Python

Programiranje u Pythonu - skripta

GEODETSKA ŠKOLA ZAGREB
slavicek@geoskola.hr
Armando Slaviček

Nastavne cjeline

1. O programiranju

a. Povijest programiranja
b. Faze razvoja programa

2. Programski jezici

a. Podjela programskih jezika
b. Odabir programskog jezika

3. Algoritmi

a. Pojam algoritma
b. Dijagram toka
c. Simboli dijagrama toka

4. Kodiranje

a. Pseudokod
b. Varijable
c. Operatori
d. Slijed
e. Grananje
f. Petlje
g. Kompajliranje
h. Testiranje

5. Izrada dokumentacije

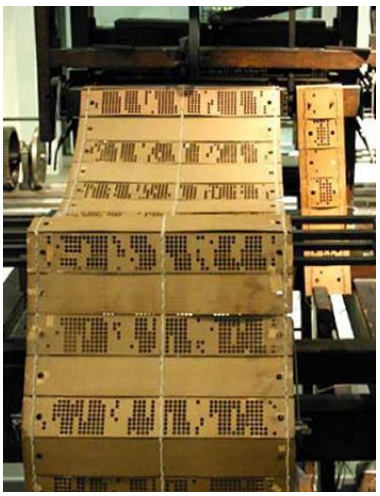
a. Korisnička dokumentacija
b. Programerska dokumentacija

Sadržaj

Povijest računala

https://informatickeigre.com/1r/povijest-racunala/

Povijest programiranja



Korijeni u tekstilnoj industriji (1801.)

Francuz Joseph Marie Charles **Jacquard** po zanimanju tkalac izradio je program za tkalački stroj, izrađen na drvenoj bušenoj kartici.

Ideja programiranja

- rašlanjivanje kompleksnih zadaća na niz nedvosmislenih i konačnih koraka koje stroj može izvesti

- stroj na temelju programa može izvršavati ponavljajuće zadatke

Prvi programer/ka:

1842. Ada Lovelace Byron

„Analitički stroj tka algebarske uzorke na isti način kako Jacquard-ov tkalački stroj tka cvjetove i listove. „

Ada je napisala skupove instrukcija koje bi se mogle izvršavati na analitičkom stroju.

Ada – prvi programer za računala. Programski jezik Ada je u njenu čast dobio ime.

Herman Hollerith (1860 – 1929), njemačko-američki statističar, izumitelj stroja na principu bušenih kartica koji se koristio pri popisu stanovništva Amerike (1890.). Smatra se začetnikom elektromehaničke obrade podataka.

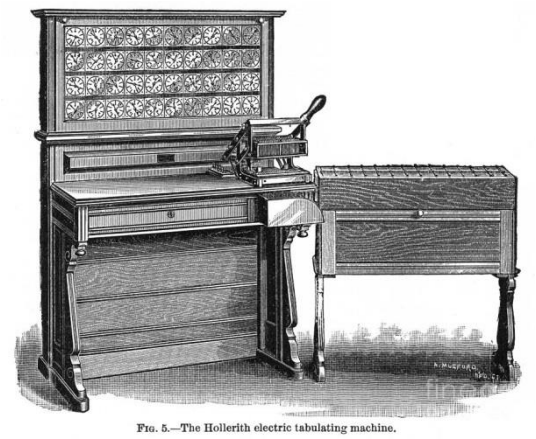
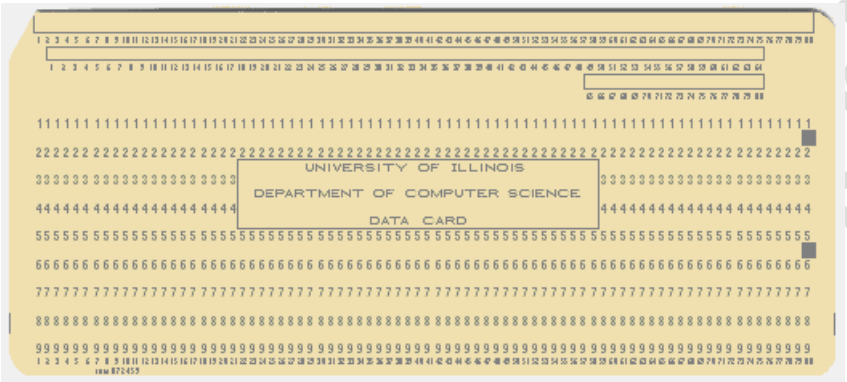
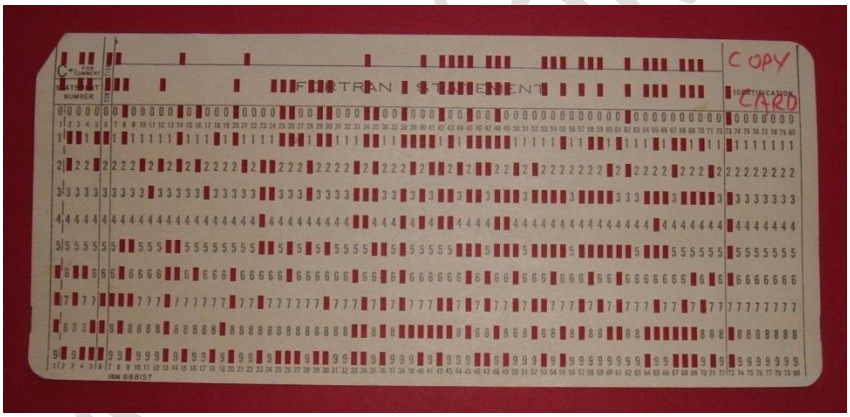


Fig. 5.—The Hollerith electric tabulating machine.

Bušene kartice





Programski jezici

Računala se također sporazumijevaju jezicima. Jezici koje računala razumiju zovu se **programski jezici**. Omogućavaju komunikaciju čovjeka i računala.

Možemo reći da je **programski jezik** skup ključnih riječi i pravila za njihovo korištenje koje „razumije“ računalo. **Sintaksa** jezika određuje pravila pisanja riječi, a **semantika** određuje značenje.

Generacije programskih jezika

1. Strojni
2. Simbolički - Asembler
3. Proceduralni
4. Problemski orijentirani
5. Prirodni

1. strojni programski jezik

- Računalni program predstavljen je nizom nula i jedinica koji računalu kazuju kako izvršiti elementarne operacije jednu po jednu
- Izuzetno je teško pisati programe u strojnom jeziku, nerazumljiv je za čovjeka i rijetko je u primjeni
- Program u strojnom jeziku procesor računala „razumije“ bez predvođenja
- primjer instrukcije:
  01110001110011100110011000011

2. simbolički - asemblerski programski jezik

- generacija programskih jezika – oko 1950. godine
- simbolički jezik (asembler)
- niži programski jezik
- jedna instrukcija strojnog jezika zamijenjena je jednom instrukcijom asemblerskog jezika (npr. ADD za zbrajanje)
- svaku naredbu prije izvođenja potrebno je prevesti u strojni jezik koju prevodi program prevoditelj

- kratice na engleskom jeziku za prikaz elementarnih operacija (mnemonički kod)

- primjer:

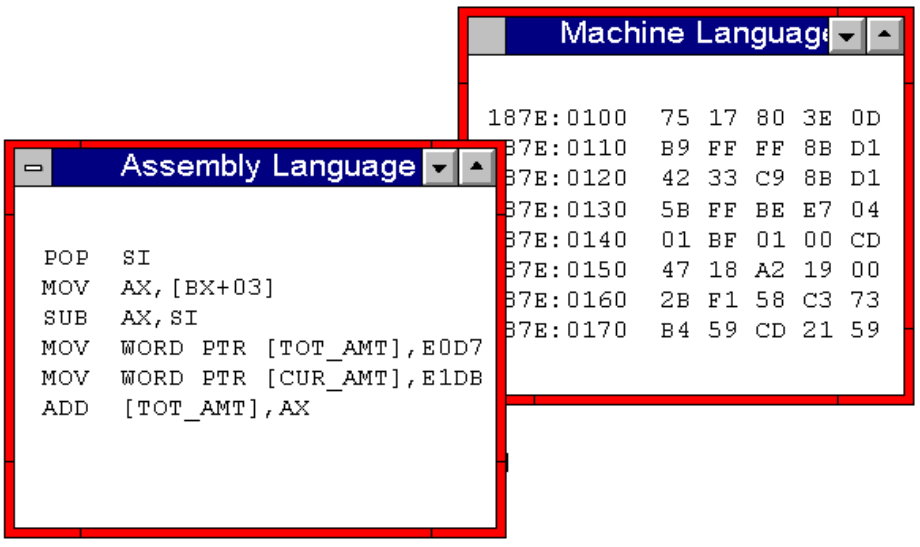
```
MOV  AX,3  ; u registar AX sprema broj 3
MOV BX, 4  ; u registar BX sprema broj 4
ADD AX,BX ; vrijednosti u registru AX dodaje vrijednost registra BX
```

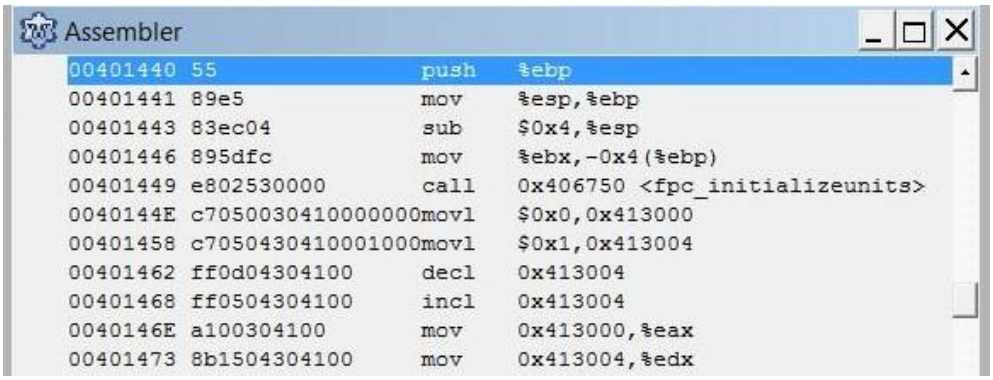
Isti program napisan u strojnom kodu u heksadekadskom zapisu.

```
B80300 (ovo je naredba MOV AX,3)
BB0400 (ovo je naredba MOV BX,4)
01D8 (naredba ADD AX, BX)
```

Asembler je čovjeku razumljiv način pisanja strojnog koda. No, čak i uz ovakav čitljiv način zapisivanja programa, programiranje u nižim programskim jezicima vrlo je teško i mukotrpno.

- Niži programski jezici imaju samo najosnovnije naredbe. Sve složenije naredbe treba izvoditi pomoću ovih osnovnih.
- Pri programiranju u nižim programskih jezicima treba izvrsno poznavati arhitekturu procesora i računala
- Program pisan za jednu vrstu procesora neće raditi na drugoj vrsti procesora
  Ovi problemi riješeni su u višim programskim jezicima.





Viši programski jezici bliži su čovjeku i njegovu načinu razmišljanja. Ne ovise o tipu računala i mogu se koristiti na bilo kojem računalu (ako na njemu postoji odgovarajući prevoditelj ili interpreter). Strojni kod se pak može izvršavati samo na onom procesoru za koji je prirađen.

3. **proceduralni programski jezici/viši programski jezici** (treća generacija)

- viši programski jezici – oko 1960. godine
- simbolički jezici u kojima se naredbe računalu pišu uporabom simbolike sličnom govornom jeziku
- više instrukcija strojnog ili asemblerskog jezika zamijenjeno je jednom instrukcijom višeg programskog jezika
- program je nužno prevesti u strojni jezik pomoću programskog prevoditelja (interpretera ili kompajlera)
    - interpreter – prevodi naredbu po naredbu, a svaka se naredba nakon provođenja odmah izvršava
    - kompajler – je program prevoditelj koji prevodi cijeli program napisan u nekom programskom jeziku i nakon toga slijedi izvršavanje programa
- naredba je izraz kojim se izvodi niz operacija na sklopovskoj razini računala
- (Logo, Basic, Visual Basic, Lisp, Prolog, Ada, Pascal, Java, Algol, Cobol, Fortran, PL/1…)

PROGRAMSKI JEZIK LOGO	PROGRAMSKI JEZIK QBASIC	PROGRAMSKI JEZIK Pascal

TO ZBROJI
MAKE ''A READ
MAKE ''B READ
MAKE ''C :A+:B
PR :C
END
	INPUT A
	INPUT B
	C=A+B
	PRINT C
	END
		Program zboji;
		var a, b: integer;
		begin;
		readln (a);
		readln (b);
		c:=a+b;
		writeln (c);
		end.

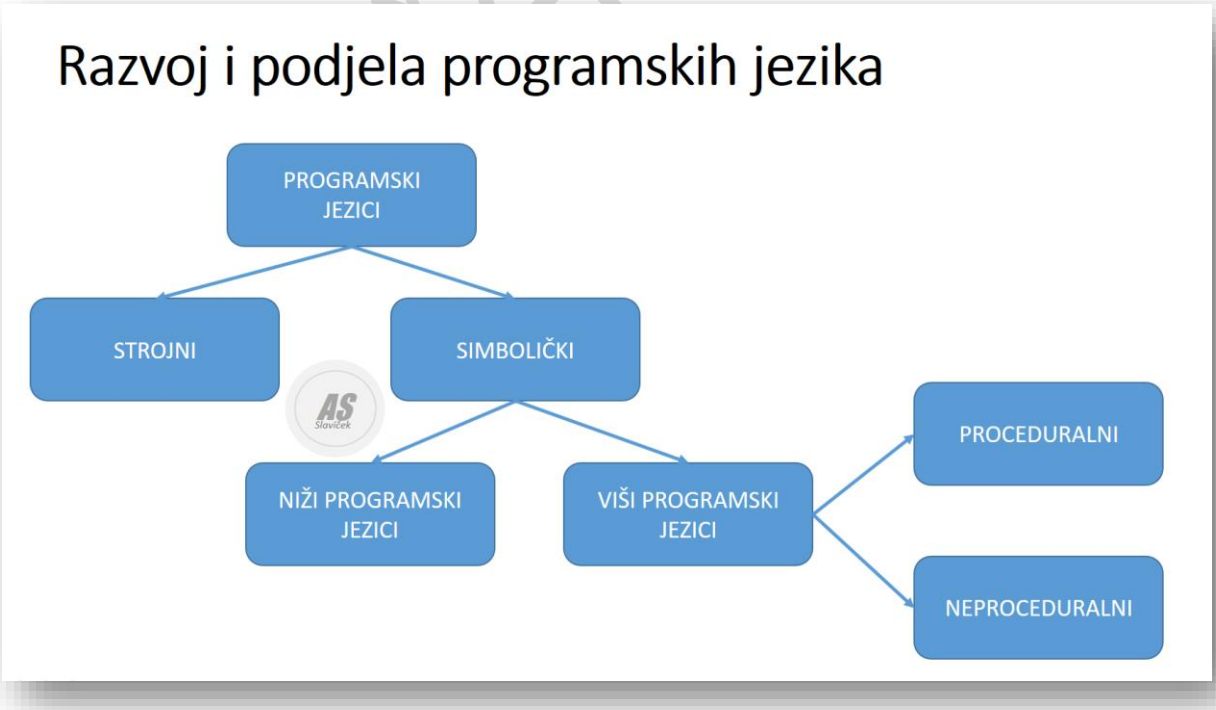
4. problemski orijentirani programski jezik/ jezici četvrte generacije

- komponente ovih jezika su baze podataka s vlastitim upravljačkim sustavom, rječnik podataka (u rječniku se nalazi detaljan opis svih podataka, izgledi ekrana) te alati koji osiguravaju zaštitu baze podataka

- **Structured Query Language (SQL)** – najpopularniji programski jezik za kreiranje, dohvat, ažuriranje i brisanje podataka iz baze podataka
- **PostScript (PS)** – jezik za opisivanje izgleda stranice i programski jezik koji se prvenstveno koristi u elektroničkom i stolnom izdavaštvu

5. prirodni programski jezik

- C++, C
- JAVA
- VisualBasic
- Generatori
- Razvojna okruženja



Jezici koji se danas traže pri zaposlenju programera

(izvor: http://www.ictbusiness.info/vijesti/koji-su-programski-jezici-najtrazeniji-u-2015)

1. **Java**

Jedan od najpopularnijih programskih jezika i onaj koji svi moraju znati ukoliko se žele baviti ovim poslom. Koristi se za sve, od weba do Android aplikacija za pametne telefone i tablete.

2. **JavaScript**

Svaki moderni website koristi JavaScript, nije na odmet znati taj programski jezik.

3. **C#**

Osnove, osnove! Svi za njega znaju, svaki programer upućen je barem donekle u njega, ali da biste raditi s Microsoft platformama morate biti detaljno upućeni u ovaj programski jezik.

4. **PHP**

Programski jezik koji služi za rad s MySQL i za stvaranje aplikacija koje rade s podacima. WordPress je primjer velikog sustava nastalog na PHP osnovama.

5. **C++**

Iako je to za "niže razine", potrebno je znanje ovog programskog jezika ako se želite spojiti direktno na hardver i izvući što je moguće više snage. Odličan programski jezik za stvaranje moćnog desktop softvera, hardvera koji izvlači maksimum pa je „gejmersko“ iskustvo bolje, a služi i za aplikacije za pametne telefone, konzole i desktop računala. Naravno, one vezane uz memoriju i njeno bolje iskorištavanje.

6. **Python**

Ovaj programski jezik može gotovo sve. Web aplikacije, korisnička sučelja, analizu podataka, statistike... Uz Javu i Python pokrili ste većinu područja i nakon toga sve je samo dodatno nadograđivanje u svrhu lakšeg pronalaska posla.

7. **C**

Ovaj programski jezik je popularan još uvijek jer je malen, brz i moćan. Ukoliko želite najviše za resurse kojima raspolažete, C je odgovor.

8. **SQL**

Programski jezik koji omogućava precizan i brz rad s ogromnom količinom podataka i kompleksnim bazama podataka.

9. **Ruby**

Programski jezik koji omogućava brzu izradu web aplikacija, sjajan za „kickstart“ *(Kickstarter je ime za web poslužitelj koji potiče financiranje iz gomile za kreativne projekte.Kickstarter financira raznovrsne projekte, od početničkih filmova rangi, početničke glazbe, drama, stripova, video igara, hrane, tehnologije, novinarstva. Ljudi ne mogu uložiti u Kickstarter projekte da bi mogli zaraditi novac.Oni mogu samo financirati projekte u zamjenu za opipljivo dobro kao recimo završni proizvod, majice, zahvalnice,…)* projekte.

10. **Objective-C**

Želite stvarati iOS aplikacije? Naučite ovaj programski jezik.

11. **Perl**

Postoji od samog početka weba i još je relevantan, koristi ga svaki IT profesionalac. Bez njega je sigurnost na internetu nezamisliva.

12. **.NET**

Iako sam po sebi nije programski jezik, ključna je Microsoftova platforma za cloud, usluge i razvoj aplikacija koje postaju sve naprednije i vrednije sa svakom novom objavom. Također, .NET uskoro dolazi i na Google i Apple platforme pa je itekako korisno znati raditi s njime.

13. **Visual Basic**

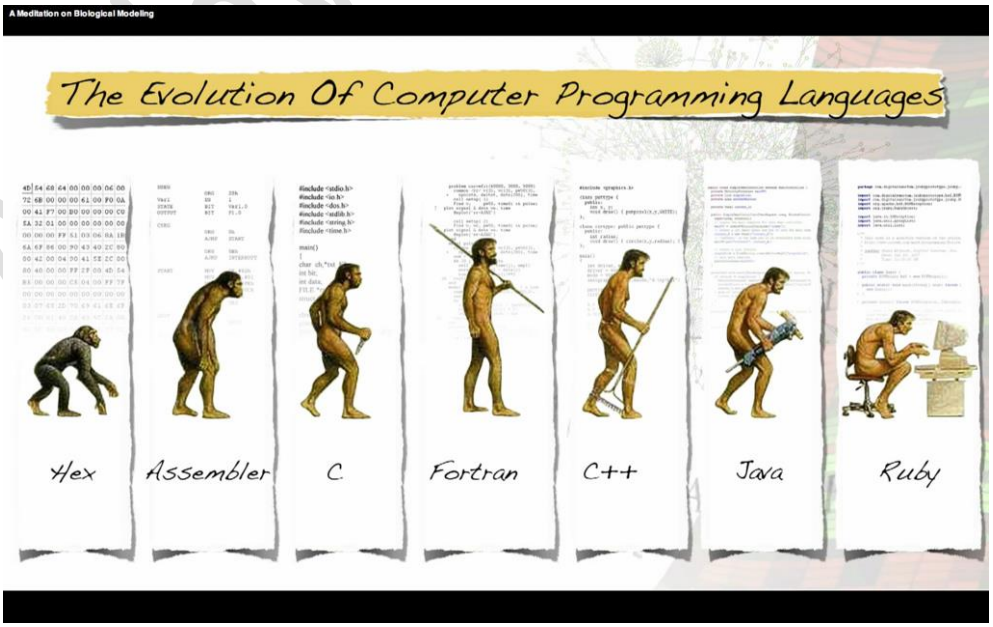
Ključni programski jezik za .NET pa ukoliko želite savladati .NET, morate i Visual Basic. Nakon toga možda dobijete posao u Microsoftu i radite na Office alikacijama kao Excel, koje postoje zahvaljujući Visual Basicu.

14. **R**

Programski jezik za baratanje velikim podacima i glavni programski jezik za 2015. godinu jer sve veće kompanije i organizacije trebat će analizu velikih podataka i statističke podatke iz njih.

15. **Swift**

Nije prisutan ni godinu dana, a već je uočen i cijenjen. Taj programski jezik služi za brz i jednostavan razvoj Mac i iOS operativnih sustava. Ukoliko posjedujete Mac, Swift vam omogućava razviti aplikacije za iOS ili Mac OS X.



https://www.flickr.com/photos/dullhunk/4833512699

Faze razvoja programa

1. **Definicija problema**

Prvi korak u rješavanju bilo kojeg opsežnijeg zadatka je definiranje problema, tj. planiranje. Dobro planiranje može uštedjeti mnogo truda i vremena. Planiranjem određujemo tko će, kada i što raditi.

2. **Analiza zadatka i skiciranje rješenja**

Analiza zadatka je potpuno razumijevanje zadatka i željenih rezultata. To je jedan od najsloženijih i najtežih koraka pri nastanku programa. Rezultat analize je specifikacija zadatka, tj. detaljan opis zadatka i željenih rezultata.

3. **Sastavljanje algoritma**

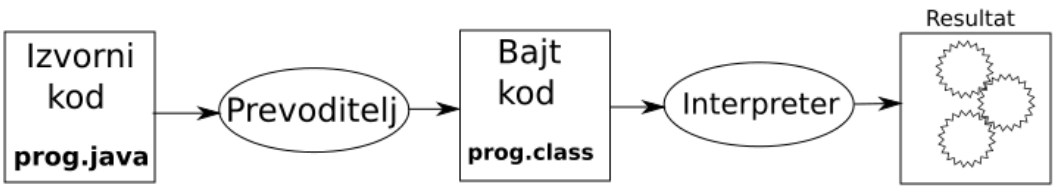
Algoritam je naputak kako riješiti neki zadatak ili obaviti neki posao. Algoritam svodi cjelokupan zadatak na rješavanje više jednostavnijih, manjih radnji.
Algoritam prikazujemo dijagramom tijeka (grafički prikaz algoritma – zadatak se vizualizira) ili **pseudojezikom**.
Pseudojezik se sastoji od kratkih izraza na govornom jeziku koji opisuju i ukratko objašnjavaju pojedine radnje algoritma.

4. **Programiranje - kodiranje**

Programiranje je postupak zapisivanja temeljnih radnji (do kojih se došlo planiranjem, algoritmom, pseudojezikom, dijagramom tijeka) naredbama odabranog programskog jezika. Zapisivanje naredbama odabranog programskog jezika zadatka koji jest u pogodnom obliku naziva se još i **kodiranje**.

5. **Prevođenje programa**

Postoje dvije vrste “prevođenja” programa, a to su: **interpretiranje** (eng. interpreting) i **kompajliranje** (eng. compiling). Interpreter je program koji “čita” program napisan programskim jezikom visoke razine i “radi što mu on kaže”. Jednostavno rečeno on prevodi program liniju po liniju i izvodi naredbe (eng. commands) koje su navedene u toj liniji. Kompajler je program koji također “čita” program napisan programskim jezikom visoke razine te “prevodi” kompletan program odjednom, bez da izvršava naredbe.



6. **Provjera i ispravljanje programa (Testiranje programa)**

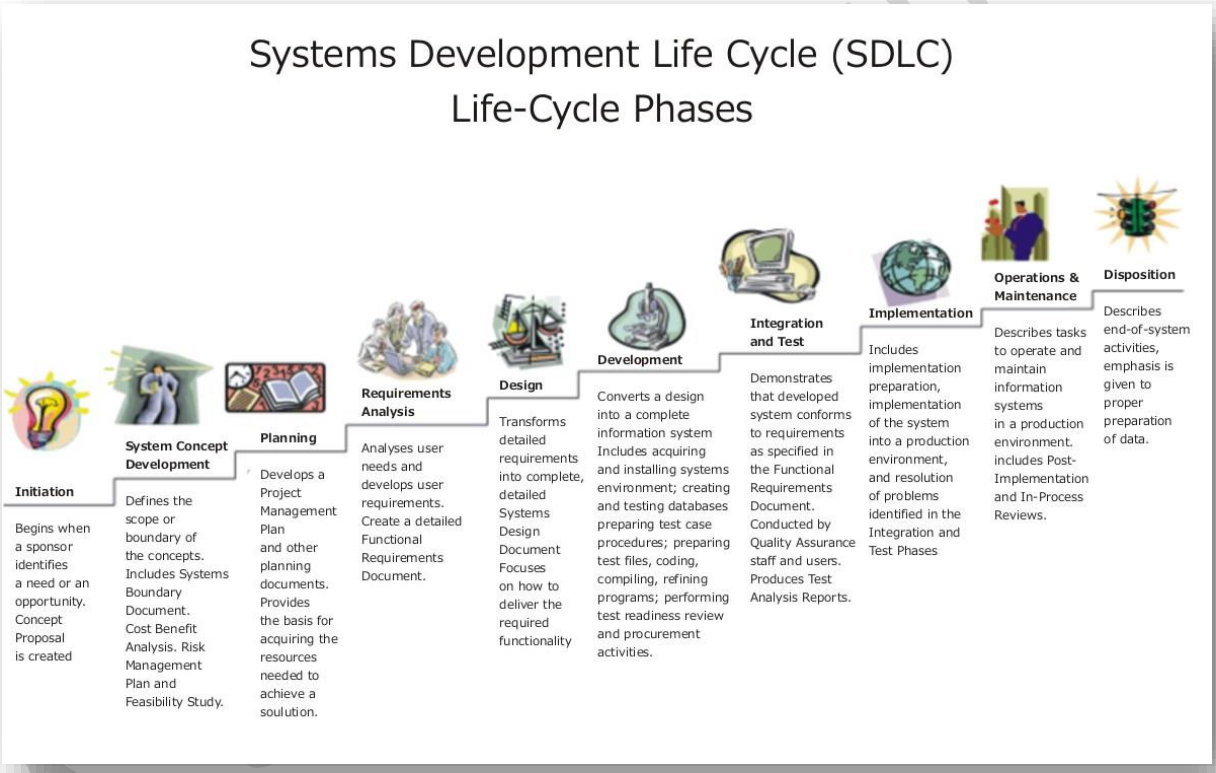
Logička ispravnost programa osigurava se provjerama ili testiranjem. Uvijek treba provjeriti pomoću ulaznih vrijednosti za koje se unaprijed zna rezultat.

7. **Dokumentiranje**

U dokumentaciju se ubrajaju upute za instaliranje programa i priručnici za korisnike.

8. **Održavanje programa**

Postupak naknadnog mijenjanja programa tijekom njegovog korištenja naziva se održavanje programa.

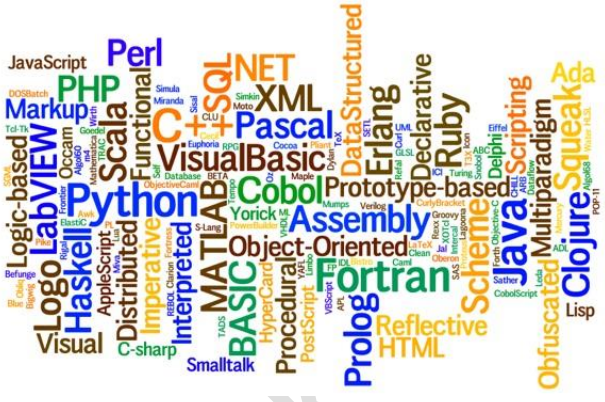


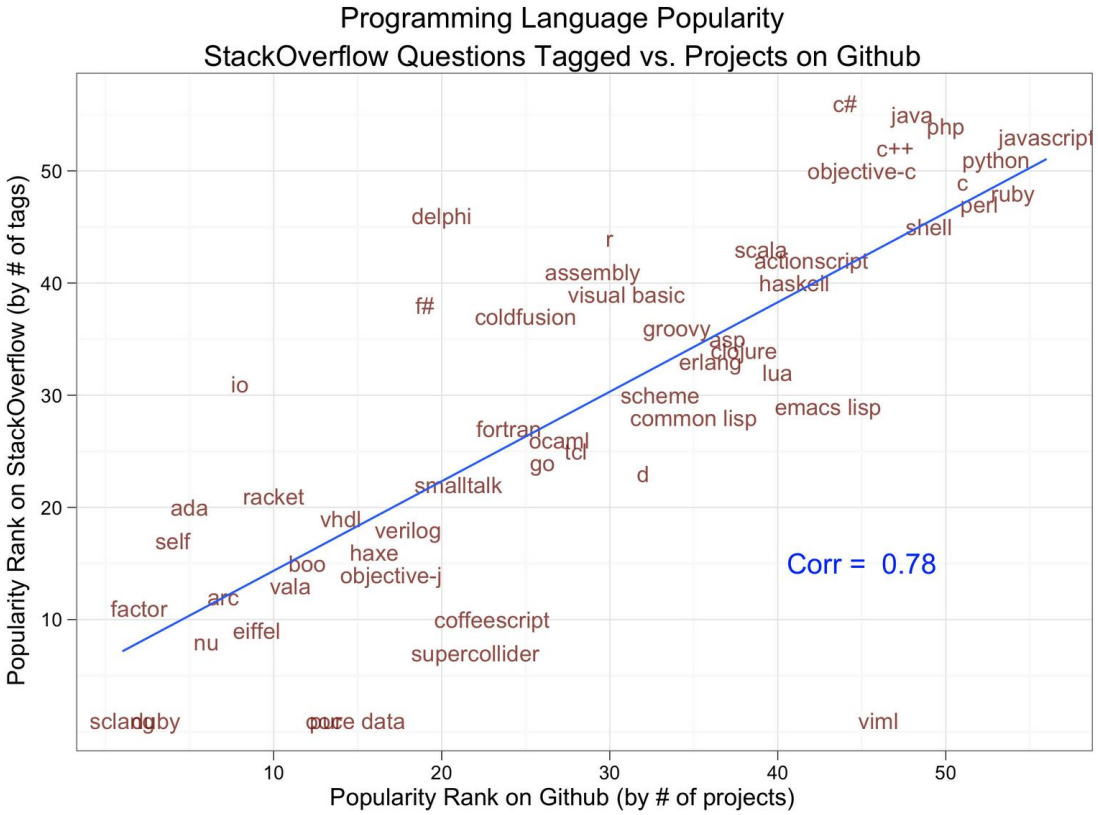
https://bn.wikipedia.org/wiki/%E0%A6%9A%E0%A6%BF%E0%A6%A4%E0%A7%8D%E0%A6%B0:Systems_Development_Life_Cycle.jpg

Odabir programskog jezika

Prostoji više različitih programskih jezika. Oni su izmišljeni da bi čovjeku olakšali komunikaciju s računalom, tj. Programiranje. Budući da bi programiranje bilo preteško kad bismo sami morali prirediti strojni kod, napravljeni su razvojni alati koji služe kao pomoć u pisanju programa. Oni podržavaju razne programerske jezike i pripremaju naše programe za izvođenje na računalu. Neki od najčešće korištenih programa su:

- FORTRAN, COBOL, CLIPPER, LOGO, PASCAL, C, C++, C#, Java, VisualBasic, HTML, CSS, PHP





Izvor: http://ethanfosse.blogspot.com/2012/03/popularity-of-programming-languages.html

Što je Python?

- programski jezik visoke razine
- razvoj začet 1991. od nizozemskog programera Guido van Rossuma
- slobodan softver
- interpreter prevodi kod tijekom izvršavanja
- multiplatformalan (izvršava se na windows, linux, macos, …, platformama)
- jednostavan
- dinamičan
- čista sintaksa – lakše čitanje koda
- velik broj modula i biblioteka
- velike mogućnosti iskorištavanja
- primjena u geoinformatici
- ne donosi neke nove revolucionarne značajke programiranju, već na optimalan način ujedinjuje sve najbolje ideje i načela rada drugih programskih jezika

Algoritam

Algoritam se najčešće predstavlja pomoću **dijagrama tijeka** i **pseudojezika**.

