Vjerujemo da ste vrlo vješti u korištenju računala. Do sada ste na računalu igrali razne igre, crtali, pisali sastavke i pregledavali mreže stranice na internetu. Da bi sve to mogli raditi, netko je morao "naučiti računalo" kako se to izvodi. Zapravo ste koristili program za pisanje, program za crtanje, itd. Svaka igra koju ste igrali jedan je računalni program. Zanima li vas kako se programira računalna igra? A možete napisati i vlastiti program. Na primjer, napisati program kojim ćete pomoću računala rješavati matematičke zadatke ili pak crtati složene likove. U ovom priručniku naučit ćete osnove programiranja u programskom jeziku Python, verzija 3.3.

Znatna je razlika između 2.x i 3.x inačica programskog jezika Python, zato je potrebno istaknuti o kojoj se inačici radi.

1. Moj prvi algoritam

Koristeći programe *Bojanje* i *Word* crtali se crteže i oblikovali tekst. Ti programi napisani su u nekom programskom jeziku. I vi možete naučiti pisati programe koje će računalo izvršavati. Da bismo napisali program koji rješava neki problem (npr. zbraja dva broja ili ih uspoređuje), moramo znati opisati taj problem na način koji računalo razumije. Za opisivanje problema kojeg treba riješiti računalo služimo se algoritmom.

Algoritam je riječ koja potječe iz arapskog jezika, a znači *postupak, pravilo, uputa.*

U svakodnevnom životu zapravo izvodite različite algoritme, a da toga često niste niti svjesni. Algoritam je postupak kojim se opisuje točan redoslijed obavljanja nekog posla ili zadaće.

Evo primjera algoritma za slaganje školske torbe:

1. Uzmi raspored sati.
2. Stavi knjige i bilježnice za svaki predmet u torbu.
3. Stavi potreban pribor u torbu.
4. Provjeri još jednom raspored sati i uvjeri se je li sve u torbi.
5. Pripremi papuče i torbu pored vrata.

algoritam - postupak, pravilo, uputa

▶ **Algoritam** je postupak kojim se opisuje točan redoslijed kojim obavljamo neki posao**.**

Skoro svaki dan pišete zadaću pri čemu činite niz uobičajenih radnji i donosite određene odluke. Napišimo taj zamišljeni slijed u obliku algoritma:

Algoritam za pisanje zadaće iz matematike

početak
    potraži bilježnicu i udžbenik
    otvori bilježnicu i udžbenik
    pripremi našiljenu olovku i gumicu
    pronađi zadatke koji su za zadaću

pročitaj zadatke
ako nešto ne razumiješ onda
prouči iz bilježnice što ste učili u školi
pronađi u udžbeniku slične riješene zadatke
riješi zadane zadatke
provjeri na kraju udžbenika jesi li točno riješio zadatke
ako imaš pogrešaka **onda**
ispravi pogreške
ako ne znaš ispraviti **onda** sutra pitaj učiteljicu
inače
zadaća je gotova
kraj

Algoritam sastavljamo od točno određenih riječi pomoću kojih kasnije pišemo program. Napisani program sadrži **naredbe** koje su razumljive računalu.

▶ **Naredba** je uputa računalu da obavi određenu operaciju.

Kao primjer, evo nekoliko naredbi napisanih u obliku koji razumije računalo:

```
upiši x
ispiši x
y=x+5
ako je y>x onda ispiši x inače ispiši y
```

Algoritam služi da detaljno opišemo problem koji računalo treba riješiti. Da bismo pisati svrhovite programe, moramo točno predvidjeti što želimo da računalo čini.

program, programiranje

▶ **Program** je konačan niz naredbi razumljivih računalu koje rješavaju neki problem. Postupak pisanja programa zovemo **programiranje.**

Program pišemo u nekom **programskom jeziku**. Kao što mi međusobno razgovaramo hrvatskim jezikom, tako računala međusobno "razgovaraju" programskim jezicima.

programski jezik

▶ **Programski jezik** je skup naredbi i pravila za njihovo pisanje razumljivih računalu.

programski jezik **Python**

Programski jezik ima mnogo manje naredbi i mnogo jednostavnija pravila za njihovo korištenje nego bilo koji govorni jezik. U ovom priručniku učit ćete programski jezik **Python**.

Da biste mogli napisati dobar program, važno je prvo sastaviti učinkovit algoritam, a iz algoritma je tada lagano napisati program u Pythonu.

Vježba 1.

Napišimo algoritam za zbrajanje dva broja i opišimo ga riječima.

algoritam:

upiši a

upiši b

zbroj = a+b

ispiši zbroj

opis:

1. zadajemo jedan broj koji će računalo zapamtiti u memorijskom mjestu nazvanom a.
2. zadajemo drugi broj, a računalo ga zapamti u memorijskom mjestu nazvanom b.
3. program računa zbroj i sprema ga u memorijsko mjesto nazvano zbroj.
4. na zaslonu će se ispisati broj koji je spremljen u mjestu zbroj.

Što će algoritam izračunati ako je prvi broj 10, a drugi broj 5?

Algoritam u memorijsko mjesto *a* stavlja 10, a u memorijsko mjesto *b* stavlja 5. Onda se u memorijsko mjesto *zbroj* stavlja 15 (rezultat zbrajanja 10 i 5), a na zaslonu se ispiše 15.

⦿ Računalo podatke pamti na određenom mjestu u svojem spremniku. To mjesto zovemo **memorijsko mjesto** ili **memorijska lokacija** (od engl. *memory location*).

Što će algoritam izračunati ako je prvi broj 365, a drugi broj 956?

Algoritam u memorijsko mjesto *a* stavlja 365, a u memorijsko mjesto *b* stavlja 956. Onda se u memorijsko mjesto *zbroj* stavlja 1 321, a na zaslonu se ispiše 1 321.

Vidimo da ovaj algoritam može zbrojiti bilo koja dva broja.

2. Od algoritma do dijagrama toka

dijagram

Algoritam se može prikazati i crtežom, odnosno grafički. Grafički prikazan algoritam nazivamo **dijagram toka**. Pri tome geometrijski likovi predstavljaju određenu vrstu naredbe.

▶ **Dijagram toka** je grafički prikaz algoritma.

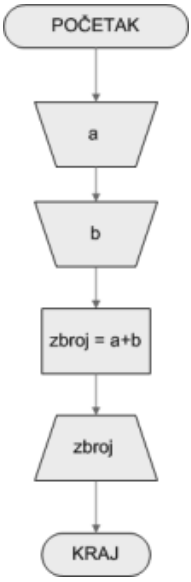
Vježba 2.

Nacrtajmo dijagram toka za zbrajanje dva broja:

Geometrijski oblici u dijagramu toka ovise o vrsti naredbe koju prikazuju. Tako prikazan algoritam vrlo je pregledan i potpuno određen. Posebno je pogodan za analizu programa, traženje sličnih rješenja ili potrebne izmjene.

Ovalni oblik služi za oznaku početka ili kraja programa.

dijagram toka:



Četverokut u obliku lijevka koristi se za ulaz ili za izlaz podataka.

trapez – četverokut u obliku lijevka

U pravokutnik pišemo naredbe kojima ćemo npr. nešto izračunati.

Postupak programiranja sastoji se od tri osnovna koraka:

Za problem koji želimo riješiti pomoću računala najvažnije je pronaći dobar algoritam. Pod pojmom "dobar" mislimo na brz i učinkovit algoritam.

Vježba 3.

Napišite algoritam i nacrtajte dijagram toka za računanje opsega i površine školskog igrališta.

Rješenje:

Školsko igralište je pravokutnik. Ako zadamo duljinu i širinu igrališta, lako ćemo izračunati opseg i površinu igrališta.

algoritam:

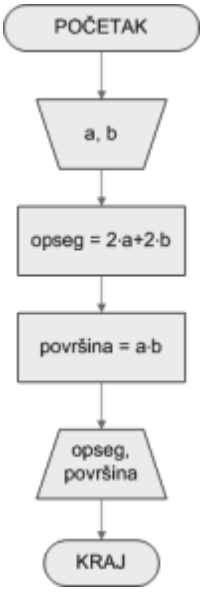
upiši a,b

opseg = $2 \cdot a+2 \cdot b$

površina=$a \cdot b$

ispiši opseg, površina

dijagram toka:



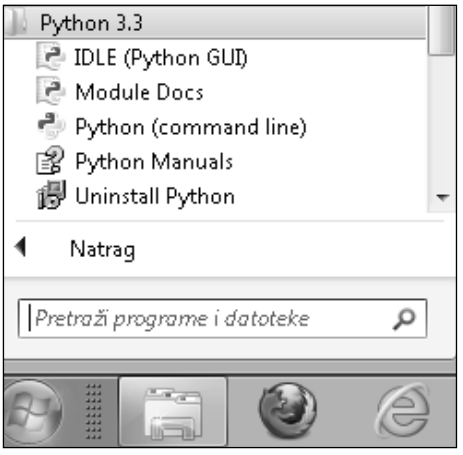
3. Upoznajmo Python

Već smo kazali da je programiranje zapisivanje algoritma u nekom od programskih jezika. U ovoj priručniku to ćete naučiti pomoću programskog jezika Python. Prije toga, pokažimo kako se pokreće Python, odnosno sučelje u kojem ćete pisati Python programe.

1. Pokretanje Pythona

Python Shell – glavni prozor IDLE sučelja

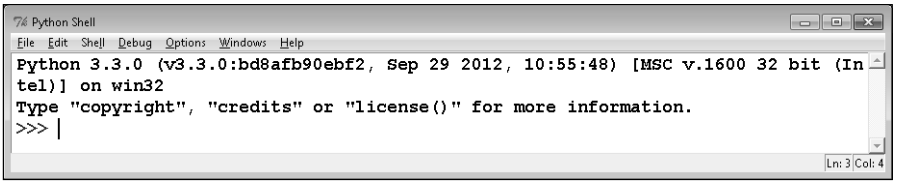
Kliknite na Windowsov gumb *Start*, odaberite *Svi programi* i u mapi **Python 3.3** kliknite na **IDLE (Python GUI).**



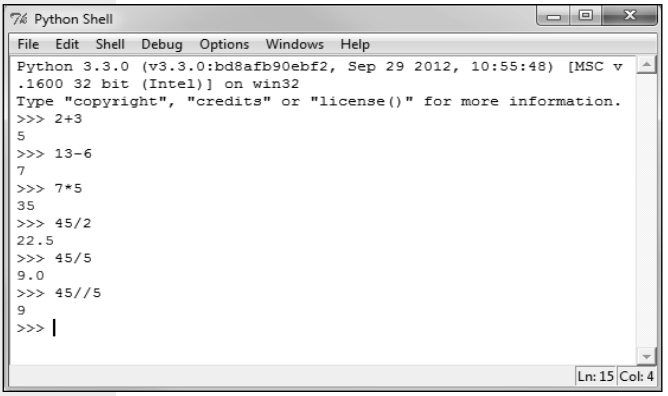
Slika 1. Pokretanje Pythona.

> Zašto Python pokrećemo klikom na *IDLE (Python GUI)*, a ne *Python (command line)*? **IDLE** je kratica od *Integrated DeveLopment Environment* (što znači *integrirano razvojno okruženje*), a **GUI** je kratica od *Graphics User Interface*, što znači *grafičko korisničko sučelje*. Za razliku od toga, *Python (command line)* nema grafičko sučelje pa je rad u njemu znatno zahtjevniji.

Nakon što se otvori ***Python Shell***, glavni prozor Pythonova sučelja (slika 2.), pokazivač teksta postavljen je iza znaka **>>>**. Tu možemo odmah pisati naredbe koje će Python izvršiti.



Slika 2. Glavni prozor Pythona – Python Shell.



Slika 3. Osnovne računske operacije.

2. Osnove mogućnosti

U prozoru *Python Shell* možemo izvoditi računske operacije i jednostavne programske naredbe.

Primijetite da znak za dijeljenje **/** uvijek daje rezultat u obliku decimalnog broja. Iako je 45 : 5 = 9, Pyhon ispisuje 45/5=9.0, dakle kao decimalni broj s jednom decimalom (u

našem primjeru 0). Želimo li odrediti djelomični količnik moramo pisati **//**. Međutim, rezultat će biti cijeli broj samo ako dijelimo cijele brojeve Ako su djeljenik ili djelitelj decimalan broj i rezultat će biti decimalan broj. Pogledajte na primjeru:

/ - dijeljenje
// - djelomični količnik

```
>>> 13//2
6
>>> 13.5//2
6.0
>>> 13//2.1
6.0
```

U tablici 1. navedeni su znakovi koji se u Pythonu koriste za pojedine računske operacije.

računska operacija	znak	primjer	
		operacija	rezultat
zbrajanje	+	2 + 3	5
oduzimanje	-	13 - 6	7
množenje	*	7 * 5	35
dijeljenje	/	45 / 2	22.5
		45 / 5	9.0
djelomični količnik	//	45 // 5	9
		14 // 3	4
		14.2 // 3.5	4.0
		14.2 // 3.7	3.0
ostatak pri dijeljenju	%	14 % 3	2

Tablica 1. Znakovi za računske operacije u Pythonu.

Osim računanja, u Pythonu možemo i ispisivati tekst.

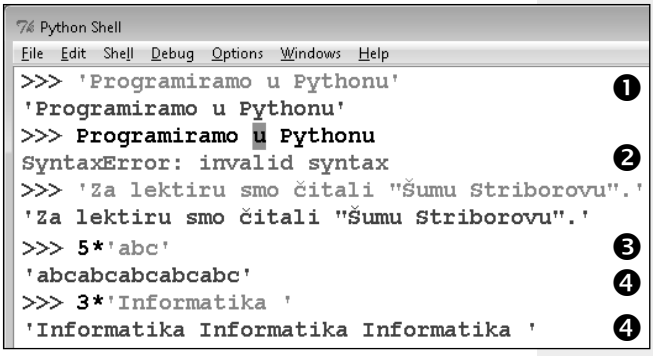
Objasnimo osnovna pravila za pisanje teksta (slijedite numeričke oznake na slici 4.):

❶ Tekst koji želimo ispisati pišemo pod jednostrukim navodnicima ‘’.

❷ Ako ih zaboravimo Python će javiti poruku o greški.

❸ Želimo li pisati tekst pod navodnicima, možemo ih koristiti unutar jednostrukih navodnika.

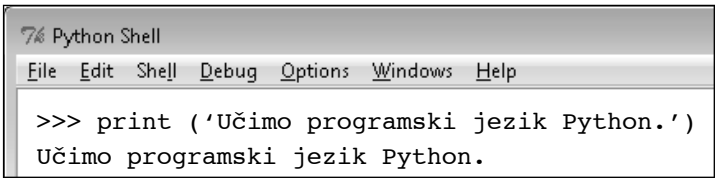
❹ Tekst možemo umnožiti pomoću znaka *. Želimo li razmak između riječi, moramo ostaviti jedan razmak na kraju teksta koji umnažamo.



Slika 4. Ispis teksta u Pythonu.

Vježba 4.

Upišite sljedeću naredbu:



Naredba **print**

⊙ Radi jednostavnosti i preglednosti, programske naredbe koje treba upisivati u prozoru Python Shell u ovom ćemo priručniku prikazivati u okviru s naslovom SHELL.

SHELL

```
>>> print (‘Učimo programski jezik Python.’)
Učimo programski jezik Python.
```

Rješenje:

Tekst koji želite ispisati treba napisati pod jednostrukim navodnicima, u okrugloj zagradi i ispred zagrade napisati naredbu `print`.

Rješenje u datoteci **vje4.py**.

▶ Naredba `print` služi za ispis teksta. Ono što želimo ispisati navedemo u okruglim zagradama pod jednostrukim navodnicima.

Vježba 5.

Kako ispisati navodnike unutar navodnika? Upišite sljedeću naredbu:

SHELL

```
>>> print ('Čitali smo "Šumu Striborovu" Ivane
↪Brlić Mažuranić.')
Čitali smo "Šumu Striborovu" Ivane Brlić Mažuranić.
```

pritisni tipku **Enter**

Rješenje:

Navodnici unutar navodnika

Želimo li ispisati tekst pod navodnicima, kao što je naslov bajke, navodnike navodimo unutar teksta. Čitav tekst koji ispisujemo ostaje pod jednostrukim navodnicima.

Rješenje u datoteci **vje5.py**.

Vježba 6.

Upišite sljedeću naredbu:

SHELL

```
>>> print ('Tekst možemo \n ispisivati u \n više
↪redaka.')
Tekst možemo
 ispisivati u
 više redaka.
```

pritisni tipku **Enter**

Rješenje:

\n – prelazak u novi redak

Želimo li neki tekst ispisati u više redova, za prelazak u novi redak pišemo **\n.**

Rješenje u datoteci **vje6.py**.

Zadaci:

1. Što će ispisati ove naredbe:

a) `>>> 10*'*'`
b) `>>> 3*'MIR '`
c) `>>> 4*'ljubav'`
d) `>>> 5*'123'`
e) `>>> 5*123`

2. Izračunaj pomoću Pythona:

a) 1234 + 5678
b) 1 000 000 – 345 678
c) 3456 · 345
d) 7006652 : 1234
e) 346 · 345 + 344 · 356
f) 746 · 45 - 244 · 356

3. Odredi djelomični količnik i ostatak sljedećih brojeva:

a) 2345 i 456
b) 12345 i 789
c) najvećeg šteroznamenkastog broja s različitim znamenkama i najmanjeg troznamenkastog broja s različitim znamenkama
d) najmanjeg šteroznamenkastog broja s različitim znamenkama i najvećeg troznamenkastog broja s različitim znamenkama

4. Što će ispisati sljedeće naredbe:

```
>>> print ('Python')
>>> print (' Čitam knjigu "Družba Pere Kvržice". ')
>>> print ('2+3=',2+3)
>>> print ('14-7=',14-7)
>>> print ('12:7=',12//7,'(',12%7,')')
>>> print ('14:3=',14//3,'(',14%3,')')
```

5. Napiši program koji ispisuje sljedeći tekst:

a) Zagreb je glavni grad Republike Hrvatske.
b) Hrvatska ima 21 županiju.
c) Ja živim u …. županiji.(Dopuni rečenicu nazivom tvoje županije.)
d) Hrvatska himna je "Lijepa naša domovina".
e) Upravo čitam knjigu " ….".(Dopuni naziv knjige koju čitaš.)
f) Najveći hrvatski gradovi:

   Zagreb
   Split
   Rijeka
   Osijek

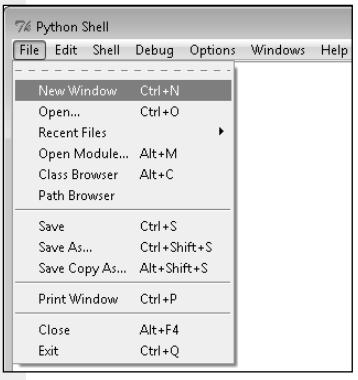
   (Uputa: koristi ispis u više redova).
g) Ispiši nazive pet predmeta koje najviše voliš, a naziv svakog predmeta u svojem redu.

4. Pisanje i izvršavanje programa

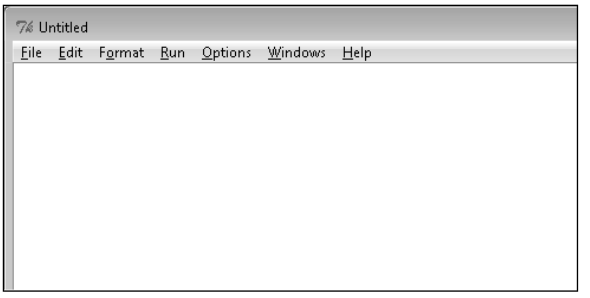
Do sada ste naučili ispisivati tekst i izvoditi osnovne računske operacije. Međutim, naredbe koje ste pisali niste mogli spremiti i kasnije ponovno koristiti. Kao što svoje crteže i tekstove na računalu možete spremiti i kasnije ih nastaviti uređivati, tako i naredbe u Pythonu možete spremiti ako ih napišete u obliku programa.

Uređivač teksta (programa)

Program pišemo u **uređivaču teksta** koji otvaramo tako da iz izbornika *File* odaberemo naredbu *New Window* (slika 5.). Nakon toga otvori se prozor u kojem pišemo program (slika 6.).



Slika 5. Otvaranje uređivača teksta.

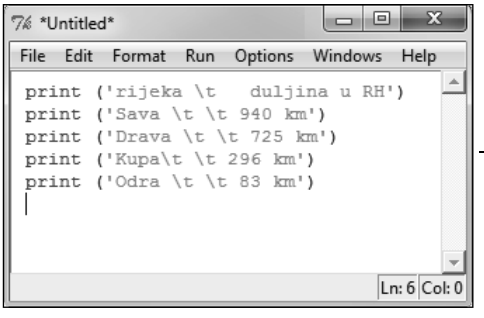


Slika 6. Uređivač teksta.

Vježba 7.

Napišite sljedeći program u uređivaču teksta:

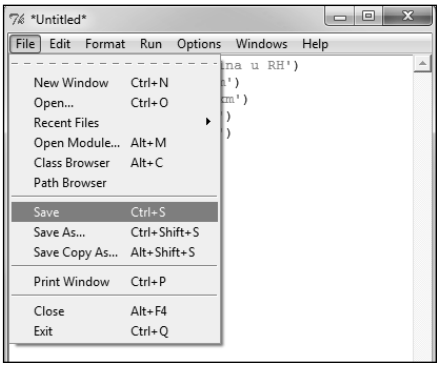
Radi jednostavnosti i preglednosti, programske naredbe koje treba upisivati u uređivaču teksta u ovom priručniku ispisivat ćemo u okviru sa sivim rubom.



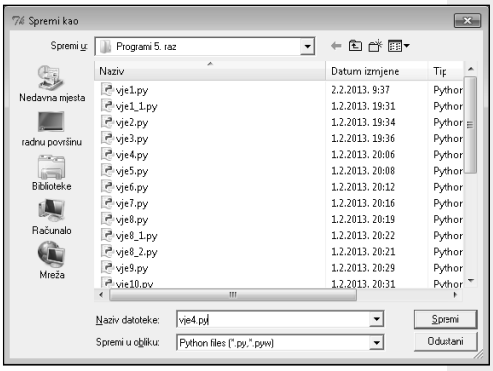

```
print (‘rijeka \t   duljina u RH’)
print (‘Sava \t \t 940 km’)
print (‘Drava \t \t 725 km’)
print (‘Kupa\t \t 296 km’)
print (‘Odra \t \t 83 km’)
```

Nakon što ste prepisali program, program treba izvršiti. Python se brine da ne izgubite svoj uradak, pa zahtjeva da program prvo spremite. Zato iz izbornika *File* odaberite ***Save*** (ako spremate prvi puta ili je već od ranije upisano ime programa).

Želite li promijeniti ime programa ili mjesto njegova spremanja odabrat ćete ***Save As*** *ili* ***Save Copy As***.



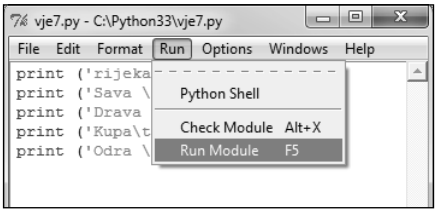
Slika 7. Spremanje programa.



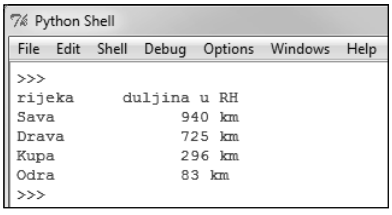
Slika 8. U polje Naziv datoteke upišite ime programa.

Sada program možete izvršiti. Iz izbornika *Run* odaberite ***Run Module*** ili jednostavno pritisnite tipku **F5**.

Run Module ili **F5** – izvršavanje programa



Slika 9. Izvršavanje programa.

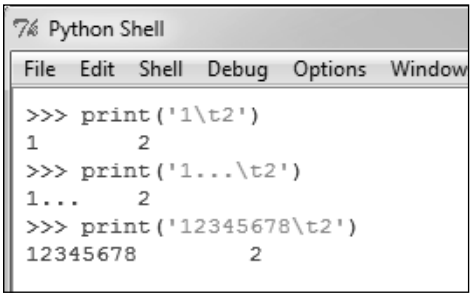


Slika 10. Rezultat programa.

Pišući program sigurno ste primijetili neobičnu oznaku **\t**. Ona omogućuje preskok za više praznih mjesta, a sljedeći znak ispisuje se iza bloka praznih mjesta. Budući da smo kod ispisa duljine rijeka trebali više razmaka, oznaku \t upotrijebili smo dvaput.

\t – naredba za tabulator, tj. preskok više praznih mjesta

Nakon oznake **\t** Python počinje ispisivati s osam mjesta razmaka, ali brojeći od prvog ispisanog znaka. Promotrite primjere na slici 11:



Slika 11. Kako radi `\t` oznaka za oblikovanje teksta.

U prvom primjeru vidi se da blok ima 8 mjesta razmaka, jer prethodni tekst ima samo jedan znak Ali, kad ispišemo tekst od 7 znakova (kao u drugom primjeru na slici 11.) ostaje samo 1 prazno mjesto i sljedeći znak ispisuje se na početku novog "bloka od 8 mjesta". I napokon, ako ispišemo tekst koji ima 8 znakova, pa stavimo oznaku `\t`, sada će Python preskočiti cijeli blok od 8

mjesta i tekst koji slijedi ispisati na početku novog bloka. Dakle, oznakom `\t` zadajemo da se tekst koji slijedi ispiše na početku sljedećeg bloka od 8 mjesta.

Rješenje u datoteci **vje7.py**.

> ▶ **Ispis podataka**
>
> Za ispis koristimo naredbu **print()**. U okruglim zagradama slijede podaci koje treba ispisati i to:
>
> • tekst pišemo pod jednostrukim navodnicima
>
> • želimo li navoditi tekst koristimo se dvostrukim navodnicima
>
> • za prelazak u novi red koristimo **\n**
>
> • želimo li ostaviti više mjesta razmaka (kao tabulatorom) koristimo **\t**

Vježba 8.

Napišimo program za zbrajanje brojeva 80 i 30. Pri tom ćemo memorijsko mjesto u koje stavljamo 80 označiti s *a*, a memorijsko mjesto u koje stavljamo 30 označiti s *b*. Oznake za memorijska mjesta u programiranju zovemo **varijable**.

Rješenje:

Algoritam:

a=80

b=30

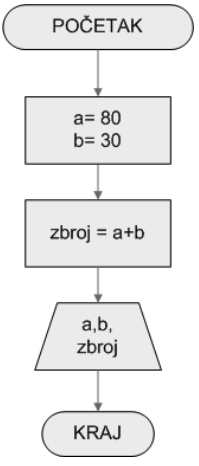
zbroj = a +b

ispiši a

ispiši b

ispiši zbroj

Dijagram toka:



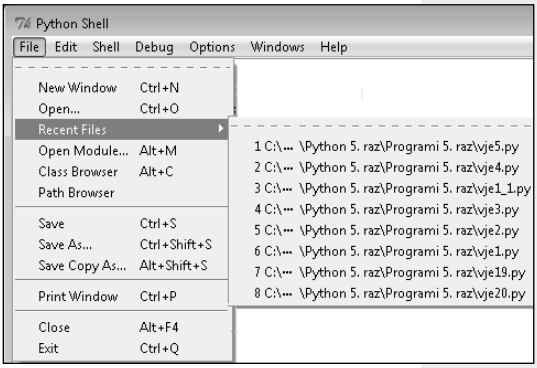
Program:

```
a=80
b=30
zbroj=a+b
print ('a=',a)
print ('b=',b)
print ('zbroj=',zbroj)
```

Prve naredbe programa (`a=80` i `b=30`) zovemo ***naredbe pridruživanja***. Ako u varijablu *a* želimo staviti vrijednost 80, tada jednostavno pišemo `a=80`. Znak jednakosti sada je ***operator pridruživanja***, jer varijabli *a* pridružuje broj 80.

> ▶ Naredba oblika `a=80` zove se **naredba pridruživanja**. Njome kažemo programu da u varijabli *a* zapamti broj 80. Varijabla *a* označava mjesto u memoriji u kojem je pohranjen broj 80.

Želimo li ovim programom zbrojiti neka druga dva broja, moramo se vratiti u uređivač teksta i izmijeniti brojeve pridružene varijablama *a* i *b*. To ćemo najlakše učiniti tako da iz izbornika *File* odaberemo ***Recent Files***.



Slika 12. Otvaranje ranije napisanih programa.

Izmijenite sada brojeve tako da bude *a* = 180 i *b* = 100. Ne zaboravite da program prije pokretanja morate prvo spremiti (*File* > *Save*), pa onda pokrenuti (*Run* > *Run Module* ili tipka **F5**.)

Rješenje u datoteci **vje8.py**.

Želite li biti brži u spremanju i pokretanju programa prilikom njihovog mijenjanja, možete to učiniti pomoću tipkovnice kombinacijom tipki **CTRL+S** za spremanje programa i **F5** za pokretanje.

Vježba 9.

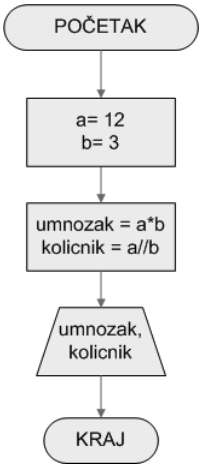
Napišimo program koji računa umnožak i količnik brojeva *a* = 12 i *b* = 3. Rezultat ispisati u obliku jednakosti: 12 * 3 = 36 i 12 : 3 = 4.

Rješenje:

Algoritam:

a=12
b=3
umnozak = a*b
kolicnik = a // b
ispiši umnozak
ispiši kolicnik

Dijagram toka:



Program:

```
a=12
b=3
umnozak=a*b
kolicnik=a//b
print (a,'*',b,'=',umnozak)
print (a,':',b,'=',kolicnik)
```

Mali rječnik

File – datoteka
Open – otvori
Save – spremi
Save As – spremi kao
Save Copy As – spremi kao
Recent Files – prethodne datoteke

Promotrite `print` naredbu. One podatke koje smo ispisivali iz varijabli *a*, *b*, *umnozak* i *kolicnik* napisali smo bez jednostrukih navodnika, a znakove računskih operacija (koje program ovdje vidi kao tekst) pod jednostrukim navodnicima.

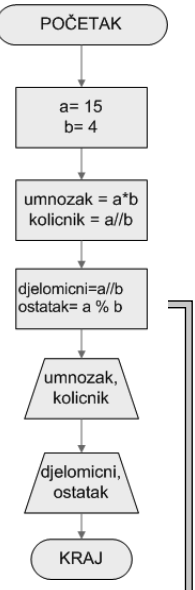
Rješenje u datoteci **vje9.py**.

***** množenje
/ dijeljenje
// djelomični količnik
% ostatak

Vježba 10.

Napišimo program koji računa umnožak i količnik brojeva *a* = 15 i *b* = 4. Količnik ispisati na dva načina: 15 : 4 = 3.75 i 15 : 4 = 3 (3)

Dijagram toka:



Rješenje:

Algoritam:

a=15
b=4
umnozak = a*b
kolicnik = a / b
djelomicni=a//b
ostatak= a % b
ispiši umnozak
ispiši kolicnik
ispiši djelomicni, ostatak

Program:

```
a=15
b=4
umnozak=a*b
kolicnik=a/b
djelomicni=a//b
ostatak= a % b
print (a,'*',b,'=',umnozak)
print (a,':',b,'=',kolicnik)
print (a,':',b,'=',djelomicni,'(',ostatak,')')
```

Rješenje u datoteci **vje10.py**.

Zadaci:

6. Što će ispisati ovi programi:

a)
```
print('Voće \n \n i \n \n povrće')
```

```
print('Voće \t \t i \t \t povrće')
```
b)

c)
```
print(4*'MIR', '\t', 6*'rad')
```

```
print(4*'MIR \t rad ')
```
d)

e)
```
print(6*'Volimo učiti.\n')
```

f)
```
print(4*'MIR \t rad \n')
```

7. Napiši program koji pomoću jedne `print` naredbe ispisuje:

a)
```
SHELL
abeceda abeceda abeceda abeceda
```

b)
```
SHELL
crvena      bijela      plava
```

c)
```
SHELL
crvena
    bijela
        plava
```

d)
```
SHELL
Učim.
Učim.
Učim.
Učim.
```

e)
```
SHELL
Čitam   i   pišem.
Čitam   i   pišem.
Čitam   i   pišem.
Čitam   i   pišem.
Čitam   i   pišem.
```

8. Prepiši u bilježnicu pa dopuni algoritam, dijagram toka i program za oduzimanje brojeva *a*=80 i *b*= 30.

Algoritam:

___=80

b=___

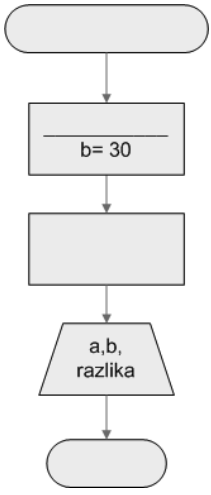
_____ = a ___b

ispiši a

ispiši _____

ispiši _____

Dijagram toka:



Program:

```
_____________
_____________
razlika = ___ - ___
print('a=',_____)
_______('b=',b)
print('razlika=',_______)
```

Koje naredbe treba izmijeniti u programu ako treba odrediti razliku brojeva *a* i *b* kao u tablici? Zatim izmijeni i program.

a	**b**	**Naredbe koje treba izmijeniti**
120	50	`a=120` `b=50`
1236	456	
4587	2364	
5987	3587	
7745	2655	

5. Upis podataka

Naučili ste pisati programe koji računaju s brojevima čije vrijednosti zadajemo naredbom pridruživanja. Međutim, kako napisati program koji će zbrojiti bilo koja dva broja? Da bi to bilo moguće, brojeve koje želimo zbrojiti treba upisati pomoću tipkovnice tijekom izvršavanja programa.

1. Naredba INPUT

Za upis podatka s tipkovnice programski jezik Python rabi naredbu `input`. Da bi se upisana vrijednost učitala u računalo morate pritisnuti tipku *Enter*.

Vježba 11.

Napišite program za izračunavanje zbroja dva broja i ispisivanje dobivenog zbroja.

Rješenje:

Algoritam:

upiši a

upiši b

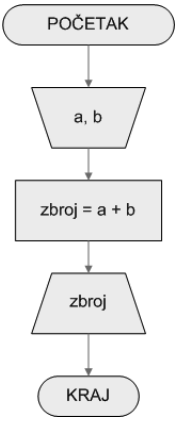
zbroj=a+b

ispiši zbroj

Program:

```
print ('Upiši prvi pribrojnik:')
a=input()
print ('Upiši drugi pribrojnik:')
b=input()
zbroj=a+b
print ('Zbroj je', zbroj)
```

Dijagram toka:

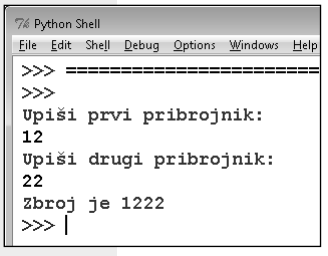


`input` - naredba za upis podataka

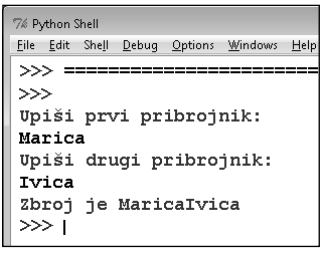
Naredba `a=input()` omogućuje da nakon pokretanja programa upišete neki broj koji se pamti u varijabli *a*. Prednost ovog načina je što ne morate mijenjati program ako želite računati s različitim brojevima. Kada program naiđe na naredbu `input` stat će i čekati podatak s tipkovnice sve dok ga ne upišete i pritisnete tipku *Enter*.

Da bi korisnik znao što se od njega očekuje, uvijek je dobro ispisati poruku o tome što treba upisati, odnosno kakav podatak program očekuje. Zato smo u ovom primjeru ispred naredbe `input` napisali naredbu `print` koja ispisuje poruku o tome što treba upisati.

Zbraja li program točno dva broja? Pokrenimo ga (**F5**) pa pogledajmo:



Slika 13. Izvršenje programa za zbrajanje za a=12 i b= 22.



Slika 14. Izvršenje programa za zbrajanje za a='Ivica' i b='Marica'.

Znamo da 12+22 nije 1222. Možete li se dosjetiti što je program učinio? Pogledajte sliku 14. Umjesto brojeva upisali smo riječi *Marica, Ivica*. Program ih je jednostavno "slijepio", odnosno združio. Sada je jasno da upravo to radi i s brojevima: ne zbraja ih, već ih sljepljuje jer ih prihvaća kao riječi, a ne kao brojeve.

Da bismo riješili opisani problem, prije zbrajanja moramo od riječi napraviti broj. To radi naredba `int`. Najlakše je samo izmijeniti naredbu za zbrajanje:

`zbroj=int(a)+int(b).`

```
print ('Upiši prvi pribrojnik:')
a=input()
print ('Upiši drugi pribrojnik:')
b=input()
zbroj=int(a)+int(b)
print ('Zbroj je', zbroj)
```

`int` – pretvaranje riječi u broj

Program možemo još malo poboljšati tako da poruku o tome koji podatak upisujemo napišemo u `input` naredbi.

```
a=input('Upiši prvi pribrojnik:')
b=input('Upiši drugi pribrojnik:')
zbroj=int(a)+int(b)
print ('Zbroj je', zbroj)
```

Rješenja u datotekama **vje11.py**, **vje11a.py**, **vje11b.py**

▶ Naredba za upis podataka je `input`. Najčešće je koristimo u obliku `x=input('poruka')`, gdje je **x** mjesto u memoriji na kojem pamtimo podatak, a `poruka` opis podatka koji učitavamo.

Naredba `input` učitani podatak shvaća kao tekst, pa ga prije računanja moramo pretvoriti u broj.

Osnovna pravila za imenovanje varijabli:

1. Ime varijable sastoji se od slova, znamenki i podvlake _
2. Python razlikuje velika i mala slova. Dakle, **zbroj**, **Zbroj** i **ZBROJ** označavaju tri različite varijable.
3. Primjeri nedozvoljenih imena:
   a) a$, x.7, 7€ - sadrže nedozvoljene znakove
   b) 7as, 90a – počinju znamenkom
   c) if, else, and, or – ključne riječi koje imaju već svoju namjenu u Pythonu

Vježba 12.

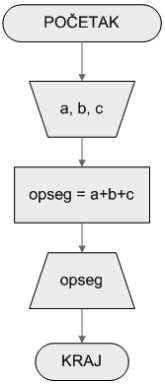
Napišite program za računanje opsega trokuta kome su zadane duljine stranica trokuta *a*, *b* i *c*.

Rješenje:

Algoritam:

upiši a
upiši b
upiši c
opseg=a+b+c
ispiši opseg

Dijagram toka:



Program:

```
a=input('a=')
b=input('b=')
c=input('c=')
opseg=int(a)+int(b)+int(c)
print ('Opseg je', opseg)
```

Rješenje u datoteci **vje12.py**.

Vježba 13.

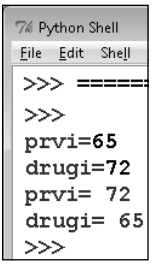
Često puta se u programiranju pojavljuje potreba zamjene vrijednosti dvije varijable. Ako imamo niz brojeva koje treba poredati po veličini tada moramo moći zamijeniti vrijednosti dviju varijabli. Napišite program koji upisuje dva broja *prvi* i *drugi* i zamjenjuje njihove vrijednosti.

Rješenje:

Promotrite algoritam i program:

upiši prvi
upiši drugi
prvi, drugi= **drugi**, **prvi**
ispiši prvi
ispiši drugi

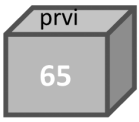
```
prvi=input('prvi=')
drugi=input('drugi=')
prvi,drugi=drugi,prvi
print ('prvi=',prvi)
print ('drugi=',drugi)
```



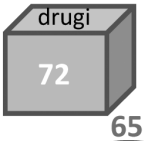
Slika 15. Izvršenje programa u vježbi 13.

Prikažimo što se događa u memoriji:

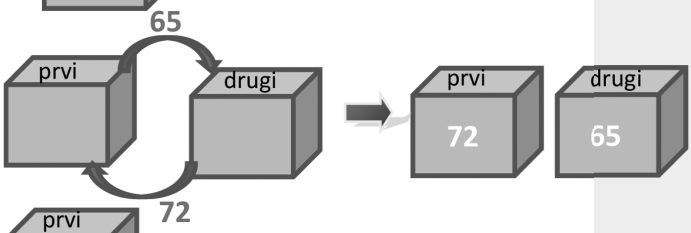
```
prvi=input('prvi=')
```



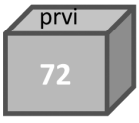

```
drugi=input('drugi=')
```



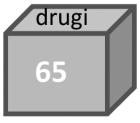

```
prvi,drugi=drugi,prvi
```




```
print ('prvi=',prvi)
```



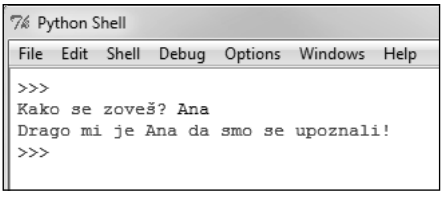

```
print ('drugi=',drugi)
```



Rješenje u datoteci **vje13.py.**

Vježba 14.

Napišimo program koji traži da upišete svoje ime a zatim vas pozdravi.



Slika 16. Primjer izvršenja programa za pozdravljanje.

Rješenje:

Algoritam:

upiši ime

ispiši *Drago mi je* ime *da smo se upoznali!*

Program:

```
ime=input('Kako se zoveš? ')
print ('Drago mi je',ime,'da smo se upoznali!')
```

Rješenje u datoteci **vje14.py.**

2. Decimalni brojevi

Python raspolaže naredbama koje možemo koristiti odmah nakon njegovog pokretanja. Pored njih, postoje i skupine naredbi koje prije korištenja moramo učitati, odnosno unijeti u Python. Na taj način proširujemo mogućnosti Pythona.

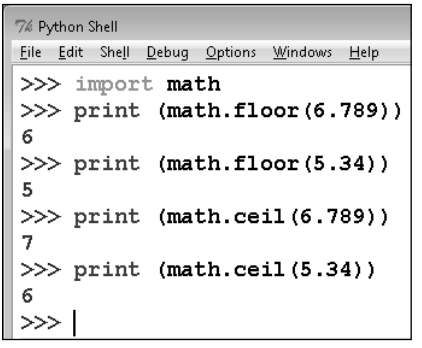
Modul – skupina programa određene namjene

Takve skupine programa zovu se **moduli**. Svaki modul ima svoje ime. Na primjer, želimo li izvršavati matematičke operacije unijet ćemo jedan modul, a želimo li crtati, unijeti ćemo drugi modul.

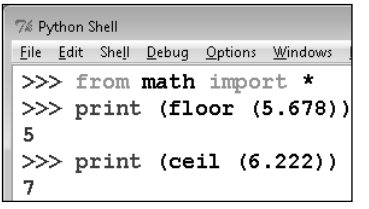
import – naredba za unošenje modula

Module unosimo u glavni program pomoću naredbe **import**.

Promotrite sljedeće slike i uočite kako se modul učitava u Python. Što rade naredbe **math.floor** i **math.ceil**?



Slika 17. Modul **math**.

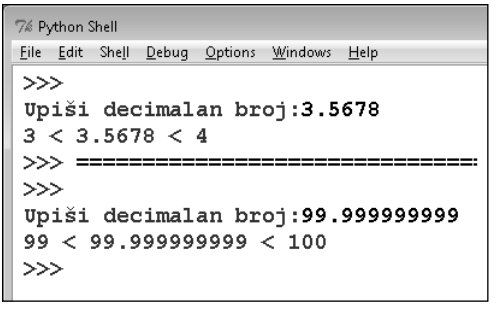


Slika 18. Modul **math** učitan na drugi način.

Sigurno ste zaključili da naredba `math.floor` daje najbliži manji cijeli broj od zadanog, a naredba `math.ceil` najbliži veći cijeli broj od zadanog. Ako vam se naredbe na slici 17. čine predugačke, možete izmijeniti način učitavanja modula (`from math import *` kao na slici 18.) pa onda više nećete morati navoditi modul iz kojeg je naredba.

Vježba 15.

Napišite program koji upisuje decimalan broj i ispisuje između kojih cijelih brojeva se on nalazi (primjer izvođenja programa prikazan je na slici).



Rješenje:

Promotrite algoritam i program:

Algoritam:

unesi *math*

upiši a

a= decimalni(a)

manji = pod(a)

veci= strop(a)

ispiši manji < a < veci

Program:

```
from math import *
a=input ('Upiši decimalan broj:')
a=float(a)
manji=floor (a)
veci=ceil (a)
print (manji,'<',a,'<',veci)
```

Ovdje osim upotrebe naredbi `floor` i `ceil` koje smo već objasnili, treba paziti da se ulazni podatak *a* iz teksta pretvori u decimalan broj. Upravo tome služi naredba `float`.

Mali rječnik:
floor – pod
ceil*ing* – strop

▶ Modul je skupina naredbi i programa koji imaju zajednička svojstva i namjenu. Modul unosimo naredbom **import**.

Naredba oblika **from math import *** omogućuje dodatne naredbe za rad s brojevima.

Neke naredbe modula **math** su:

`floor (a)` – daje nabliži manji prirodan broj zadanog broja *a*

`ceil (a)` - daje nabliži veći prirodan broj zadanog broja *a*

Rješenje u datoteci **vje15.py**.

Oblikovanje ispisa decimalnih brojeva

Postoje razlomci koji se mogu prikazati u obliku konačnog decimalnog broja, npr. $\frac{1}{2}, \frac{3}{50}$, …, ali postoje razlomci čiji je zapis beskonačan, npr. $\frac{2}{3}, \frac{1}{7}$, … Python će ispisati te razlomke sa točnosti od 16 decimalnih mjesta. Želimo li smanjiti broj decimalnih mjesta ispis oblikujemo metodom `.format` unutar `print` naredbe. Pogledajmo ispis razlomka $\frac{1}{7}$ s različitim brojem decimalnih mjesta:

```
>>> print ('{0:13.10f}'.format (1/7))
 0.1428571429
```

Razlomak $\frac{1}{7}$ napisan u obliku decimalnog broja je beskonačan periodičan decimalan broj s periodom od 6 decimala: "142857". U ovom `print` naredbom razlomak $\frac{1}{7}$ zapisan je s točnosti od 10 decimala, a za ispis decimalnog broja upotrijebili smo ukupno 13 mjesta (`{0:13.10f}`). Budući da se broji i decimalna točka, za cijeli dio broja ostalo je 2 mjesta Oznaka `f` znači da se ispisuje broj u decimalnom

zapisu. Posljednja ispisana decimala je 9, a ne 8, jer se ona zaokružuje. Ako broj ima manje cijelih mjesta nego smo zadali ispisat će se prazna mjesta. Naredba

```
>>> print ('{0:12.7f}'.format (1/7))
   0.1428571
```

će prvo ispisati 3 prazna mjesta, pa cijeli dio (0), točku i 7 decimala (ukupno 12 mjesta).

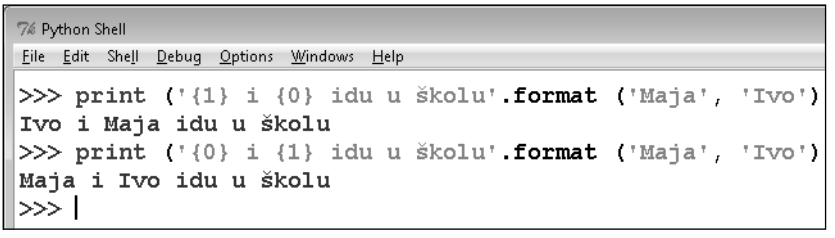
Grafički to možemo pokazati ovako:

			0	**.**	**1**	**4**	**2**	**8**	**5**	**7**	**1**

Ostaje objasniti čemu služi **0** prije oblikovanja. Python započinje brojati od 0, a ne od 1, tako da prvi podatak označava s 0, drugi s 1, itd. Upišite u prozor *Python Shell* sljedeće naredbe i promotrite kako oznake {0}, {1} i {2} utječu na redoslijed ispisa brojeva:

```
>>> print ('{0} + {1} + {2} = {3}'.format(3,4,5,3+4+5))
3 + 4 + 5 = 12
>>> print ('{1} + {2} + {0} = {3}'.format(3,4,5,3+4+5))
4 + 5 + 3 = 12
>>> print ('{2} + {0} + {1} = {3}'.format(3,4,5,3+4+5))
5 + 3 + 4 = 12
```

U format naredbi su četiri podatka: prvi: 3 – u ispisu je označen s {0}, drugi 4 – u ispisu je označen s {1}, treći 5 – u ispisu je označen s {2} i četvrti podatak 3+4+5 – u ispisu je označen s {3} (prilikom ispisa će se zbroj izračunati.



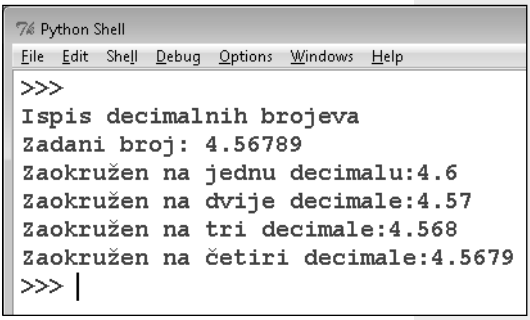
Slika 19. Oblikovanje ispisa: {0} označava prvi podatak u format naredbi,a {1} označava drugi podatak.

Koristeći ispis decimalnih brojeva na zadani broj decimala, možete zaokruživati decimalne brojeve:

```
print ('Ispis decimalnih brojeva')
x=4.56789
print ('Zadani broj:',x)
print ('Zaokružen na jednu decimalu:{0:3.1f}'.format(x))
print ('Zaokružen na dvije decimale:{0:4.2f}'.format(x))
print ('Zaokružen na tri decimale:{0:5.3f}'.format(x))
print ('Zaokružen na četiri decimale:{0:6.4f}'.format(x))
```

Rješenje u datoteci **vje15a.py**.

Slika 20. Oblikovanje ispisa decimalnog broja.



Zadaci

9. Prepiši u bilježnicu, pa dopuni rečenice:

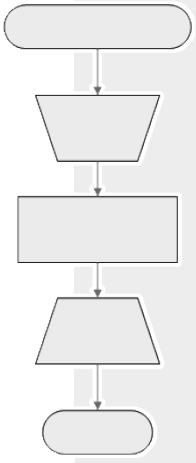
Naredba za upis podatak s tipkovnice je _____. Želim li u varijablu **broj** upisati neki broj napisat ću naredbu:

```
_______ = _______ ('Upiši neki broj')
```

Naredba upisani podatak shvaća kao _____, pa ga prije računanja moramo pretvoriti u _____ naredbom `broj= _____ (broj).`

10. Prepiši u bilježnicu pa dopuni algoritam, dijagram toka i program za oduzimanje dva broja **a** i **b**.

Dijagram toka:



Algoritam:

upiši a

_____ b

_____=a___b

ispiši _____

Program:

```
a=input('Upiši umanjenik:')
b=input('Upiši umanjitelj:')
razlika = int(a) _____
print('Razlika je', _____)
```

11. Ivanov tata želi kupiti zemljište u obliku kvadrata. Zemljište mora ograditi žicom jer na njemu želi saditi jabuke tako da na svakom kvadratnom metru posadi jedno stablo jabuka. Ivan mu treba pomoći odlučiti koliko veliko zemljište treba kupiti. Ako je duljina zemljišta *a*, dopuni algoritam, i program za računanje opsega i površine kvadrata stranice duljine *a*. Nacrtaj dijagram toka. (*Uputa;* o=4 · a, P= a · a)

Algoritam:

upiši ___

o= _____

_____=a___a

ispiši _____

ispiši _____

Program:

```
a=_____('_____')
o=_____
P = int(a) _____
print('Duljina žice:', _____)
_____ ('Broj stabala jabuka:',_____)
```

12. U zemlji Nigdjezemskoj dosjetili su se kako pomoći gladnoj djeci. Ukinuli su novčanice manje od 10 tako da se svaki iznos računa plaća zaokružen na veći iznos, a višak novaca ide za gladnu djecu. Napiši program (dopuni naredbe koje nedostaju) i nacrtaj dijagram toka koji za svaki upisani iznos koji bi trebalo platiti (*cijena*), računa koliko stvarno treba platiti (*plati*) i iznos koji ide gladnoj djeci (*djeci*). Promotri primjere:

cijena	plati	djeci
23	30	7
27	30	3
30	40	10
55	60	5
50	60	10

Algoritam:

upiši cijena

ostatak= cijena % 10

djeci=10-ostatak

plati = cijena+djeci

ispiši plati

ispiši djeci

Program:

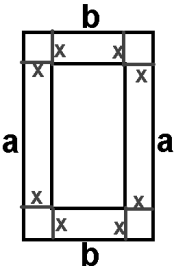
```
a =_____________('_________________')
ostatak =_____________________
djeci = _____________________
plati = _____________________
print('Treba platiti:', _____)
_____ ('Za gladnu djecu:',_____)
```

13. Napiši program koji upisuje brojeve u varijable *a*, *b* i *c* i računa **a*****b**+**c**.

14. Napiši program koji upisuje brojeve u varijable *a*, *b* i *c* i računa **a*****b**+**a*****c**.

15. Napiši program koji upisuje brojeve u varijable *a*, *b*, *c* i *d* i računa **a*****b**+**c**-**d**.

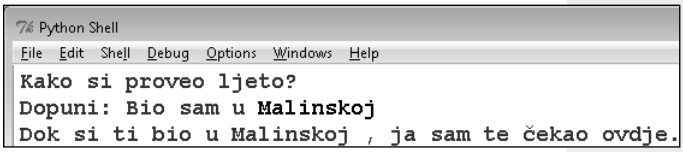
16. Ana ima okvir za slike oblika pravokutnika čija je duljina *a*, a širina *b*. Debljina okvira je *x*. Ana želi po unutarnjem i vanjskom rubu okvira nalijepiti zlatnu traku. Kolika duljina trake treba Ani?



17. Napiši program koji traži da upišeš ime svoje škole u varijablu *ime_skole* (poruka prije upisa: *Kako se zove tvoja škola?*) i mjesto u varijablu *mjesto* (zbog smisla rečenice naziv mjesta upiši u lokativu (poruka prije upisa: *Dopuni: Živim u*), pa ispisuje rečenicu: Ideš u školu *ime_skole* u mjestu *mjesto*.

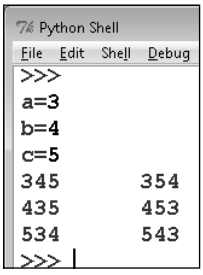
18. Napiši program koji omogućuje razgovor s računalom.

U prvom retku računalo pita: *Kako si proveo ljeto?* U drugom retku traži da dopuniš rečenicu, pa ispisuje: *Dopuni: Bio sam u*. Ovdje treba upisati naziv mjesta u varijablu *mjesto* (na slici 21. upisali smo *Malinskoj*). Zatim računalo ispisuje rečenicu oblika: Dok si ti bio u *mjesto*, ja sam te čekao ovdje.



Slika 21. Plavi tekst ispisuje program, a crni tekst se upisuje.

19. Napiši program koji upisuje 3 znamenke ***a*, *b*** i *c* i ispisuje sve troznamenkaste brojeve koji se od njih mogu napisati tako da se brojevi sa istom znamenkom stotica nalaze u istom retku. Brojevi se ispisuju u dva stupca, tako da su u istom retku brojevi sa istom znamenkom stotica. Razmak među brojevima u retku je jedan tabulator. Pretpostavimo da su *a*, *b* i *c* različiti prirodni brojevi manji od 10. Koristi se algoritmom.

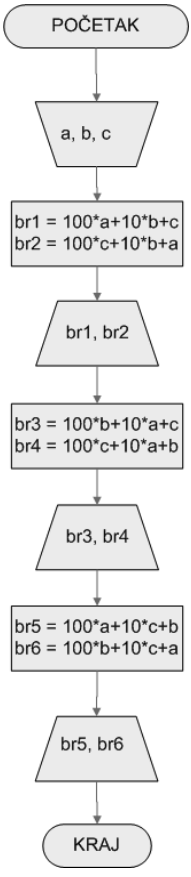


Slika 22. Primjer izvršenja zadatka 19.

Uputa:

Ovaj zadatak možete riješiti na dva načina. Razmišljajući matematički (kako je zadatak i zadan) algoritam i dijagram toka napravili biste ovako:

upiši a,b, c
br1= 100*a+10*b+c
br2= 100*a+10*c+b
ispiši br1, br2
br1= 100*b+10*a+c
br2= 100*b+10*c+a
ispiši br3, br4
br1= 100*c+10*a+b
br2= 100*c+10*b+a
ispiši br5, br6



Na drugi način zadatak se može riješiti tako da znamenke ostavimo u obliku teksta i jednostavno ih „slijepimo“: ako je *a*=3, *b*=4, *c*=5, tada će naredba `br1=a+b+c` u varijablu *br1* staviti 345, a naredba `br3=b+a+c` u varijablu *br3* staviti 435. Riješi program i na taj način.

20. Napiši program koji upisuje 4 znamenke: *a*, *b*, *c* i *d* i ispisuje sve troznamenkaste i četveroznamenkaste brojeve koji se od njih mogu napisati. Pretpostavimo da su a, *b*, *c* i *d* različiti prirodni brojevi manji od 10.

21. Ivičina majka ima samo novčanice od 100 kn, 20 kn te kovanice od 5 kn i 1 kn. Ivici uvijek daje točan iznos novca koji mu treba, ali tako da mu dâ najmanji mogući broj novčanica i kovanica.

    Promotri tablicu:

kn	100 kn	20 kn	5 kn	1 kn
237	2	1	3	2
1289	12	4	1	4
993	9	4	2	3

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
>>>
Upiši iznos u kunama:135
1 novčanica po 100 kn.
1 novčanica po 20 kn.
3 kovanica po 5 kn.
0 kovanica po 1 kn.
>>> ========================
>>>
Upiši iznos u kunama:244
2 novčanica po 100 kn.
2 novčanica po 20 kn.
0 kovanica po 5 kn.
4 kovanica po 1 kn.
>>>
```

Slika 23. Primjer izvršenja zadatka 21.

    Evo objašnjenja prvog retka tablice. Ako majka Ivici treba dati 237 kn, tada će mu dati 2 novčanice po 100 kn, jer je 237 // 100 = 2, a ostaje mu 237 % 100 = 37 kn. Tih 37 kn pretvorite prvo u novčanice po 20 kn: 37 // 20 =1, tj. 1 novčanica od 20 kn, a ostaje mu 37 % 20 = 17 kn. Njih pretvorite u kovanice po 5 kn: 17 // 5 = 3 kovanice od 5 kn. Ostaju 17 % 5 = 2 kovanica po 1 kn.

    Napiši program koji će Ivici ispisati s kojim novčanicama treba platiti.

22. Izmijeni program o novčanicama iz zadatka 21. tako da:

    a) ispisuje i ukupan broj novčanica od 100 kn i 20 kn, te kovanica od 5 kn i 1 kn.

    b) zadani iznos u kunama pretvori u najmanji broj novčanica od 200, 50, 10 kn i kovanica po 1 kn.

    c) zadani iznos *kune* pretvori u najmanji broj novčanica od *broj1*, *broj2* i *broj3* kn, ako su *broj1*, *broj2* i *broj3* vrijednosti novčanica koje upisujemo na početku programa. Pretpostavi da takve novčanice postoje i da je *broj1* > *broj2* > *broj3*. Ispisati koliki iznos preostaje, tj. ne može se pretvoriti u zadane novčanice. Promotri primjere:

Ulaz				Izlaz			
kune	**br1**	**br2**	**br3**	Broj novčanica po **br1** kuna	Broj novčanica po **br2** kuna	Broj novčanica po **br3** kuna	Ne može se isplatiti
1500	200	100	50	7	1	0	0
1527	500	200	100	3	0	0	27
1527	50	20	10	30	1	0	7
1527	1000	5	1	1	105	2	0