SVE FUNKCIJE KOJE SMO KORISTILI U IZRADI PYTHON I PYGAME PROGRAMA IZ INDIVIDUALNIH PROJEKATA I IGRICA

Sadržaj

[NISKE (STRING) 2](#_Toc40030704)

[UNOS I ISPIS STRINGOVA U KONZOLU 2](#_Toc40030705)

[DUŽINA NISKE, IZDVAJANJE DELOVA NISKE 3](#_Toc40030706)

[PRETRAGA NISKE 3](#_Toc40030707)

[TORKE, LISTE, SKUPOVI 4](#_Toc40030708)

[FUNKCIJE ZA RAD SA TORKAMA I LISTAMA 4](#_Toc40030709)

[PRETRAGA 4](#_Toc40030710)

[SORTIRANJE 5](#_Toc40030711)

[IZMENE LISTE I RANGE FUNKCIJA 5](#_Toc40030712)

[SKUPOVI 5](#_Toc40030713)

[MATEMATIČKE FUNKCIJE 6](#_Toc40030714)

[PYGAME 6](#_Toc40030715)

[INICIJALIZACIJA IGRE I OBJEKATA OSNOVNE FUNKCIJE U IGRI 6](#_Toc40030716)

[FUNKCIJE ZA INICIJALIZACIJU OBJEKTA CLOCK I PODEŠAVANJE OSVEŽAVANJA EKRANA (FPS) FREJMOVI U SEKUNDI 7](#_Toc40030717)

[FUNKCIJE ZA CRTANJE GEOMETRIJSKIH OBLIKA NA EKRANU 7](#_Toc40030718)

[DOGAĐAJI ODNOSNO EVENTS 8](#_Toc40030719)

[ZVUCI 8](#_Toc40030720)

[DETEKCIJA KOLIZIJE 9](#_Toc40030721)

[LOOP FUNKCIJE ZA PONAVLJANJE FUNKCIJA ZA ISCRTAVANJE OBJEKATA NA EKRANU ODNOSNO ZA POSTIZANJE ANIMACIJE 9](#_Toc40030722)

[FUNKCIJE ZA RAD SA POZICIJOM I VIDLJIVOŠĆU MIŠA 9](#_Toc40030723)

[FUNKCIJE ZA RAD SA STILOVIMA SLOVA I PRIKAZOM ISTIH 10](#_Toc40030724)

[FUNKCIJE ZA IZRAČUNAVANJE I ZAUSTAVLJANJE VREMENA 10](#_Toc40030725)

[FUNKCIJE ZA IZLAZAK IZ IGRICE 10](#_Toc40030726)

# NISKE (STRING)

## UNOS I ISPIS STRINGOVA U KONZOLU

FUNKCIJE: **input(niska), int(niska), float(niska), print(niska)**

text = **input("unesite tekst, nemojte unositi brojeve\n")**

**input** funkcija koja daje mogućnost unosa teksta u promenljivu uz pomoć konzole, između zagrada unosimo tekst koji bi trebalo korisniku da pojasni koje vrednosti treba da unese u konzolu, \n je znak za novi red pa će korisnik unositi tekst iz novog reda, možemo staviti i razmak, pa će u konzoli biti razmaka između poruke i teksta koji unosimo

Pomoću **input()** unosi se uvek tekst tj. rezultat ove operacije je uvek niska, čak iako taj tekst sadrži samo cifre.

Ako tekst sadrži samo cifre, onda se broj predstavljen tim ciframa može dobiti pomoću **int()** funkcije. Stoga se učitavanje broja može postići pomoću **int(input("Unesi broj: "))**.