Dijagram tijeka je grafički način predstavljanja algoritma skupom grafičkih simbola koji označuju pojedine operacije u algoritmu.

Pseudojezik oponaša sintaksu programskih jezika koristeći izraze, tj. govorni jezik kojim oponašamo naredbe i sintaksu programskih jezika.

- Pseudo jezik ili pseudokod je „lažni“ (govorni) program (grč. pseudos - laž).
- Jezik koji nije zapisan u programskom jeziku koji bi se mogao izravno primijeniti na računalu
- Sastoji od kratkih izjava na govornom jeziku koji opisuju i ukratko objašnjavaju pojedine zadatke algoritma.

Varijabli se vrijednost pridružuje s pomoću operatora pridruživanja. Za ovu inačicu pseudo jezika to je :=.

Npr. izraz x:=13 se može čitati "varijabli x se pridružuje broj 13".

Svaka naredba ove inačice pseudo jezika završava znakom točka-zarez (;).

funkcija	operator
• Zbrajanje	+
• Oduzimanje	-
• Množenje	*
• Dijeljenje	/
• Cjelobrojno dijeljenje	DIV
• Cjelobrojni ostatak dijeljenja	MOD

a je 2,5	b je 2	c je 1

Logički podaci su podaci koji mogu poprimiti samo jednu od dvije vrijednosti, npr.: 1 ili 0.

Za rad s logičkim podacima, postoje logičke funkcije.

Logičke funkcije

funkcija	operator
• Logički I	I
• Logički ILI	ILI
• Logički NE	NE

LOGIČKE FUNKCIJE

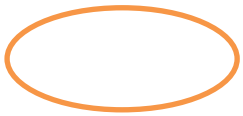
A	NE A
1	0
0	1

A	B	A I B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A ILI B
0	0	0
0	1	1
1	0	1
1	1	1

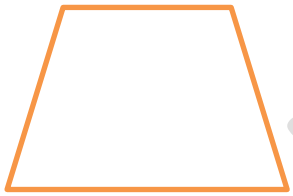
Dijagram tijeka

Algoritme za rješavanje problema najčešće pokazujemo grafički pomoću dijagrama tijeka. Dijagram tijeka grafički je prikaz algoritma. Tako prikazan algoritam vrlo je pregledan i potpuno određen. Posebno je pogodan za analize programa, traženje sličnih rješenja ili potrebne izmjene. Pri crtanju dijagrama služimo se posebnim simbolima.



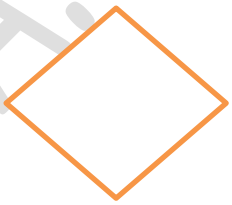
početak/kraj

ulaz podataka



izlaz podataka

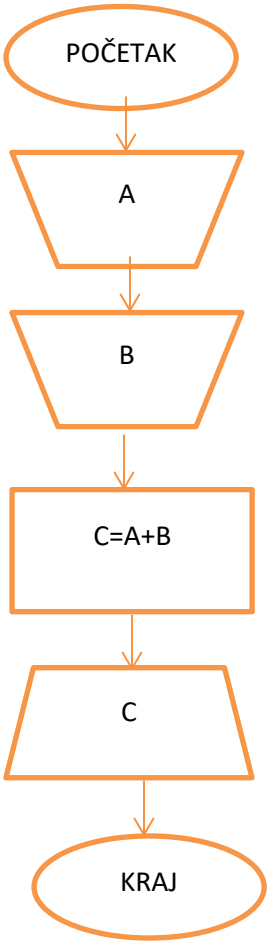
naredba



odluka

OSNOVNI ALGORITAMSKI POSTUPCI

SLIJED



početak
unesi broj A
unesi broj B
C= A+B
Ispiši C
kraj

GRANANJE

Početak

Ispis „želite li registrirati program“

Učitaj odgovor

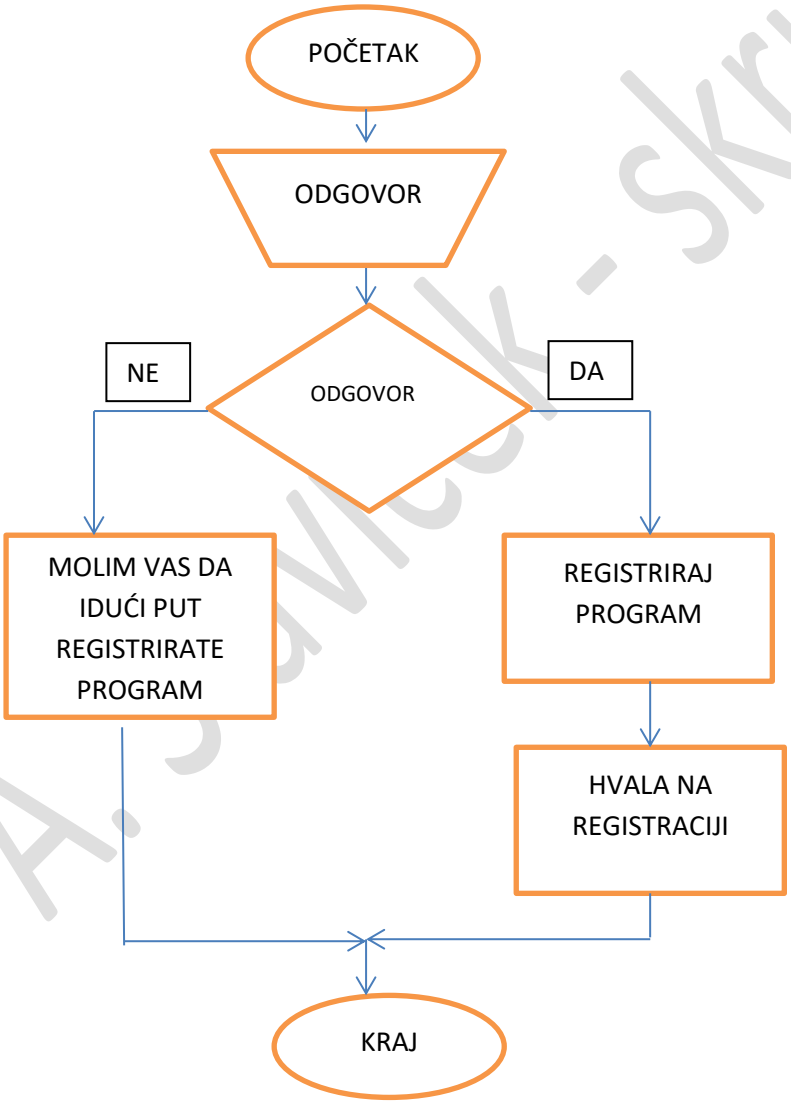
Ako je odgovor „da“ registriraj program

Ispis „hvala na registriranju“

Inače

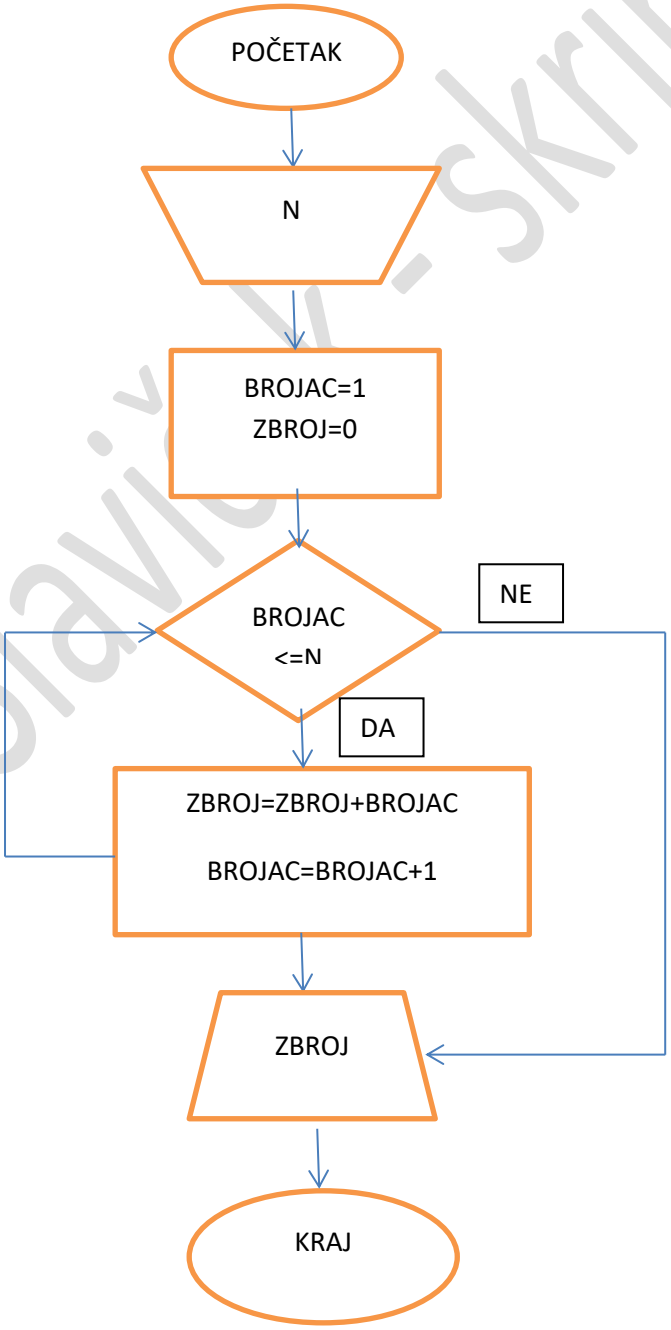
Ispis „molimo da idući put registrirate program“

Kraj



PETLJA

Početak
Učitaj N
Brojac=1
Zbroj=0
Dok je brojac <=N
Zbroj=zbroj+brojac
Brojac=brojac+1
Ispiši zbroj
Kraj

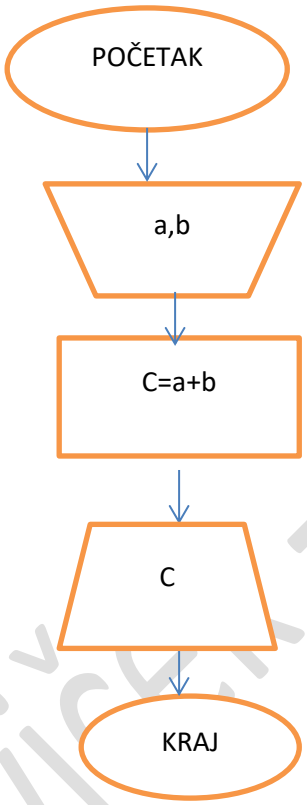


Algoritamski zadaci

Zadatak 1. Napiši algoritam i dijagram tijeka za izračunavanje zbroja dva broja i ispisivanje dobivenog zbroja.

Pseudokod

Početak
 unesi broj a
 unesi broj b
 C=a+b
 ispiši C
kraj



Python

Python Shell

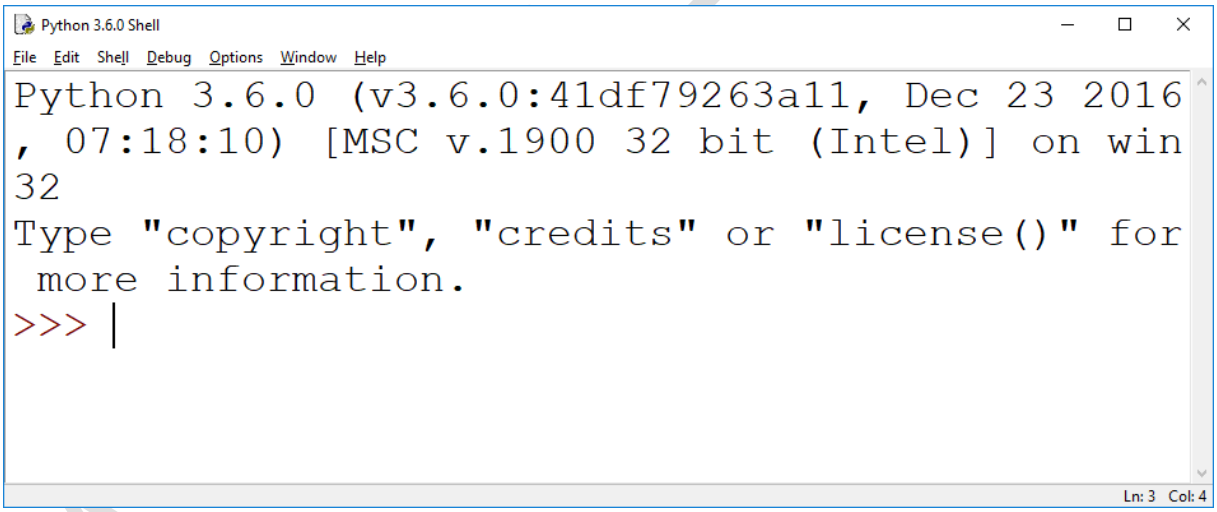
Pokretanje Python-a

Python pokrećemo otvaranjem razvojnog okruženja za Python nazvanog **IDLE** (Integrated DeveLopment Environment) na sljedeće načina:

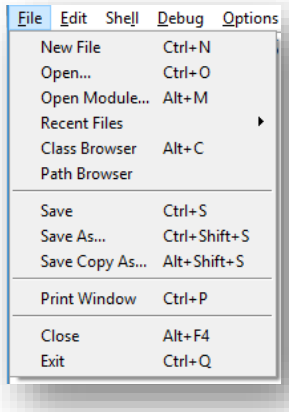
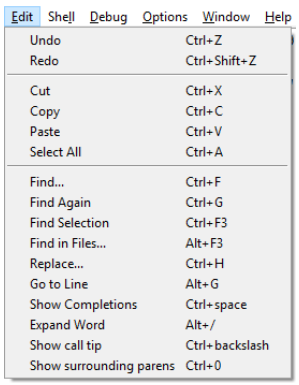
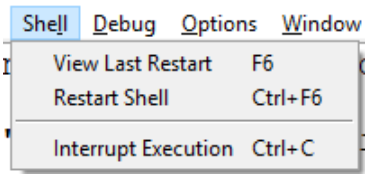
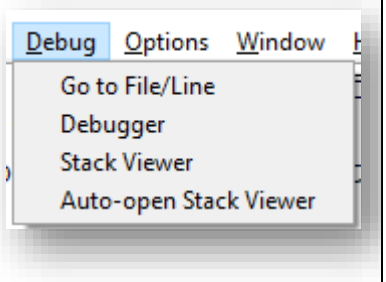
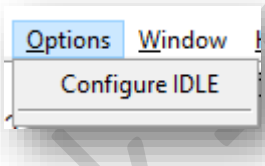
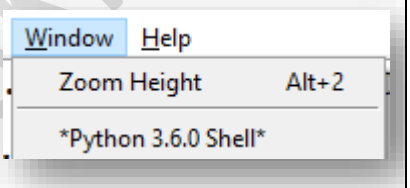
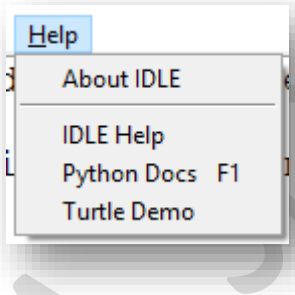
1. Start > Svi programi (All Programs) > Python 3.6 > IDLE (Python GUI)

2. Dvostrukim klikom na ikonu  *prečice na radnoj površini*

Kada se pokrene razvojno okruženje IDLE otvori se prozor koji čini interaktivno sučelje s Pythonom i naziva se Python Shell.



Izbornici u Pythonu

File	Edit	Shell
		
Debug	**Options**	**Window**
		
Help		
		

Na početku novog retka pojavljuju se tri strelice i to mjesto nazivamo znak upita (*engl*. *prompt*)

```
>>>
```

gdje unosimo neku naredbu ili podatak.

Osnovni tipovi podataka u Pythonu su:

- Cijeli brojevi
- Decimalni brojevi
- Znakovni niz ili string

Cijeli brojevi

Ako se nakon znaka upita utipka neka vrijednost, Python ispisuje tu istu vrijednost:

```
>>> 1024
1024
```

>>> Python ispisuje vrijednost plavom bojom tako da možemo razlikovati ono što smo utipkali od Pythonovog ispisa.

Naziv za cjelobrojni podatak u Pythonu je **int**.

Broj znamenki cijelog broja nije ograničen:

```
>>> 5568
5568
>>> -552698
-552698
>>> 123456789123456789
123456789123456789
```

Decimalni brojevi

Prilikom upisivanja decimalnog broja koristimo decimalnu točku. Po toj točki Python prepoznaje decimalni broj, tj. da je tip **float**.

Nekoliko primjera:

```
>>> 3.14
3.14
>>> 0.245689
0.245689
>>> 1.896484
1.896484
>>> 0.000456
0.000456
>>>
```

Pretvaranje decimalnih brojeva u cijele i obratno

Python ima ugrađene funkcije int() i float() pomoću kojih možemo pretvoriti decimalni broj u cijeli broj i obratno.

Za pretvaranje decimalnog broja u cijeli koristimo funkciju int():

```
>>> int (13.14)
13
>>> int (3.596652)
3
>>>
```

Pretvaranje se sastoji u zaokruživanju decimalnog broja u prvi manji cijeli broj.

Za pretvaranje cijelog broja u decimalni koristi se funkcija **float**():

```
>>> float (56)
```

56.0

>>> float (123456)

123456.0

Pretvaranje se svodi na dodavanje decimalne točke i prikazu broja s decimalnom nulom.

Python kao kalkulator

Operator	Opis djelovanja	Primjer
+	Zbrajanje	>>> 649+749 1398
-	Oduzimanje	>>> 1455.36-2469.85 -1014.49
*	Množenje	>>>12.56*15.63 196.3128
/	Dijeljenje	>>> 638.49/27.32 23.370790629575403
//	Cjelobrojno dijeljenje	>>> 68//4.324 15.0
%	Ostatak dijeljenja	>>> 27%5 2

Znakovni niz ili string

Osnovni tip podataka za prikaz teksta je znakovni niz ili string i naziva se str. U Pythonu se taj tip zove str, a vrijednost znakovnog niza obilježava se jednostrukim ili dvostrukim navodnim znacima na početku i na kraju niza.

Znakovni niz ili string

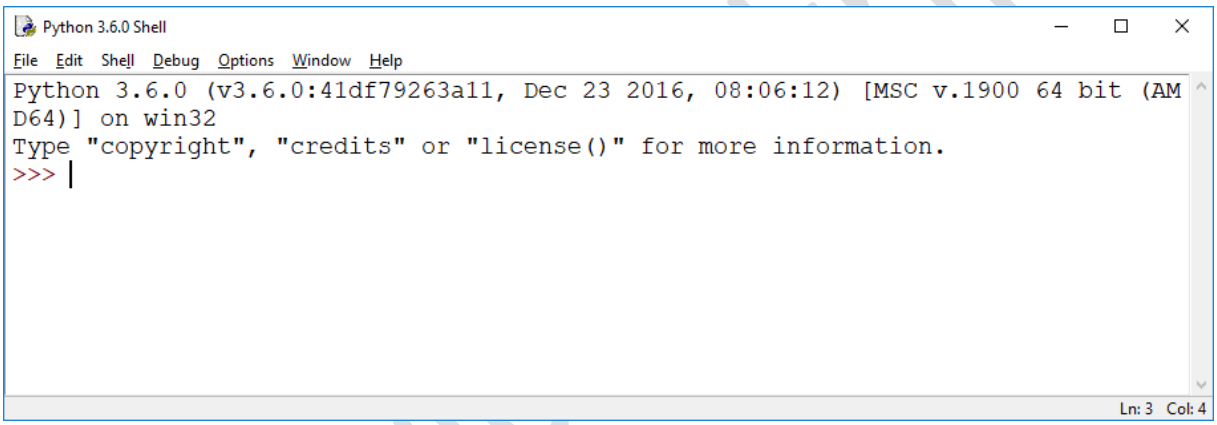
```
>>> g=s=z="Geodetska škola, Zagreb"
>>> g
'Geodetska škola, Zagreb'
>>> s
'Geodetska škola, Zagreb'
>>> z
'Geodetska škola, Zagreb'
>>>
```

Python – radno okruženje

- Python Shell ili IDLE (Integrated DeveLopment Environment)
- GUI – Graphical User Interface

Dva načina rada:

1. Interaktivni
2. Skriptni

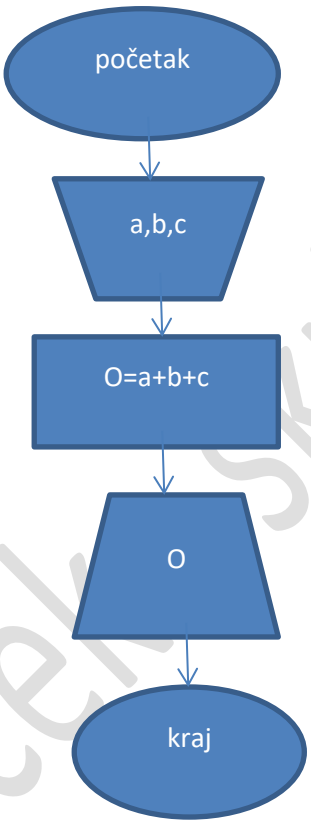


Zadatak 2. Napišimo algoritam i dijagram tijeka za izračunavanje opsega raznostraničnog trokuta, kome su zadane duljine stranica a,b i c.

Pseudokod

Početak
 unesi a,b,c
 O=a+b+c
 ispiši = O

kraj




```
>>> a=int(input("stranica a je velika="))
stranica a je velika= 5
>>> b=int(input("stranica b je velika="))
stranica b je velika=8
>>> c=int(input("stranica c je velika="))
stranica c je velika=9
>>> print("opseg jednakostraničnog trokuta iznosi =", a+b+c)
opseg jednakostraničnog trokuta iznosi = 22
```

```
>>> a=int(input("broj a iznosi="))
broj a iznosi= 5
>>> b=int(input("broj b iznosi="))
broj b iznosi= 6
>>> print("zbroj tih brojeva iznosi=", a+b)
zbroj tih brojeva iznosi= 11
>>>
```

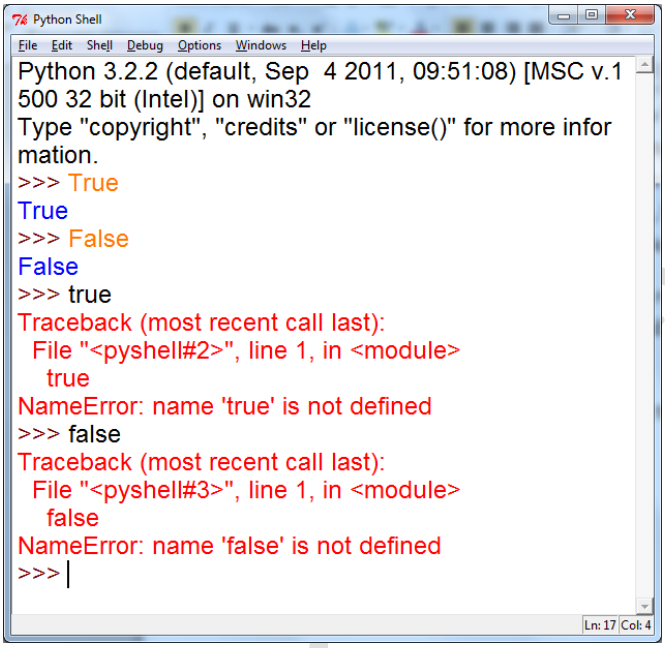
Logički ili Booleov tip podataka

Naziv logički tip potječe iz logike – grane filozofije koja se bavi oblicima ispravne misli i metodama spoznaje. Osnovni pojam u logici je **logički sud**. Pod tim nazivom podrazumijeva se svaka tvrdnja koja se ocjenjuje samo s gledišta **istinitosti** ili **lažnosti**. Neka druga svojstva sudova (primjerice: lijepo, ružno, dobro, loše) u logici se ne razmatraju. logički sudovi moraju biti valjani. Valjani sudovi moraju biti tako oblikovani da mogu biti **istiniti ili lažni**.

Za istraživanje sudova i složenih sudova razvijena je posebna grana matematike – matematička logika. Osnovne matematičke logike čini algebra sudova znana po nazivivima **logička algebra** ili **Booleova algebra**.

U Pythonu se logički tip naziva **bool**. Taj tip može primiti samo dvije vrijednosti:

True i **False**.



Python ima ugrađene funkcije **int()** i **bool()** pomoću kojih možemo ustanoviti cjelobrojnu vrijednost neke logičke vrijednosti i obrnuto, kakva je logička vrijednost neke cjelobrojne vrijednosti:

```
>>> int(True)
1
>>> int(False)
0
>>> bool(1)
True
>>> bool(0)
False
>>>
```

Pokazuje se da je cjelobrojni ekvivalent vrijednosti **True** jednak **1** te vrijednosti **False** jednak 0.

LOGIČKI OPERATORI

Najčešće se koriste tri operatora: **AND**, **NOT** i **OR**.

AND - operator koji je istinit samo ako su uvjeti s obje njegove strane istiniti (1 AND 1).

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

NOT – operator koji pretvara 0 u 1 i obrnuto.

A	NOT A
0	1
1	0

OR – operator koji nije istinit samo ako uvjeti s obje njegove strane nisu istiniti (0 OR 0).

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Vidimo da operator **NOT** uvijek invertira vrijednost (0 postaje 1 i 1 postaje 0), operator **AND** je istinit samo ako su oba parametra istinita, a **OR** je istinit kad god je barem jedan parametar istinit.

PONAVLJANJE:

1. Nabroji generacije programskih jezika
2. Nabroji faze razvoja programa
3. Nabroji neke od najčešće korištenih programa
4. Razvoj Pythona započet je _____ godine.
5. Algoritam se najčešće predstavlja pomoću _____ i _____
6. Dijagram tijeka je _____ prikaz programa
7. Nacrtaj grafičke simbole i objasni njihovo značenje koji označuju pojedine operacije u algoritmu
8. Osnovni algoritamski postupci su:
9. Napiši pseudokod za unos dva broja i izračunavanje njihove sume i ispis sume.
10. Napiši algoritam i dijagram tijeka za izračunavanje opsega raznostraničnog trokuta, kome su zadane duljine stranica a,b i c.
11. Osnovni tipovi podataka u Pythonu su:
12. Za ispis cijelog broja u binarnom odnosno heksadekadskom obliku postoje funkcije _____ i _____
13. U Pythonu se logički tip naziva _____, koji može poprimiti dvije vrijednosti _____ i _____
14. Int (True) = ?, Int (False)= ?
15. Posebni znak za oblikovanje teksta \t znači rijelaz u novi redak? Da-ne
16. Naredba True i true je ista naredba? DA – NE
17. Znakovni nizovi mogu se ograditi ili jednostrukim ili dvostrukim znakovima? DA – NE

Rješavanje problema i stjecanje novih spoznaja računalom

Koraci izgradnje programa su:
-analiza zahtijeva,
-specifikacija problema,
-odabir algoritma i zasnivanje programa
-pisanje programa
-provjeravanje i ispitivanje programa
-održavanje programa.

Algoritam - konačan niz koraka koji vodi prema rješenju nekog problema .

Algoritamski proces - obavljanje programa

Algoritam može izvoditi čovjek,životinja ili neki uređaj.

Izvori algoritma su:
- Praktično iskustvo-imitacijski algoritmi,
- Znanstvena teorija- teorijski algoritmi,
- Skup postojećih algoritma- konstrukcijski algoritmi,
- Domišljatost stvaratelja.

Svojstva algoritma su:
-Ispravnost (korektnost)- radi ispravno za sve dopuštene ulaze,
-Trajanje-broj osnovnih operacija od kojih se algoritam sastoji.

Dva su algoritma ekvivalentna ako su:
-Dopuštanje klase objekata i za jedan i za drugi algoritam jednake,
-Završna stanja jednog i drugog algoritma jednake za jednaka početna stanja.

PONAVLJANJE

Zadatak 1.

Unesi stranicu a i b pravokutnika i izračunaj površinu i opseg pravokutnika.

```
>>> a=int(input("stranica a iznosi="))
stranica a iznosi=45
>>> b=int(input("stranica b iznosi="))
stranica b iznosi=12
>>> P=a*b
>>> P
540
>>> O=2*a+2*b
>>> O
114
>>>
```

Zadatak 2.

Unesi radius kruga i izračunaj opseg i površinu kruga.

```
>>> r=float(input("unesi radius r="))
unesi radius r=12.5
>>> pi=float(input("unesi pi="))
unesi pi=3.14
>>> P=r*r*pi
>>> P
490.625
>>> O=2*r*pi
>>> O
78.5
>>>
```

Zadatak 3.

Unesi četiri broja i ispiši aritmetičku sredinu.

```
>>> b1=int(input("stranica a iznosi="))
stranica a iznosi=12
>>> b=24
>>> c=45
>>> d=5
>>> aritsred=(a+b+c+d)/4
>>> print("aritmetička sredina četiri broja=", aritsred)
aritmetička sredina četiri broja= 29.75
>>>
```

Zadatak 4.

Unesi jedan broj od 20. Provjeri dali je veći od 10 ili manji ispiši komentar („broj je veći ili broj je manji“).

```
>>> broj=int(input("unesi jedan broj do broja 20="))
unesi jedan broj do broja 20=9
>>> if broj>10:
        print("broj je veci od 10")
else:
        print("broj je manji od 10")


broj je manji od 10
>>>
```

FORMATIRANI ISPIS

Formatirani ispis izuzetno je pogodan kada jednim ispisom želimo ispisati više varijabli popraćene odgovarajućim tekstom.

Funkcija **print()** omogućuje precizniji ispis, a sastoji se od varijabli i stringova.

Oblik pisanja:

Print('st_1{br_1},st_2{br_2},…,st_n{br_n}',format(v_1,v_2,…,v_n))

Print('st1{br1),st2{br2},….,stn{brn}',format(v1,v2,…vn))

St_1,st_2,…,st_n – proizvoljan tekst

V_1, v_2,…, v_n – popis varijabli koje želimo ispisati

Br_1, br_2,…, br_n – predstavljaju redne brojeve varijabli unutar popisa

```
>>> print('zbroj brojeva {0} i {1} je: {2}'.format (4,5,4+5))
zbroj brojeva 4 i 5 je: 9
>>> print('brojevi zapisani redoslijedom su {3}, {2}, {1}, {0}'.format(5,4,3,2))
brojevi zapisani redoslijedom su 2, 3, 4, 5
>>> print('brojevi zapisani redoslijedom su {0}, {1}, {2}, {3}'.format(5,4,3,2))
brojevi zapisani redoslijedom su 5, 4, 3, 2
```

```
>>> print('zbroj brojeva {1} i {2} je {3}, a  {2} i {1} je {3}' .format (25, 35,
 25+35, 35+25))
zbroj brojeva 35 i 60 je 60, a  60 i 35 je 60
>>> print('zbroj brojeba {0} i {1} je isto što {1} + {0} = {2}'.format(25,35, 25
+35))
zbroj brojeba 25 i 35 je isto što 35 + 25 = 60
>>>
```

```
>>> print('zbroj brojeva {1} i {2} je {3}, a  {2} i {1} je {3}' .format (25, 35,
 25+35, 35+25))
zbroj brojeva 35 i 60 je 60, a  60 i 35 je 60
>>> print('zbroj brojeba {0} i {1} je isto što {1} + {0} = {2}'.format(25,35, 25
+35))
zbroj brojeba 25 i 35 je isto što 35 + 25 = 60
>>>
>>> print('zbroj brojeva {} i {} je: {}'.format (127,423,127+423))
zbroj brojeva 127 i 423 je: 550
>>> print('zbroji {0} i {1} je: {4}, a {2} i {3} je: {5}'.format (25,47,52,117,
25+47, 52+117))
zbroji 25 i 47 je: 72, a 52 i 117 je: 169
>>>
```

Unutar vitičastih zagrada brojeve možemo izostaviti ako se podrazumijeva da podaci idu uzlaznim redom.

Ukoliko želimo pobliže označiti koji tip podataka ispisujemo i u tom ćemo slučaju unutar vitičastih zagrada iza rednog broja varijable napisati dvotočku i nakon toga oznaku tipa:

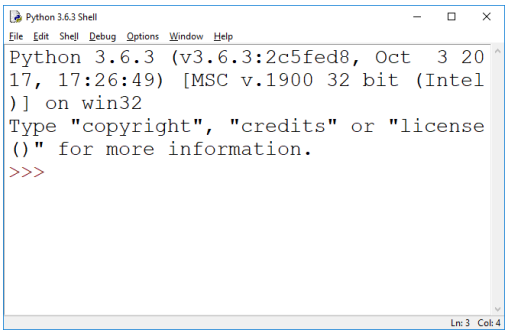
- d – ako ispisujemo cijeli broj
- f – ako ispisujemo realni broj
- s – ako ispisujemo string

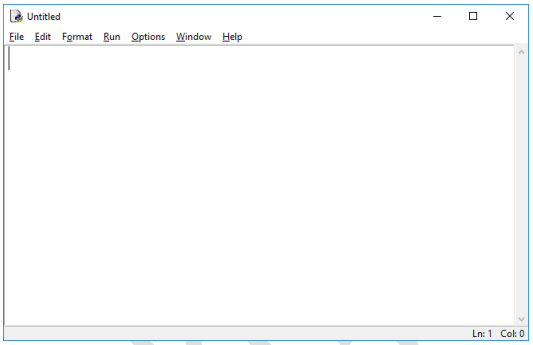
```
>>> print('{2s} brojeva {0:d} i {1:d} je {3:f}' .format(3,4, 'Količnik', 3/4))
Količnik brojeva 3 i 4 je 0.75000
```

Neka osnovna pravila pisanja programa

U Pythonu možemo raditi s dva sučelja. To su?

- Interaktivno sučelje (Python Shell)
- Uređivački dio sučelja IDLE - editor (Integrated DeveLopment Enviroment - Integrirano razvojno okruženje)





Osnovni tipovi podataka u Pythonu jesu:

- cijeli broj - int
- broj s pomičnom točkom - float
- logički ili Booleov tip – bool (ugrađena funkcija bool() – može poprimiti samo dvije vrijednosti: True i Fals.
- znakovni niz ili string - str

Kako se piše tekst u Pythonu (znakovni niz)?

Koje su aritmetičke operacije u Pythonu?

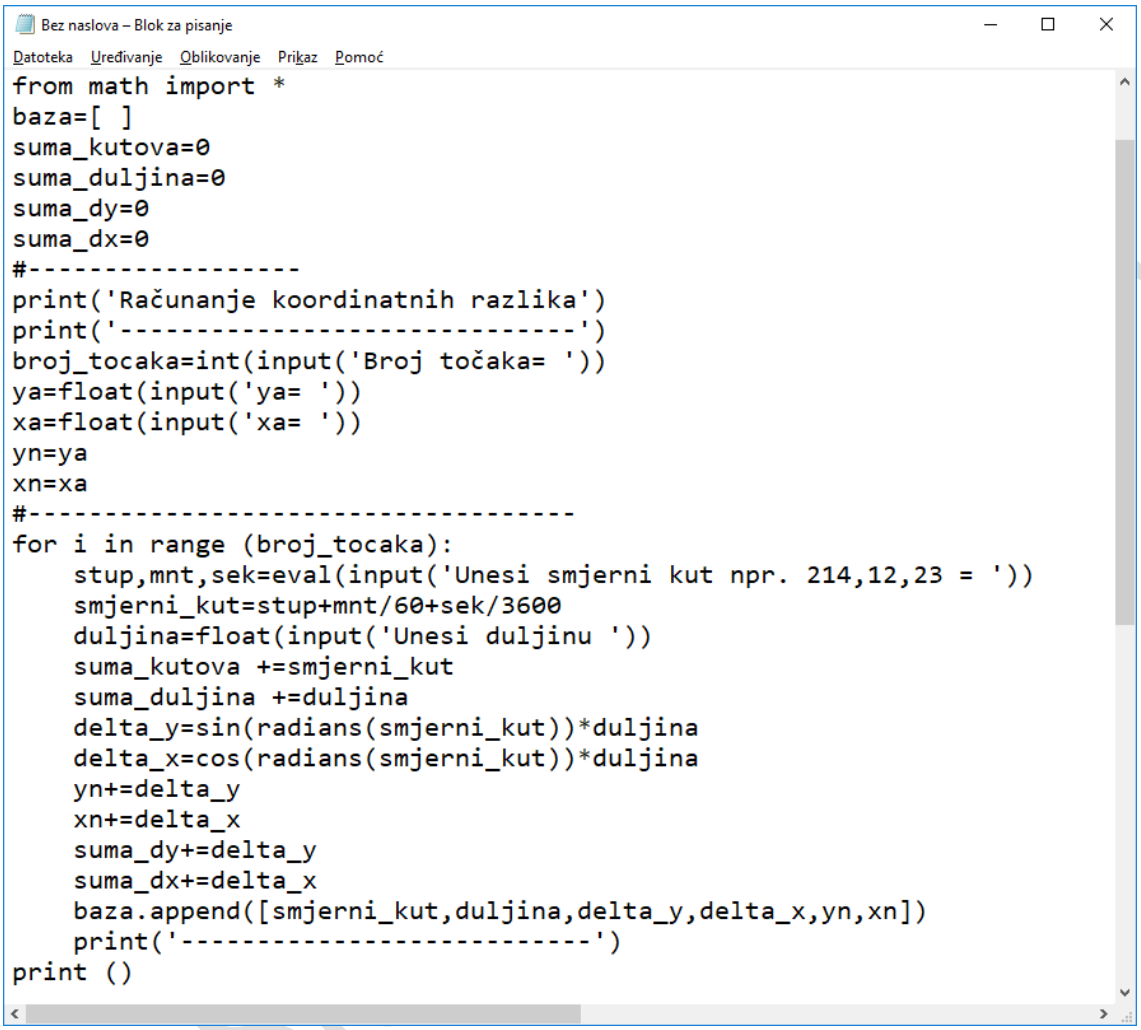
Odgovor: + zbrajanje , - oduzimanje, ** potenciranje, * množenje, / dijeljenje, // cjelobrojno dijeljenje , % računanje ostatka dijeljenja.

Red prvenstva aritmetičkih operatora je:

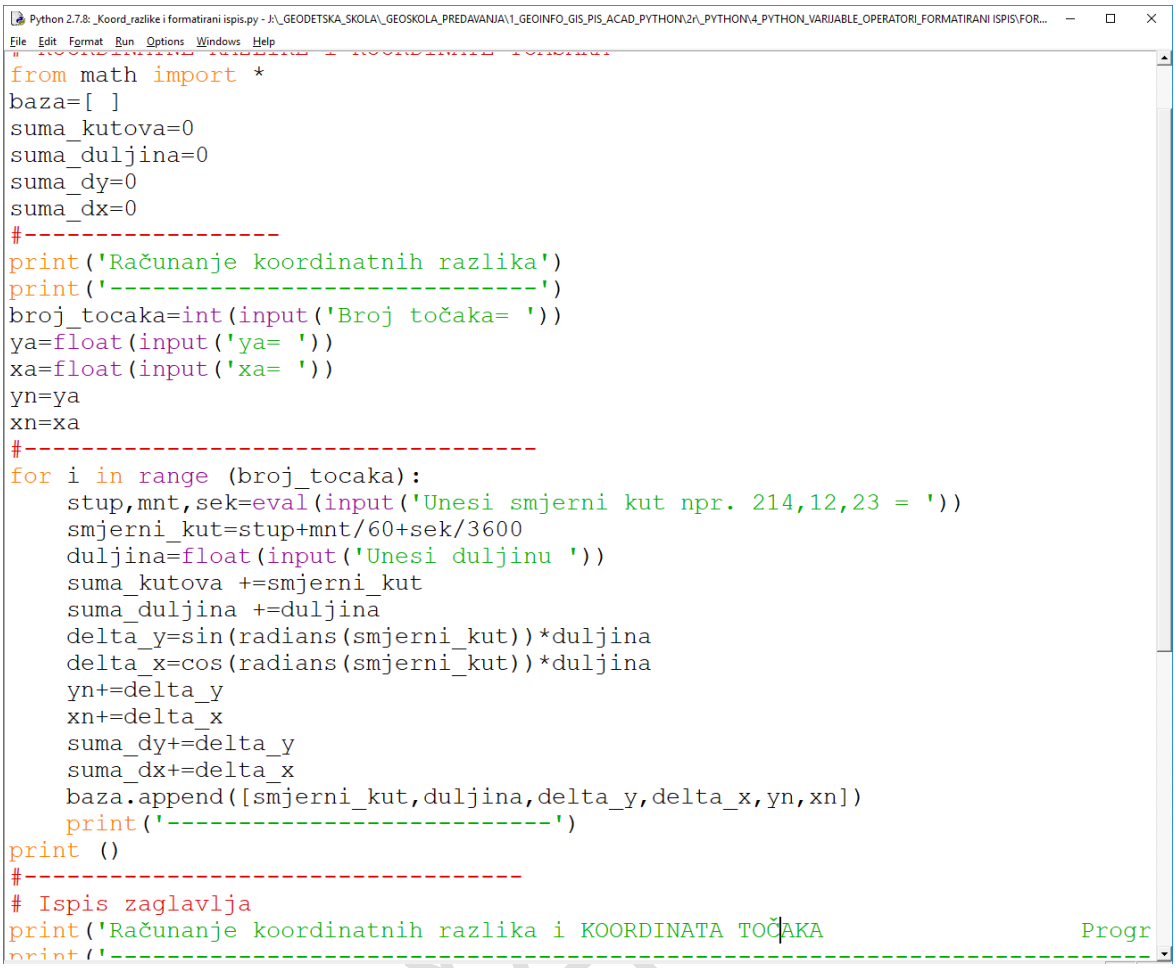
a) ** potenciranje
b) – negacija
c) *, /, //, %, množenje, dijeljenje, ostatak
d) Zbrajanje, oduzimanje

Tekst je obojen različitim bojama kako bi se lakše prepoznali karakteristični jezični elementi i održavala pravila pisanja programa (pravila sintakse). To obojenje čini programe čitljivijim.

Program napisan u bloku za pisanje



Program napisan u Pythonu



Ako se, primjerice, po završetku utipkavanja nekog stringa ne promijeni boja znakova, znači da nismo napisali završni navodni znak.

Primjer:

```
>>> ime=Ivan
Traceback (most recent call last):
  File "<pyshell#56>", line 1, in <module>
    ime=Ivan
NameError: name 'Ivan' is not defined
>>> ime='Ivan
SyntaxError: EOL while scanning string literal
>>> ime='Ivan'
>>> ime
'Ivan'
>>>
```

Pravila za pisanje imena varijabli

```
>>> a=7
>>> A=9
>>> a
7
>>> A
9
>>>
```

U Pythonu se imena sastoje od proizvoljnog broja slova, znamenki i podvlaka, s tim da se smiju početi znamenkom Python ima posebna imena koja počinju s dvije podvlake i to treba izbjegavati. Najbolje je odabrana imena započinjati slovom.

Već smo naučili da Python razlikuje velika i mala slova, tako da su a i A različita imena.

Python upotrebljava neke riječi, zovemo ih ključnim riječima, za svoje svrhe i korisnik ih ne smije upotrebljavati kao imena.

Susreli smo već neka od tih imena: int, float, bool, str, True, False.

Evo popisa svih ključnih riječi u Pythonu:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	it	or	yield
assert	else	import	pass	
break	except	in	raise	

Primjer:

```
>>> global=157
SyntaxError: invalid syntax
>>> pass=356
SyntaxError: invalid syntax
>>> raise=256
SyntaxError: invalid syntax
>>>
```

Navodimo nekoliko valjanih imena:

x, X, x_1, X_1, x_2, X_2, visina, Točka, reper, visina_repera

```
>>> x=51
>>> x
51
>>> X=23
>>> X
23
>>> X_1=546
>>> X_1
546
```

```
>>> visina=123.52
>>> visina
123.52
>>> Točka=129
>>> Točka
129
```

Sljedeća imena nisu valjana:

x.5, Z(2), kta_$	jer se u imenu upotrebljavaju ostali znakovi
3_x, 155_reper, 3_mjerenje	jer je prvi znak u imenu znamenka
as, raise, not	jer su to ključne riječi

```
>>> x.1=222
SyntaxError: invalid syntax
>>> x.5=9
SyntaxError: invalid syntax
>>> kta_$=500
SyntaxError: invalid syntax
>>> 3_x=333
SyntaxError: invalid token
>>> 3_mjerenje=123.65
SyntaxError: invalid token
>>> raise=525
SyntaxError: invalid syntax
>>>
```

Iako se sastoji od niza znakova, ime nije znakovni niz, odnosno string. Ustanovili smo da je string tip podataka. Poslije ćemo naučiti da imena imaju i druge važne funkcije u Pythonu, no jedna od najvažnijih uloga je pohranjivanje vrijednosti kako bi se vrijednost kasnije mogla upotrijebiti.

Posebno treba istaknuti da Python upotrebljava za kodiranje znakova Unicode i da su naši dijakritički znakovi dopušteni. Prema tome, u imenima možemo upotrebljavati sve naše znakove, što kod drugih jezika nije moguće.

U hrvatskom jeziku dijaktrički znakovi u abecedi - su č, ć, đ, š i ž, a dijakritici su č, ć, đ, dž, š i ž (sastavljeni od dijaktritičkih znakova).

```
>>> čestica_152_m2=1567
>>> čestica_152_m2
1567
>>>
```

Iako se možemo koristiti i našim znakovima, preporučeno je, posebice za imena klasa, funkcija, metoda itd., koristi se područjem koji je definirano ASCII kodom.

Postavlja se pitanje kako odabrati imena. Kraća imena su jednostavnija i brže ih zapisujemo, ali nakon nekog vremena teško ćemo se sjetiti što nam ona predstavljaju.

Dulja imena zahtijevaju više tipkanja, ali će nas i nakon duljeg vremena podsjećati što nam ona predstavljaju. Preporučljivo je izabrati imena koja će nam olakšati uporabu programa kroz neko dulje vrijeme.

NAREDBE PRIDRUŽIVANJA

Opći oblik naredbe produživanja je:

varijabla = izraz

pri čemu varijabla mora biti ime odabrano u skladu s pravilima oblikovanja imena u Pythonu.

Znak = ovdje ima značenje pridruživanja, pa ga zovemo znakom pridruživanja (*u matematici ga zovemo znakom jednakosti)*.

Vrijednost koja se piše desno od znaka = bit će pridružena varijabli čije smo ime naveli s lijeve strane znaka pridruživanja.

```
>>> a=15
>>> b=21
>>> c=32
>>> kateta_a=22
>>> kateta_b=25
>>> hipotenuza=33.30
>>> a
15
>>> b
21
>>> c
32
>>> kateta_a
22
>>> kateta_b
25
>>> hipotenuza
33.3
```

Ako želimo ispisati vrijednost koju pokazuje a (b,c,…), tada treba utipkati a i Python će je jednostavno ispisati.

Varijabla se može upotrijebiti i u izrazu s desne strane znaka pridruživanja. U tom će se slučaju za izračunavanje rabit vrijednost na koju varijabla trenutno pokazuje. Dakle, isto se može pojaviti i s lijeve i desne strane znaka pridruživanja. U Pythonu se može pisati x=x+15, *što u matematici nema smisla.* Treba zapamtiti da Python izračunava izraz s desne strane znaka jednakosti i kada je to obavio, dobivenoj vrijednosti pridružuje tu novu vrijednost.

Naredba x=x+15 uvećat će vrijednost na koju pokazuje x za 15.

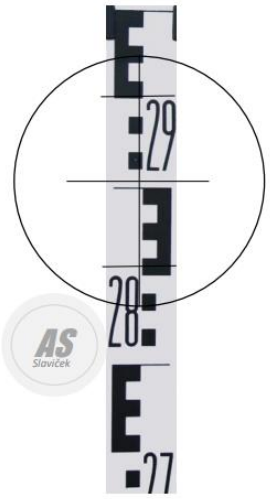
Primjeri:

```
>>> x=20
>>> x
20
>>> x=x+15
>>> x
35
>>>
```

```
>>> stranica_a=15
>>> stranica_b=stranica_a*2
>>> stranica_b
30
>>> a=12
>>> a=a**2
>>> a
144
```

Zadatak:

Izračunaj u interaktivnom sučelju duljinu do letve i kontrolu očitanja.



Gornja nit	g = 2,951
Srednja nit	s = 2,903
Donja nit	d = 2,856
	g –d = 0,095

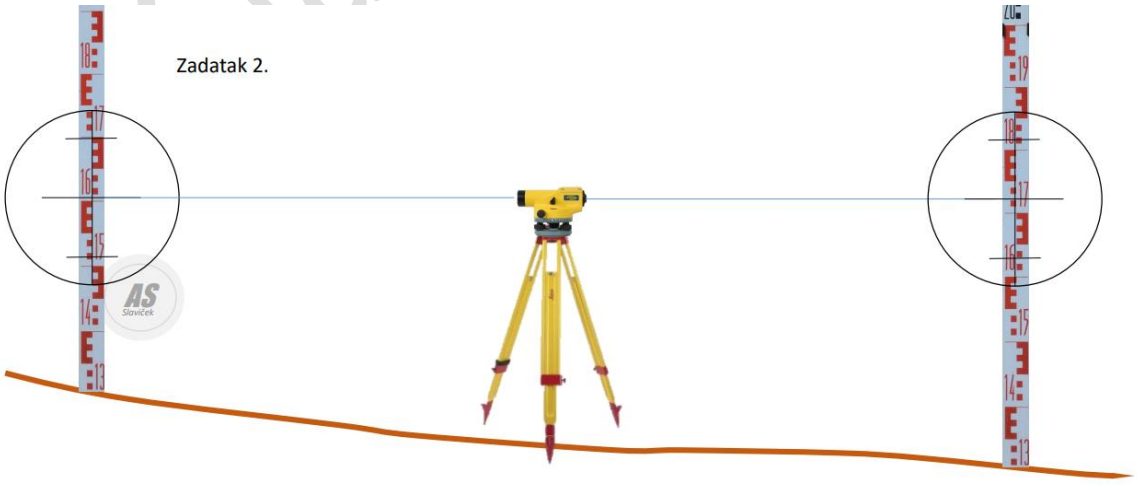
Kontrola očitanja: (g +d)/2
(2,951-2,856)/2= 2,9035

D = 0,095 * 100 = 9,50 m

Očitanje letve
Metri, decimetri i centimetri se čitaju direktno, a milimetre procjenjujemo.
Npr. gornja nit: 2 m 9 decimetara 5 centimetara i 2 mm (procjenjujemo).

```
>>> g=2.951
>>> d=2.856
>>> kontrola=(g+d)/2
>>> kontrola
2.9035
>>> duljina=(g-d)*100
>>> duljina
9.500000000000002
>>> round(duljina,3)
9.5
```

Zadatak:



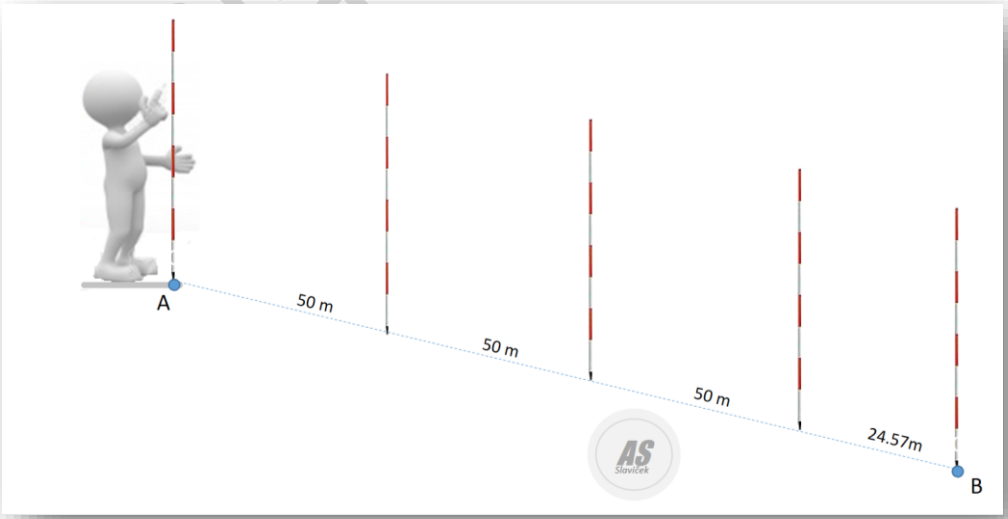
Letva_z

Letva_p

Visinska razlika (Δh) = Letva_z – Letva_p

```
>>> letva_z=1.608
>>> letva_p=1.722
>>> delta_h=letva_z-letva_p
>>> delta_h
-0.11399999999999988
>>> round(delta_h,3)
-0.114
>>>
```


Zadatak: Odredi aritmetičku sredinu mjerenih duljina od točke A do B.



Duljina između točaka A i B mjerena je tri puta:

l1 = 174.57

l1 = 174.54

l1 = 174.59

```
>>> l1=174.57
>>> l2=174.54
>>> l3=174.59
>>> arit_sredina=(l1+l2+l3)/3
>>> arit_sredina
174.5666666666667
>>> round(arit_sredina,3)
174.567
>>>
```

Možemo li ovaj program ponavljati proizvoljan broj puta i zadavati različite vrijednosti duljina?

Da. Program trebamo napisati u editoru (uređivačkom dijelu) Pythona.

Kada se program pokrene otvara interaktivni prozor *Python Shell* kao ulazno-izlazni prostor.

Naredbe višestrukog pridruživanja

U Pythonu je moguće jednom naredbom obaviti više pridruživanja.

Takva naredba u općem slučaju ima sljedeći oblik:

varijabla_1, varijabla_2, varijabla_3,… = izraz_1, izraz_2, izraz_3,…

S desne strane znaka pridruživanja mora se nalaziti onoliko izraza odvojenih zarezima koliko je s lijeve strane imena varijabli.

Primjeri u interaktivnom sučelju:

```
>>> a,b=12,35
>>> a
12
>>> b
35
>>> kateta_a,kateta_b=156.45,124.69
>>> kateta_a
156.45
>>> kateta_b
124.69
>>> l1,l2,l3,l4=125.36,125.32,125.39,125.34
>>> l1
125.36
>>> l2
125.32
>>> l3
125.39
>>> l4
125.34
>>>
```

Ponovimo

Python razlikuje velika i mala slova u pisanju imena varijabli.

Python upotrebljava ključne riječi koje nije dopušteno upotrijebiti kao imena. Neke od ključnih riječi su: int, float, bool, str, True, False,.. Te su riječi naglašene drugom bojom.

Nekoj varijabli vrijednost pridružujemo znakom =, npr. a=255.

Tijekom istog programa varijabli se mogu pridruživati različiti tipovi podataka (int, float).

Moguće je upotrebljavati i naredbu višestrukog pridruživanja (a,b,c = 12,25,36).

Zadatak 1. Napiši program koji će učitati ime korisnika i broj njegovih godina. Ispiši ime, broj godina i broj dana života korisnika.

```
>>> ime=input("kako se zoveš?")
kako se zoveš? Tihana
>>> godina=int(input("koliko imas godina?"))
koliko imas godina? 16
>>> print(ime, "imaš", godina, "a to iznosi", 365*16, "dana")
 Tihana imaš 16 a to iznosi 5840 dana
>>> print(ime, "imaš", godina, "a to iznosi", 365 * godina, "dana")
 Tihana imaš 16 a to iznosi 5840 dana
>>> print(ime, "imaš", godina, "godina, a to iznosi", 365 * godina, "dana")
 Tihana imaš 16 godina, a to iznosi 5840 dana
```

Zadatak 2. Napiši program koji će učitati veličinu kuta u stupnjevima, a zatim stupnjeve pretvoriti u radijane.

```
>>> stupnjevi=int(input("unesi stupnjeve"))
unesi stupnjeve 35
>>> minute=int(input("unesi minute"))
unesi minute45
>>> sekunde=int(input("unesi sekunde"))
unesi sekunde37
>>> dstupnjeva=stupnjevi + minute/60 + sekunde/3600
>>> radijan=dstupnjeva*3.14159/180
>>> print("kut ima", dstupnjeva, "stupnjeva")
kut ima 35.76027777777778 stupnjeva
>>> print("kut ima", radijan, "radijana")
kut ima 0.6241340614660494 radijana
>>>
```

Primjeri:

```
>>> x=100
>>> x
100
>>> X
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module
    X
NameError: name 'X' is not defined
>>> x
100
>>> A=1000
>>> A
1000
>>> B=10
>>> C=A+B
>>> C
1010
>>> C=A/B
>>> C
100.0
>>> C=A//B
>>> C
100
>>> X=10
>>> Y=100
>>> Z=1000
>>> X=X+1
>>> X
11
>>> Y=Y+X
>>> Y
111
>>> Z=Z+Y
>>> Z
1111
>>> baza=10
>>> dvostruko=2*baza
>>> dvostruko
20
>>> baza=100
>>> dvostruko
20
>>> dvostruko=2*baza
>>> dvostruko
200
>>>
```

NAREDBE VIŠESTRUKOG PRIDRUŽIVANJA

```
>>> a=100
>>> b=15
>>> zbroj, razlika = a+b, a-b
>>> zbroj
115
>>> razlika
85
>>>
```

```
>>> a=100
>>> b=15
>>> zbroj, razlika = a+b, a-b
>>> zbroj
115
>>> razlika
85
>>>
>>> količnik, ostatak = a//b, a%b
>>> količnik
6
>>> ostatak
10
>>>
```

```
>>>zbroj=15
>>> razlika=17
>>> print('zbroj={} razlika={}'.format(zbroj, razlika))
zbroj=15 razlika=17
>>> količnik=15
>>> ostatak=10
>>> print('količnik={} ostatak = {}' .format(količnik,razlika))
količnik=15 ostatak = 17
>>> print('količnik={}\nostatak={}' .format(količnik,ostatak))
količnik=15
ostatak=10
>>>
```

ZAMJENA VRIJEDNOSTI VARIJABLI

```
>>> a=100
>>> b=10
>>> c=a
>>> a=b
>>> b=c
>>> print(a,b)
10 100
>>>
```

NAREDBA VIŠESTRUKOG PRIDJELJIVANJA

```
>>> x=100
>>> y=10
>>> x,y=y,x
>>> print(x,y)
10 100
>>> x=1
>>> y=2
>>> z=3
>>> x,y,z=z,x,y
>>> print(x,y,z)
3 1 2
>>>
```

PONAVLJANJE

1. Napiši program koji će unositi stranicu a i b pravokutnika i ispisivati površinu i opseg pravokutnika.

2. Napiši program koji ce unositi prirodni broj n i ispisivati površinu kvadrata čija stranica ima duljinu n.

3. Cijena računala iznosi n kuna. Ti imas m kuna (m<n). Napiši program koji će unositi cijenu računala n i iznos m koji ti imaš a ispisivati iznos koji je još potreban za kupnju računala.

4. Napiši program koji će unositi dva prirodna broja a i b te ispisivati njihov zbroj, razliku, umnožak i količnik (drugi je broj uvijek različit od nule). Ispis programa treba biti „punog“ oblika, primjerice, za unos brojeva 5 i 7 te operaciju + ispis treba biti 5+7=12.

```
>>> a=int(input("broj a iznosi="))
broj a iznosi= 5
>>> b=int(input("broj b iznosi="))
broj b iznosi= 7
>>> print(a,"+", b, "=", a+b)
5 + 7 = 12
>>> print(a,"-", b, "=", a-b)
5 - 7 = -2
>>> print(a,"/", b, "=", a/b)
5 / 7 = 0.7142857142857143
>>> print(a,"*", b, "=", a*b)
5 * 7 = 35
>>>
```

5. Napiši program koji će unositi iznos odobreno g potrošačkog kredita c, godišnju kamatnu stopu p, broj mjeseci m, a ispisivati kamate prema formuli: k=cp(m+a)/2400
primjer:
unos:1000, 8, 12
ispis: 43.33

```
>>> c=int(input("iznos odobrenog potrošačkog kredita iznosi="))
iznos odobrenog potrošačkog kredita iznosi= 1000
>>> p=int(input("godišnja kamatna stopa iznosi="))
godišnja kamatna stopa iznosi= 8
>>> m=int(input("broj mjeseci iznosi="))
broj mjeseci iznosi=12
>>> print("kamate iznose=", c*p*(m+1)/2400)
kamate iznose= 43.333333333333336
>>>
```

6. Ivica ima n kuna za koje želi kupiti prijateljima čokolade. Jedna čokolada stoji m kuna. Ivicu zanima koliko će najviše čokolada moći kupiti te koliko će mu novca nakon toga preostati. Pomogni Ivici i napiši program koji će unositi iznos novca kojim Ivica raspolaže te cijenu jedne čokolade, a ispisivati koliko maksimalno čokolada Ivica može kupiti te koliko će mu novca nakon toga ostati.

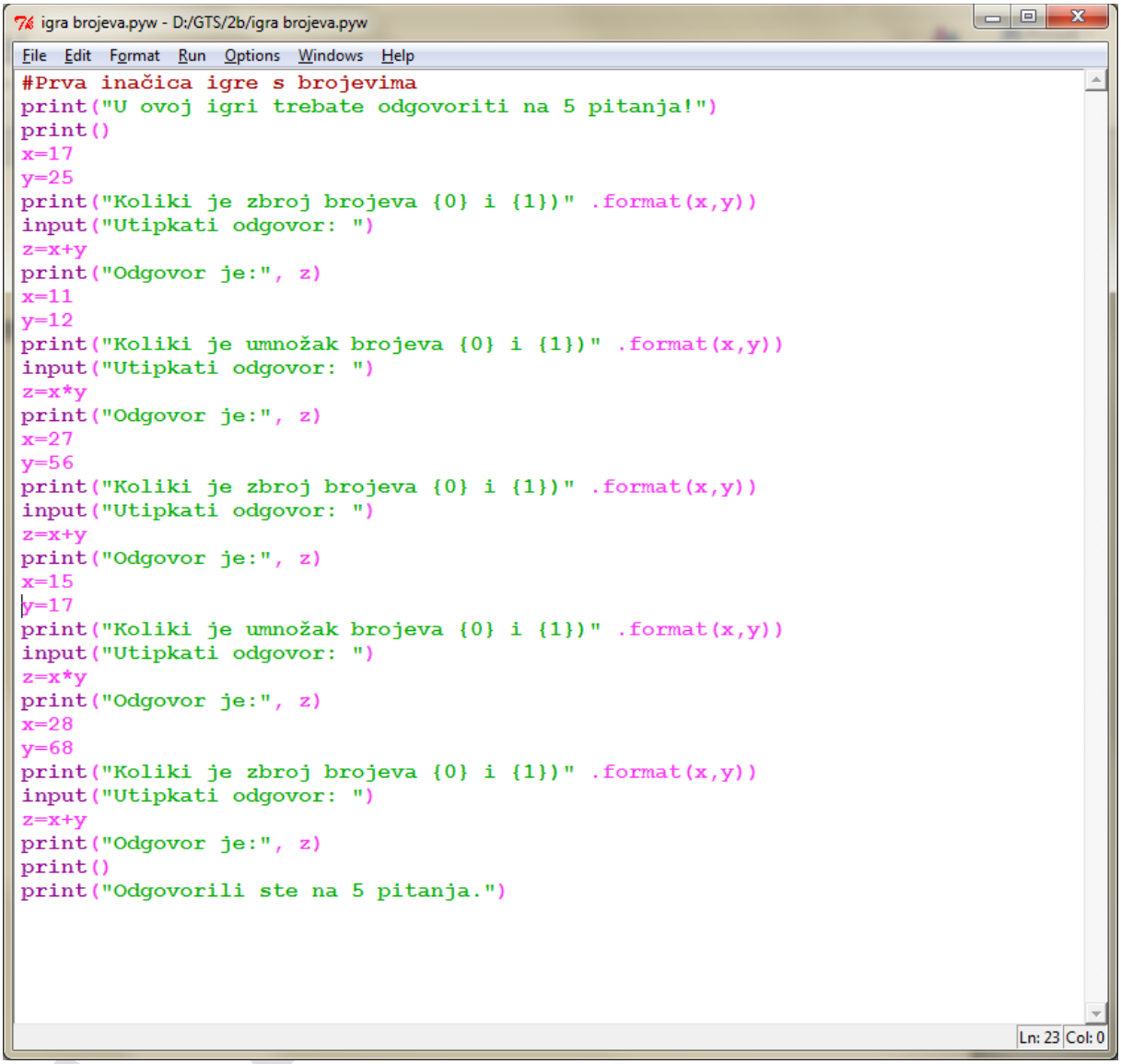
```
>>> n=int(input("ivica ima="))
ivica ima=12
>>> m=int(input("jedna čokolada košta="))
jedna čokolada košta=8
>>> cokolada=n//m
>>> print("ivica može kupiti",cokolada,
"cokolada")
ivica može kupiti 1 cokolada
>>> ostatak=n%m
>>> print("ivici ce ostati", ostatak, "kuna")
ivici ce ostati 4 kuna
>>>
```

7. Napiši program koji će unositi dva prirodna broja n i m. ispisivati koji je veći, a koji manji.

```
>>> n=int(input("broj n je="))
broj n je= 10
>>> m=int(input("broj m je="))
broj m je= 12
>>> if n>m:
        print(n, "je veći od", m)
else:
        print(n, "je manji od", m)
10 je manji od 12
>>>
```

JEDNOSTAVNI PROGRAMI

Zamišljena igra brojevima (1):



Ovakav program ima niz nedostataka:

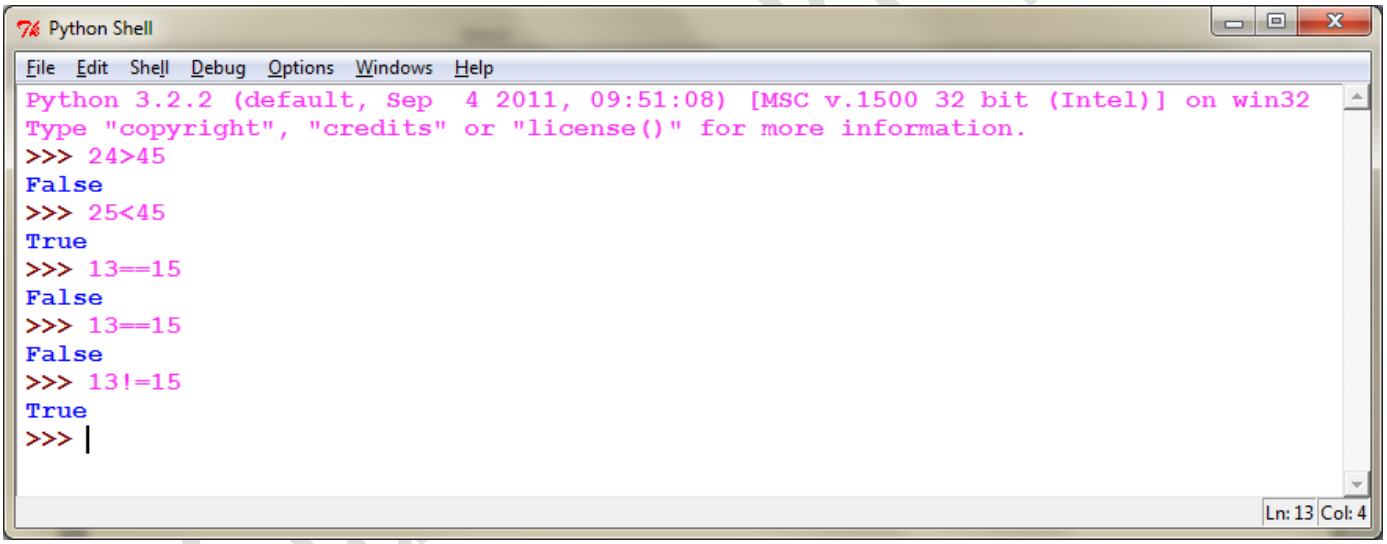
- Nemamo mogućnost uspoređivanja točnog rezultata
- Iste naredbe moramo prepisivati
- Nemamo mogućnost upita igrača želi li igrati dalje
- Igru možemo pokrenuti ponovno pokretanjem programa, ali će se ponoviti zadaci

DONOŠENJE ODLUKA I GRANANJA U PROGRAMIMA

Relacijski operatori (operatori uspoređivanja)

Relacijski operatori uspoređuju dva operanda. Rezultat usporedbe ima vrijednosti True ili False. Prema tome, izraz usporedbe koji se sastoji od dva operanda i relacijskog operatora je logički sud koji može biti istinit ili lažan.

