_len=0.5)
ploča_l.color('white')
ploča_l.goto(-350,0)
ploča_l.speed(0)


#desna ploča, tj. desni igrač (izgled,pozicija u okviru,...)
ploča_d=turtle.Turtle()
ploča_l.penup()
ploča_d.shape('square')
ploča_d.shapesize(stretch_wid=4, stretch_len=0.5)
ploča_d.color('white')
ploča_d.goto(350,0)
ploča_d.speed(0)


#loptica (izgled,pozicija u okviru,...)
loptica=turtle.Turtle()
ploča_l.penup()
loptica.shape('circle')
loptica.color('white')
loptica.goto(0,0)
loptica.speed(0)
```



Sada sam morao „pokrenuti" ploče i lopticu. Ploče sam pokrenuo tako što sam definirao funkciju *def ploča_l_gore()* kojom ću odrediti za koliko će se pomaknuti lijeva ploča prema gore pritiskom tipke na tipkovnici. Zatim sam napisao *y=ploča_l.ycor()* (*.ycor()* očitava koordinatu na y osi na kojoj se nalazi središte ploče). Pomoću *y+=20* sam odredio za koliko će se piksela ploča pomaknuti kada uvedem da se ploča kontrolira pomoću klika tipke na tipkovnici. Na kraju sam dodao *ploča_l.sety(y)*, što služi da se taj novi *y* koji je nastao klikom na tipkovnicu pohrani kao novi *y*. Isto sam napravio i za pomak lijeve ploče dolje (samo što sam napisao *y-=20* jer se ploča pomiče prema dolje). Kôd koji sam napisao je izgledao ovako:

```
#za koliko se lijeva ploča pomiče gore pritiskom tipke na tipkovnici
def ploča_l_gore():
        y=ploča_l.ycor()
        y+=20
        ploča_l.sety(y)


#za koliko se lijeva ploča pomiče dolje pritiskom tipke na tipkovnici
def ploča_l_dolje():
        y=ploča_l.ycor()
        y-=20
        ploča_l.sety(y)

#za koliko se desna ploča pomiče gore pritiskom tipke na tipkovnici
def ploča_d_gore():
        y=ploča_d.ycor()
        y+=20
        ploča_d.sety(y)


#za koliko se desna ploča pomiče dolje pritiskom tipke na tipkovnici
def ploča_d_dolje():
        y=ploča_d.ycor()
        y-=20
        ploča_d.sety(y)
```

Sada sam morao omogućiti da se klikom tipke na tipkovnici ploče pomiču za vrijednosti koje su prethodno određene. Prvo sam napisao *okvir.listen ()*, da okvir „sluša“ naredbe tipkovnice, tj. klikom tipke na tipkovnici se neka animacija može dogoditi u okviru. Zatim sam napisao *okvir.onkeypress (ploča_l_gore, 'w')*. To znači da sam se pozvao na funkciju *ploča_l_gore* i da će se *y* ploče povećati za 20, tj. ploča ide prema gore za 20 piksela, ako se klikne tipka W. To mi je omogućila naredba *.onkeypress*, a unutar zagrada sam upisao na koju se funkciju pozivam te potom klikom na koju tipku će se izvršiti radnja u toj funkciji. Tako sam napravio i za pomak prema dolje za lijevu ploču i za pomake prema gore i dolje za desnu ploču. Tipke su klasične kao i na svim igricama za dva igrača, lijevi igrač, tj. ploča, se upravlja pomoću tipki W i S, a desni igrač, tj. ploča, pomoću strelica s desne strane tipkovnice. Kôd:

```
#pomicanje ploča pomoću tipkovnica-kontrole igre
okvir.listen()
okvir.onkeypress(ploča_l_gore,'w')
okvir.onkeypress(ploča_l_dolje,'s')
okvir.onkeypress(ploča_d_gore,'Up')
okvir.onkeypress(ploča_d_dolje,'Down')
```

Da pokrenem lopticu, morao sam tamo gdje sam već definirao lopticu kao objekt dodati *loptica.dx=0.3* i *loptica.dy=-0.3*. To je pomak po x i y osi (slovo „d” ispred x i y označava „delta”-promjena) koji će prelaziti loptica spontano prelaziti (pomicat će se „sama od sebe”, bez da joj dajemo neke nove naredbe). Vrijednosti uz .dx i .dy označavaju koliko se brzo mijenjaju koordinate x i y osi. Što su te vrijednosti veće, promjene koordinata x i y su veće, tj. loptica se brže kreće:

```
#loptica (izgled,pozicija u okviru,...)
loptica=turtle.Turtle()
loptica.penup()
loptica.shape('circle')
loptica.color('white')
loptica.goto(0,0)
loptica.speed(0)
loptica.dx=0.3
loptica.dy=-0.3
```

Zatim ću u petlju za igru dodati da se trenutačne x i y koordinate (*loptica.xcor()* i *loptica.ycor()*) loptice zbrajaju s upravo spomenutim pomakom loptice po x i y (*loptica.dx()* i *loptica.dy()*) osi te da taj novi x i y postaju koordinate na koje će se loptica trenutačno pomaknuti (loptica se pomiče na te nove koordinate pomoću *loptica.setx()* i *loptica.sety()*, uz naravno napisan kôd unutar zagrada koji se nalazi na fotografiji):

```
#petlja za igru
while True:
    okvir.update()


    loptica.setx(loptica.xcor()+loptica.dx)  #pomak loptice
    loptica.sety(loptica.ycor()+loptica.dy)
```

Loptica se sada spontano pomiče, ali izlazi izvan okvira. Kako bismo napravili da ne izlazi iz okvira koristit ćemo „if" petlje. U petlju za igru ćemo upisati slijedeće:

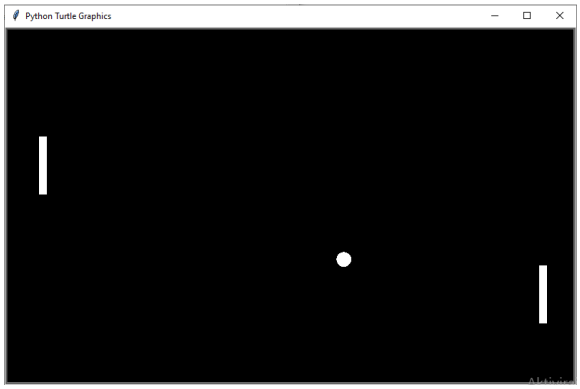
```
    if loptica.ycor()>240:  #loptica ne može otići iz okvira na gore/dolje
        loptica.sety(240)
        loptica.dy*=-1

    if loptica.ycor()<-240:
        loptica.sety(-240)
        loptica.dy*=-1

    if loptica.xcor()>390:  #loptica ne može otići iz okvira lijevo/desno
        loptica.goto(0,0)
        loptica.dx*=-1

    if loptica.xcor()<-390:
        loptica.goto(0,0)
        loptica.dx*=-1
```

Objasnit ću na primjeru prve „if" petlje kako sam napravio da loptica ne izlazi izvan okvira s gornje strane. Petlja govori ako y koordinata središta loptice bude veća od 240 piksela na gornjoj strani ploče (y=240), y loptice postane 240 (y=240 zbog *.sety()*) te da se loptica počne kretati u suptotnom smjeru od onog iz kojeg je došla, odbit će se (to je omogućilo *loptica.dy*=-1*. *.dx* i *.dy* su pomaci loptice koje smo prije definirali pa će *.dx* i *.dy* će se pomnožiti s -1, tj. predznak te promjene koordinate će se promijeniti, npr. ako je *dx*=1, objekt kojemu je ovaj *dx* pridružen će se gibati prema pozitivnom smjeru osi x, a množenjem s -1, *dx* će iznositi -1 pa će se gibati prema negativnom dijelu osi x). Objašnjenje kôda da loptica ne izlazi iz okvira s donje stane: ako se središte koordinate loptice nađe na koordinatama y=-240, trenutačni y loptice će postati -240 i loptica će se opet kretati u suprotnom smjeru od onog iz kojeg je došla (ponovno zahvaljujući *loptica.dy*=-1*). Međutim, kako bi igra funkcionirala, kada loptica dotakne lijevu i desnu, ona mora doći točno u središte ploče i gibati se u suprotnome smjeru od onoga iz kojeg je došla (npr. ako se loptica kretala slijeva udesno prilikom dodira okvira, mora se stvoriti na sredini okvira i početi gibati zdesna nalijevo). To sam napravio kao što je već vidljivo na gornjoj fotografiji tako što sam pomoću „if" petlje odredio da ako je x koordinata loptice veća od 390, ona će se premjestiti na centar (*loptica.goto(0,0)*) i početi gibati u suprotnome smjeru (*loptica.dx*=-1*). Objašnjenje množenja *.dx* i *.dy* s -1: *.dx* i *.dy* su pomaci loptice koje smo prije definirali pa će se *.dx* i *.dy* pomnožiti s -1, tj. predznak te promjene koordinate će se promijeniti, npr. ako je *dx*=1, objekt kojemu je ovaj *dx* pridružen će se gibati prema pozitivnom smjeru osi x, a množenjem s -1, *dx* će iznositi -1 pa će se gibati prema negativnom dijelu osi x. Isto je i za os y.

Sada kada loptica ne izlazi izvan okvira, morao sam napraviti da se loptica odbija od ploča. Kôd sam upisao u petlju za igru. Objasnit ću samo kôd prve „if“ petlje, tj. kôd da se loptica odbije od desne ploče, jer je za odbijanje od lijeve ploče sve gotovo isto (razlika je samo u koordinatama na kojima se nalazi lijeva ploča). Ako je x koordinata loptice u intervalu od 340 do 350 (širina ploče) ***i*** ako je y koordinata loptice veća od y koordinate desne ploče kojoj je oduzeta ili pribrojena njena duljina koja iznosi 50 piksela (iako nisam uspio pronaći kolike su dimenzije oblika 'square' koji sam koristio kao osnovu za svoje ploče (tom sam kvadratu bio 4 uveća duljinu, a širinu duplo smanjio), eksperimentalno sam otkrio da bi duljina ploče koja ima oblik pravokutnika u mojoj igrici trebala biti 50 piksela, a širina 10 piksela), onda će loptica promijeniti smjer kretanja. Za lijevu ploču moramo samo promijeniti da se x koordinata loptice nalazi u intervalu od -350 do -340. *Loptica.setx()*, odnosno *.setx()*, sam već objasnio na jednom od prethodnih primjera u ovoj dokumentaciji:

```
if (loptica.xcor()>340 and loptica.xcor()<350) and (loptica.ycor()<ploča_d.ycor()+50 and loptica.ycor()>ploča_d.ycor()-50):  #loptica se odbija od desne ploče
    loptica.setx(340)
    loptica.dx*=-1


if (loptica.xcor()<-340 and loptica.xcor()>-350) and (loptica.ycor()<ploča_l.ycor()+50 and loptica.ycor()>ploča_l.ycor()-50):  #loptica se odbija od lijeve ploče
    loptica.setx(-340)
    loptica.dx*=-1
```

Fotografijom ne mogu pokazati kako se loptica odbija i ploče pomiču klikom na tipku, ali sam priložio fotografiju na kojoj se vidi da ploče i loptica nisu u položaju u koji su prvotno stavljene. Pokretanje objekata i kretanje loptice ću prikazati u screencastu.



Dalje sam napravio sustav bodovanja. To sam napravio tako da sam odredio da će uz gornji rub okvira pisati koliko koji igrač ima bodova. Obzirom da je to još jedan novi objekt, koristio sam klasu. Sve sam napravio slično kao i za ostale objekte pa neću dodatno pojašnjavati, već ću pojasniti samo ono što nije spomenuto:

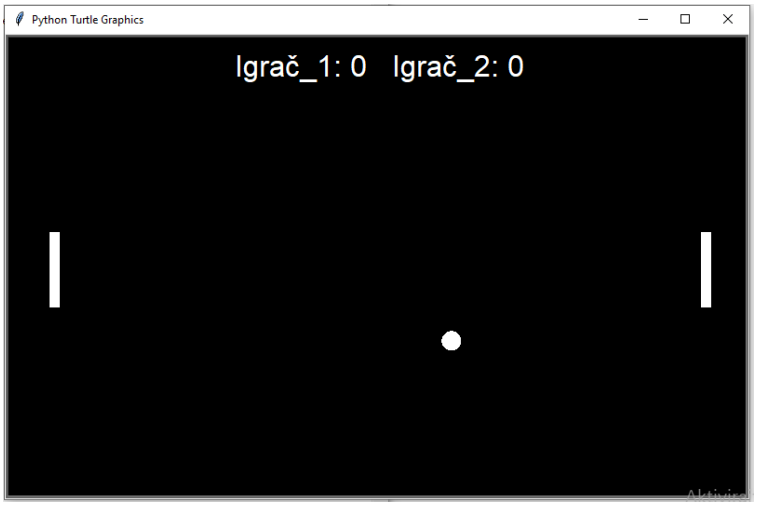
```
#zbrajanje bodova
bodovi=turtle.Turtle()
bodovi.color('white')
bodovi.penup()
bodovi.goto(0,200)
bodovi.speed(0)
bodovi.write('Igrač_1: 0   Igrač_2: 0',align='center',font=('Calibri Body',24,'normal'))
```

Obzirom da želim da rezultat bude vidljiv u obliku da piše *„Igrač_1: (ovdje će umjesto ove zagrade biti broj bodova koji je na početku nula) Igrač_2: (ovdje će umjesto ove zagrade biti broj bodova koji je*

na početku nula)“, moram nakon što sam definirao bodove pomoću klase, upisati *bodovi.write((ovdje upisujem što želim da piše, a to je u mome slučaju: „Igrač_1: (ovdje će umjesto ove zagrade biti broj bodova koji je na početku nula) Igrač_2: (ovdje će umjesto ove zagrade biti broj bodova koji je na početku nula)“))*, moram poravnati s neke strane to što će pisati (ja sam upisao „*center*“ jer želim da tekst bude na sredini), i na kraju moram odrediti veličinu i vrstu slova te hoće li biti „normalna“ ili podebljana, odnosno nakošena):

Primjećujemo da se kod ispisa rezultata nalazi mala strelica koja tu ne bi trebala biti. To je zapravo kornjača koja cijelo vrijeme crta ove objekte, a maknut ćemo je, odnosno sakriti, tako da dopišemo *bodovi.hideturtle()*, kao što je vidljivo na priloženim fotografijama:

```
#zbrajanje bodova
bodovi=turtle.Turtle()
bodovi.color('white')
bodovi.penup()
bodovi.goto(0,200)
bodovi.speed(0)
bodovi.hideturtle()
bodovi.write('Igrač_1: 0   Igrač_2: 0',align='center',font=('Calibri Body',24,'normal'))
```



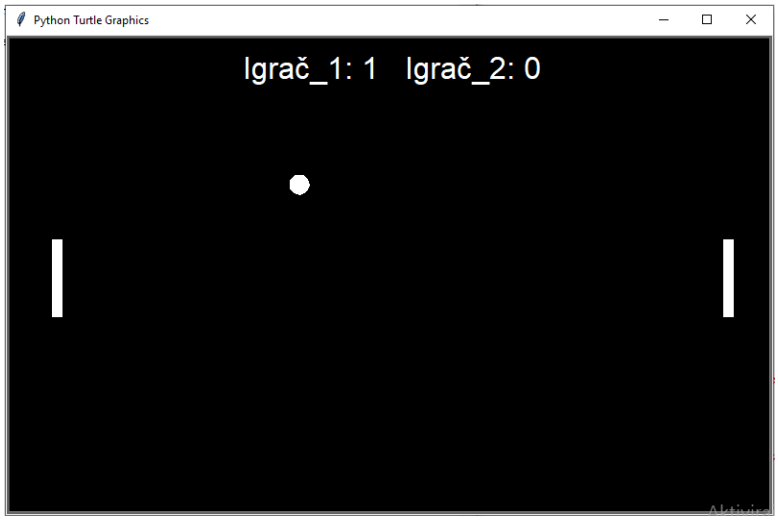
Sada još moramo povezati da se igraču upisuju bodovi kada njegov protivnik ne uspije odbiti lopticu. Da bismo to napravili, prvo moramo postaviti neki brojač bodova koji će biti nula u početku i kojemu će se kasnije pribrajati bodovi:

```
#dodavanje bodova kad ih igrač osvoji, tj. kad loptica lupi iza ploče
bodovi_1=0
bodovi_2=0
```

Potom se moramo vratiti u petlju za igru gdje smo upisali da loptica ne može izlaziti iz okvira s lijeve i desne strane okvira te dopisati kôd vidljiv na fotografiji:

```
if loptica.xcor()>390:  #loptica ne može otići iz okvira lijevo/desno
    loptica.goto(0,0)
    loptica.dx*=-1
    bodovi_1+=1
    bodovi.clear()
    bodovi.write('Igrač_1: {}   Igrač_2: {}'.format(bodovi_1,bodovi_2),align='center',font=('Calibri Body',24,'normal'))