Niska može da sadrži i decimalni zapis nekog broja i tada se broj predstavljen tom niskom može dobiti pomoću **float().**

**print(“Ana voli Milovana”)**

* ispisuje tekst između zagrada u konzolu
* tekst može biti dopunjen i nekom promenljivom ili sa više njih, npr.   
  **dodatak = 123**   
  **print(“Ana voli “ + str(dodatak) + “Milovana”)**   
  ovo će na konzoli odštampati **“Ana voli 123 Milovana”** ,   
  primetite da se koristi **str()** to je takođe funkcija koja parametar između zagrada pokušava da pretvori u string ( a parametar je jedna niska sačinjena od tri dela koji su svi niske odnosno stringovi i spojeni su plusevima u jednu nisku ), naravno mi moramo proslediti parametar koji se može pretvoriti u string, tu neće biti problema, ali šta će se desiti ako pozovemo sličnu funkciju **int()** koja konvertuje promenljivu između zagrada u integer, tj. celobrojni tip  
  ako nam je **promenljiva između zagrada** u vidu rečenice **“Ana voli Milovana”** onda se ona  
  **ne može pretvoriti u celi broj** i tako će se javiti greška u programu   
  Value error: Invalid literal for int() with base 10

Kod sve četiri funkcije prosleđuje se niska, kod **print** mogu i drugi tipovi koji se konvertuju u nisku.

## DUŽINA NISKE, IZDVAJANJE DELOVA NISKE

FUNKCIJA: **len(niska)**

Dužinu niske tj. broj njenih karaktera možemo dobiti pomoću funkcije **len()**. Tako je **len("Zdravo")** jednako 6, jer niska "Zdravo" ima tačno 6 karaktera. P.S. Broje se i prazna mesta. Parametar funkcije je niska.

**print(len(“Zdravo svete”))**

**ovo će odštampati u konzoli 12**

## PRETRAGA NISKE

FUNKCIJE (metode): **niska.find(karakter/niska), (podniska\_1, ..., podniska\_n) = niska.split(),** **niska.count(karakter/niska)**

Često je potrebno da proverimo da li jedna niska sadrži neki karakter ili sadrži neku drugu nisku. To možemo uraditi korišćenjem **find()** metode.

Umesto **find()** možemo upotrebiti i **index()** (kao i kod torki i lista), međutim, **kada se ono što se traži ne nalazi u niski, find() vraća vrednost -1,** dok **index() daje grešku pri izvršavanju programa**.

PODELA NISKE NA LISTU NISKI

Funkcija **split()** formira listu reči date rečenice (vrši podelu niske na podniske na osnovu razmaka koji se javljaju između podniski).

listaStringova = **nekiString.split()**

* ovo je metoda klase str i poziva se iz promenljive koju delimo a vraća listu stringova
* deli nekiString kome mora da bude dodeljena vrednost str tipa, na listu stringova koji su u nekiString rećenici odvojeni razmacima, tako ćemo od rečenice **“Ana voli Milovana”** dobiti listu čiji su **prvi element “Ana” drugi “voli” i treći “Milovana”**
* može se koristiti sa parametrom koji će označavati po kojem karakteru ili listi karaktera da podelimo nekiString, tako da ako nam je nekiString jednak rečenici   
  **“Volim da programiram, volim nauku i da citam”**   
  i iskucamo **x = nekiString.split(“,”)**  
  **x** će nam biti **[‘Volim da programiram’,’ volim nauku i da citam’]**
* Primetite da drugi element liste x počinje razmakom, da smo stavili **x = nekiString.split(“, “)** sa razmakom podelio bi ovo kako treba i ne bismo imali razmak na početku drugog elementa
* Dakle videli smo da promenljiva u funkciji split može da bude i string
* Možemo staviti i dve promenljive s tim što bi druga promenljiva bila tipa int i bila bi broj deljenja koje izvršavamo
* Tako bi u recenici **“Ana voli Milovana” split(“ “, 1)** izbacio rezultat **[‘Ana’,’voli Milovana’]**

**text = str.upper(text)**

* **str.upper** pretvara sva slova text promenljive koja je tipa niska u velika slova

**text = str.lower(text)**

* **str.lower** pretvara sva slova text promenljive u mala slova

Osim što možemo pronaći poziciju na kojoj se neko slovo ili neka podniska javlja unutar niske, možemo i izbrojati koliko se puta ono pojavljuje. Za to možemo koristiti **count()**.

**nekiString = “Ja živim u Nišu, Niš je lep grad.”**

**print(nekiString.count(”Niš”))**

će odštampati u konzoli 2

**ord('a')**

vraća redni broj malog slova a iz ASCII tabele karaktera

# TORKE, LISTE, SKUPOVI

## FUNKCIJE ZA RAD SA TORKAMA I LISTAMA

FUNKCIJE: **len(lista), sum(lista), min(lista), max(lista)**

Funkcijom **len()** izračunavamo dužinu liste, funkcijom **sum()** izračunavamo zbir elemenata liste, funkcijom **min()** najmanji element u listi, a funkcijom **max()** najveći. Kod svih ovih funkcija, kao argument prosleđuje se lista nad kojom želimo da se izvrši funkcija.

### PRETRAGA

FUNKCIJA (metoda): **lista.index(element)**

Pronalaženje najmanje pozicije (indeksa) na kojoj se nalazi neki traženi element može se uraditi korišćenjem metode index, kojoj se kao jedini parametar prosleđuje element koji se traži.

Metode su funkcije koje se pozivaju na nekoj promenljivoj kucanjem tačke iza promenljive, i definišu se u klasite promenljive i imaju obavezni parametar self koji se odnosi na tu promenljivu iz koje se poziva, a funkcije se pozivaju bez promenljive i tačke ispred funkcije i definišu se u glavnom programu. Dakle metode su funkcije napisane u klasi.

### SORTIRANJE

FUNKCIJA: **sorted(lista)**

Elemente torke tj. liste je veoma jednostavno urediti od najmanjeg do najvećeg (pa i obratno, od najvećeg do najmanjeg). Funkcija **sorted(list)** se može primeniti i na torke i na liste i ona na osnovu datih podataka gradi novu torku, tj. listu koja sadrži iste podatke, u sortiranom redosledu. Metoda **list.sort()** primenljiva je samo na liste i ona menja listu tako što joj elemente ređa u sortiranom redosledu. Ovo je efikasnije nego da se pravi nova lista, ali se originalni redosled elemenata gubi. **sorted** prosleđujemo listu kao argument a **sort** pozivamo kao metodu liste.

### IZMENE LISTE I RANGE FUNKCIJA

FUNKCIJE (metode): **lista.append(element), del lista[indeks]**

Element na kraj liste možemo dodati metodom **append()**. Parametar funkcije je element koji dodajemo listi, a ova metoda se izvršava nad željenom listom.

Elemente iz liste možemo brisati pomoću **del lista[0]**. Prosleđuje se indeks željenog elementa, **u ovom slučaju brišemo prvi element liste jer u pythonu brojimo od 0**. Ako imamo listu.

**lista = [‘crveno’,’plavo’,’zeleno’,’zuto’]**

**lista.remove(‘plavo’) će ukloniti plavo iz lista liste**

**range(6)**

* **kreira listu brojeva od 0 do 5, dakle ukupno 6 brojeva** i koristimo je obično kada hoćemo for petljom da prođemo 6 puta kroz neki kod, ali će nam promenljiva “i” u prolazima petlje for(i in range(6)) uzimati vrednosti 0 u prvom prolazu, 1 u drugom i tako dalje sve do zadnjeg prolaza gde će vrednost promenljive “i” biti 5

### SKUPOVI

FUNKCIJE: **set(lista/torka)**

**set(lista)**

* kreira set od liste/torke, koja se prosleđuje kao argument
* set je struktura u pythonu kao skup u matematici, **elementi nemaju redosled, nema duplih elemenata**, **i ne možemo im pristupiti pomoću indeksa**, rednog broja elementa kao u listi
* set se može napraviti i kao
* **voce = { ‘jabuka’, ‘kruska’, ‘tresnja’ }** koriste se vitičaste zagrade

# MATEMATIČKE FUNKCIJE

**x = random.randint(0,255)**

* x dobija vrednost nasumicno izabranog broja od 0 do 255 pre toga moramo napisati negde u programu **import random** za importovanje random biblioteke uz pomoc koje cemo moci da koristimo ovu funkciju

**x = math.floor(4.6)**

* zaokruživanje broja na donju vrednost u ovom primeru će x dobiti vrednost 4
* kada bismo hteli da koristimo **matematičko zaokruživanje** koristili bismo **math.round**, **math.round(4.4) bi zaokružio i vratio vrednost** **4** a **math.round(4.6) bi vratio vrednost 5**, **math.ceil(4.4) bi zaokružio ovaj broj izmedju zagrada na 5** jer se ovde radi o zaokruživanju na veću vrednost naravno pre svega ovoga moramo da importujemo biblioteku math naredbom **import math**

# PYGAME

## INICIJALIZACIJA IGRE I OBJEKATA OSNOVNE FUNKCIJE U IGRI

**prozor = pygamebg.open\_window(sirina, visina, niska)**

* otvaranje prozora, agrumenti su širina, visina, naziv programa

**pygame.display.set\_mode((width, height))**

* inicijalizuje prozor ili ekran za prikaz

**prozor.fill(pygame.Color(“gray”))**

* bojimo pozadinu prozora u onu boju koja je unesena kao argument npr. “red” ili “gray”

**pygame.init()**

* **init()** inicijalizuje sve importovane pygame module i to je prikladan način da se sve započne.   
  **init()** metoda za pojedine module će izbaciti grešku ako se neusmešno učitaju. Možda biste hteli da inicijalizujete različite module odvojeno da biste ubrzali vaš program ili da ne biste upotrebljavali module koje vaša igrica ne koristi.

**slika = pygame.image.load(“images/bird/frame-1.png”)**

* učitavanje slike iz putanje do datoteke unesene kao argument tipa niska odnosno string

**slika = pygame.transform.scale(slika, (x, y))**

* transformisanje slike u navedenu veličinu - (x, y) su širina i visina

**slikaRect = slika.get\_rect()**

* get\_rect metoda vraća novi pravougaonik koji se nalazi na koordinatama 0, 0 i ima širinu i visinu slike

**playerRect.move\_ip(playerMoveRate, 0)**

* pomera playerRect objekat tako što x koordinatu pomeri za plazerMoveRate vrednost, a y koordinatu u ovom slucaju ne pomera tj. pomera je za 0

**prozor.blit(slika, (x, y))**

* prikazivanje slike na određenoj poziciji, x i y su koordinate gornjeg levog temena slike

**pygame.display.set\_caption("Naziv\_prozora")**

* davanje imena prozoru

**pygame.display.update()**

* pozivanje funkcije **display.update()** kako bi se sve opisane promene iznad ove funkcije prikazale na ekranu

**pygamebg.wait\_loop()**

* prikazujemo prozor i čekamo da ga korisnik isključi

## FUNKCIJE ZA INICIJALIZACIJU OBJEKTA CLOCK I PODEŠAVANJE OSVEŽAVANJA EKRANA (FPS) FREJMOVI U SEKUNDI

**clock = pygame.time.Clock()**

* dodeljuje promenljivoj clock promenljivu tipa Clock, dakle i Clock je konstruktor funkcija

**clock.tick(60)**

* iz promenljive clock tipa Clock pozivamo metodu **tick** koja će nam odrediti osvežavanje ekrana ili izvršavanje while petlje 60 puta u sekundi, tako će naš program raditi normalnom brzinom

## FUNKCIJE ZA CRTANJE GEOMETRIJSKIH OBLIKA NA EKRANU

**pygame.draw.line(prozor, pygame.Color(“red”), (x1, y1), (x2, y2), debljina)**

* crtanje linije, prvi argument je površina na kojoj se crta (prozor), drugi je boja, treći i četvrti su koordinate duži (linije) koju crtamo, i peti argument je debljina linije

**pygame.draw.polygon(prozor, pygame.Color(“red”), [(x1, y1), (x2, y2), (x3, y3)], debljina)**

* crtanje poligona je slično crtanju linije, samo navodimo sve tačke koje nam trebaju za crtanje poligona i one su spojene linijama, ako stavimo debljinu na kraju poligon ce biti iznutra providan i imaće crveni okvir debljine koju zadamo u pikselima, ako stavimo bez parametra debljina onda će polygon biti ceo popunjen crvenom bojom i nece imati ivicu druge boje.

**pygame.draw.rect(prozor, pygame.Color("gray"), (x, y, sirina, visina), debljina)**

* naredba za crtanje pravougaonika sa parametrima koordinata gornjeg leve tacke pravougaoniika kao i sirina i visina i border kao debljina ivice, ako izostavimo border, pravougaonik ce biti ceo obojen izabranom bojom, za parametar debljina isto vazi kao i za polygon

**pygame.draw.circle(ime\_prozora, pg.Color("boja"), (x,y), r, debljina)**

* naredba crtanja kruga sa parametrima boje kruga, koordinata centra kruga i poluprečnik kruga  
  za parametar debljina isto vazi kao i za polygon

**pygame.draw.ellipse(prozor, pygame.Color("gray"), (x, y, duzina, visina), debljina)**

* u prozoru koji smo definisali na početku programa crtamo elipsu čija je boja siva, čije su koordinate x i y, a duzina i visina elipse se računaju desno i dole od koordinata elipse, ako stavimo 2 na kraju elipsa ce biti iznutra providna i imaće sivi okvir debljine 2 piksela, ako stavimo bez zadnjeg parametra onda će elipsa biti cela popunjena sivom bojom.

## DOGAĐAJI ODNOSNO EVENTS

**pygame.event.get()**

* metoda **pygame.event.get()** vraća listu svih pritisnutih tastera na tastaturi ako uzmemo jedan event objekat iz te liste možemo proveriti sledeće **if event.type == pygame.KEYDOWN:** da li je pritisnut neki taster na tastaturi  **if event.key == pygame.K\_ESCAPE:** da li je pritisnut taster escape

**pressed\_keys = pygame.key.get\_pressed()**

* metoda **pygame.key.get\_pressed()** u ovom slučaju vraća rečnik sa svim događajima tj. eventima koji su se dogodili i mi možemo uz pomoć ključeva koji su u vidu pygame.K\_LEFT, upotrebom **pressed\_keys[pygame.K\_LEFT]** proveriti da li je pritisnut taster leve strelice, ovu metodu ne možemo izvršiti bez prethodno izvršene metode **pygame.event.get()**

**pygame.time.set\_timer(GEN\_EVENT, GEN\_OBJ\_PERIOD)**

* tajmer gde je **GEN\_OBJ\_PERIOD** broj milisekundi za koje će se generisati događaj **GEN\_EVENT**

## ZVUCI

**coin\_sound = pygame.mixer.Sound('sounds/coin.wav')**

* dodeljuje promenljivoj coin\_sound promenljivu tipa Sound, funkcija **Sound** je i konstruktor koji kreira promenljivu tipa Sound. Dakle konstruktori su funkcije. Prepoznaćete ih po velikom početnom slovu, ne mora da znači uvek, ali tako bi trebalo da se pišu konstruktori

**pygame.mixer.music.load('sounds/background-music.mp3')**

* ova metoda se koristi za učitavanja pozadinske muzike u igricu

**pygame.mixer.music.play(-1)**

* ova metoda se koristi za puštanje pozadinske muzike ali na repeat

**self.drop\_sound.play()**

* ova metoda pušta drop\_sound zvuk

**gameOverSound.stop()**

* ova metoda stopira gameOverSound zvuk

## DETEKCIJA KOLIZIJE

**self.mask = pygame.mask.from\_surface(self.sprite[self.current\_sprite // 4])**

* pravi masku od spritea za detektovanje kolizije

**if(self.mask.overlap(obj.mask,[abs(int(self.x - obj.x)),abs(int(self.y - obj.y))])):**

* ovde proveravamo da li je došlo do kolizije uz pomoć maske objekta i metode overlap i koordinata objekta iz kojeg zovemo ovu metodu (u ovom slučaju bird) i obj objekta kojeg smo dobili iz liste svih objekata

**if playerRect.colliderect(b['rect']):**

* colliderect proverava da li se playerRect objekat u vidu pravougaonika preklapa sa b[‘rect’] objektom niza b koji je isto pravougaonik vraca True ako se preklapaju inace False

## LOOP FUNKCIJE ZA PONAVLJANJE FUNKCIJA ZA ISCRTAVANJE OBJEKATA NA EKRANU ODNOSNO ZA POSTIZANJE ANIMACIJE

**pygamebg.frame\_loop(60, crtaj)**

* 60 puta u sekundi se poziva funkcija **crtaj** i tako mozemo simulirati animaciju

**pygamebg.event\_loop(crtaj, obradi\_dogadjaj)**

* započinjemo petlju zasnovanu na događajima, parametri su funkcija **crtaj** za crtanje i funkcija  
  **obradi\_dogadjaj** obrade događaja koja prima jedan parametar koji mozemo nazvati po izboru  
  npr. **dogadjaj**  i iz koga mozemo uhvatiti dogadjaje, npr. pritisak na taster kao u primeru ispod  
  **if pg.K\_a <= dogadjaj.key and dogadjaj.key <= pg.K\_z:**ova **if** naredba uslovnog grananja **proverava da li je pritisnut taster između a i z**

**pygamebg.frame\_loop(frekvencija,novi\_frejm,obradi\_dogadjaj)**

* u nekim situacijama želimo da kombinujemo animacije i događaje. Na primer, želimo da mišem pomeramo animirane likove po ekranu. korišćenje biblioteke PyGameBg omogućava da se funkciji **frame\_loop** kao treći parametar zada i funkcija **obradi\_dogadjaj** čime se postiže da se prilikom prelaska na svaki naredni frejm obrade svi događaji koji su nastupili između prikazivanja dva frejma. Naglasimo da se događaji ne obrađuju čim nastupe, pa ako je **frekvencija** smenjivanja frejmova niska, tada korisnik može osetiti mali zastoj u odzivu programa.

## FUNKCIJE ZA RAD SA POZICIJOM I VIDLJIVOŠĆU MIŠA

**pygame.mouse.set\_pos(playerRect.centerx, playerRect.centery)**

* postavlja x koordinatu misa na playerRect.centerx, i y koordinatu na playerRect.centery vrednost, tj. u igrici ovo stavlja miš na poziciju igrača

**(mis\_x, mis\_y) = pygame.mouse.get\_pos()**

* uzimamo x i y koordinate pozicije miša i dodeljujemo ih mis\_x i mis\_y promenljivama

[**pygame.mouse.set\_visible**](https://www.pygame.org/docs/ref/mouse.html#pygame.mouse.set_visible)**(False)**

* set\_visible metoda prikazuje kursor misa ili ga sakriva u zavisnosti od parametra, u ovom slucaju ce ga sakriti jer je parametar False

## FUNKCIJE ZA RAD SA STILOVIMA SLOVA I PRIKAZOM ISTIH

**font = pygame.font.SysFont("Arial", velicina)**

* kreira font promenljivu iz sistemskih fontova odredjene veličine

**text= font.render(“Neki tekst”, True, pygame.Color("black"))**

* dodeljivanje prikaza “Neki tekst” stringu, sa izabranim fontom iz prethodnog pasusa promenljivoj text i odredjivanje boje teksta
* ovaj tekst ćemo prikazati funkcijom **prozor.blit(text,(100,200))** gde je drugi parametar x i y koordinate leve gornje tacke prikaza ovog teksta na prozoru, x koordinata je 100 a y koordinata je 200

## FUNKCIJE ZA IZRAČUNAVANJE I ZAUSTAVLJANJE VREMENA

**end = time.time()**

* ovde promenljiva end dobija vrednost proteklih sekundi od 1. januara 1970 godine, ovo nam je korisno kada uzmemo pre toga negde u programu promenljivu sa istom funkcijom npr.  
  **start = time.time()** i od promenljive **end oduzmemo start** i dobijamo broj sekundi proteklih između start i end

**time.sleep(5)**

* uspavljuje, odnosno zaustavlja program na 5 sekundi

## FUNKCIJE ZA IZLAZAK IZ IGRICE

**pygame.quit()**

* kraj programa i izlazak iz igrice

**sys.exit()**

* gašenje glavnog procesa programa