Operator	Značenje simbola
>	Veće od
<	Manje od
>=	Veće od ili jednako
<=	Manje od ili jednako
==	Jednako
!=	Nije jednako



#1. Želimo li znati je li neki broj x veći od ili jednak nekoj donjoj granici x_d i istovremeno manji ili jednak nekoj gornjoj granici x_g, primjerice za zadane vrijednosti:

x_d=0, x_g=1000, x=700 ili za x=1100.

```
>>> x_d=0
>>> x_g=1000
>>> x=700
>>> x>=x_d
True
>>> x<=x_g
```

```
True
>>> x=1100
>>> x>=x_d
True
>>> x<=x_g
False
```

LOGIČKI OPERATORI I LOGIČKI IZRAZI

Za stvaranje bilo kojeg složenog suda dovoljna su nam 3 operatora.

Operator	Naziv operacije	Algebarski simboli
and	I operacija, konjugacija	^
or	ILI operacija, disjunkcija	v
not	NE operacija, komplementiranje	¬

```
>>> False and False
False
>>> False and True
False
>>> True and False
False
>>> True and True
True
```

Prema tome, I operacija daje rezultat True samo onda kada oba operatora imaju vrijednost True.

Umjesto vrijednosti False i True mogu se upotrebljavati vrijednosti 0 i 1.

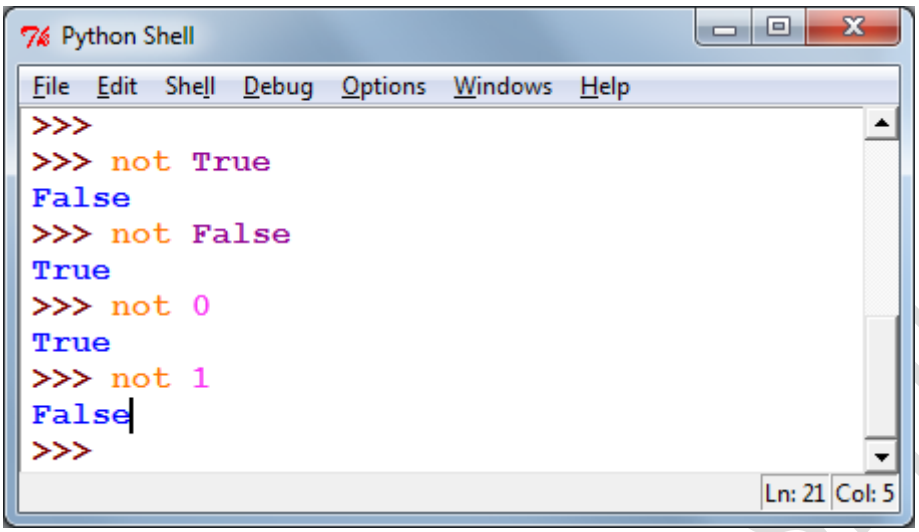
```
>>> 0 and 0
0
>>> 0 and 1
0
>>> 1 and 0
0
>>> 1 and 1
1
>>>
```


DJELOVANJE OPERATORA OR:

>>> False or False
False
>>> False or True
True
>>> True or False
True
	>>> 0 or 0
	0
	>>> 0 or 1
	1
	>>> 1 or 0
	1

>>> True or True True	>>> 1 or 1 1 >>>

Operator negacije NOT djeluje na jedan operand i invertira njegovu vrijednost.



Nedosljednost u jeziku Python.

```
>>> not 0

True

>>> not 1

False

>>>
```

Prioritet izvođenja logičkih operacija

Redoslijed	Operacija
1.	not
2.	and
3.	or

```
>>> True or False and True
True
>>> False or True and False
False
>>> False or not False
True
>>>
```

Redoslijed	Operacija
1.	aritmetički
2.	relacijski
3.	logički

```
>>> 2+3<4

False

>>> 150+150<290

False

>>> 2+3-1>3*3-5

False

>>> 2*3-1>3*3-5

True

>>> 3+4>5 and 3+5>4 and 4+5>3

True

>>> 3+4>5 and 3+5>4 and 4+5>13

False

>>>
```

Ispitivanje je li broj x veći od ili jednak donjoj granici x_d i istovemeno manji od ili jednak gornjoj granici x_g može se provesti s ovim složenim uvjetom:

(x_d<=x) and (x=> x_g)

Uvjet će biti istinit ako se vrijednost varijable x nalazi u zadanim granicama.

x_d <= x and x <= x_g

isti se uvjet može napisati na sljedeći način:

x_d <= x <= x_g

```
>>> xd=0
>>> xg=1000
>>> x=750
>>> (x>=xd) and (x<=xg)
True
>>> x>=xd and x<=xg
True
>>> x=1500
>>> x>=xd and x<=xg
False
>>> x=-5
>>> x>=xd and x<=xg
False
>>>
```

Donošenje odluka u programima

Pseudojezik (nije riječ o stvarnom programskom jeziku)

…

```
Ako je uvjet onda
{
      Naredba 1_1;
      …
      Naredba 1_n;
}
inače
{
     Naredba 2_1
     …
     Naredba 2_m
}
…
```

U **Pythonu** se odabir alternativnih blokova naredbi obavlja naredbama u kojima se rabe sljedeće ključne riječi: if, else i elif iza kojih se stavlja dvotočka.

Blokovi naredbi ne ograđuju se posebnim simbolima.

Iza zadnje naredbe bloka mora pisati naredba koja nije uvučena. Blok je na taj način jasno grafički određen u tekstu programa.

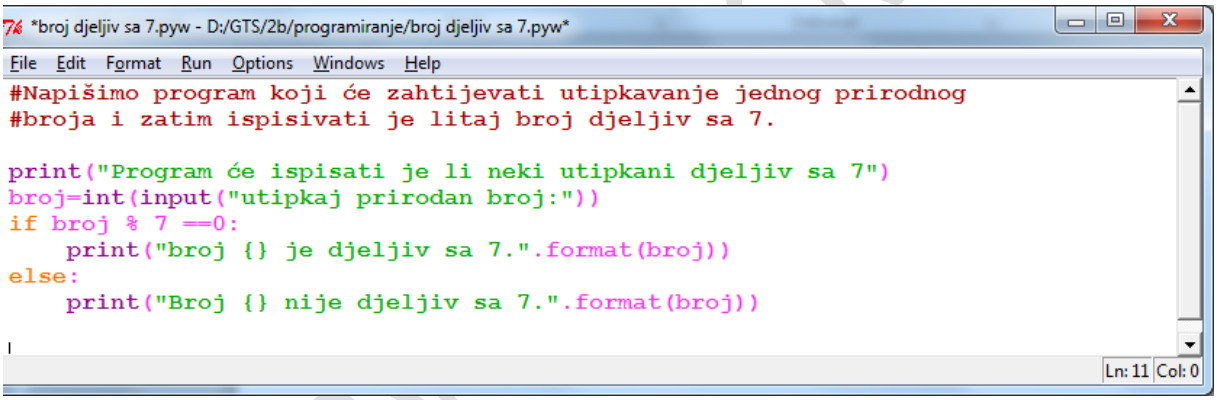
```
if uvjet:
        naredba 1_1
        …
        naredba 1_n
else:
        naredba 2_1
        …
        naredba 2_m
```

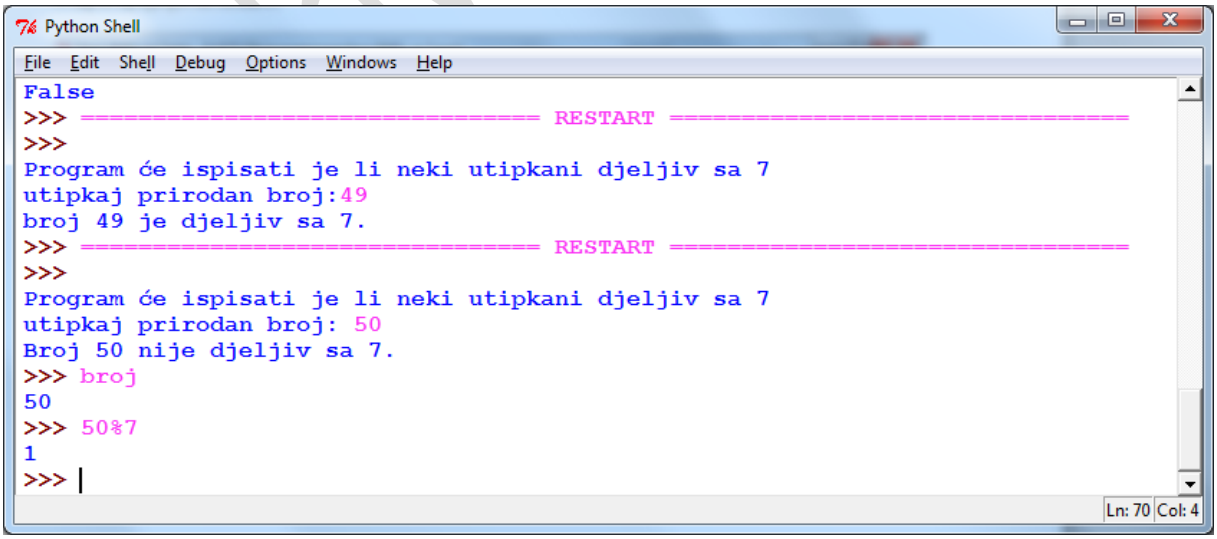
Niz naredbi koje smo označili sa: naredba 1_1, naredba 1_2,…, naredba 1_n zvat čemo blok naredbi i on će se izvesti ako je uvjet istinit. Primijetimo da je isto tako nizz naredbi naredba 2_1, naredba 2_2,…, naredba 2_m jedan blok naredbi koji se izvodi ako je uvjet lažan.

Na osnovi navedenog slijedi opći oblik naredbe if koji zapisujemo na slijedeći način:

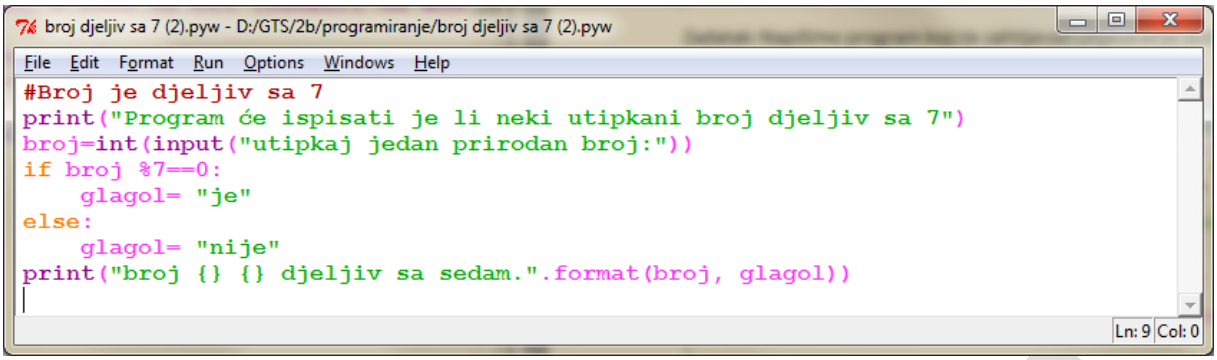
```
…
if uvjet:
        blok_naredbi_1
else:
       blok_naredbi_2
…
```

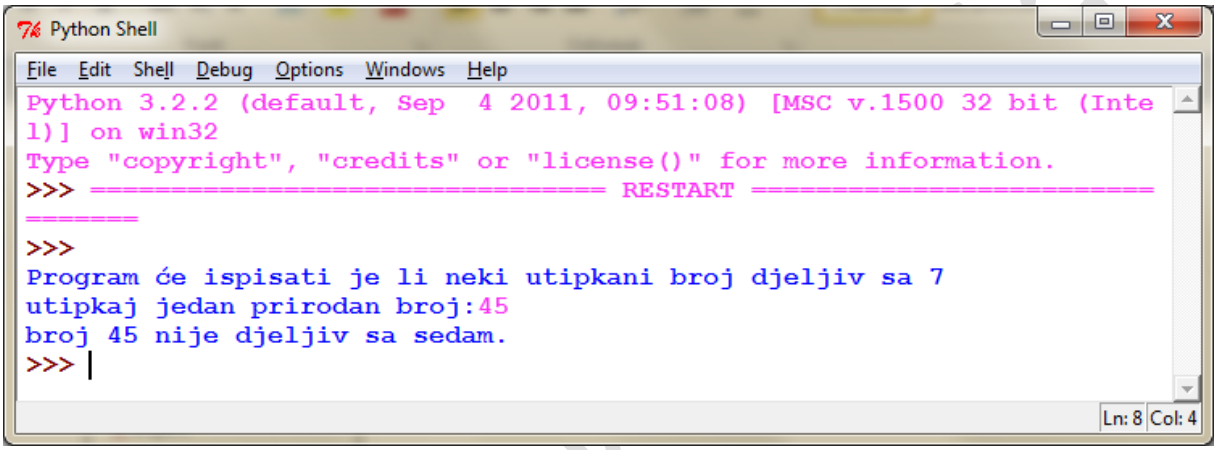
Zadatak: Napišimo program koji će zahtijevati utipkavanje jednog prirodnog broja i zatim ispisivati je li taj broj djeljiv sa sedam.





Zadatak:





Uporabe ključne riječi elif:

-elif za izgradnju programskih struktura višestrukih izbora

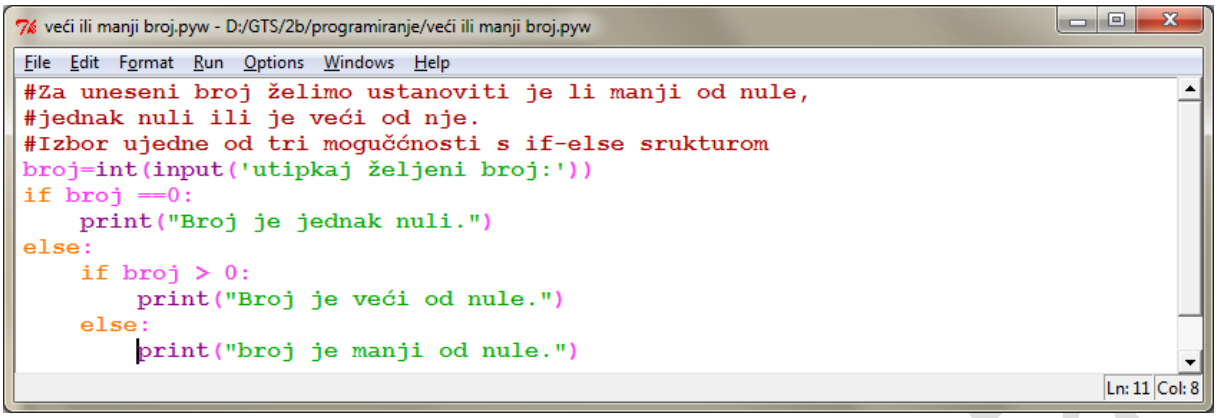
Način izgradnje takve strukture može se opisati na sljedeći način:

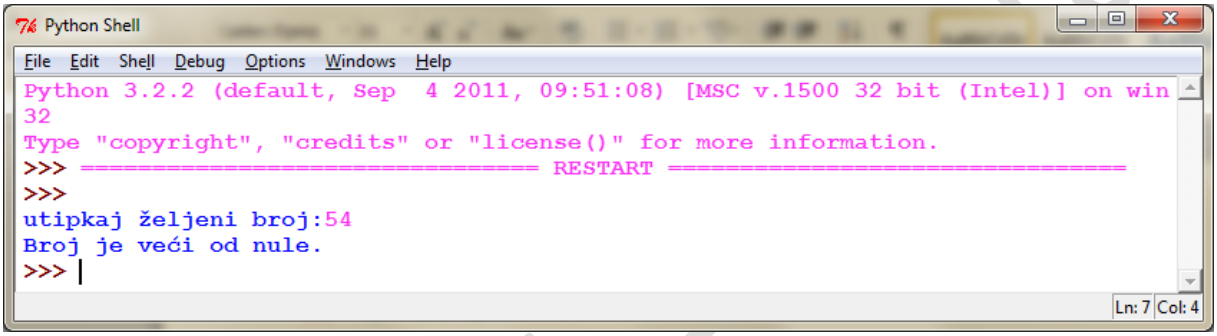
…

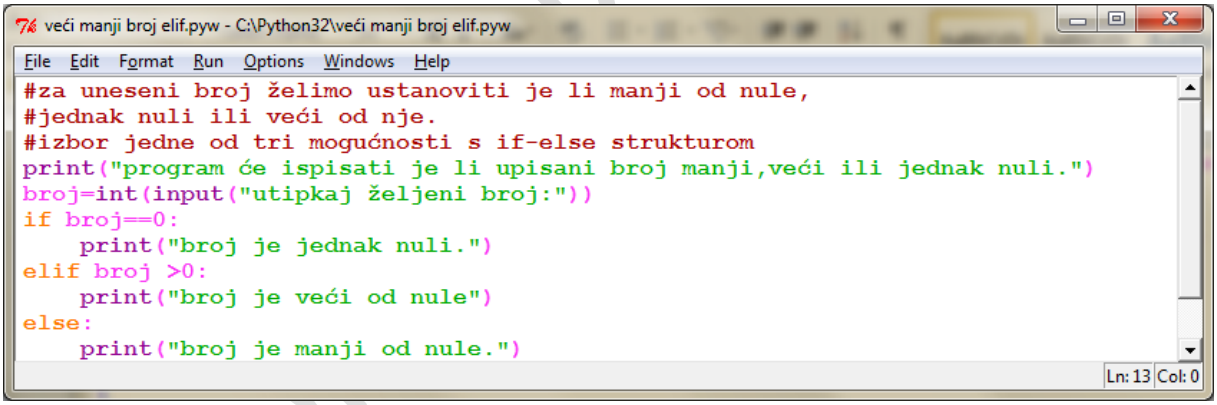
```
If logički izraz 0:
      blok naredbi 0
elif logički izraz 1
      blok naredbi 1
elif logički izraz 2
      blok naredbi 2
…
elif logički izraz n:
     blok naredbi n
else:
     blok naredbi
```

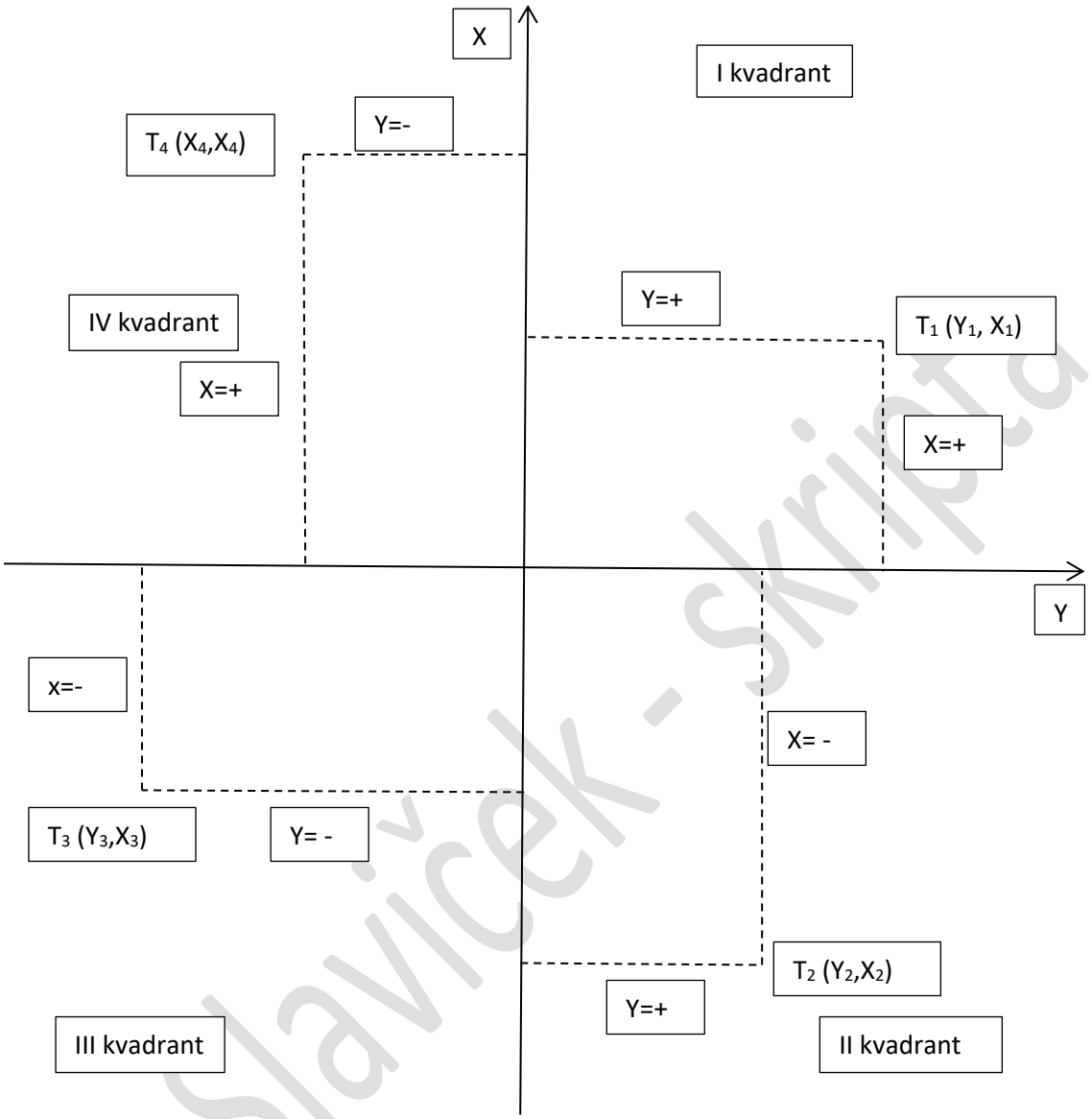
…

Zadatak:


veći ili manji broj.pyw - D:/GTS/2b/programiranje/veći ili manji broj.pyw
File Edit Format Run Options Windows Help
#Za uneseni broj želimo ustanoviti je li manji od nule,
#jednak nuli ili je veći od nje.
#Izbor ujedne od tri mogučćnosti s if-else srukturom
broj=int(input('utipkaj željeni broj:'))
if broj ==0:
print("Broj je jednak nuli.")
else:
if broj > 0:
print("Broj je veći od nule.")
else:
print("broj je manji od nule.")
Ln: 11 Col: 8


Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
utipkaj željeni broj:54
Broj je veći od nule.
>>>
Ln: 7 Col: 4


veći manji broj elif.pyw - C:\Python32\veći manji broj elif.pyw
File Edit Format Run Options Windows Help
#za uneseni broj želimo ustanoviti je li manji od nule,
#jednak nuli ili veći od nje.
#izbor jedne od tri mogućnosti s if-else strukturom
print("program će ispisati je li upisani broj manji,veći ili jednak nuli.")
broj=int(input("utipkaj željeni broj:"))
if broj==0:
print("broj je jednak nuli.")
elif broj >0:
print("broj je veći od nule")
else:
print("broj je manji od nule.")
Ln: 13 Col: 0



Ako je y>0 and x>0
print 1. Kvadrant

Ako je y>0 and x<0
print 2. Kvadrant

Ako je y<0 and x<0
print 3. Kvadrant

Ako je y<0 and x>0
print 4. Kvadrant

kvadranti.pyw - D:\GTS\2b\programiranje\kvadranti.pyw

File Edit Format Run Options Windows Help

```
y=int(input("y je:"))
x=int(input("x je:"))
if y>0 and x>0:
    print("točka je u 1. kvadrantu")
elif y>0 and x<0:
    print("točka je u 2. kvadrantu")
elif y<0 and x<0:
    print("točka je u 3. kvadrantu")
elif y<0 and x>0:
    print("točka se nalazi u 4. kvadrantu")
```

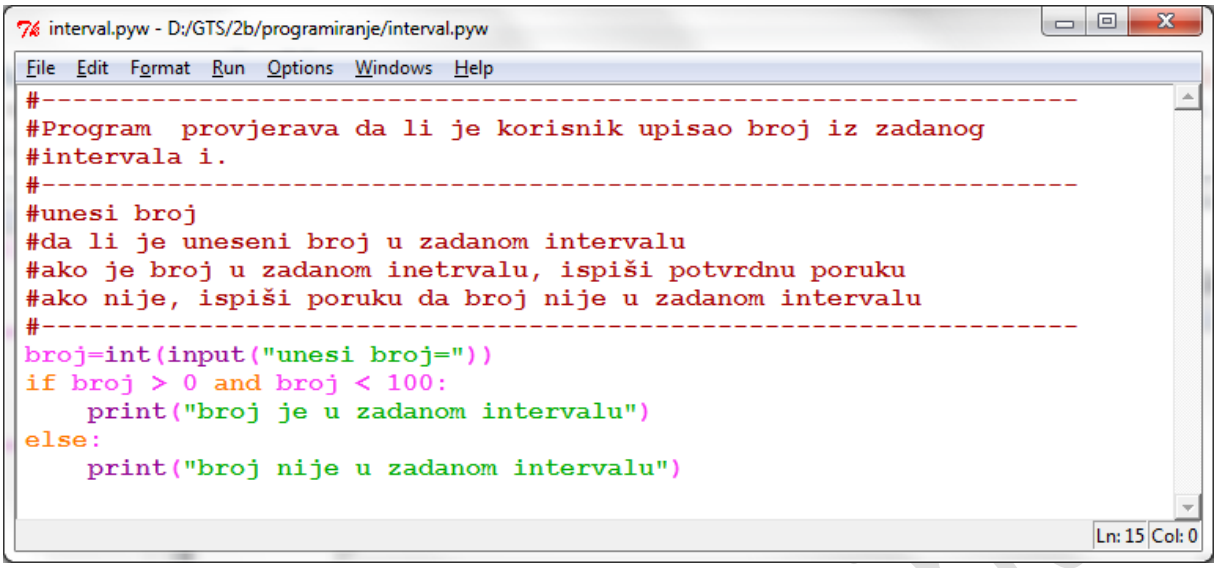
Ln: 11 Col: 0

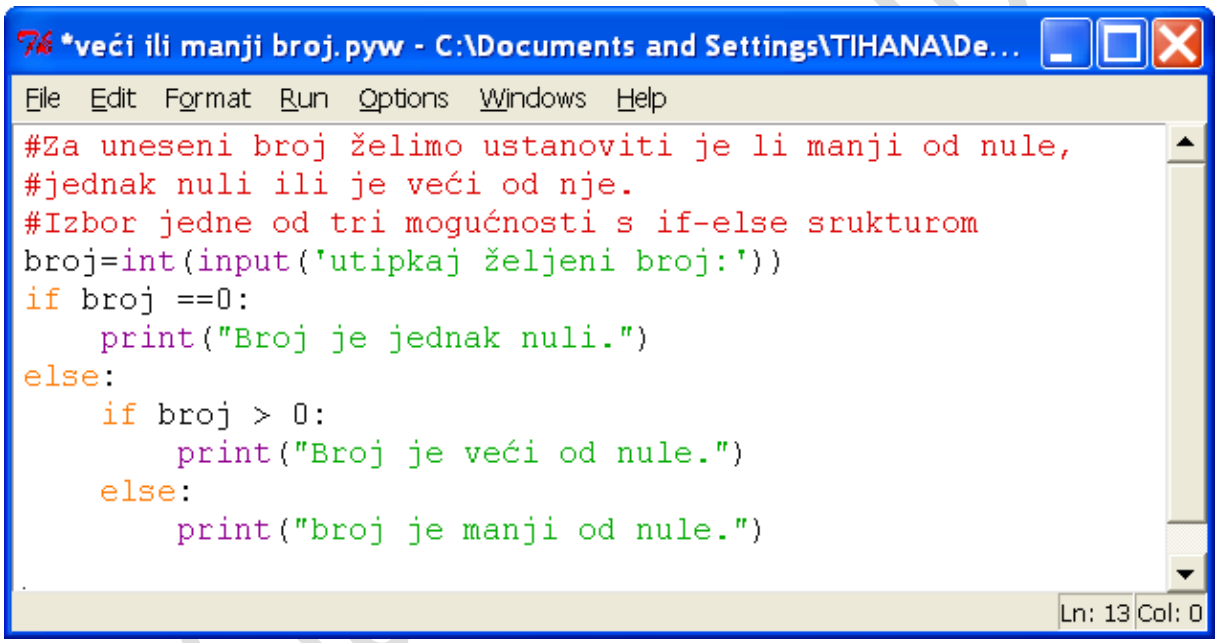
korijen.pyw - D:/GTS/2b/programiranje/korijen.pyw

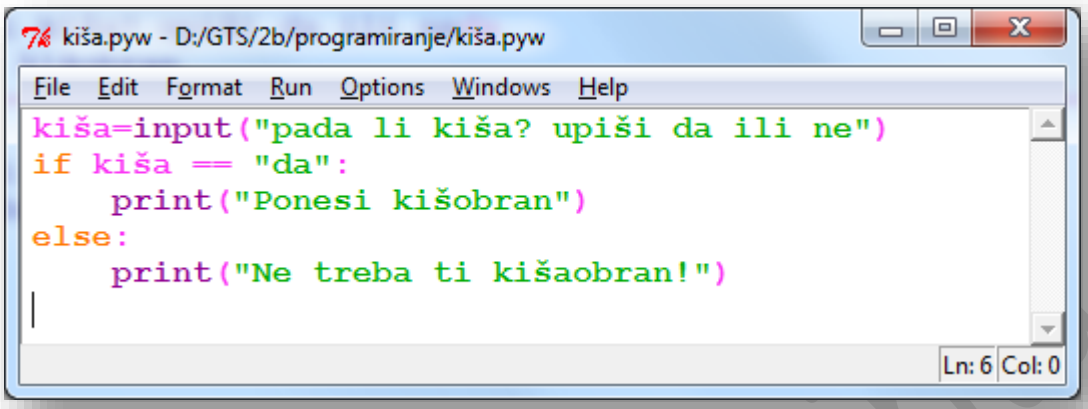
File Edit Format Run Options Windows Help

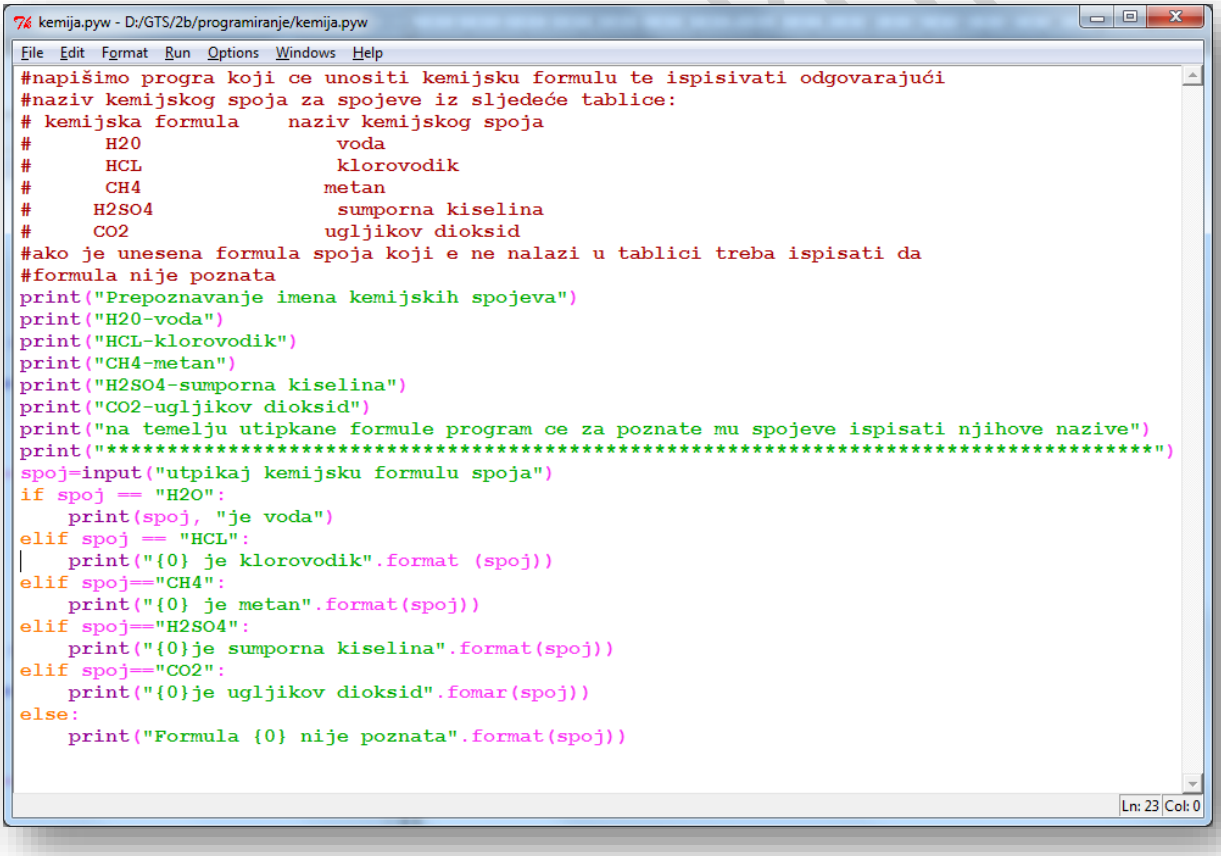
```
#program upisuje jedan pozitivan broj.
#Ako je dobro uisan ispisat će njegov drugi korijen,a ako nije
#upozorit ce na grešku
#-------------------------------------------------------------------
#algoritam
#unesi broj
#ako je broj pozitivan izračunaj drugi korijen
#ako nije program će upozoriti na grešku
#*napomena: ako želimo izračunati drugi korijen potrebno je umenuti
#matematičke funkcije naredbom import mat
#program
#-------------------------------------------------------------------
import math
broj=int(input("unesi jedan broj="))
if broj>0:
    print("drugi korijen od",broj,"iznosi",math.sqrt(broj))
else:
    print("broj je negativan")
```

Ln: 12 Col: 67


interval.pyw - D:/GTS/2b/programiranje/interval.pyw
File Edit Format Run Options Windows Help
#--
#Program provjerava da li je korisnik upisao broj iz zadanog
#intervala i.
#--
#unesi broj
#da li je uneseni broj u zadanom intervalu
#ako je broj u zadanom inetrvalu, ispiši potvrdnu poruku
#ako nije, ispiši poruku da broj nije u zadanom intervalu
#--
broj=int(input("unesi broj="))
if broj > 0 and broj < 100:
    print("broj je u zadanom intervalu")
else:
    print("broj nije u zadanom intervalu")
Ln: 15 Col: 0


*veći ili manji broj.pyw - C:\Documents and Settings\TIHANA\De...
File Edit Format Run Options Windows Help
#Za uneseni broj želimo ustanoviti je li manji od nule,
#jednak nuli ili je veći od nje.
#Izbor jedne od tri mogućnosti s if-else srukturom
broj=int(input('utipkaj željeni broj:'))
if broj ==0:
    print("Broj je jednak nuli.")
else:
    if broj > 0:
        print("Broj je veći od nule.")
    else:
        print("broj je manji od nule.")
Ln: 13 Col: 0


kiša.pyw - D:/GTS/2b/programiranje/kiša.pyw
File Edit Format Run Options Windows Help
kiša=input("pada li kiša? upiši da ili ne")
if kiša == "da":
print("Ponesi kišobran")
else:
print("Ne treba ti kišaobran!")
Ln: 6 Col: 0


kemija.pyw - D:/GTS/2b/programiranje/kemija.pyw
File Edit Format Run Options Windows Help
#napišimo progra koji ce unositi kemijsku formulu te ispisivati odgovarajući
#naziv kemijskog spoja za spojeve iz sljedeće tablice:
kemijska formula naziv kemijskog spoja
H20 voda
HCL klorovodik
CH4 metan
H2SO4 sumporna kiselina
CO2 ugljikov dioksid
#ako je unesena formula spoja koji e ne nalazi u tablici treba ispisati da
#formula nije poznata
print("Prepoznavanje imena kemijskih spojeva")
print("H20-voda")
print("HCL-klorovodik")
print("CH4-metan")
print("H2SO4-sumporna kiselina")
print("CO2-ugljikov dioksid")
print("na temelju utipkane formule program ce za poznate mu spojeve ispisati njihove nazive")
print("**********")
spoj=input("utpikaj kemijsku formulu spoja")
if spoj == "H2O":
print(spoj, "je voda")
elif spoj == "HCL":
print("{0} je klorovodik".format (spoj))
elif spoj=="CH4":
print("{0} je metan".format(spoj))
elif spoj=="H2SO4":
print("{0}je sumporna kiselina".format(spoj))
elif spoj=="CO2":
print("{0}je ugljikov dioksid".fomar(spoj))
else:
print("Formula {0} nije poznata".format(spoj))
Ln: 23 Col: 0

predmeti.pyw - C:\Documents and Settings\TIHANA\Desktop\Programiranje\predmeti.pyw

File Edit Format Run Options Windows Help

```
print("Napiši program koji će ispisivati kad uneseš predmet tko ga predaje.")
print("matematika- prof. Portner")
print("povijest -  prof. Bašić")
print("tjelesni - prof. Kantar")
print("fizika - prof. Šarić")
print("hrvatski - prof.Varović")
print("geodezija - prof. Jurković-Koren")
print("Kartografija - prof. Jurković- Koren")
print("fotogrametrija - prof. Tomasović")
print("programiranje - prof. Slaviček")
print("etika - prof. Bašić")
print("geodetska izmjera - prof. Jurković-Koren")
print("engleski - prof. Plemenčić")
print("nacrtna geometrija - prof. Žibrat")
print("OGI - prof. Šurina")
print("zemljopis - prof. Lovrec")
predmet=input("Unesi predmet=")
if predmet=="matematika":
    print(predmet, "U GTŠ predaje prof. Portner")
elif predmet=="povijest":
    print("{} u GTŠ predaje prof. Bašić".format(predmet))
elif predmet=="tjelesni":
    print("{} u GTŠ predaje prof. Kantar".format(predmet))
elif predmet=="fizika":
    print("{} u GTŠ predaje prof. Šarić".format(predmet))
elif predmet=="hrvatski":
    print("{} u GTŠ predaje prof. Varović".format(predmet))
elif predmet=="geodezija":
    print("{} u GTŠ predaje prof. Jurković- Koren".format(predmet))
elif predmet=="geodetska izmjera":
    print("{} u GTŠ predaje prof. Jurković- Koren".format(predmet))
```

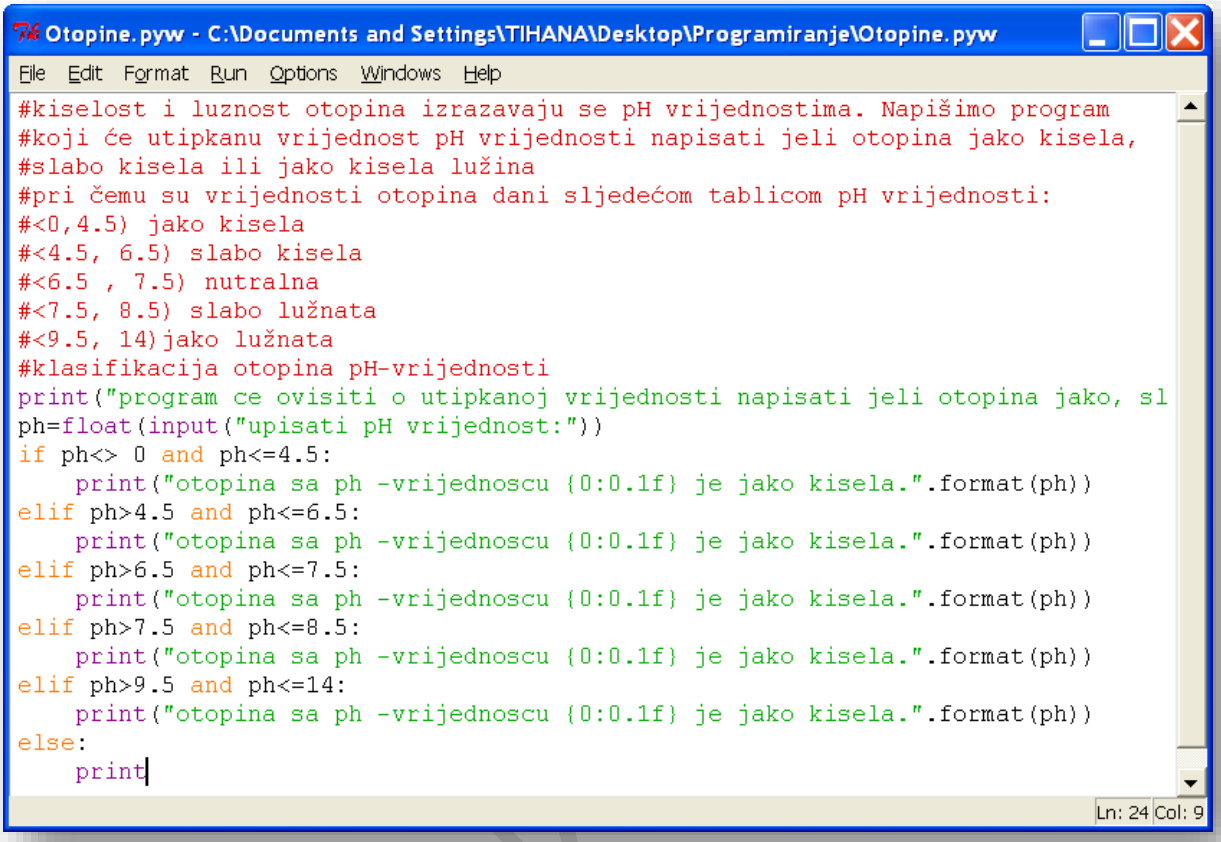
Ln: 31 Col: 0

predmeti.pyw - C:\Documents and Settings\TIHANA\Desktop\Programiranje\predmeti.pyw

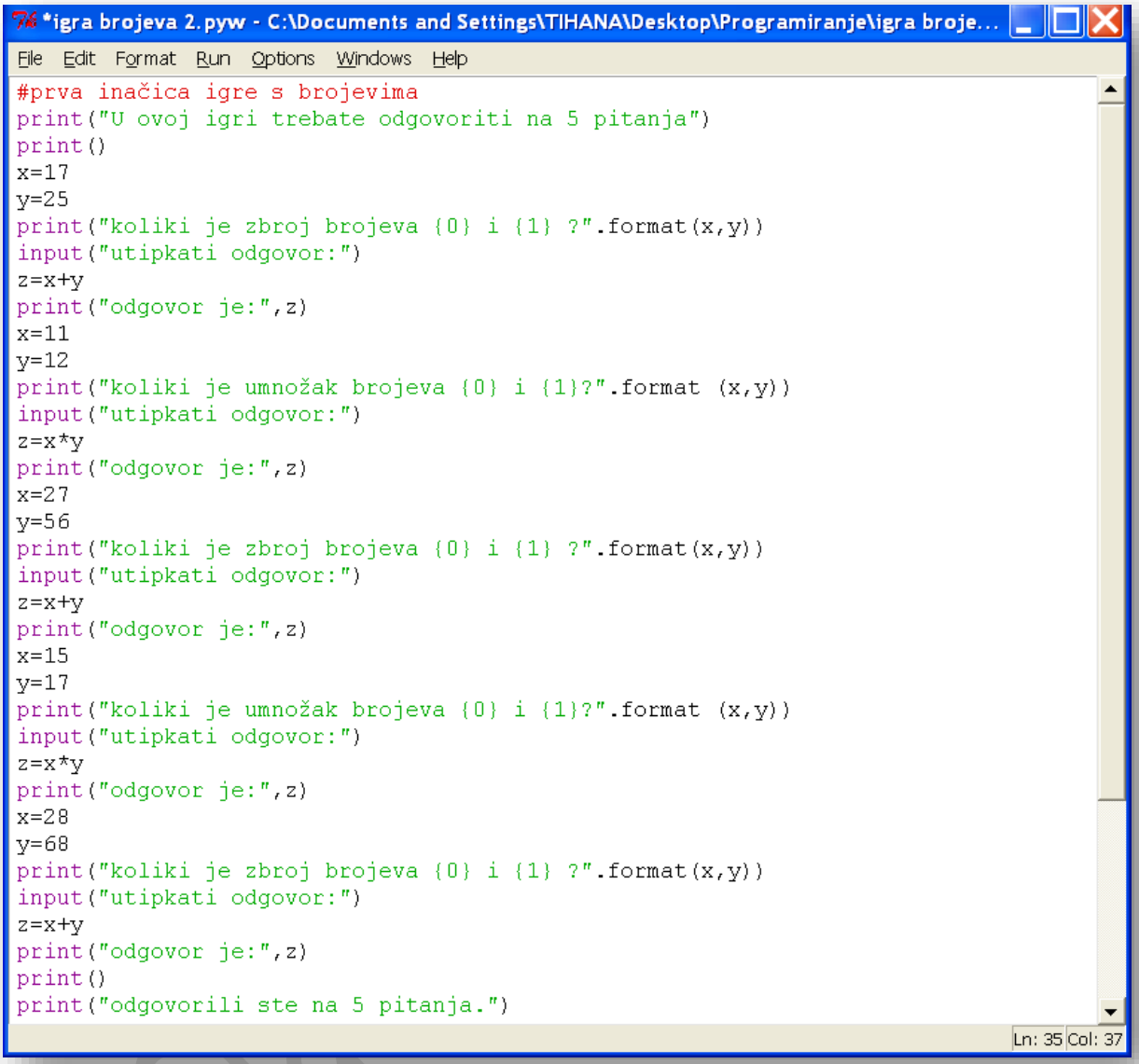
File Edit Format Run Options Windows Help

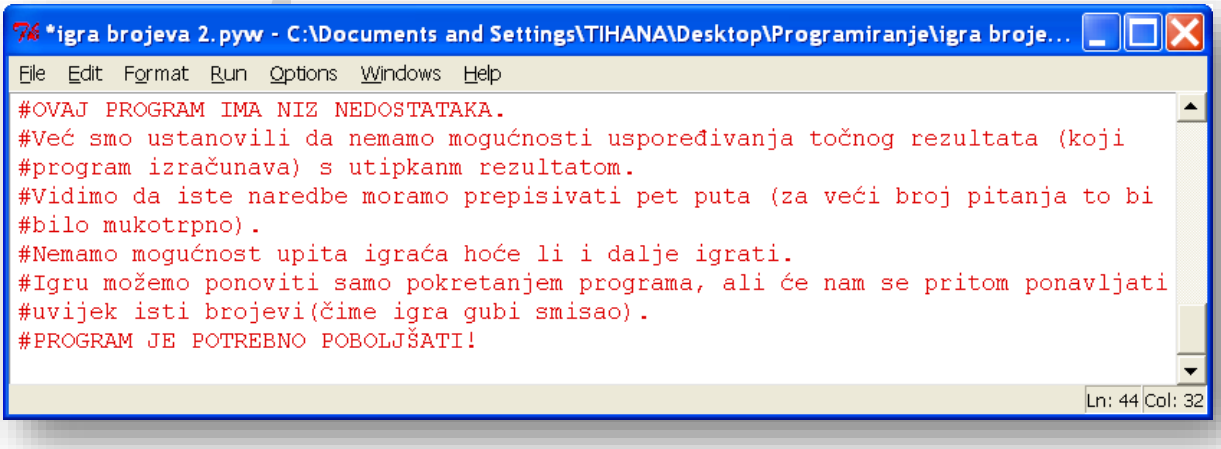
```
elif predmet=="kartografija":
    print("{} u GTŠ predaje prof. Jurković- Koren".format(predmet))
elif predmet=="fotogrametrija":
    print("{} u GTŠ predaje prof. Tomasović".format(predmet))
elif predmet=="programiranje":
    print("{} u GTŠ predaje prof. Slaviček".format(predmet))
elif predmet=="etika":
    print("{} u GTŠ predaje prof. Bašić".format(predmet))
elif predmet=="engleski":
    print("{} u GTŠ predaje prof. Plemenčić".format(predmet))
elif predmet=="nacrtna geometrija":
    print("{} u GTŠ predaje prof. Žibrat".format(predmet))
elif predmet=="OGI":
    print("{} u GTŠ predaje prof. Šurina".format(predmet))
elif predmet=="zemljopis":
    print("{} u GTŠ predaje prof. Lovrec".format(predmet))
else:
    print("tog predmeta nema u školi.")
```

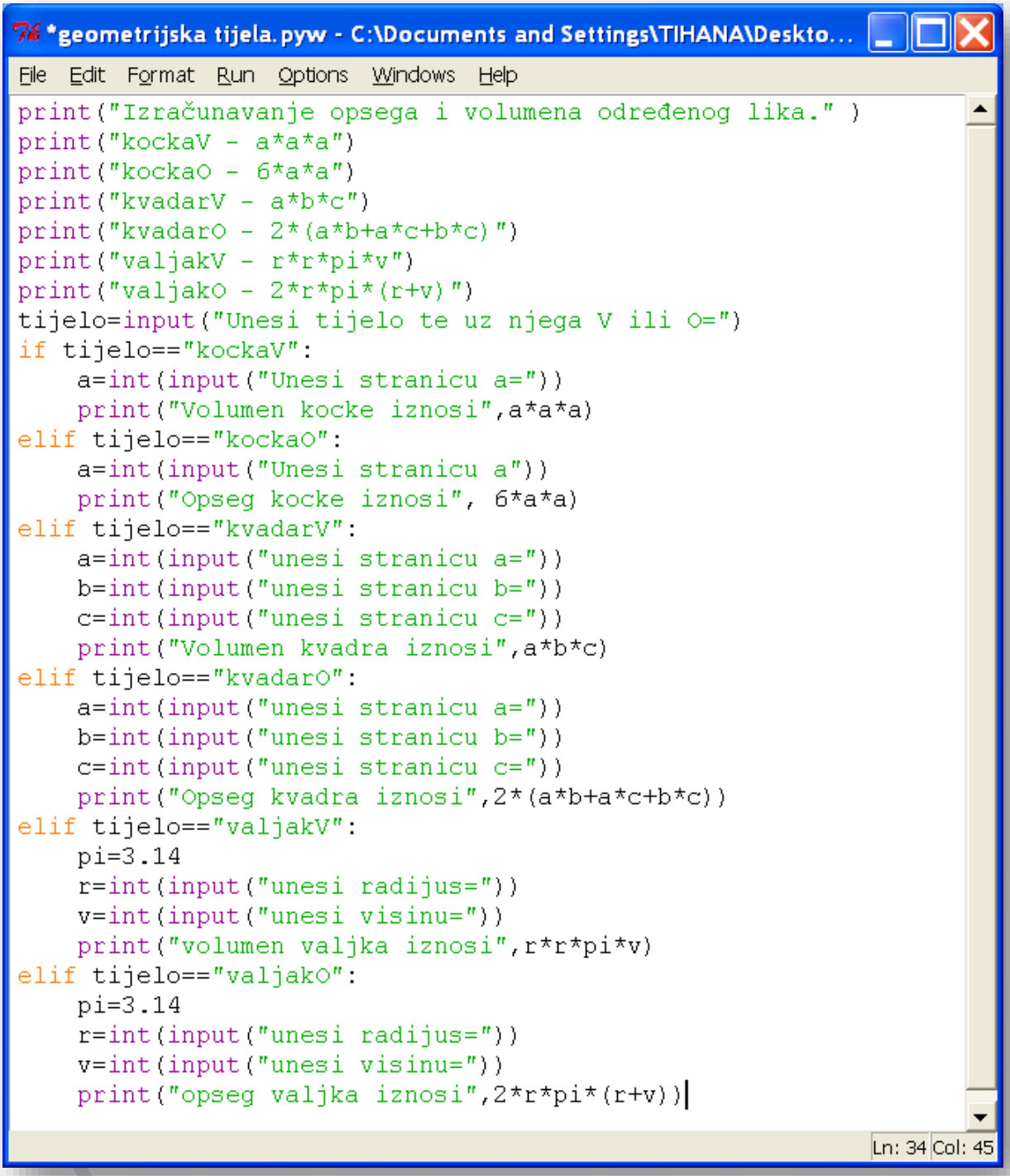
Ln: 49 Col: 0


Otopine.pyw - C:\Documents and Settings\TIHANA\Desktop\Programiranje\Otopine.pyw
File Edit Format Run Options Windows Help
#kiselost i luznost otopina izrazavaju se pH vrijednostima. Napišimo program
#koji će utipkanu vrijednost pH vrijednosti napisati jeli otopina jako kisela,
#slabo kisela ili jako kisela lužina
#pri čemu su vrijednosti otopina dani sljedećom tablicom pH vrijednosti:
#<0,4.5) jako kisela
#<4.5, 6.5) slabo kisela
#<6.5 , 7.5) nutralna
#<7.5, 8.5) slabo lužnata
#<9.5, 14)jako lužnata
#klasifikacija otopina pH-vrijednosti
print("program ce ovisiti o utipkanoj vrijednosti napisati jeli otopina jako, sl
ph=float(input("upisati pH vrijednost:"))
if ph<> 0 and ph<=4.5:
print("otopina sa ph -vrijednoscu {0:0.1f} je jako kisela.".format(ph))
elif ph>4.5 and ph<=6.5:
print("otopina sa ph -vrijednoscu {0:0.1f} je jako kisela.".format(ph))
elif ph>6.5 and ph<=7.5:
print("otopina sa ph -vrijednoscu {0:0.1f} je jako kisela.".format(ph))
elif ph>7.5 and ph<=8.5:
print("otopina sa ph -vrijednoscu {0:0.1f} je jako kisela.".format(ph))
elif ph>9.5 and ph<=14:
print("otopina sa ph -vrijednoscu {0:0.1f} je jako kisela.".format(ph))
else:
print
Ln: 24 Col: 9

Zamišljena igra brojevima (2):





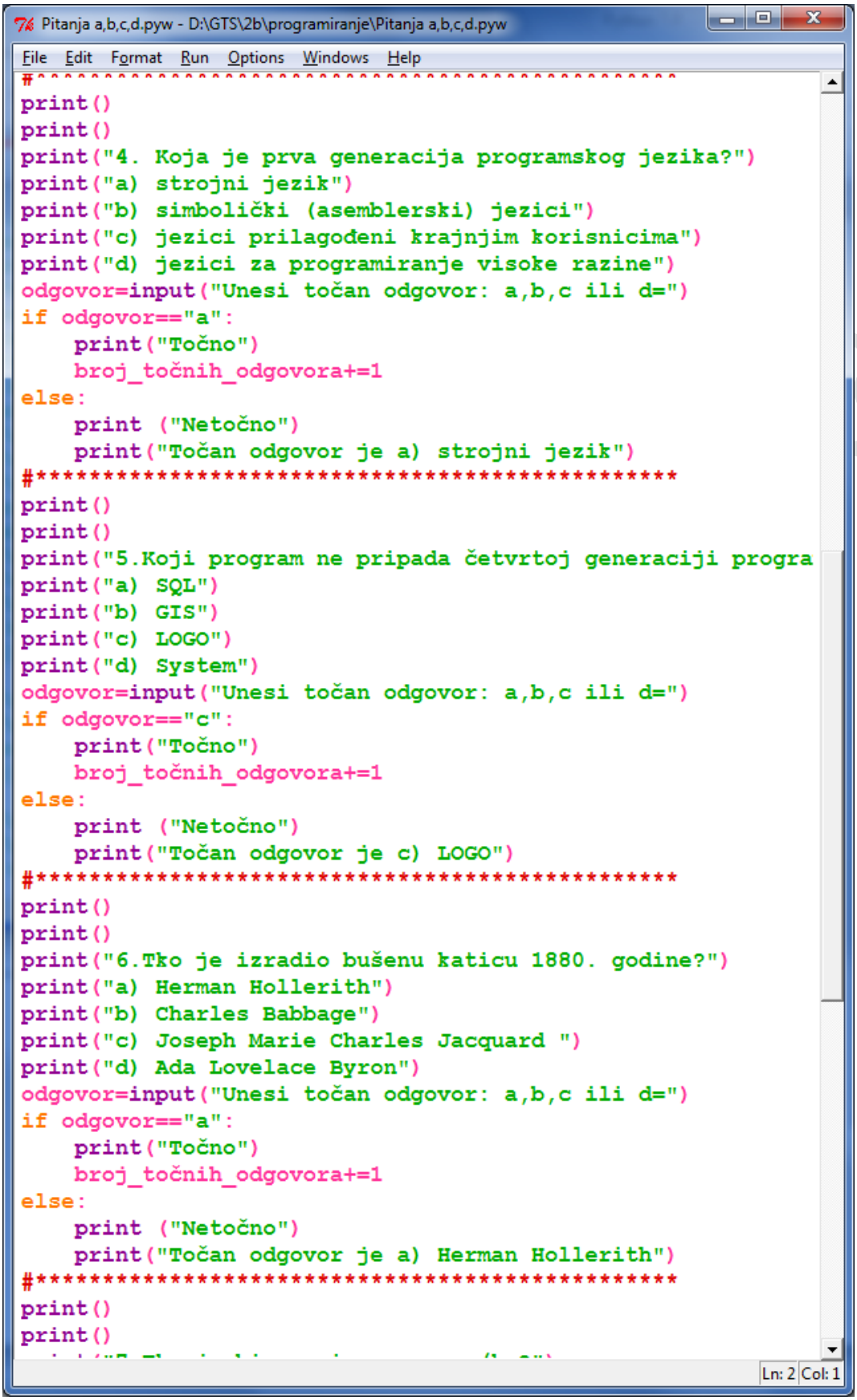

*geometrijska tijela.pyw - C:\Documents and Settings\TIHANA\Deskto...
File Edit Format Run Options Windows Help
print("Izračunavanje opsega i volumena određenog lika.")
print("kockaV - a*a*a")
print("kockaO - 6*a*a")
print("kvadarV - a*b*c")
print("kvadarO - 2*(a*b+a*c+b*c)")
print("valjakV - r*r*pi*v")
print("valjakO - 2*r*pi*(r+v)")
tijelo=input("Unesi tijelo te uz njega V ili O=")
if tijelo=="kockaV":
a=int(input("Unesi stranicu a="))
print("Volumen kocke iznosi",a*a*a)
elif tijelo=="kockaO":
a=int(input("Unesi stranicu a"))
print("Opseg kocke iznosi", 6*a*a)
elif tijelo=="kvadarV":
a=int(input("unesi stranicu a="))
b=int(input("unesi stranicu b="))
c=int(input("unesi stranicu c="))
print("Volumen kvadra iznosi",a*b*c)
elif tijelo=="kvadarO":
a=int(input("unesi stranicu a="))
b=int(input("unesi stranicu b="))
c=int(input("unesi stranicu c="))
print("Opseg kvadra iznosi",2*(a*b+a*c+b*c))
elif tijelo=="valjakV":
pi=3.14
r=int(input("unesi radijus="))
v=int(input("unesi visinu="))
print("volumen valjka iznosi",r*r*pi*v)
elif tijelo=="valjakO":
pi=3.14
r=int(input("unesi radijus="))
v=int(input("unesi visinu="))
print("opseg valjka iznosi",2*r*pi*(r+v))
Ln: 34 Col: 45

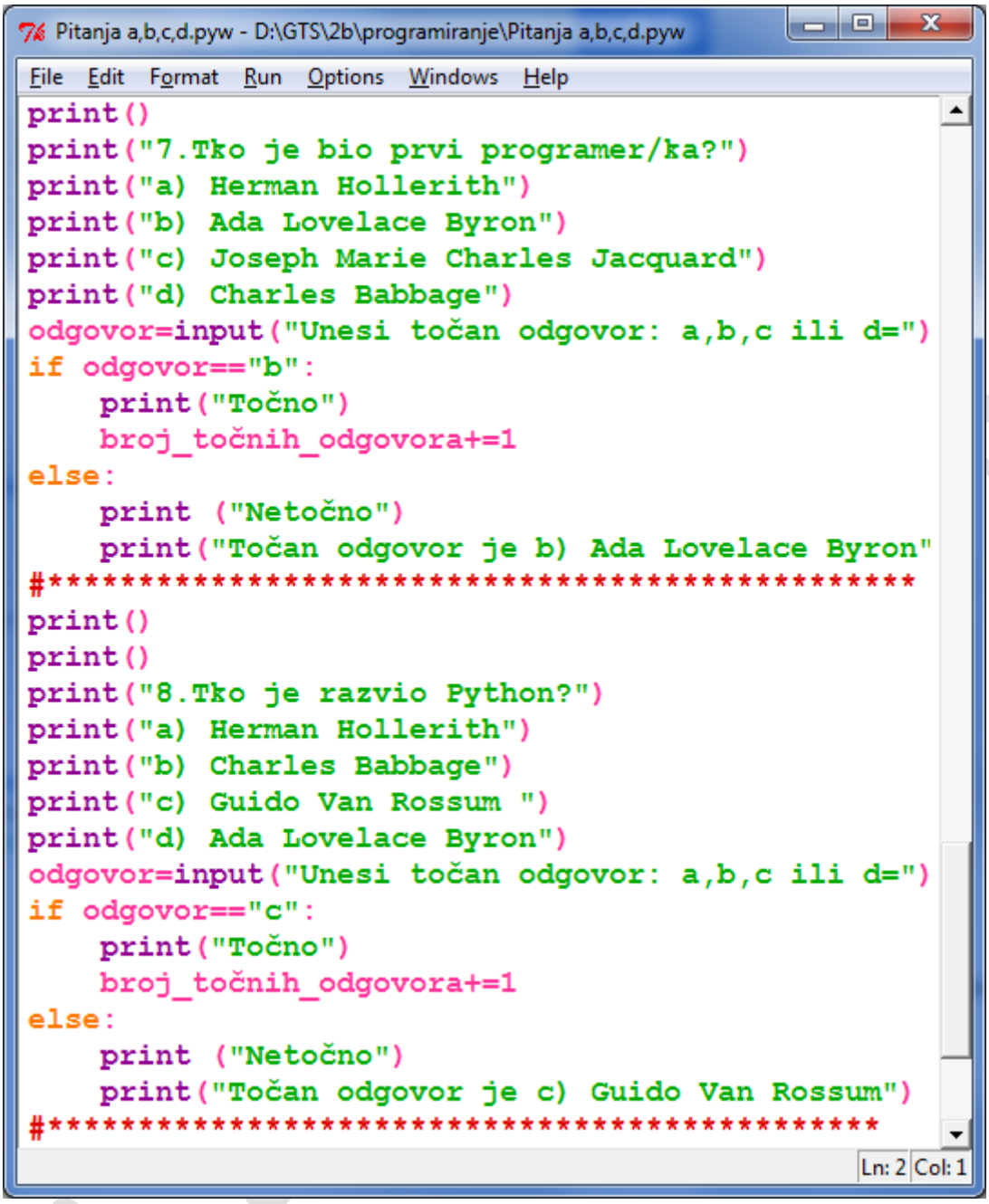
Pitanja a,b,c,d.pyw - D:\GTS\2b\programiranje\Pitanja a,b,c,d.pyw

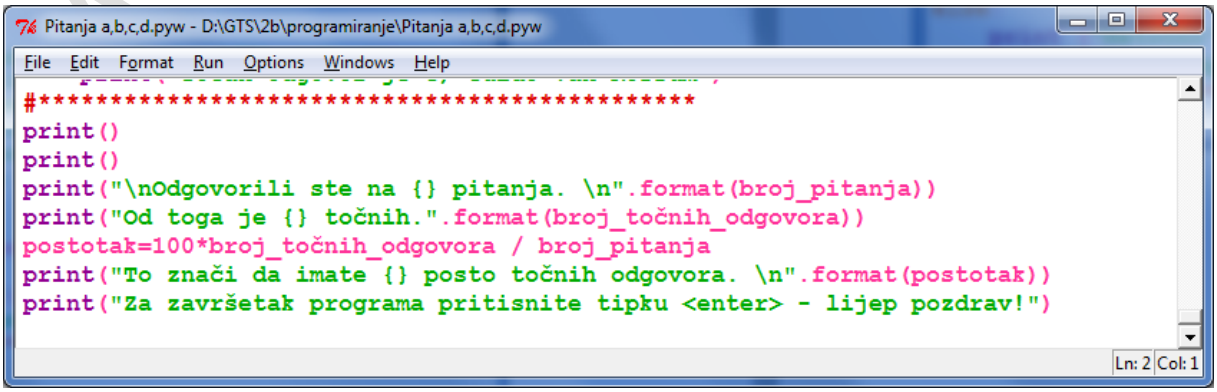
File Edit Format Run Options Windows Help

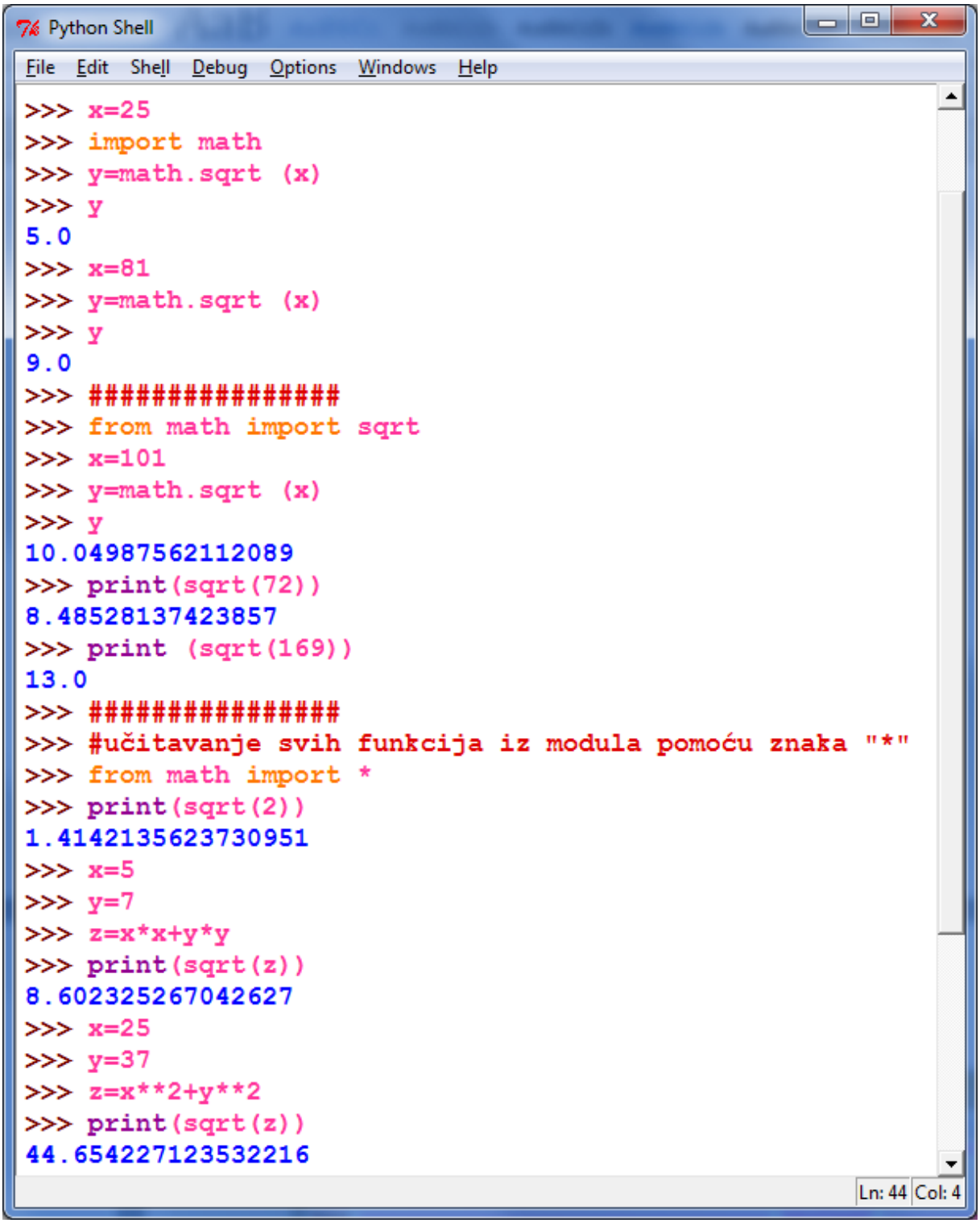
```
#pitanja a,b,c,d
#
#Broj_točnih_odgovora
broj_pitanja=8
print("Trebate odgovoriti na {0} pitanja \n".format(broj_pi
broj_točnih_odgovora=0
print("1.Koja je najpoznatija društvena mreža?")
print("a) MSN")
print("b) Facebook")
print("c) Twitter")
print("d) Account")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="b":
    print("Točno")
    broj_točnih_odgovora+=1
else:
    print ("Netočno")
    print("Točan odgovor je b) Facebook")
#***********************************************
print()
print()
print("2.Kada je započeo razvoj Pythona?")
print("a) 1991. god")
print("b) 2007. god")
print("c) 1987. god")
print("d) 1992. god")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="a":
    print("Točno")
    broj_točnih_odgovora+=1
else:
    print ("Netočno")
    print("Točan odgovor je a) 1991. god")
#***********************************************
print()
print()
print("3.Pravokutnik u dijagramu tijeka označava?")
print("a) početak")
print("b) ulaz podataka")
print("c) odluka")
print("d) naredba")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="d":
    print("Točno")
    broj_točnih_odgovora+=1
else:
    print ("Netočno")
    print("Točan odgovor je d) naredba")
```

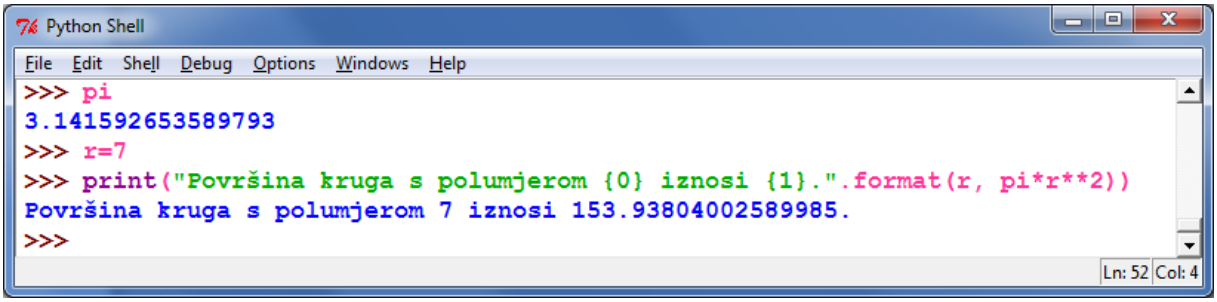
Ln: 2 Col: 1


Pitanja a,b,c,d.pyw - D:\GTS\2b\programiranje\Pitanja a,b,c,d.pyw
File Edit Format Run Options Windows Help
#**********
print()
print()
print("4. Koja je prva generacija programskog jezika?")
print("a) strojni jezik")
print("b) simbolički (asemblerski) jezici")
print("c) jezici prilagođeni krajnjim korisnicima")
print("d) jezici za programiranje visoke razine")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="a":
broj_točnih_odgovora+=1
else:
print ("Netočno")
print("Točan odgovor je a) strojni jezik")
#**********
print()
print()
print("5.Koji program ne pripada četvrtoj generaciji progra
print("a) SQL")
print("b) GIS")
print("c) LOGO")
print("d) System")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="c":
broj_točnih_odgovora+=1
else:
print ("Netočno")
print("Točan odgovor je c) LOGO")
#**********
print()
print()
print("6.Tko je izradio bušenu katicu 1880. godine?")
print("a) Herman Hollerith")
print("b) Charles Babbage")
print("c) Joseph Marie Charles Jacquard ")
print("d) Ada Lovelace Byron")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="a":
broj_točnih_odgovora+=1
else:
print ("Netočno")
print("Točan odgovor je a) Herman Hollerith")
#**********
print()
print()
Ln: 2 Col: 1


Pitanja a,b,c,d.pyw - D:\GTS\2b\programiranje\Pitanja a,b,c,d.pyw
File Edit Format Run Options Windows Help
print()
print("7.Tko je bio prvi programer/ka?")
print("a) Herman Hollerith")
print("b) Ada Lovelace Byron")
print("c) Joseph Marie Charles Jacquard")
print("d) Charles Babbage")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="b":
print("Točno")
broj_točnih_odgovora+=1
else:
print ("Netočno")
print("Točan odgovor je b) Ada Lovelace Byron"
#**********
print()
print()
print("8.Tko je razvio Python?")
print("a) Herman Hollerith")
print("b) Charles Babbage")
print("c) Guido Van Rossum ")
print("d) Ada Lovelace Byron")
odgovor=input("Unesi točan odgovor: a,b,c ili d=")
if odgovor=="c":
print("Točno")
broj_točnih_odgovora+=1
else:
print ("Netočno")
print("Točan odgovor je c) Guido Van Rossum")
#**********
Ln: 2 Col: 1


Pitanja a,b,c,d.pyw - D:\GTS\2b\programiranje\Pitanja a,b,c,d.pyw
File Edit Format Run Options Windows Help
#**********
print()
print()
print("\nOdgovorili ste na {} pitanja. \n".format(broj_pitanja))
print("Od toga je {} točnih.".format(broj_točnih_odgovora))
postotak=100*broj_točnih_odgovora / broj_pitanja
print("To znači da imate {} posto točnih odgovora. \n".format(postotak))
print("Za završetak programa pritisnite tipku <enter> - lijep pozdrav!")
Ln: 2 Col: 1


Python Shell
File Edit Shell Debug Options Windows Help
>>> x=25
>>> import math
>>> y=math.sqrt (x)
>>> y
5.0
>>> x=81
>>> y=math.sqrt (x)
>>> y
9.0
>>> ################
>>> from math import sqrt
>>> x=101
>>> y=math.sqrt (x)
>>> y
10.04987562112089
>>> print(sqrt(72))
8.48528137423857
>>> print (sqrt(169))
13.0
>>> ################
>>> #učitavanje svih funkcija iz modula pomoću znaka "*"
>>> from math import *
>>> print(sqrt(2))
1.4142135623730951
>>> x=5
>>> y=7
>>> z=x*x+y*y
>>> print(sqrt(z))
8.602325267042627
>>> x=25
>>> y=37
>>> z=x**2+y**2
>>> print(sqrt(z))
44.654227123532216
Ln: 44 Col: 4


Python Shell
File Edit Shell Debug Options Windows Help
>>> pi
3.141592653589793
>>> r=7
>>> print("Površina kruga s polumjerom {0} iznosi {1}.".format(r, pi*r**2))
Površina kruga s polumjerom 7 iznosi 153.93804002589985.
>>>
Ln: 52 Col: 4

Moduli

Zbog toga je preporučljivo za uvoz modula koristiti se naredbom:

import ime_modula

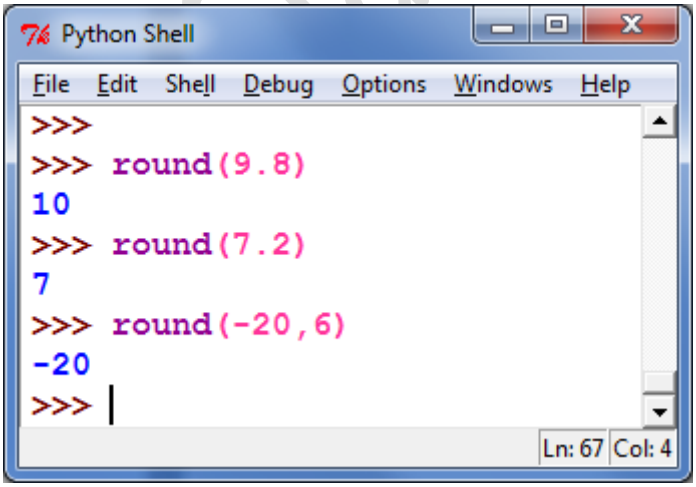
i zatim ispred imena svake funkcije u prefiksu napisati ime_modula ili uz uvoz modula naznačiti funkcije kojima ćemo se koristiti:

from ime_modula import ime_funkcije_1, ime_funkcije_2,…

S obzirom na to da ćemo funkcije modula math često upotrebljavati, u tablici 5.7. nalazi se popis najčešće korištenih funkcija ovog modula:

Funkcija	Opis	Primjer
ceil(x)	najmanji cijeli broj veći od ili jednak x	ceil(3.4)=4 ceil(4.9)=5
floor(x)	najveći cijeli broj manji od ili jednak x	floor(3.2)=3 floor(7.8)=7
exp(x)	e^x	exp(1)=2.71828
log(x,b)	$\log_b x$	log(100,10)=2.0
sin(x)	sin x	sin(pi/2)=1.0
cos(x)	cos x	cos(pi)=1.0
tan(x)	tg x	tan(0)
asin(x)	Arcus sinusa	asin(1)=1.57079
acos(x)	Arcus cosinus	acos(-1)=3.14159

Funkcija **round(x)** vraća cijeli broj koji je najbliži realnom broju.



Modul random

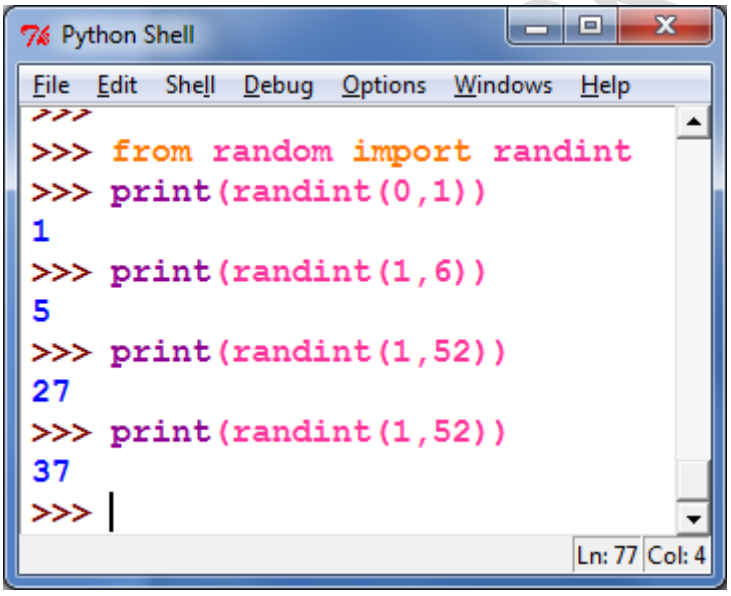
Za generiranje brojeva koristit ćemo se funkcijama iz modula koji se zove **random**. Modul random sadržava funkciju kojima se može generirati nasumične brojeve i modelirati slučajne događaje i pojave.

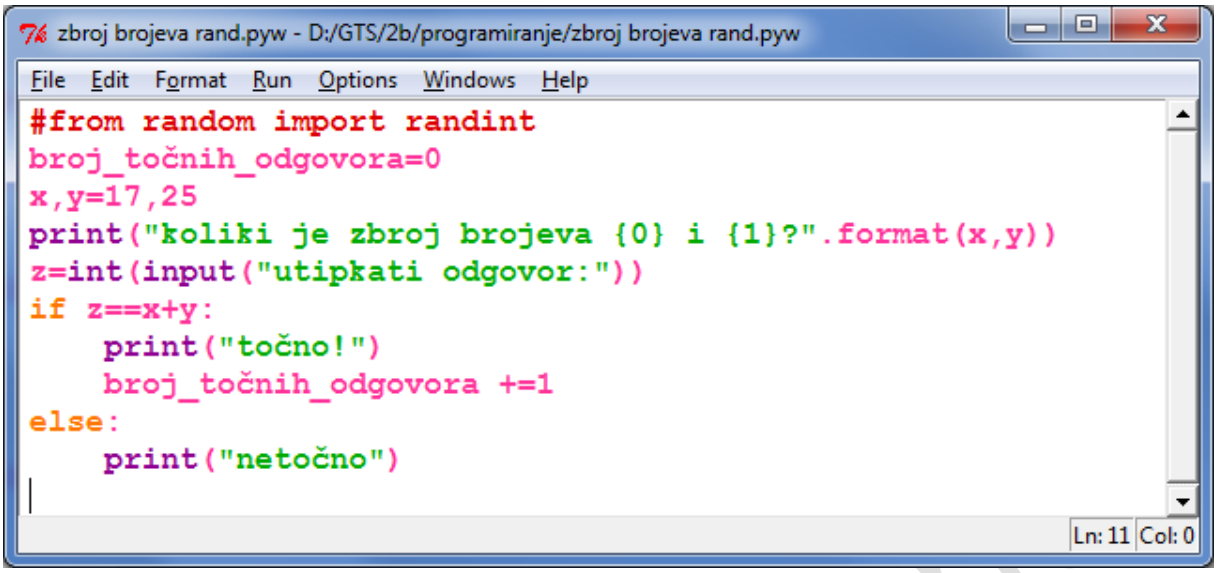
U modulu random postoji funkcija **randint(a,b)** kojom možemo oponašati različite primjere zadataka. Svaki put kad se ta funkcija pozove ona će vratiti jedan cijeli broj iz zatvorenog intervala (a,b) (tj. i brojeve koji su jednaki granicama a i b).

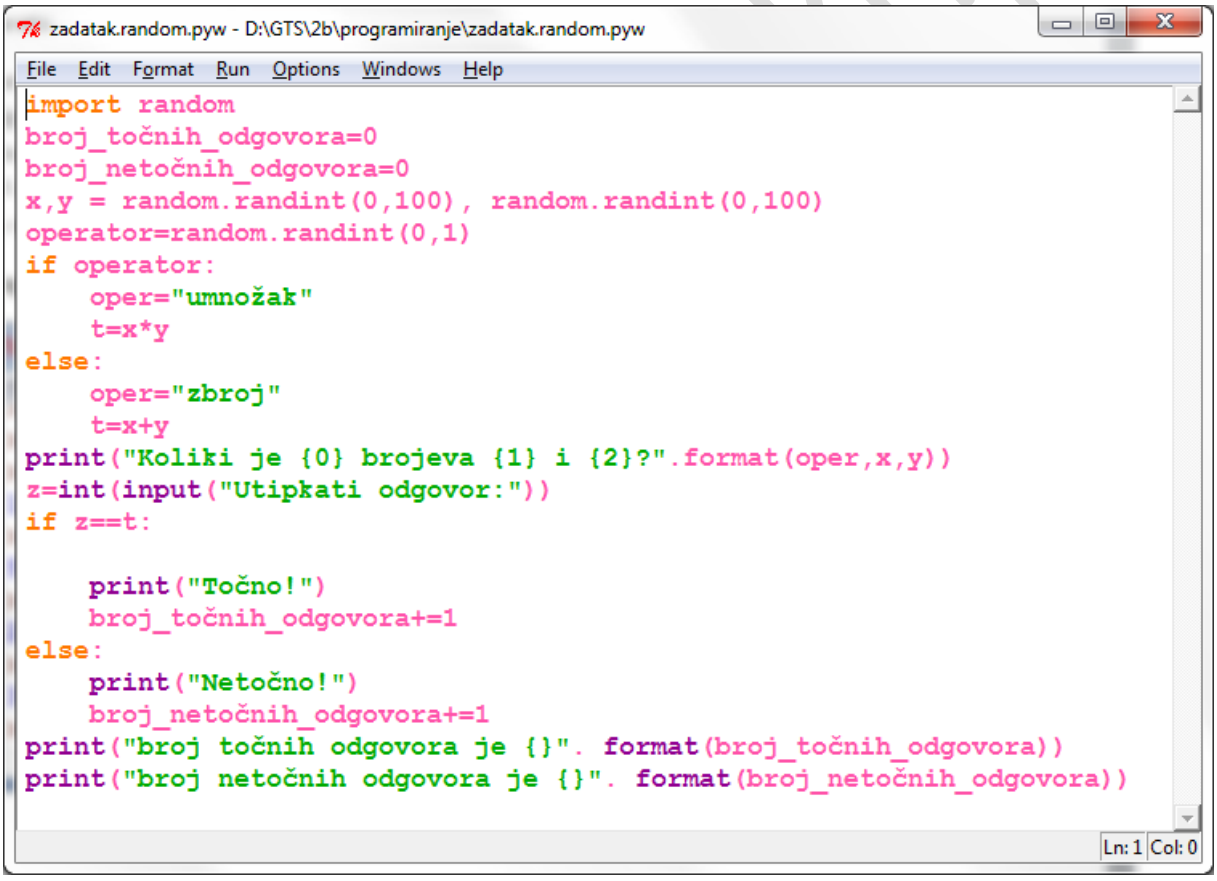
Prema tome, funkcija randint(0,1) može oponašati novčić (ako glavu označimo brojem 0 i pismo brojem 1), funkcija randint(1,6) može oponašati kocku, a funkcija randint(1,N) snop karata.

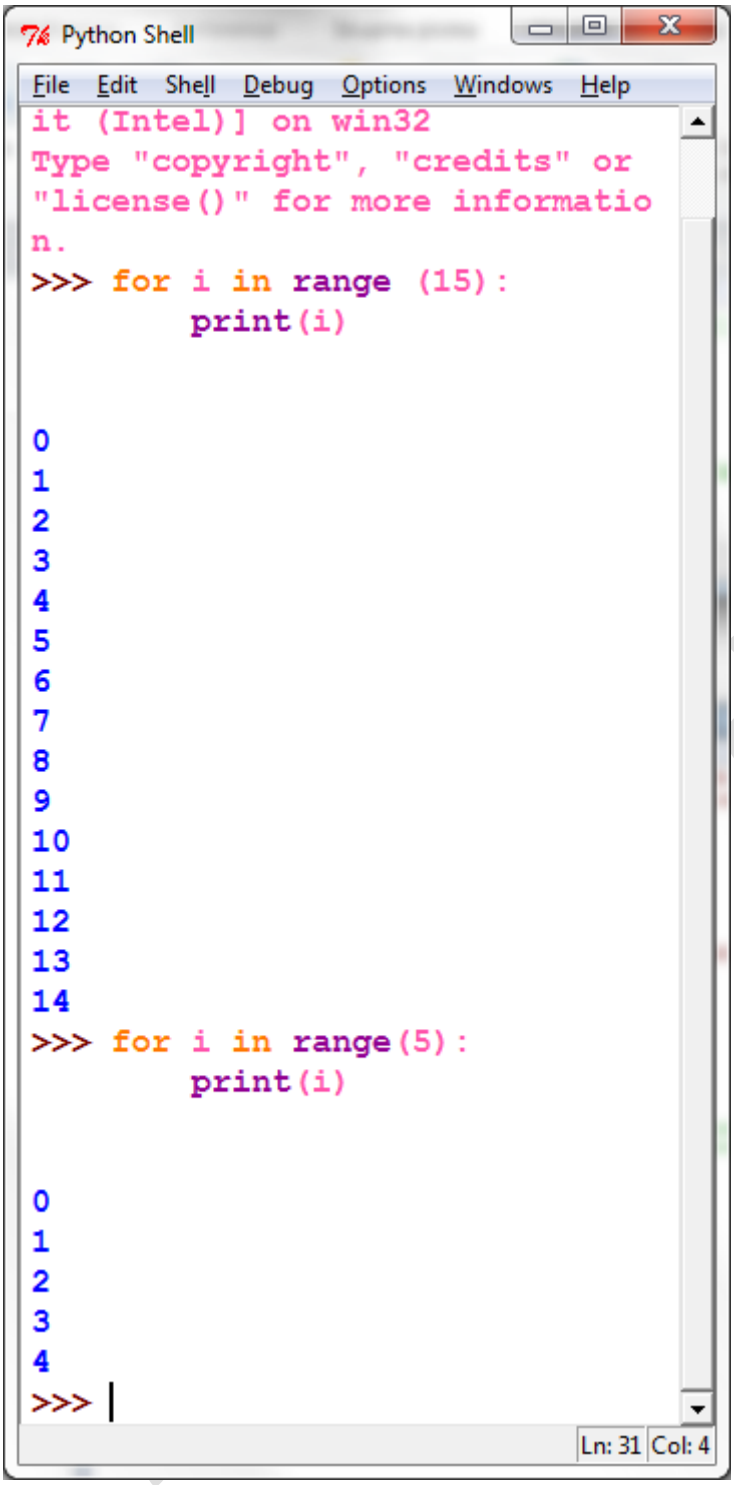
Funkcija randit(a,b) ne određuje ishode sasvim slučajno- ona ih izračunava nekim algoritmom koji jako dobro oponaša slučajnost.
Zbog toga se brojevi koje ona vraća mogu nazvati *pseudoslučajnim*.
Mi ćemo ih nazvati *nasumično odabranim brojevima* ili kraće samo *nasumičnim brojevima*.
Funkcija randint(a,b) može nam poslužiti kao generator nasumičnih brojeva.

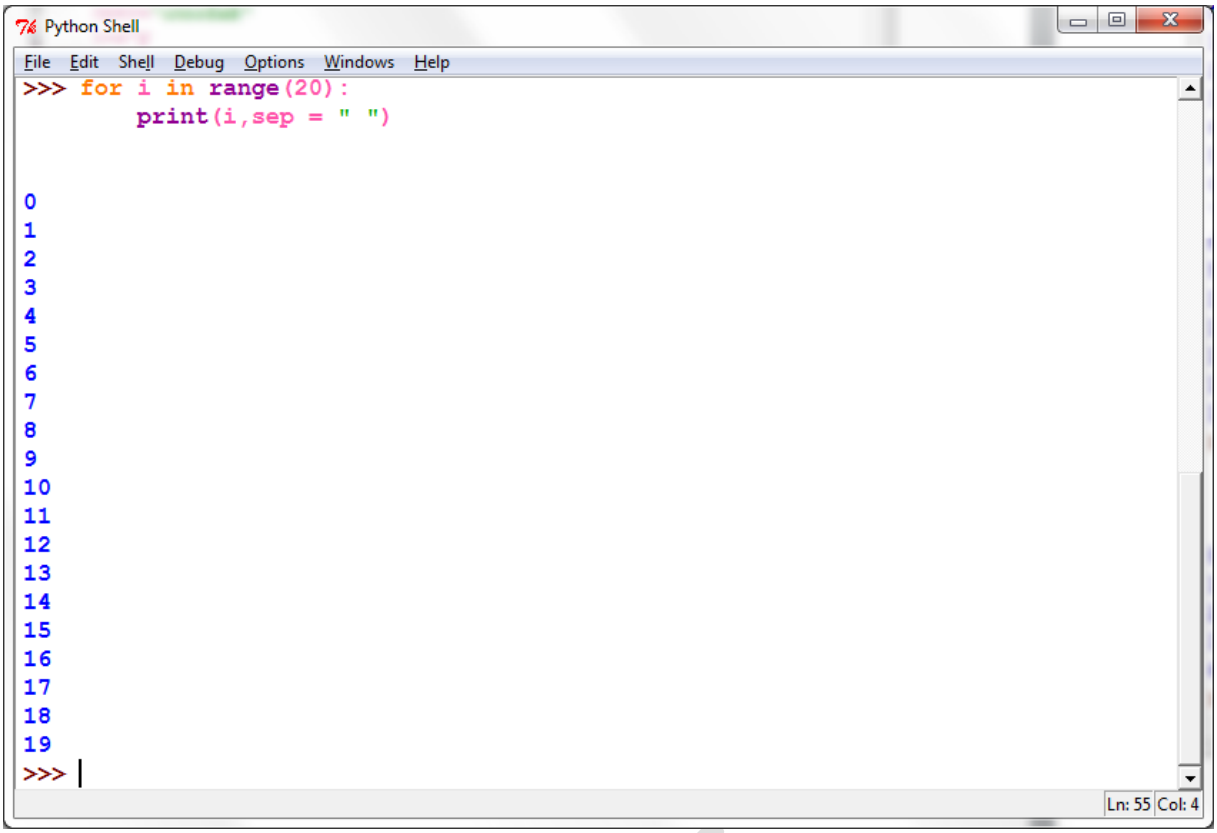
Pogledajmo brojeve koje dobivamo u interaktivnom sučelju:

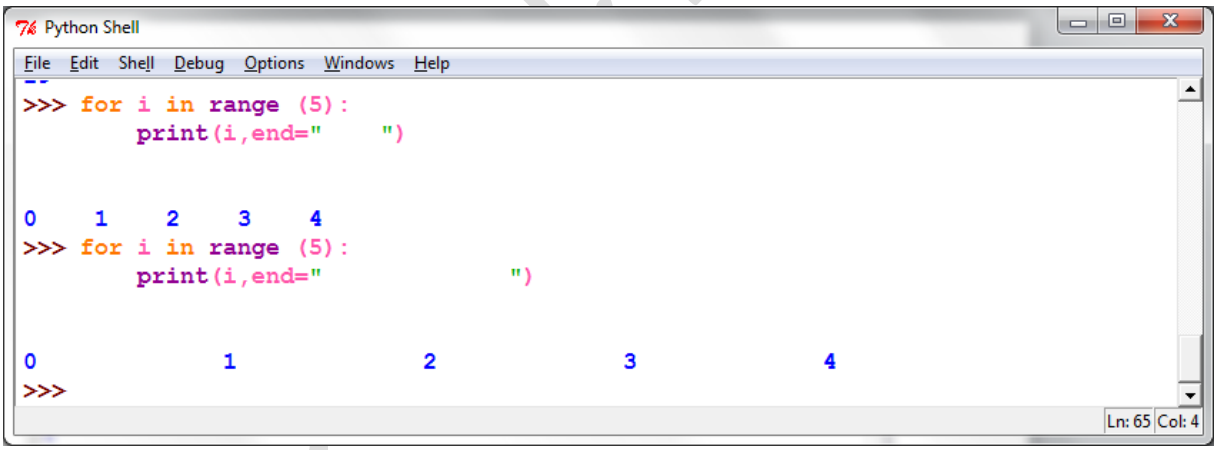


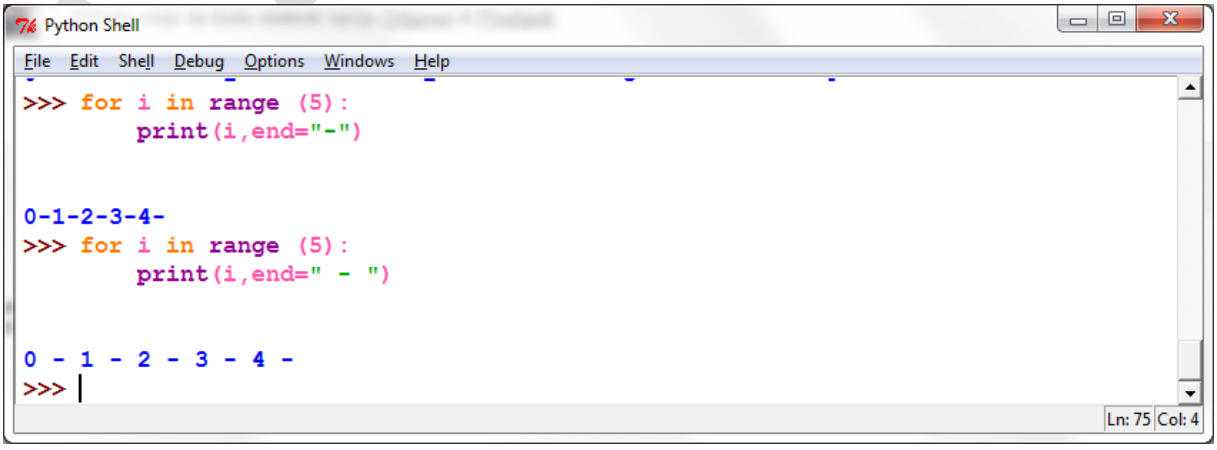

zbroj brojeva rand.pyw - D:/GTS/2b/programiranje/zbroj brojeva rand.pyw
File Edit Format Run Options Windows Help
#from random import randint
broj_točnih_odgovora=0
x,y=17,25
print("koliki je zbroj brojeva {0} i {1}?".format(x,y))
z=int(input("utipkati odgovor:"))
if z==x+y:
print("točno!")
broj_točnih_odgovora +=1
else:
print("netočno")
Ln: 11 Col: 0

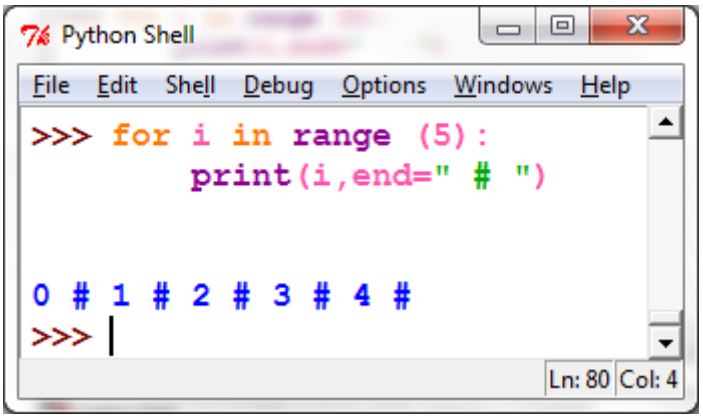

zadatak.random.pyw - D:\GTS\2b\programiranje\zadatak.random.pyw
File Edit Format Run Options Windows Help
import random
broj_točnih_odgovora=0
broj_netočnih_odgovora=0
x,y = random.randint(0,100), random.randint(0,100)
operator=random.randint(0,1)
if operator:
oper="umnožak"
t=x*y
else:
oper="zbroj"
t=x+y
print("Koliki je {0} brojeva {1} i {2}?".format(oper,x,y))
z=int(input("Utipkati odgovor:"))
if z==t:
print("Točno!")
broj_točnih_odgovora+=1
else:
print("Netočno!")
broj_netočnih_odgovora+=1
print("broj točnih odgovora je {}". format(broj_točnih_odgovora))
print("broj netočnih odgovora je {}". format(broj_netočnih_odgovora))
Ln: 1 Col: 0

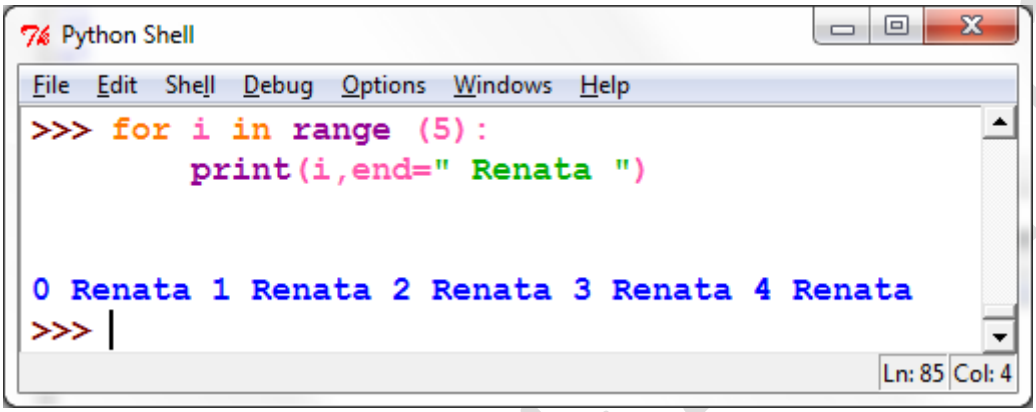

Python Shell
File Edit Shell Debug Options Windows Help
it (Intel)] on win32
Type "copyright", "credits" or
"license()" for more informatio
n.
>>> for i in range (15):
print(i)
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
>>> for i in range(5):
print(i)
0
1
2
3
4
>>>
Ln: 31 Col: 4

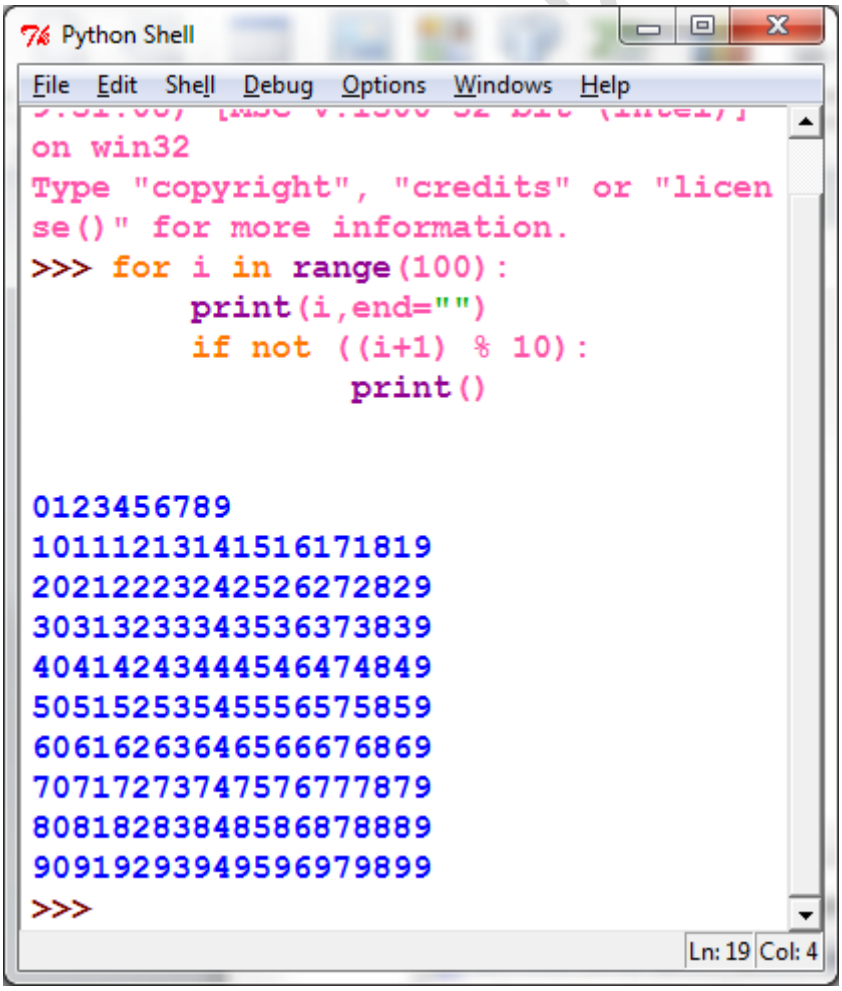

Python Shell
File Edit Shell Debug Options Windows Help
>>> for i in range(20):
print(i,sep = " ")
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
>>>
Ln: 55 Col: 4

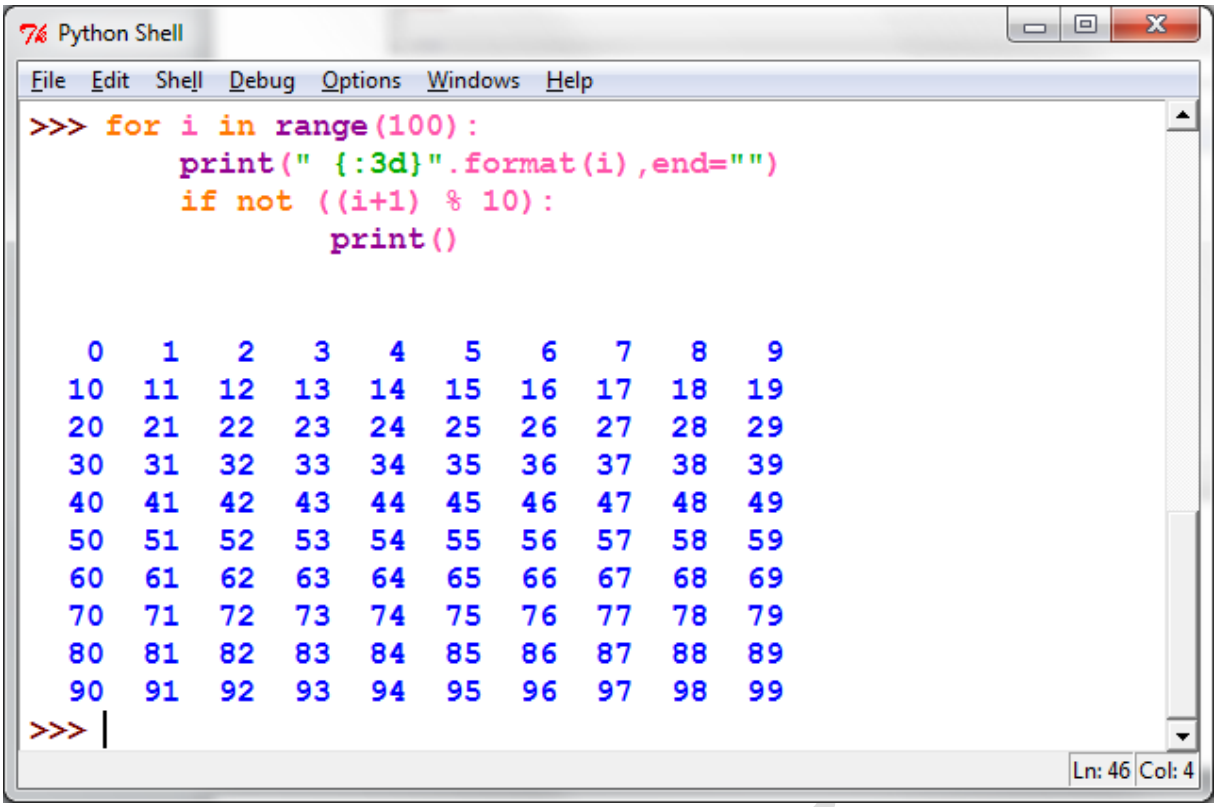

Python Shell
File Edit Shell Debug Options Windows Help
>>> for i in range (5):
print(i,end=" ")
0 1 2 3 4
>>> for i in range (5):
print(i,end=" ")
0 1 2 3 4
>>>
Ln: 65 Col: 4


Python Shell
File Edit Shell Debug Options Windows Help
>>> for i in range (5):
print(i,end="-")
0-1-2-3-4-
>>> for i in range (5):
print(i,end=" - ")
0 - 1 - 2 - 3 - 4 -
>>>
Ln: 75 Col: 4


Python Shell
File Edit Shell Debug Options Windows Help
>>> for i in range (5):
print(i,end=" # ")
0 # 1 # 2 # 3 # 4 #
>>>
Ln: 80 Col: 4

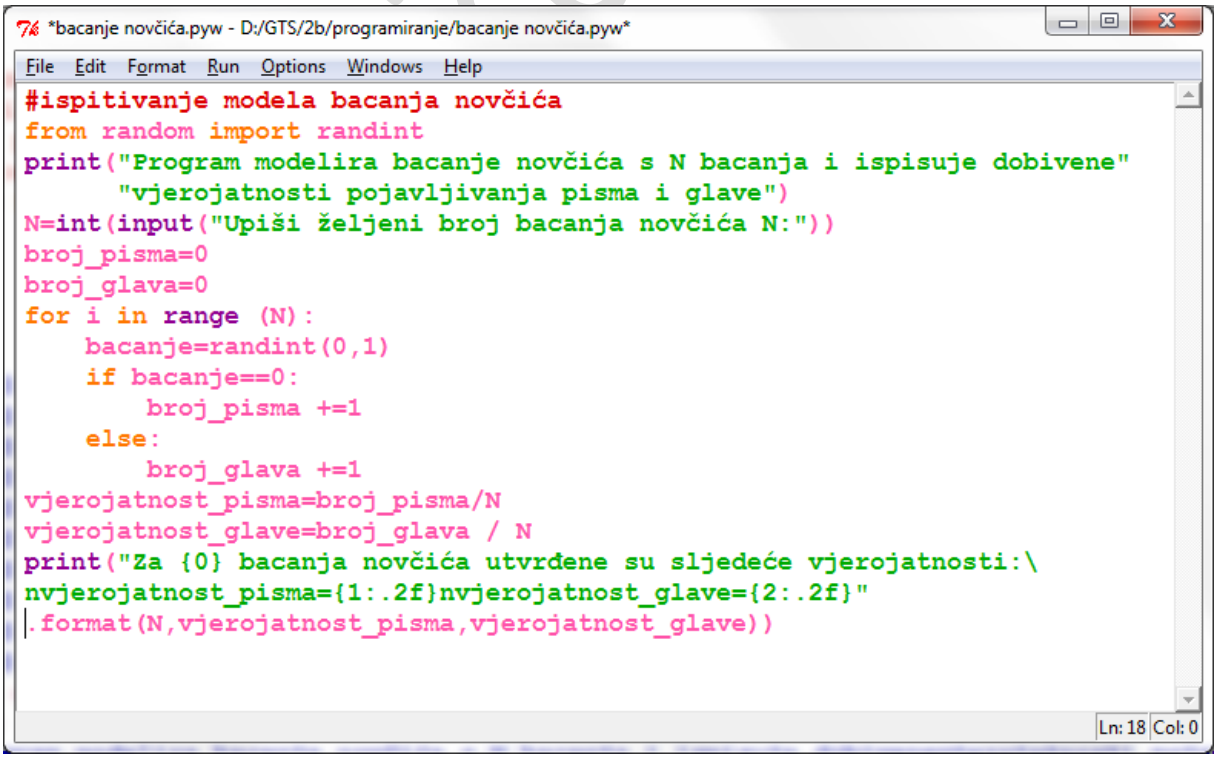

Python Shell
File Edit Shell Debug Options Windows Help
>>> for i in range (5):
print(i,end=" Renata ")
0 Renata 1 Renata 2 Renata 3 Renata 4 Renata
>>>
Ln: 85 Col: 4


Python Shell
File Edit Shell Debug Options Windows Help
on win32
Type "copyright", "credits" or "licen
se()" for more information.
>>> for i in range(100):
print(i,end="")
if not ((i+1) % 10):
print()
0123456789
10111213141516171819
20212223242526272829
30313233343536373839
40414243444546474849
50515253545556575859
60616263646566676869
70717273747576777879
80818283848586878889
90919293949596979899
>>>
Ln: 19 Col: 4



Zadatak 1.

Napišimo program za ocjenu generatora nasumičnih brojeva. Program treba simulirati bacanje novčića n puta te treba ispisati vjerojatnost pojavljivanja pisma odnosno glave.



visina-while.pyw - D:/GTS/2b/programiranje/visina-while.pyw

File Edit Format Run Options Windows Help

```
#Izračunavanje srednje visine petljom while
print("Izračunaj srednje visine proizvoljnog broja osoba")
brojilo=0
zbroj=0.0
imabrojeva="da"
while imabrojeva=="da":
    broj=float(input("Upisati visinu u m:"))
    zbroj+= broj
    brojilo+=1
    imabrojeva=input("Ima li jos unosa(utipkaj da ili ne)?")
print('\nSrednja visina {} osoba u metrima{:.2f}'.format(brojilo, zbroj/brojilo)
```

Ln: 11 Col: 0

igra brojevima4.pyw - D:/GTS/2b/programiranje/igra brojevima4.pyw

File Edit Format Run Options Windows Help

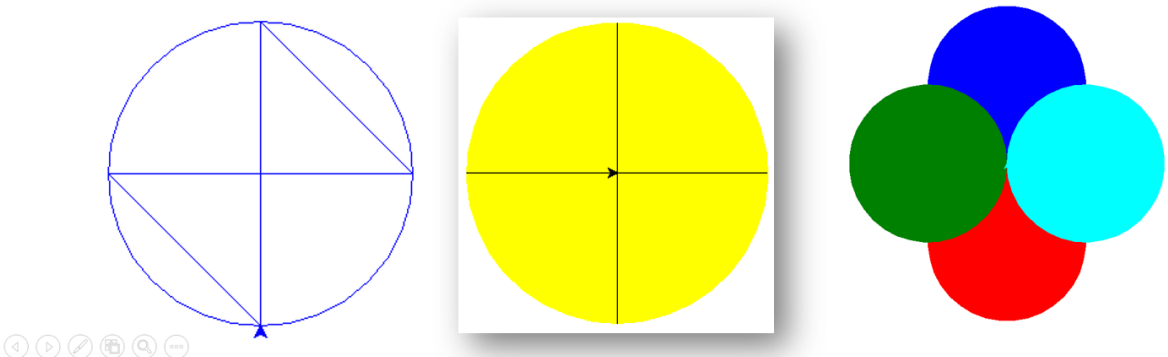
```
#Igra s brojevima(četvrta inačica)
import random
random.seed()
nastavak="da"
while nastavak == "da":
    brojpitanja=random.randint(2,5)
    print("U ovoj igri trebate odgovoriti na {} pitanja!\n".format(brojpitanja))
    brojtočnihodgovora=0
    for i in range (brojpitanja):
        x,y=random.randint(10,20), random.randint(10,20)#interval operanada
        operator=random.randint(0,1)
        if operator:
            oper="umnožak"
            t=x*y
        else:
            oper="zbroj"
            t=x+y
        print("Koliki je {0} brojeva {1} i {2}?".format(oper,x,y))
        z=int(input("Utipkati odgovor:"))
        if z==t:
            print("Točno!")
            brojtočnihodgovora+=1
            print("-----------")
        else:
            print("Netočno!")
            print("-----------")
    print("\nOdgovorili ste na {} pitanja. \n".format(brojpitanja))
    print("Od toga je {} točnih.".format(brojtočnihodgovora))
    postotak=100*brojtočnihodgovora/brojpitanja
    print("To znači da imaš {}% točnih odgovora.\n".format(postotak))
    nastavak=input("Želite li nastaviti igru (da ili  ne)?")
input("Za završetak programa pritisnite tipku <Enter> - Lijep pozdrav!")
```

Ln: 27 Col: 40

Grafički modul

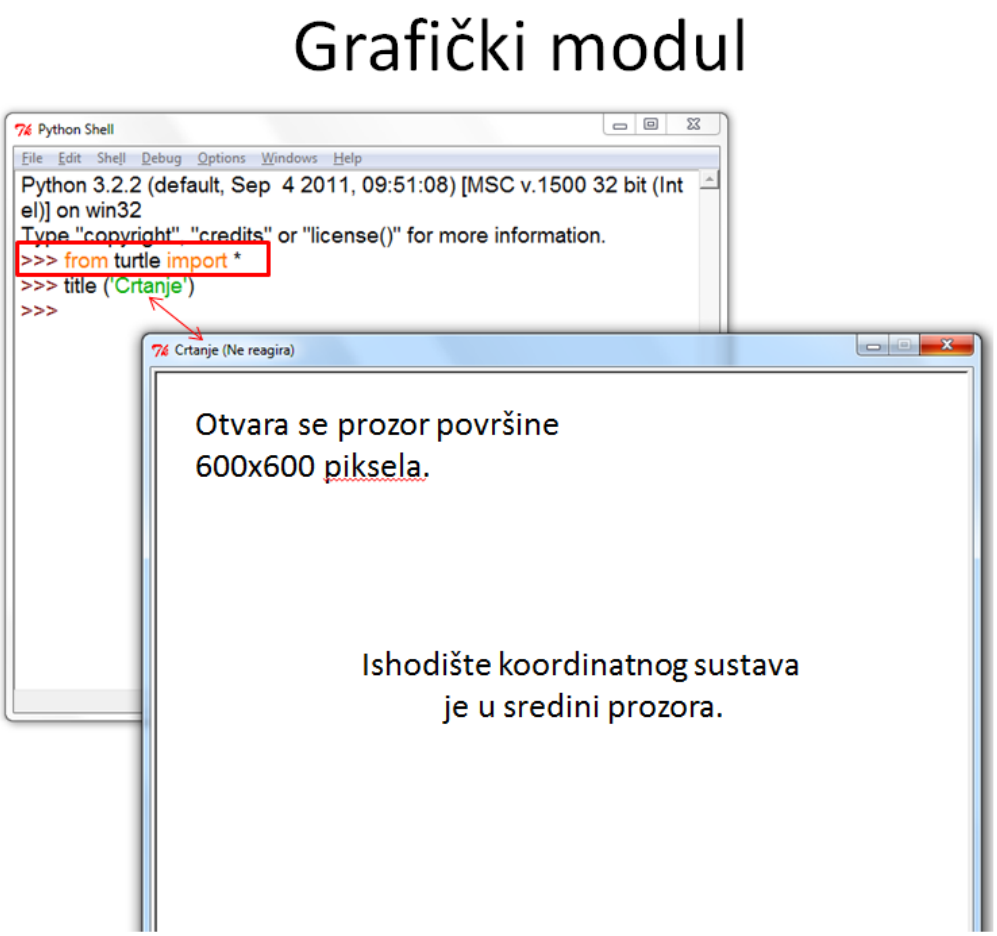
Python – grafika

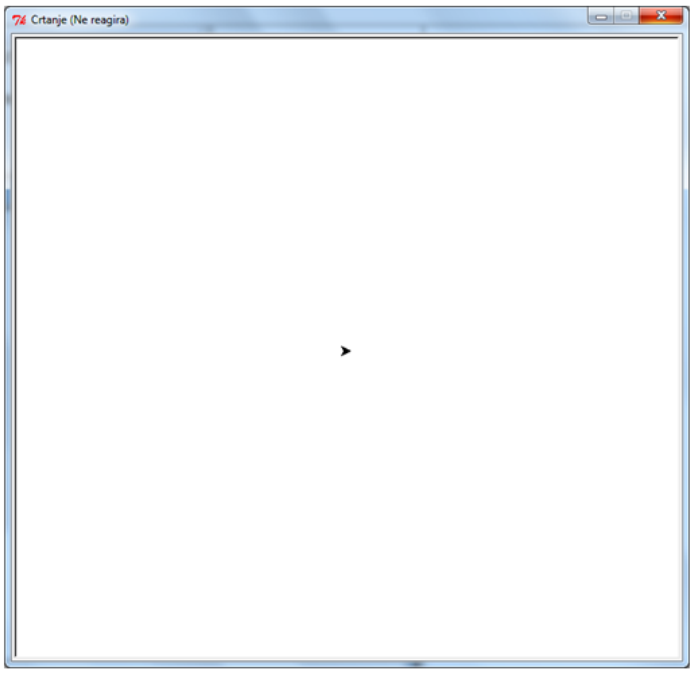
Armando Slaviček



Osnove računalne grafike

- Modul **turtle** omogućava crtanje u Pythonu
- Crtanje se odvija u grafičkom prozoru
- Naredbom reset() postavlja se strelica (olovka za crtanje) u sredini grafičkog prozora
- Pomicanjem strelice (olovke) ostavljamo trag u grafičkom prozoru
- Grafički prozor je veličine 600x600 piksela (px)
- Na početku rada u grafičkom prozoru moramo učitati modul turtle: from turtle import *






```
>>> reset()
```

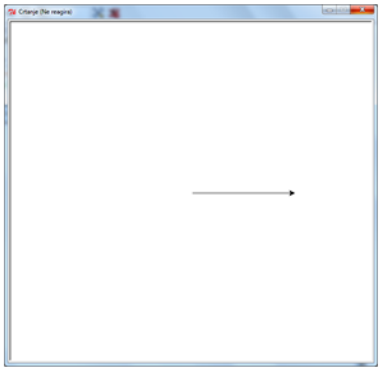
- Pojavit će se u ishodištu pero (strelica glave kornjače) usmjerena u smjeru pozitivne osi x.

Osnovne funkcije za gibanje pera

Funkcija	Alternativni naziv	Opis djelovanja funkcije
forward (d)	fd(d)	Pomiče pero za d jedinica naprijed
backward (d)	back (d), bk (d)	Pomiče pero za d jedinica unatrag
right(kut)	rt(kut)	Zakreće pero za kut stupnjeva udesno
left(kut)	lt(kut)	Zakreće pero za kut stupnjeva ulijevo

Funkcije za relativna gibanja pera (vektorska grafika)

```
>>> from turtle import *
>>> title ('Crtanje')
>>> pd()
>>> fd(200)
>>>
```



Modul turtle ima i funkcije za gibanje pera određeno apsolutnim koordinatama

Funkcija	Alternativni naziv	Opis djelovanja funkcije
goto(x,y)	setpos(x,y)	Pomiče pero na točku s koordinatama (x,y)
goto (t)	setpos(t)	Pomiče pero na točku s koordinatama t, gdje je t par brojeva (koordinate)
setx(x)		Postavlja prvu koordinatu na x, druga koordinata ostaje nepromijenjena
sety(y)		Postavlja prvu koordinatu na y, druga koordinata ostaje nepromijenjena
setheading(kut)	seth(kut)	Usmjerava pero tako da pokazuje u smjer kuta kut

Upravljanje perom i crtežom

Funkcije upravljanja perom

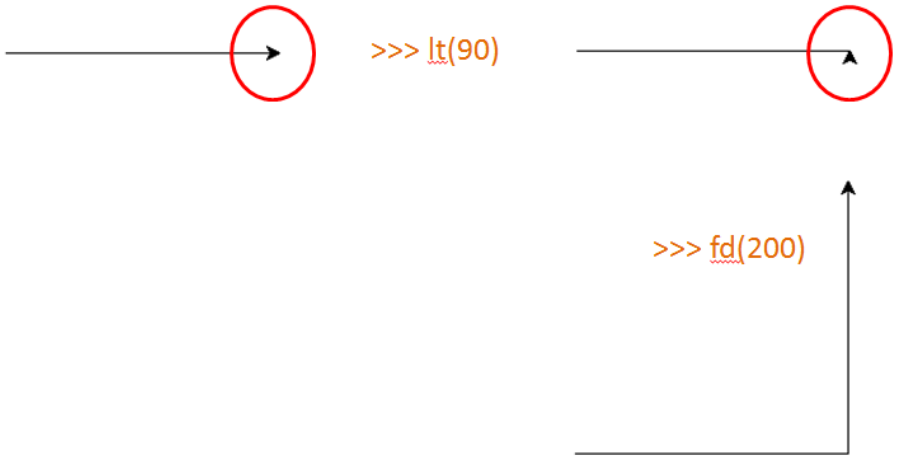
Funkcija	Alternativni naziv	Opis djelovanja funkcije
pendown ()	pd(), down()	Pero se spušta i ostavlja trag
penup()	pu(), up()	Pero se podiže i ne ostavlja trag
pensize(d)	width (d)	Pero ima debljinu *d* jedinica
showturtle()	st ()	Pero postaje vidljivo
hideturtle()	ht ()	Pero postaje nevidljivo
home()		Postavlja pero u početni položaj
undo()		Poništava zadnju akciju pera; ta se funkcija može upotrijebiti višekratno

Funkcije upravljanja crtežom

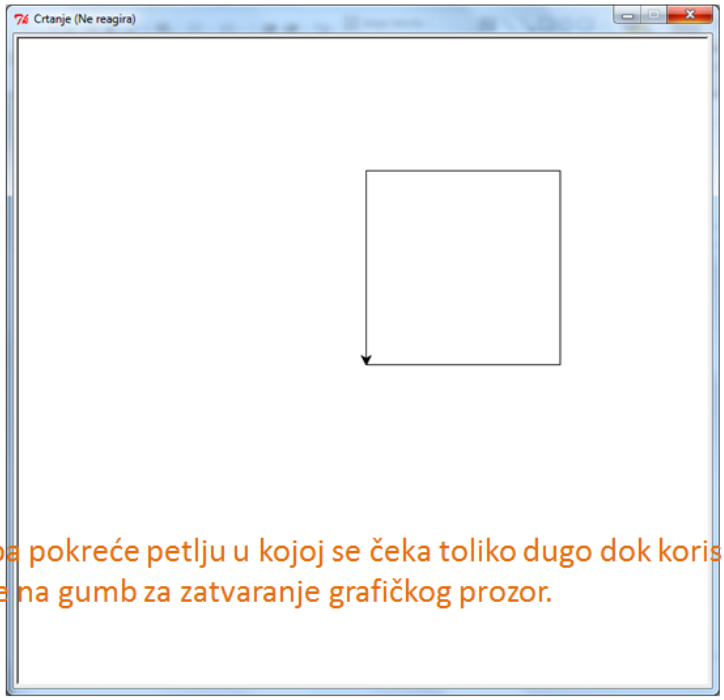
Funkcija	Opis djelovanja funkcije
reset()	Briše sve crteže u grafičkom prozoru; postavlja pero u početni položaj s početnim atributima
clear()	Briše se crtež u grafičkom prozoru; pero ostaje nepromijenjeno

Gibanje pera

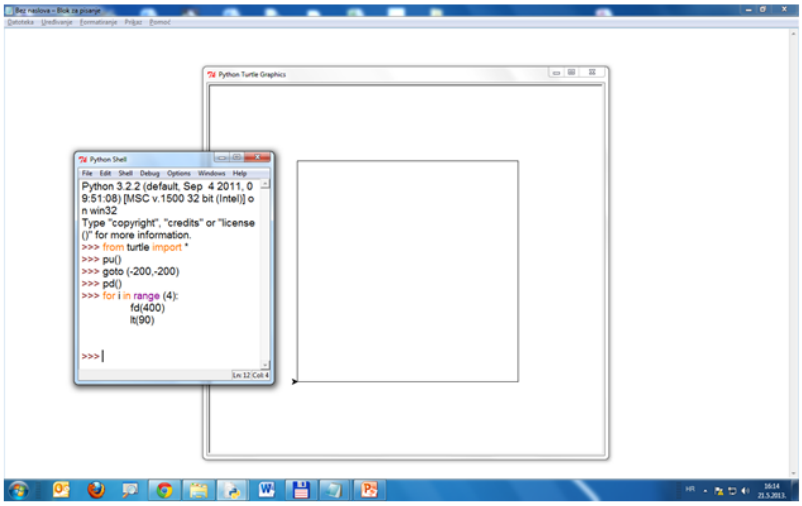
```
>>> from turtle import *
>>> title ('Crtanje')
>>> pd()
>>> fd(200)
>>>
```




```
>>> from turtle import *
>>> title ('Crtanje')
>>> pd()
>>> fd(200)
>>> lt(90)
>>> fd(200)
>>> lt(90)
>>> fd(200)
>>> lt(90)
>>> fd(200)
>>> mainloop()  # ta naredba pokreće petlju u kojoj se čeka toliko dugo dok korisnik
                ne klikne na gumb za zatvaranje grafičkog prozor.
```

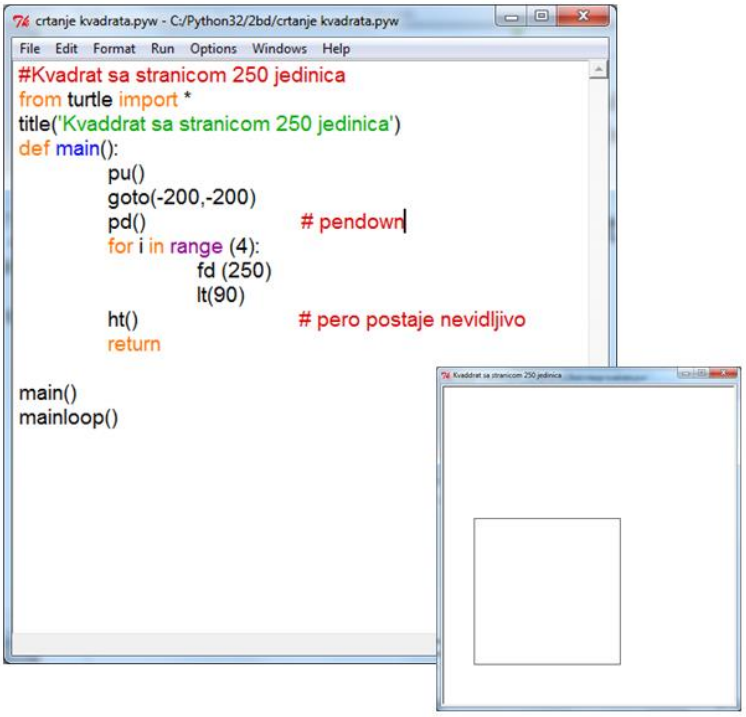


```
>>> from turtle import *
>>> pu()
>>> goto (-200,-200)
>>> pd()
>>> for i in range (4):
        fd(400)
        lt(90)


>>>
```

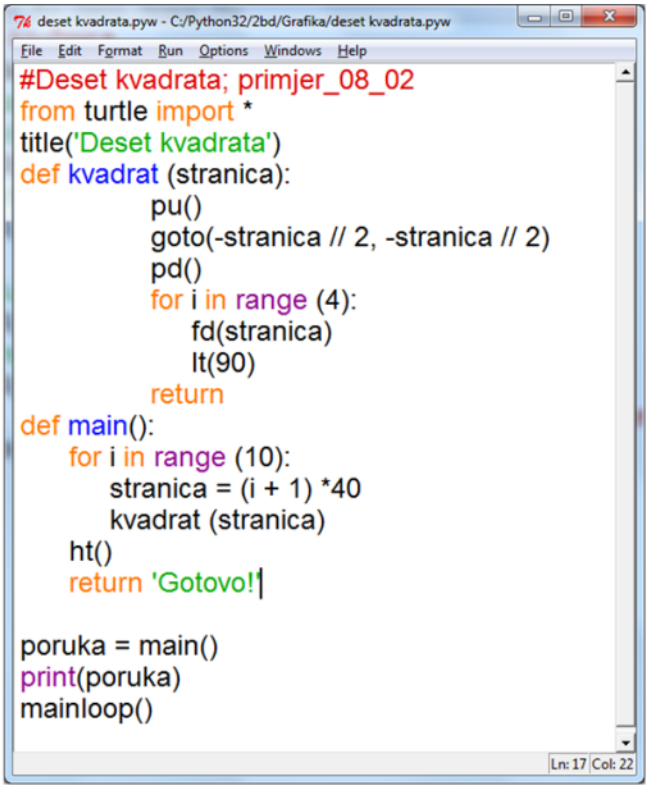


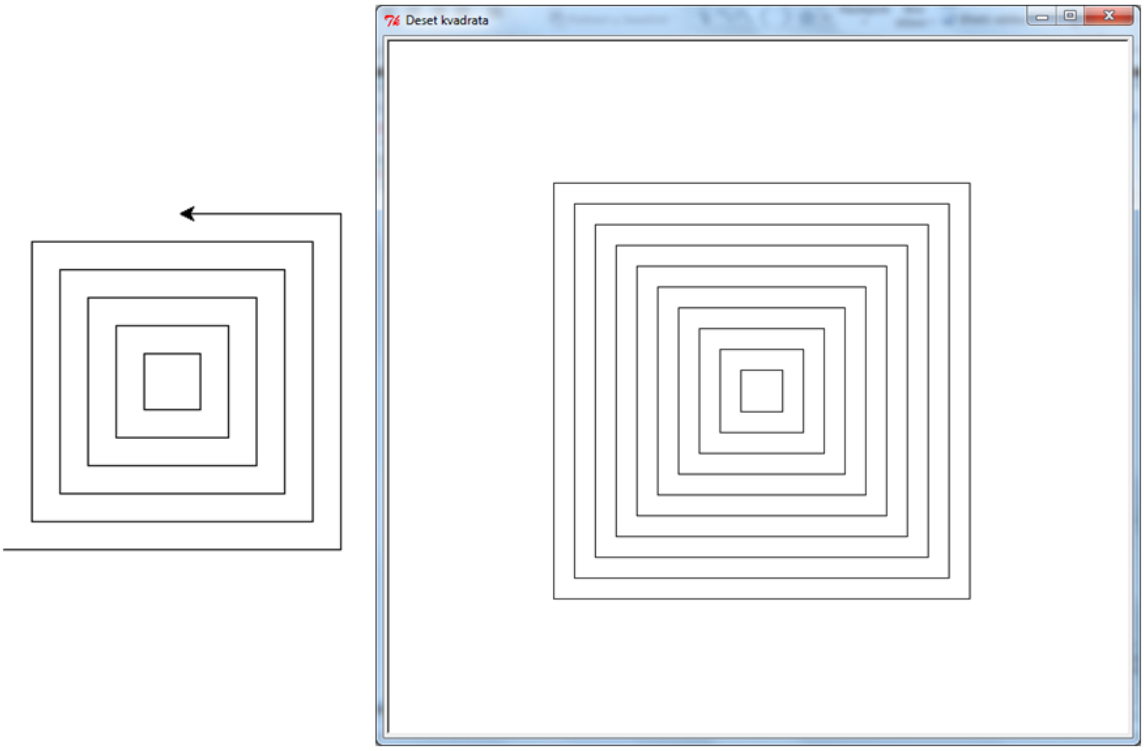
Napiši program koji će crtati kvadrat stranice duljine 250i dodati mu naslov: „Kvadrat sa stranicom 250 jedinica”, a na kraju će sakriti kornjaču.

```
#Kvadrat sa stranicom 250 jedinica
from turtle import *
title('Kvaddrat sa stranicom 250 jedinica')
def main():
        pu()
        goto(-200,-200)
        pd()                    # pendown
        for i in range (4):
                fd (250)
                lt(90)
        ht()                    # pero postaje nevidljivo
        return

main()
mainloop()
```



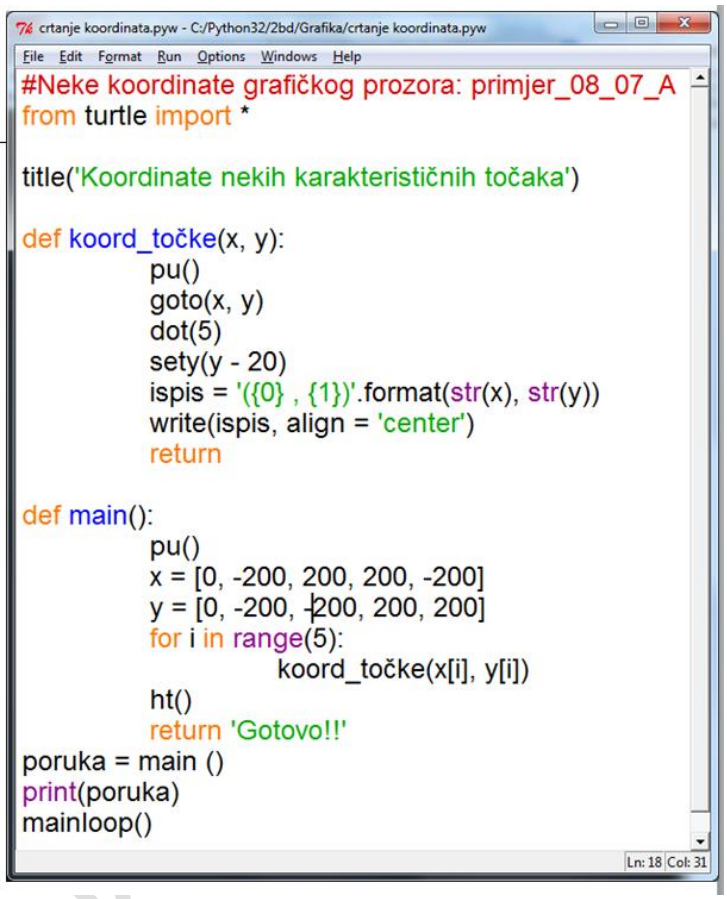
```
#Deset kvadrata; primjer_08_02
from turtle import *
title('Deset kvadrata')
def kvadrat (stranica):
        pu()
        goto(-stranica // 2, -stranica // 2)
        pd()
        for i in range (4):
            fd(stranica)
            lt(90)
        return
def main():
    for i in range (10):
        stranica = (i + 1) *40
        kvadrat (stranica)
    ht()
    return 'Gotovo!'

poruka = main()
print(poruka)
mainloop()
```

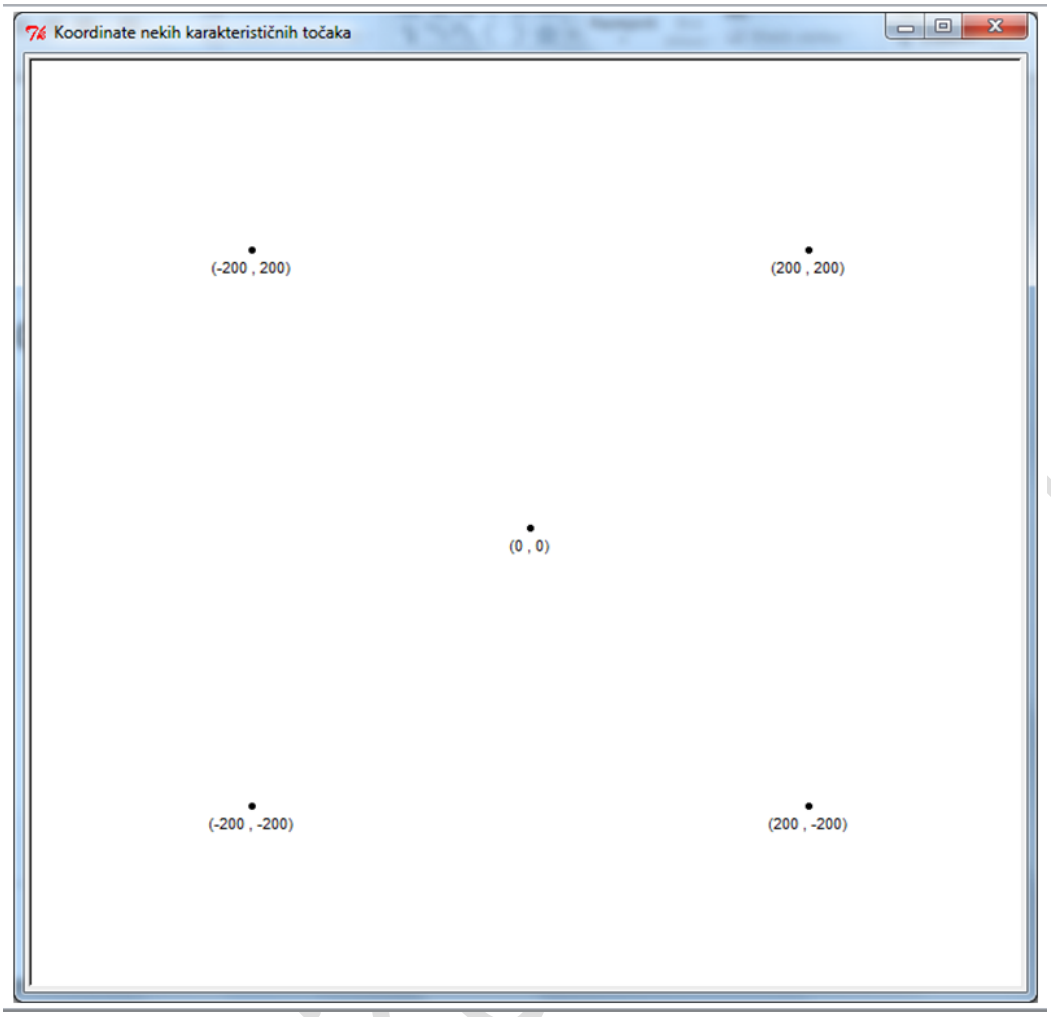


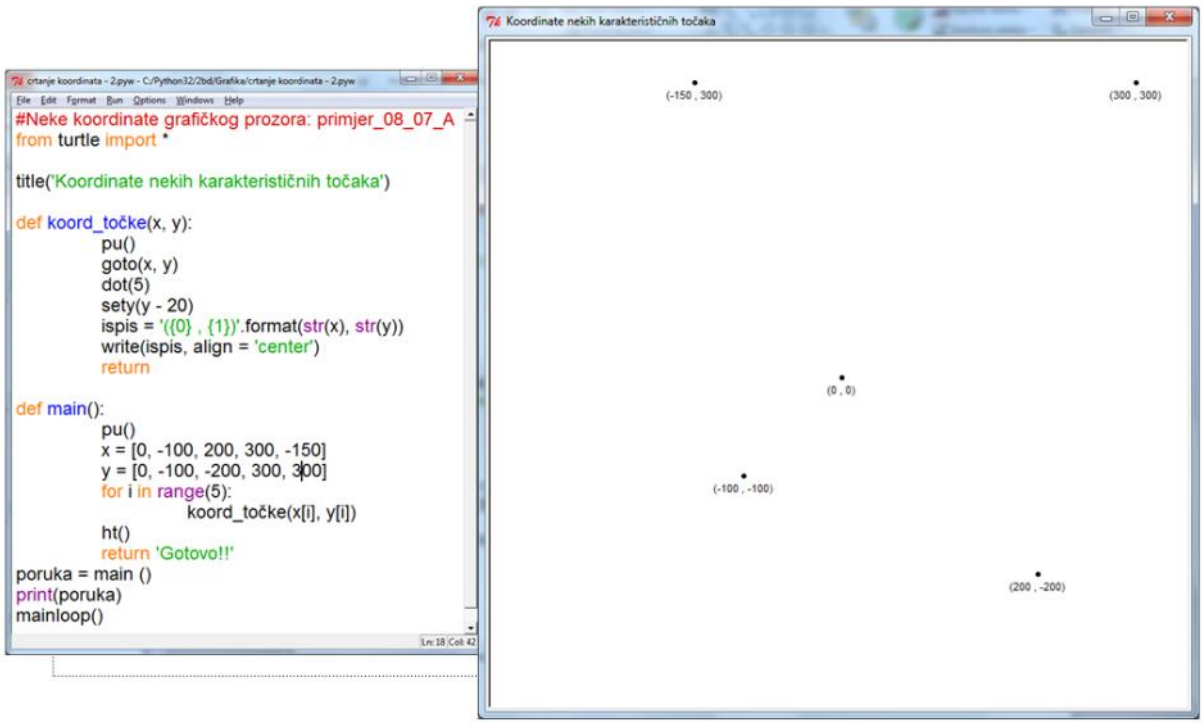


KOORDINATE GRAFIČKOG PROZORA

```
#Neke koordinate grafičkog prozora: primjer_08_
from turtle import *

title('Koordinate nekih karakterističnih točaka')

def koord_točke(x, y):
        pu()
        goto(x, y)
        dot(5)
        sety(y - 20)
        ispis = '({0} , {1})'.format(str(x), str(y))
        write(ispis, align = 'center')
        return

def main():
        pu()
        x = [0, -200, 200, 200, -200]
        y = [0, -200, -200, 200, 200]
        for i in range(5):
                koord_točke(x[i], y[i])
        ht()
        return 'Gotovo!!'
poruka = main ()
print(poruka)
mainloop()
```

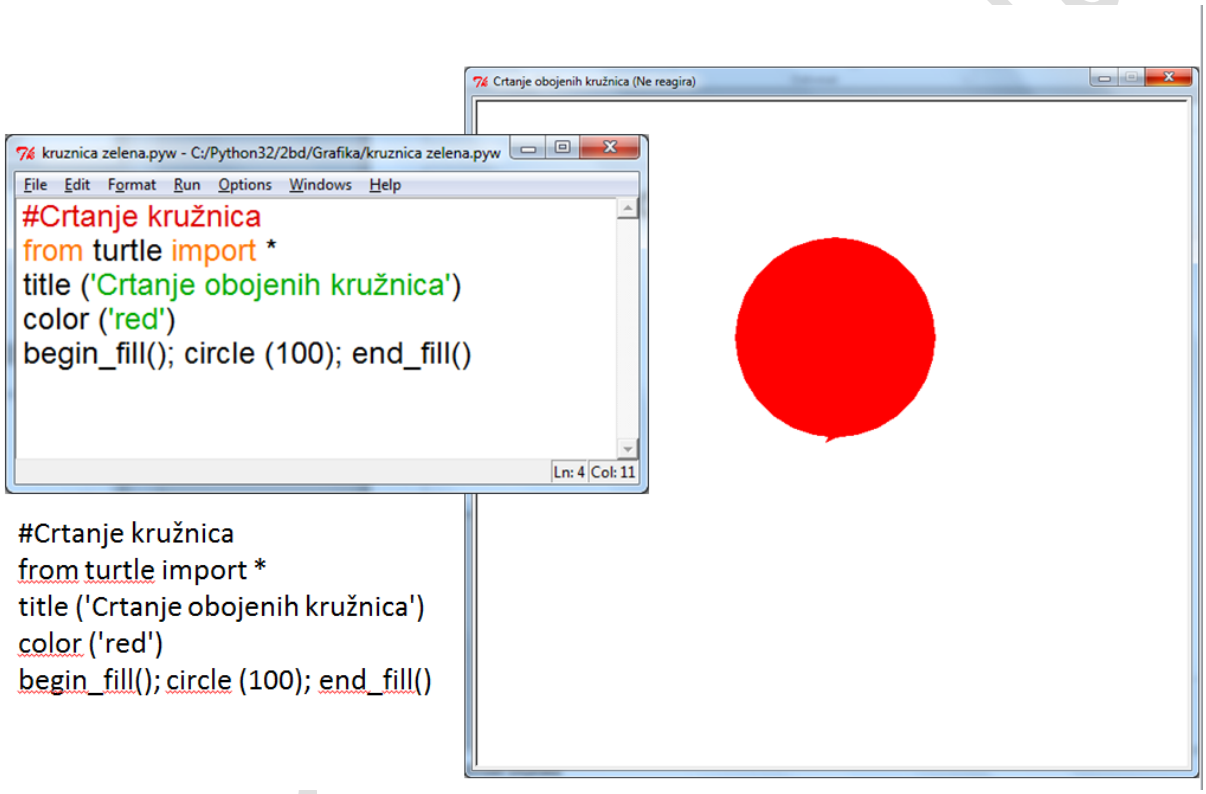



Koordinate nekih karakterističnih točaka
(-200 , 200)
(200 , 200)
(0 , 0)
(-200 , -200)
(200 , -200)


crtanje koordinata - 2.pyw - C:/Python32/2bd/Grafika/crtanje koordinata - 2.pyw
File Edit Format Run Options Windows Help
#Neke koordinate grafičkog prozora: primjer_08_07_A
from turtle import *
title('Koordinate nekih karakterističnih točaka')
def koord_točke(x, y):
pu()
goto(x, y)
dot(5)
sety(y - 20)
ispis = '({0} , {1})'.format(str(x), str(y))
write(ispis, align = 'center')
return
def main():
pu()
x = [0, -100, 200, 300, -150]
y = [0, -100, -200, 300, 300]
for i in range(5):
koord_točke(x[i], y[i])
ht()
return 'Gotovo!!'
poruka = main ()
print(poruka)
mainloop()
Ln: 18 Col: 42
Koordinate nekih karakterističnih točaka
(-150 , 300)
(300 , 300)
(0 , 0)
(-100 , -100)
(200 , -200)

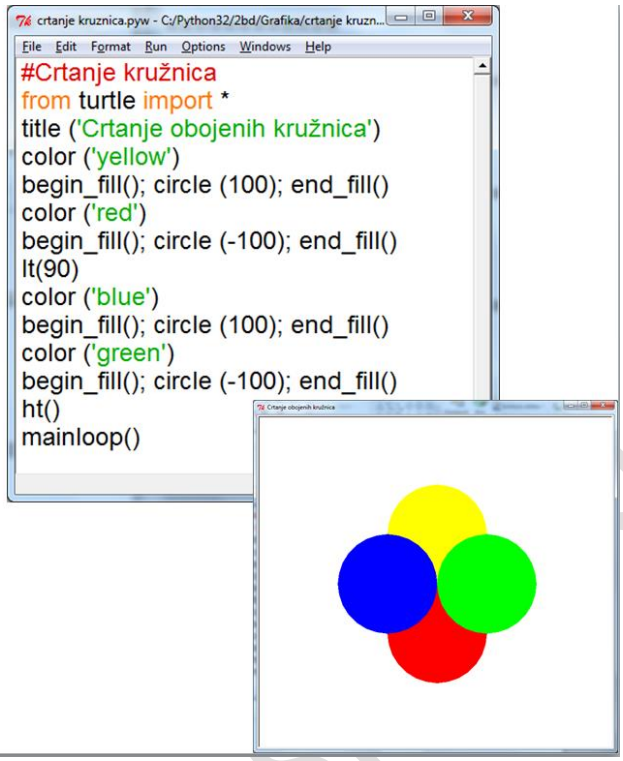
Crtanje u bojama

Funkcija	Opis djelovanja
pencolor (boja)	Boja traga koji ostavlja strelica (olovka)
fillcolor (boja)	Boja ispune
begin_fill ()	Počni ispunjavati neki lik (područje) koje ćemo crtati zadanom bojom
end_fill ()	Prekid ispunjavanja lika zadanom bojom
pensize (d), width (d)	Određujemo debljinu linije

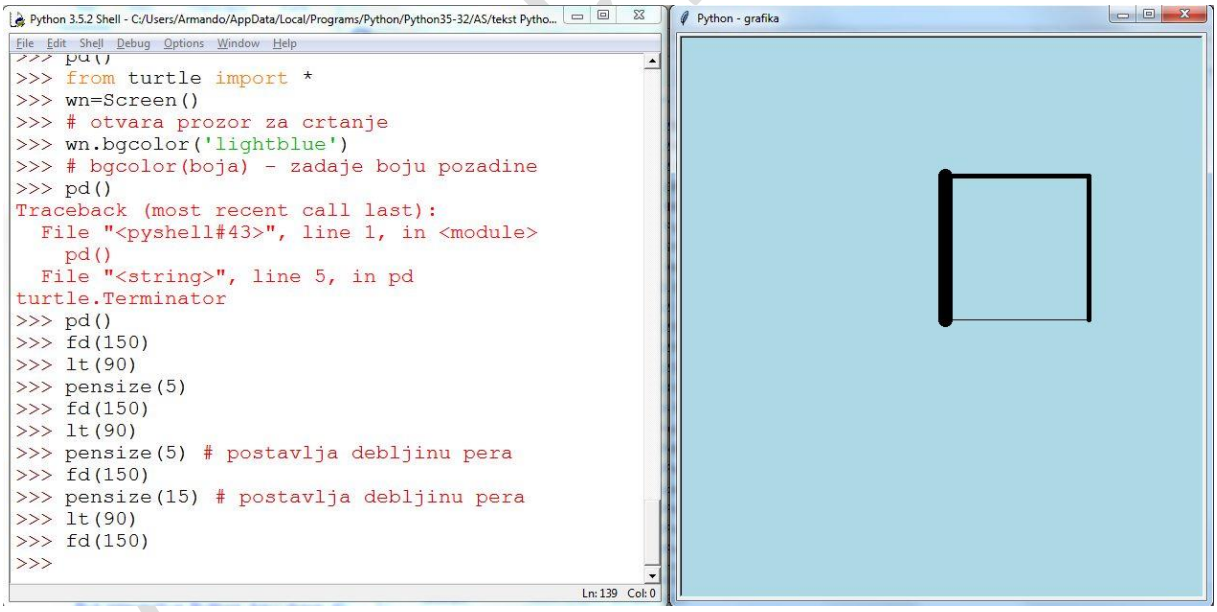



```
#Crtanje kružnica
from turtle import *
title ('Crtanje obojenih kružnica')
color ('red')
begin_fill(); circle (100); end_fill()
```

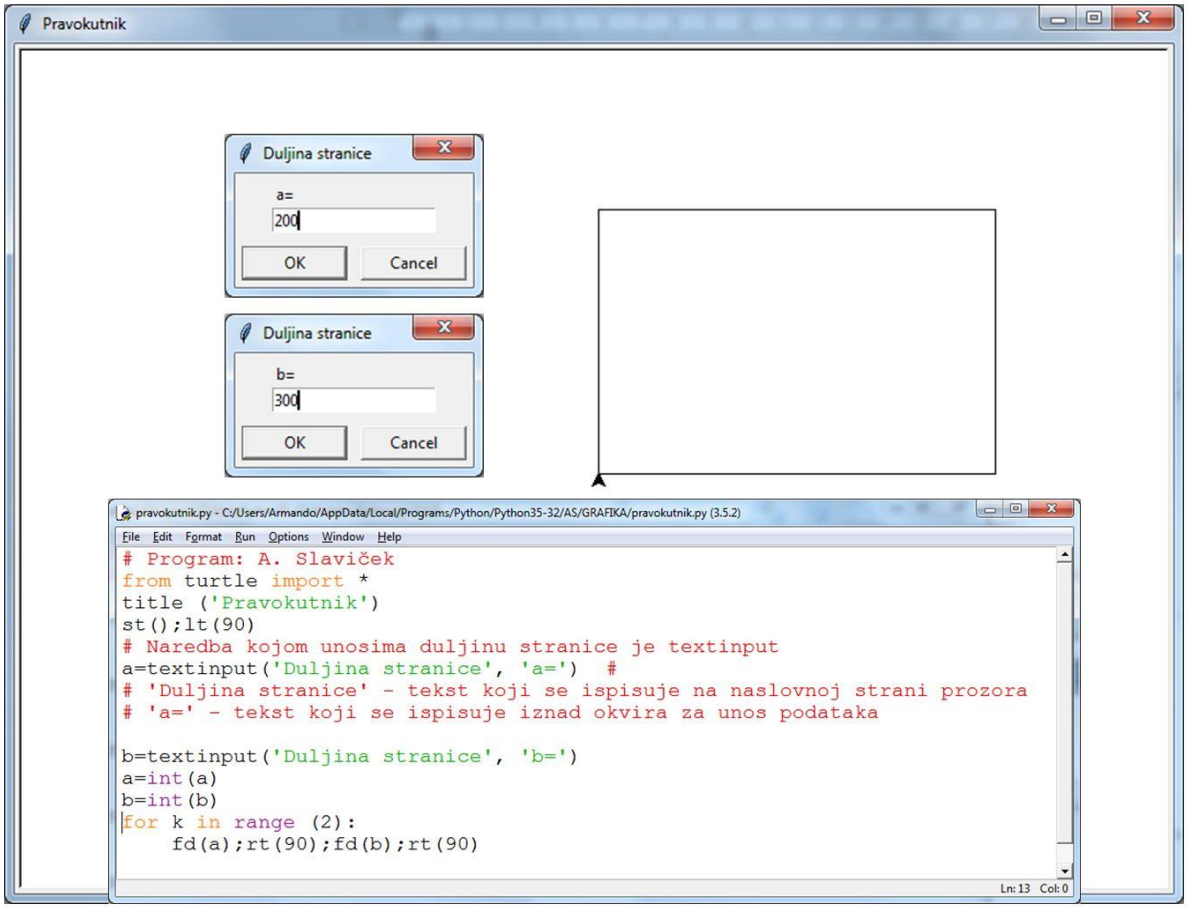
```
#Crtanje kružnica
from turtle import *
title ('Crtanje obojenih kružnica')
color ('yellow')
begin_fill(); circle (100); end_fill()
color ('red')
begin_fill(); circle (-100); end_fill()
lt(90)
color ('blue')
begin_fill(); circle (100); end_fill()
color ('green')
begin_fill(); circle (-100); end_fill()
ht()
# pero postaje nevidljivo
mainloop()
```



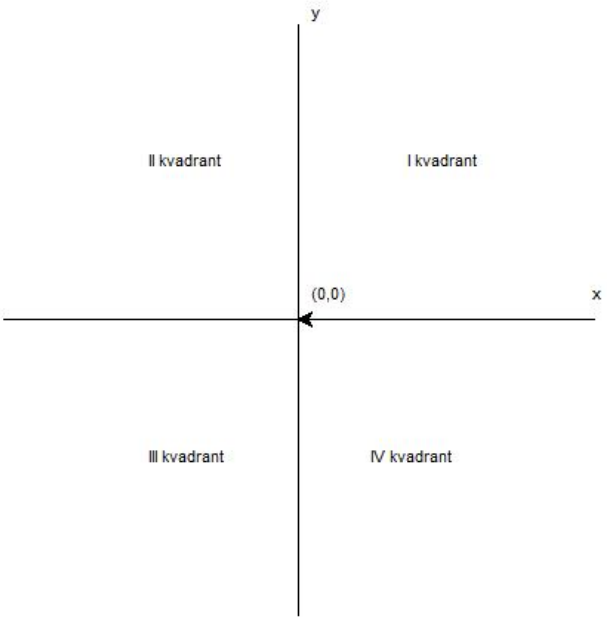
Debljina linije



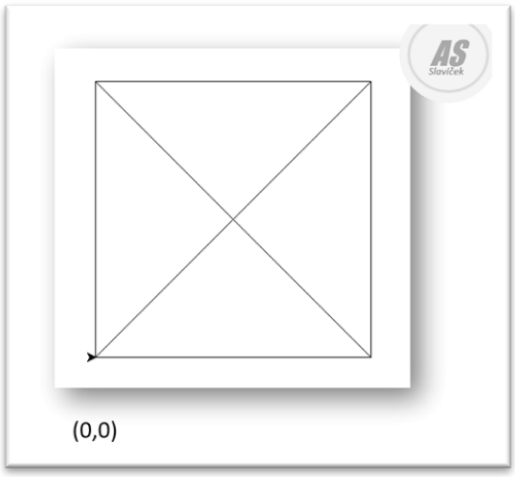
Unos podataka preko dijaloškog okvira



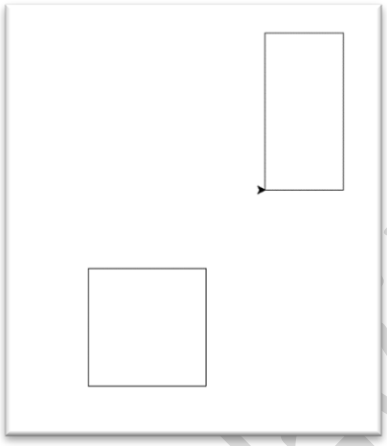
Zadatak: Nacrtaj osi pravokutnog koordinatnog sustava



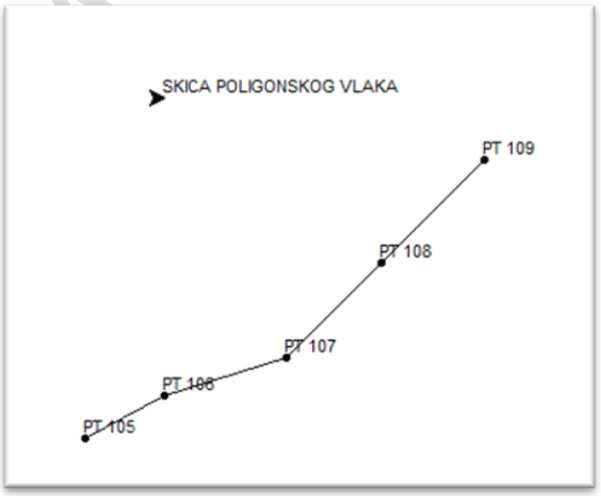
Zadatak: Nacrtaj kvadrat (a=320) i dijagonale kvadrata.



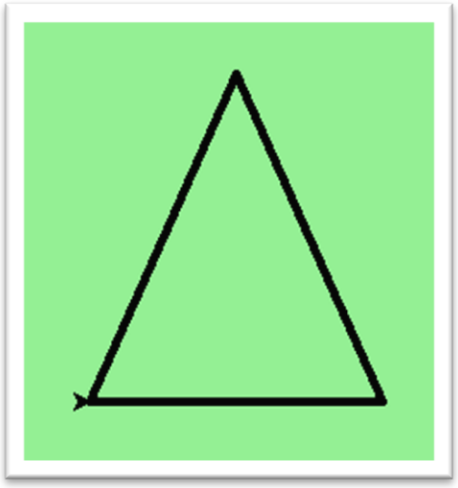
Zadatak: Nacrtaj kvadrat (a=150, početna koordinata je (25,25)) i pravokutnik (a=100, b=200, početna koordinata je (250,275).



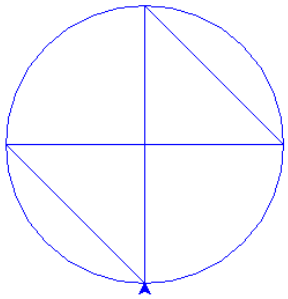
Zadatak: Nacrtaj skicu poligonskog vlaka (točka je veličine 5). Koordinate točaka: PT 105(25,35), PT 106(75,62), PT 107(152,86), PT 108(212,146) i PT 109(277,211).



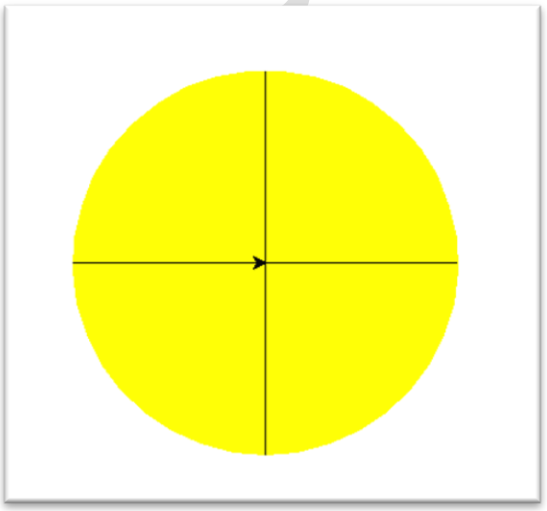
Zadatak: Nacrtaj trokut prema zadanim elementima: duljina osnovice = 170, visina trokuta = 190, debljina pera = 5, pozadina = svijetlozelena.



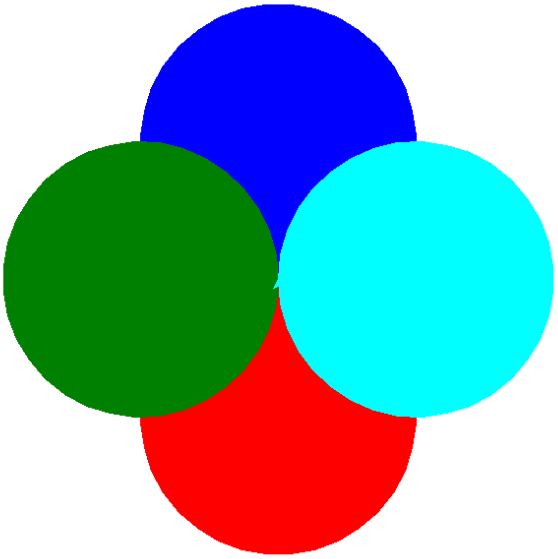
Zadatak: Nacrtaj kružnicu (r=120).



Zadatak: Nacrtaj kružnicu i oboji je kao na slici (r=150).

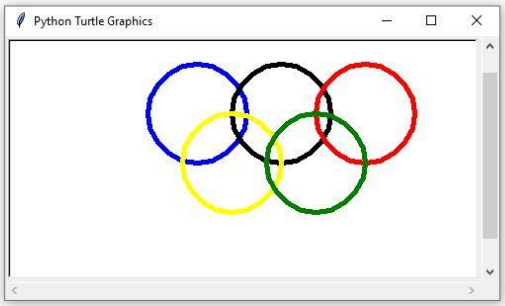


Zadatak: Nacrtaj četiri obojene kružnice polumjera (r=140) koje se dodiruju u jednoj točci.



Zadatak: Nacrtaj olimpijske krugove

```
#ASLavicek
# Zadatak: Nacrtaj olimpijske krugove
# Plavi krug predstavlja Europu, žuti Aziju,
# crni predstavlja Afriku, crveni Ameriku, a zeleni Australiju.
from turtle import *
r=50
pensize(5)
boja = ['blue','black','red','yellow','green']
pu()
goto(-50,0)
pd()
b=0
end = 3
for i in range(2):
        for j in range(end):
            color(boja[b])
            circle(r)
            b=b+1
            pu()
            fd(85)
            pd()
        pu()
        goto(-15,-50)
        pd()
        end = end-1
ht()                # hideturtle() - pero će biti nevidljivo
mainloop()
```



Literatura

- SRCE, Osnove programiranja (Python)
- Programski jezik Python - abc tutorijal - Znanje.org
- A. Maksimović: Python
- Sysprint, Udžbenik 7r
- Sysprint, Udžbenik 8r
- Sysprint, Informatika 1
- Z. Hercigonja, PROGRAMIRANJE U PYTHONU
- Z. Hercigonja, Python osnove programiranja
- Osnove programiranja u Pythonu
- CARNET, Priručnik "Programiranje"
- Python - w3schools
- Galešev, Sokol: Informatika i računalstvo - radna bilježnica
- Dlačić, Pilipović i dr: Računalno razmišljanje i programiranje
- IPAQ PETA, Dijaloški prozori