if loptica.xcor()<-390:
    loptica.goto(0,0)
    loptica.dx*=-1
    bodovi_2+=1
    bodovi.clear()
    bodovi.write('Igrač_1: {}   Igrač_2: {}'.format(bodovi_1,bodovi_2),align='center',font=('Calibri Body',24,'normal'))
```

Objanit ću što ovaj kôd radi samo za jednu (u ovome slučaju prvu) „if" petlju jer je za drugu gotovo sve isto (objasnit ću samo ono što nije jednako). Ako loptica dodirne okvir iza desne ploče, to znači da će Igrač_1, odnosno lijevi igrač, dobiti bod pa pišemo *bodovi_1+=1*. Pomoću *bodovi.clear()*, briše se cijeli objekt bodovi, ali se on *pomoću bodovi.write('Igrač_1: {} Igrač_2: {}'.format(bodovi_1,bodovi_2),align='center',font=('Calibri Body',24,'normal'))* ponovno vraća (kornjača ga ponovno crta), ali uz pomoć formatiranog zapisa, broj bodova koji je bio uz Igrača_1 se povećava za jedan. Za drugu „if" petlju na fotografiji je sve jednako osim što se na kraju bodovi pribrajaju Igraču_2. Iako se objekt bodovi obrisao i kornjača ga je ponovno upisala, mi to nismo vidjeli zahvaljujući *okvir.tracer(0)* te je upravo ovo primjer zašto se naredba *.tracer(0)* uopće koristi. Evo kako to izgleda kada Igrač_1 dobije bod:

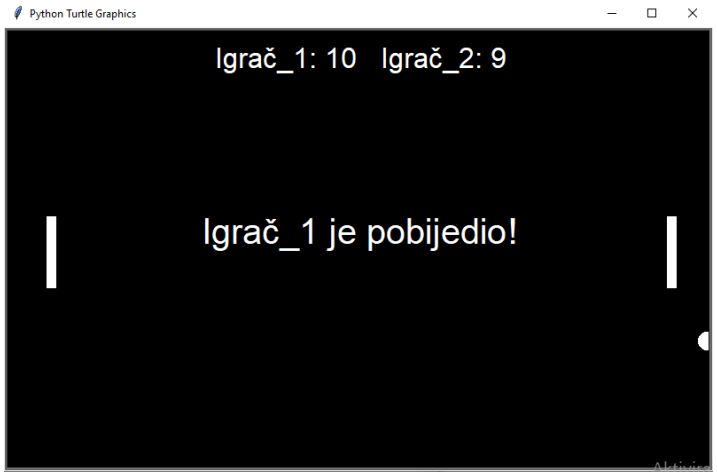


Kako izgleda kada Igrač_2 dobije bod:

Na kraju je ostalo samo napraviti da program sam zaustavi igru i ispiše tko je pobjednik unutar samog okvira. Ja sam odredio da igra završava kada jedan od igrača prvi dođe do deset bodova. Objasnit ću kako radi ovo što sam naveo na način da objasnim kôd za Igrača_1 jer je kôd za Igrača_2 gotovo identičan.

```
if bodovi_1 == 10:
    poruka=turtle.Turtle()
    poruka.color('white')
    poruka.penup()
    poruka.goto(0,0)
    poruka.speed(0)
    poruka.write('Igrač_1 je pobijedio!',align='center',font=('Calibri Body',30,'normal'))
    okvir.exit()


if bodovi_2 == 10:
    poruka=turtle.Turtle()
    poruka.color('white')
    poruka.penup()
    poruka.goto(0,0)
    poruka.speed(0)
    poruka.write('Igrač_2 je pobijedio!',align='center',font=('Calibri Body',30,'normal'))
    okvir.exit()
```

To sam napravio pomoću „if” petlje koju sam upisao u petlju za igru. Odredio sam ako bodovi_1 (bodovi Igrača_1) budu točno jednaki 10, da se pomoću klase stvori objekt poruka. To je poruka koju će ispisati program kada Igrač_1 pobijedi. Odredio sam boju, koordinate u okviru na kojem će se poruka ispisati, brzinu kojom se poruka ispiše. Sve kao i za ostale objekte. Predzadnji red prve „if” petlje predstavlja poruku koja će se ispisati, pokazuje da će se centrirati središnje, te font, vrstu i veličinu slova. Na kraju sam dodao *okvir.exit()*. To služi da se okvir „zamrzne”, tj. pritskom tipke na tipkovnici ploče se neće pomicati, a nema više niti spontanoga gibanja loptice. Za Igrača_2 je sve jednako samo da program ispiše poruku da je on pobjednik, on mora prvi skupiti deset bodova. Kako izgleda kada Igrač_1 pobijedi:



Kako izgleda kada Igrač_2 pobijedi: