Podloge za stručno usavršavanje učitelja osnovnih škola
za domenu
Računalno razmišljanje i programiranje

02
Prozor za pisanje programa – editor, stil pisanja programa

Uz dozvolu izdavača korišteni su sadržaji iz priručnika:

Leo Budin, Predrag Brođanac, Zlatka Markučič
Smiljana Perić, Dejan Škvorc, Magdalena Babić

Računalno razmišljanje i programiranje u Pythonu
Element, Zagreb, 2017

Računalni programi

Dosad smo sve naredbe pisali u prozoru interaktivnog sučelja. Naredbu smo mogli pisati kada se u novom redu pojavio znak prompt >>>. Nakon utipkavanja svih znakova (koje smo mogli i mijenjati) pritiskom na tipku (*Unos*) naredba se odmah i izvela. Na taj način možemo rješavati neke jednostavnije zadatke i probleme.

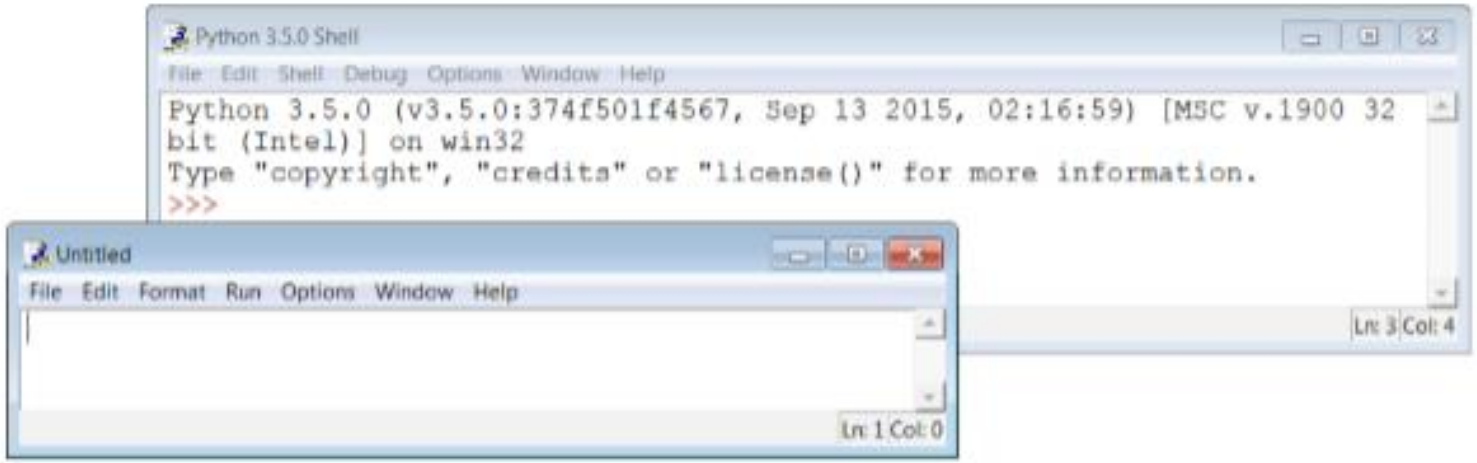
Međutim, sve će se te naredbe izgubiti kada zatvorimo interaktivno sučelje *Pythona*. Ako želimo ponovno rješavati zadatak koji smo riješili nizom naredbi u interaktivnom prozoru, morat ćemo te naredbe ponovno utipkavati u prozor interaktivnog sučelja.

Zbog toga je korisno niz naredbi koji nam rješava neki problem trajno pohraniti u neku datoteku koju možemo pohraniti na disk i kad god poželimo jednostavno ih pokrenuti.

Znamo da se niz tako pripremljenih nardbi naziva ***računalnim programom***.

Otvaranje prozora za pisanje programa, prozor editora

Naše ćemo programe pripremati (pisati) u novom prozoru koji ćemo otvoriti iz prozora interaktivnog sučelja. To činimo tako da u prozoru `Python 3.x.y Shell` mišem aktiviramo padajući izbornik *File* i u njemu kliknemo mišem na *New File*. U tom će se trenutku otvoriti novi prozor s natpisom *Untitled* (engl. *untitled* – bez naslova) kako je to prikazano na slici 2.5.

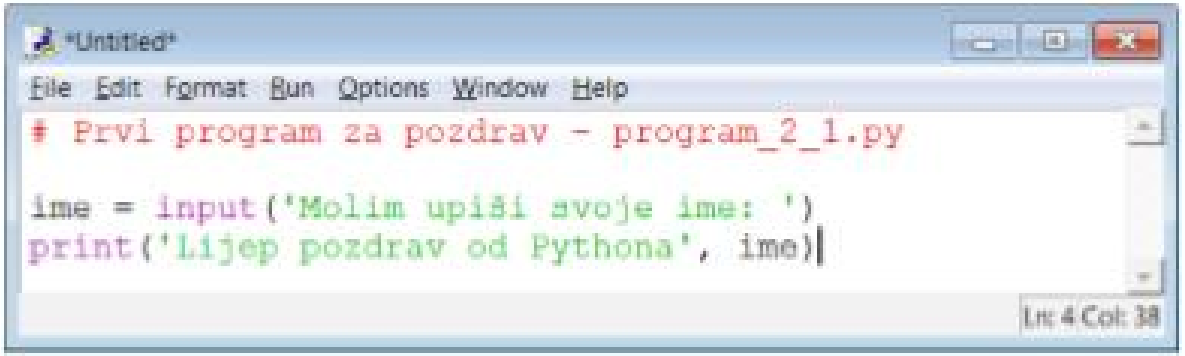


Taj se prozor razlikuje od interaktivnog prozora `Python 3.x.y Shell`. Primijetimo da u tom prozoru nema znaka >>> te da podsjeća na prozor nekog programa za obradu teksta (Word, Notepad). Uistinu se radi o svojevrsnom uređivaču teksta koji ćemo kraće nazvati **editorom**. Postupno ćemo naučiti kako se editor rabi.

Pisanje programa

Kada u prozoru editora počnemo nešto pisati, u naslovnoj traci pojavit će se natpis **Untitled**. Zvjezdice označavaju da sadržaj koji je zapisan u prozoru nije pohranjen u datoteku.

U tom ćemo prozoru napisati program za pozdravljanje. Pritom ćemo rabiti funkcije `input()` i `print()` čije smo djelovanje već provjerili u interaktivnom prozoru. Taj bi program mogao izgledati kao na slici 2.6.



Slika 2.6. *Program za pozdrav prije pohranjivanja na disk*

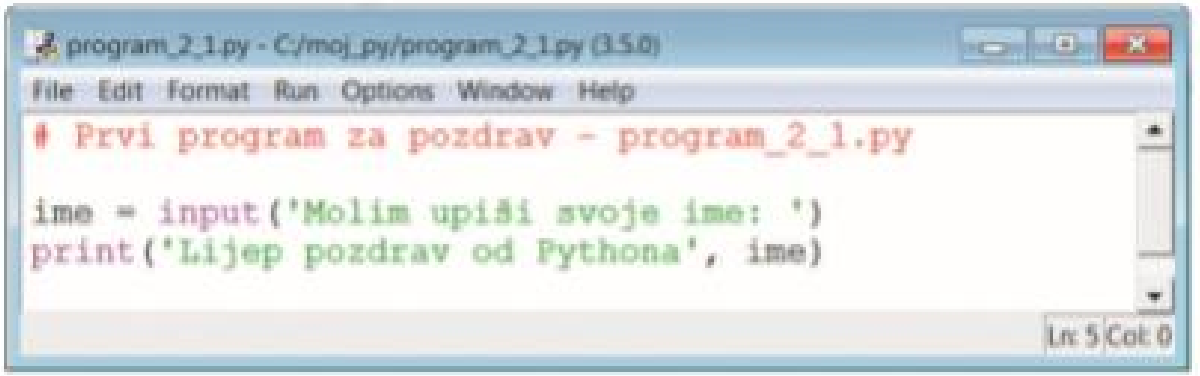
Kada u prozoru editora napišemo pojedine naredbe, one se ne izvedu tj. ništa se ne dogodi. One će početi djelovati tek kada ih pohranimo u neku programsku datoteku i pokrenemo izvođenje pripremljenog programa.

Prije nego datoteku pohranimo, moramo se odlučiti u koju ćemo je mapu smjestiti. Najbolje rješenje je organizirati posebnu mapu koju možemo nazvati primjerice `moj_py`.

Pohranjivanje programa

Pohranjivanje se obavlja tako da se u padajućem izborniku *File* odabere funkcija *Save as* i nakon što se pojavi mali prozorčić s pregledom mogućih mapa odabere mapa `moj_py` te u okvir s *File name* upiše odabrano ime datoteke u koju će biti smješten naš program. Nazovimo naš prvi program `program_2_1`. *Python* će automatski dodati sufiks `.py`.

Nakon što to obavimo, promijenit će se tekst u naslovnoj traci kao na slici 2.7.



Slika 2.7. *Pohranjeni program za prvi pozdrav*

Iz natpisa je vidljivo da se program nalazi u datoteci `program_2_1.py` te da se ta datoteka nalazi u mapi `moj_py` na disku `C`.

Pokretanje programa tijekom njegove pripreme

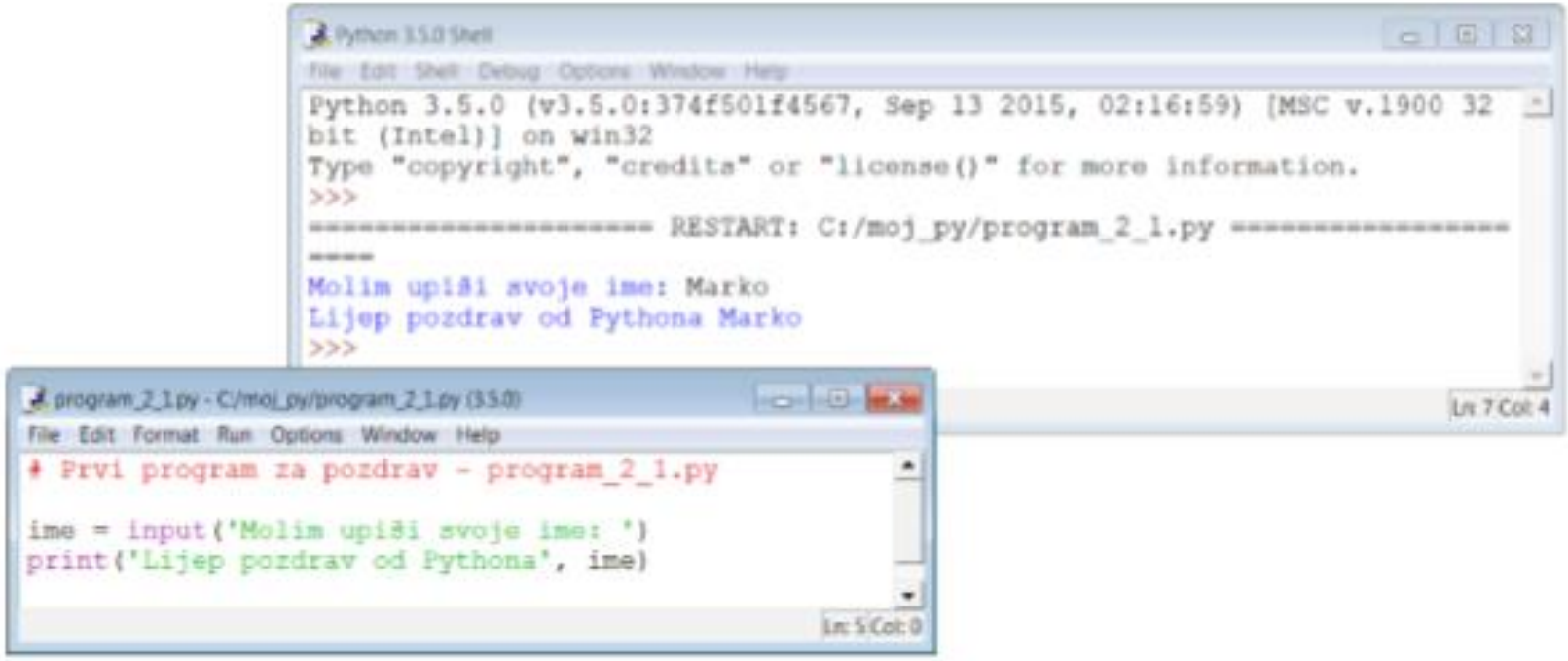
Nakon što je program pohranjen, on se može i pokrenuti. Pokretanje se obavlja tako da se u padajućem izborniku *Run* klikne mišem na *Run Module* ili da se na tipkovnici pritisne tipka *(F5)* (modulom se naziva datoteka u koju je smješten program, na neki način nazivi *modul* i *program* imaju podjednako značenje).

Nakon toga *Python* počinje izvoditi redom naredbe iz datoteke, a cijeli ispis tekstova (stringova) iz naredbi obavlja u interaktivnom prozoru.

Dakle, mi prilikom izvođenja programa vidimo dva prozora:

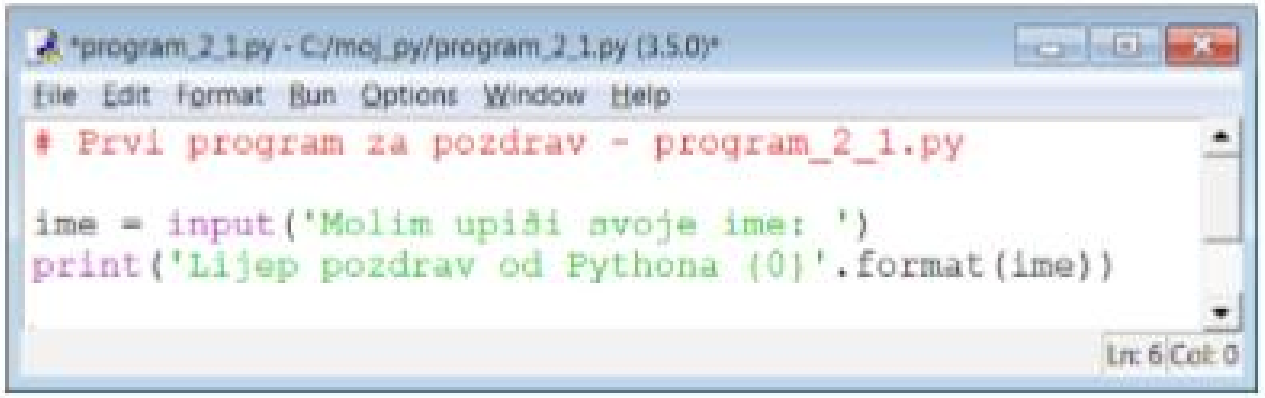
- jedan u kojem se nalaze naredbe i
- drugi u kojem se ispisuju tekstovi prilikom izvođenja naredbi.

Izgled tih prozora nakon što smo tipkovnicom utipkali ime *Marko* prikazuje slika 2.8.



Slika2.8. *Izgled interaktivnog prozora i prozora s prikazom naredbi programa*

Ako u primjeru sa slike 2.8. želimo ispis promijeniti tako da ispisni string oblikujemo metodom `format()`, dobit ćemo izgled prozora kao na slici 2.9.



Slika 2.9. *Zvjezdice u naslovnoj traci ukazuju na promjene teksta programa*

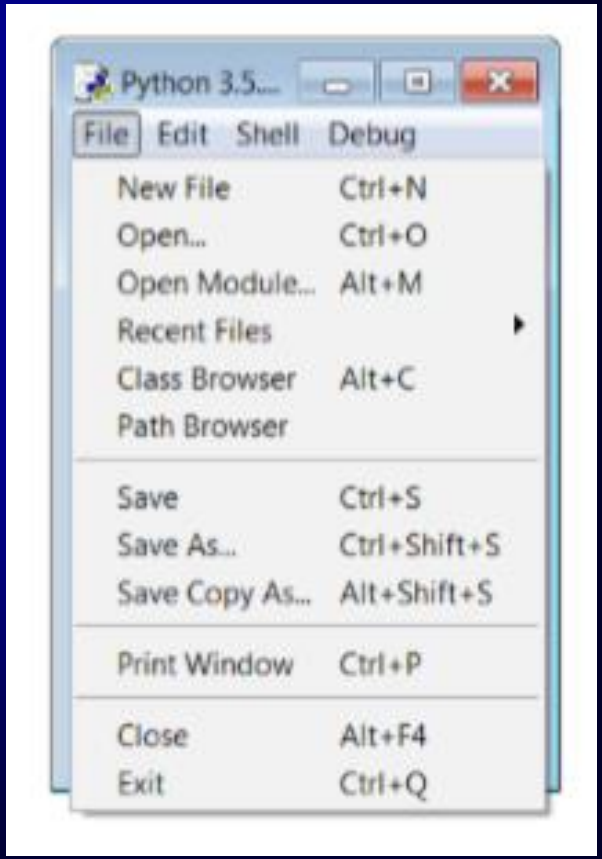
Izmijenjeni program možemo pohraniti pod novim imenom ako se u padajućem izborniku *File* odabere funkcija *Save as*. No, ako želimo promijenjeni program pohraniti pod istim imenom, u padajućem izborniku treba odabrati funkciju *Save* ili pritisnuti tipke *(Ctrl)* + *(S)*. Nakon što to učinimo, nestat će zvjezdice u naslovnoj traci.

Pokušamo li pokrenuti program koji je promijenjen prije pohranjivanja, *Python* će nas posebnom porukom upozoriti da ga trebamo pohraniti.

Pokretanje prethodno pohranjenih programa

Velika je razlika između pisanja niza naredbi u interaktivnom prozoru i njihova pohranjivsanja u datoteke u tome što su ti nizovi trajno pohranjeni i možemo ih uvijek ponovno pokrenuti.

Otvaranje se obavlja pomoću padajućeg izbornika interaktivnog sučelja.



Programsku datoteku otvorit ćemo tako da u padajućem izborniku *File* interaktivnog sučelja odaberemo funkciju *Open* i odaberemo mapu `moj_py` te u toj mapi datoteku `program_2_1.py`.

Treba obratiti pažnju na funkciju *Recent Files* jer će se klikom miša na nju pojaviti popis od dvadesetak programskih datoteka koje su zadnje bile aktivne.

Učenje programiranja nije jednostavno

Priprema računalnog programa koji služi za rješavanje nekog problema nije jednostavna. U prvom redu potrebno je osmisliti moguće načine njihova rješavanja i odabrati prikladne postupke (algoritme).

Nakon toga treba te postupke zapisati u odabranom programskom jeziku. Neki koraci postupaka (pa čak i cijeli postupak kod jednostavnijih problema) mogu se pripremiti i ispitati u interaktivnom sučelju, a konačni program treba posredstvom uređivača teksta trajno pohraniti u obliku datoteke.

Jasno je da nam se prilikom pisanja naredbi u interaktivnom sučelju, kao i u uređivaču teksta programa potkradaju pogreške. Naredbe koje pišemo u interaktivnom sučelju odmah će se izvesti i pogreška će nam se pojaviti odmah. Pogreške koje smo unijeli u pripremljeni program zapazit ćemo tek tijekom izvođenja programa.

Pogreške koje se pojave možemo ispraviti. Kada *Python* prepozna pogrešku, on će iza mjesta gdje je otkrio pogrešku ispisati poruku u kojoj objašnjava razlog nastajanja pogreške. Ponekad nam objašnjenje neće biti dovoljno jasno te ćemo se morati dodatno potruditi kako bismo otkrili mjesto i vrstu pogreške, no pažljivo pregledavanje naredbi koje su napisane ispred ispisa teksta o pogrešci će nam pomoći u tome. Treba naglasiti da će *Python* lako prepoznati i javiti narušavanje pravila pisanja tzv. **sintaksne pogreške**.

Međutim, *Python* nažalost ne može prepoznati pogrešno odabrani algoritam, odnosno pogrešno osmišljeno programsko rješenje. Ispitivanje programa za prepoznavanje takvih tzv. **semantičkih pogrešaka** moramo osmisliti sami.

Preporuke za lijepo pisanje programa

Kada pišemo neki program, možemo ga zapisivati na različite načine, a da ispravnost naredbi ostane ista. Kako je kod programiranja dosta važna i činjenica da programski kôd treba biti čitljiv, uredan i pregledan, dogovorene su neke preporuke za lijepo pisanje programa.

Prvo ćemo se upoznati s preporukama za pisanje izraza i naredbe pridruživanja. Kod pisanja izraza ćemo prije i poslije operanda, a onda i znaka pridruživanja dodati po jedan razmak. Korištenje razmaka će nam program učiniti čitljivijim i preglednijim. Pogledajmo primjere:

Loše	Dobro
a=24+43	a = 24 + 43
b=7	b = 7
c=a//b%3	c = a // b % 3

Uočimo da između dvije kose crte koje predstavljaju cjelobrojno dijeljenje nismo stavili razmak. Razmak ne stavljamo ni između operatora koji se sastoji od dvaju znakova kao što su +=, -=...

Kod pozivanja funkcija i definiranja funkcija ispred i iza zagrada ne piše se znak razmaka. Kada unutar poziva funkcije nabrajamo više parametara (varijable, stringovi, vrijednosti), iza zareza ćemo pisati razmak.

Loše	Dobro
`print ( x )`	`print(x)`
`print(x , y )`	`print(x, y)`
`print(x,y)`	`print(x, y)`

U editoru je duljina retka ograničena na 80 znakova. To nam u nekim situacijama može biti ograničavajuće, odnosno ponekad će se dogoditi da nam neka od naredbi ne stane u tih 80 znakova, posebice što se veći dio naredbi ne piše od prvog znaka u nekom retku već se uvlači za četiri ili više mjesta. Kada se radi o izrazima, taj ćemo problem riješiti tako da dodamo zagrade prije i poslije elemenata koji se nalaze neposredno prije i neposredno poslije mjesta na kojem želimo prelomiti redak i jednostavno na mjestu gdje želimo prelomiti redak pritisnemo (*Unos*).

```
rezultat = varijabla_1 + varijabla_2 + vrijabla_3 + varijabla_4
rezultat = varijabla_1 + (varijabla_2 +
           varijabla_3) + varijabla_4
```

Drugi način rješavanja tog problema je upisivanje znaka lijevo nagnute kose crte (/) neposredno prije prelaska u novi redak, obavezno iza operatora, i potom se pritisne *(Unos)*.