SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

PyCharm

Seminarski rad iz kolegija Uporaba računala u nastavi

Marko Zeman,

PMF-FO, nastavnički smjer

8. semestar

doc. dr. sc. Dalibor Paar

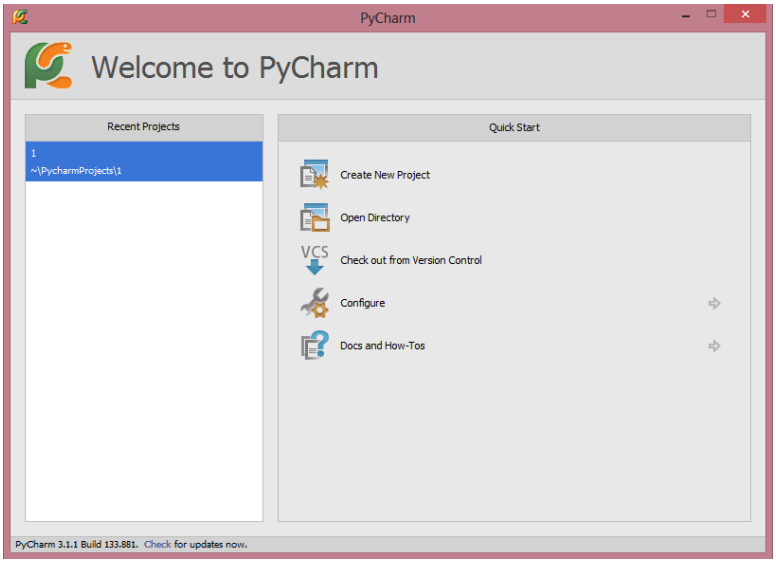
Zagreb, svibanj, 2014.

Uvod

PyCharm je razvojno sučelje (IDE) koje se koristi za programiranje u Pythonu. Pruža nam različite mogućnosti; pomoć kod pisanja koda (analizira i dopunjuje kod, podcrtava greške), vrlo lagana navigacija u cijelom programu (može se skakat s jednog mjesta na drugo, strukturiran izgled), Python *Refactoring* (pomaže kod izmjene postojećeg koda), dobra integracija s Python web framework-cima (*Django, Web2Py*), integriran Python *Debugger* te podrška za „*unit testing*“, integracija s *Google App Engine* servisom, *Version Control System* integracija, podrška za najpopularnije sustave za verzioniranje (Git, Mercurial, Subversion, CVS). Radi na više platforma kao što su Windows, Mac OS X, Linux. Upravo to ga plasira na sam vrh tržišta. Da biste ga mogli pokrenuti, trebate imati instaliranu verziju Pythona.

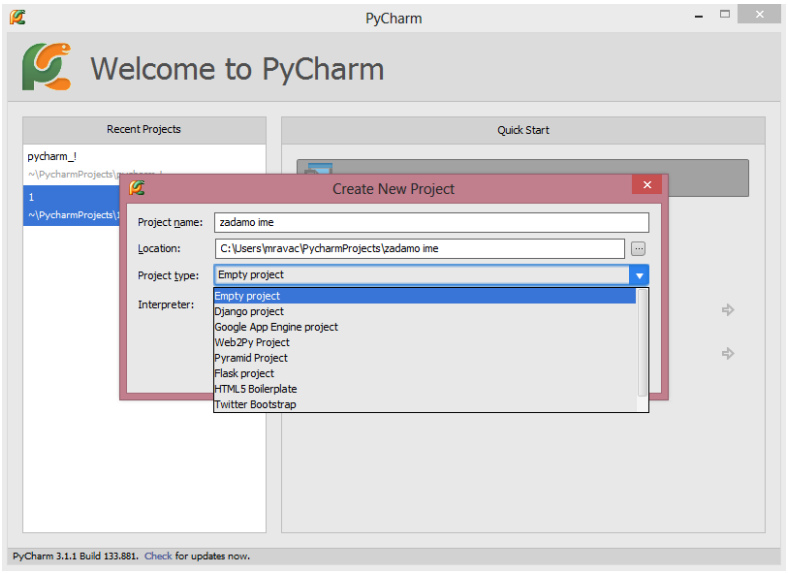
Primjeri

Instalirali smo PyCharm (u daljnjem tekstu paket) na naše računalo te ga sada idemo koristiti. Pokrenemo paket i otvori se prozor (sl.1).



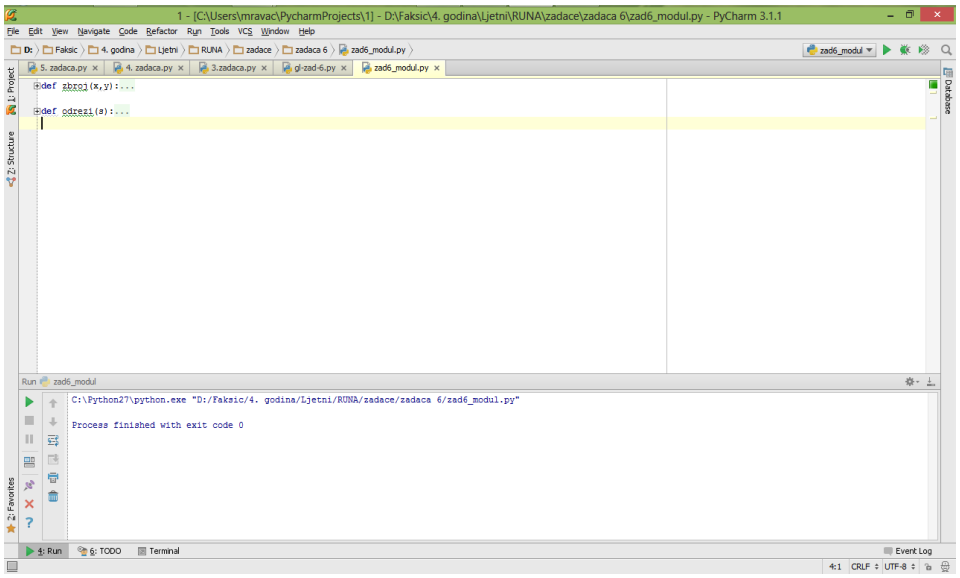
Slika 1: Početni prozor

Želimo otvoriti novi projekt te kliknemo na „*create new project*“. Otvorio nam se novi prozor (sl.2) na kojem vidimo veliku mogućnost paketa. Možemo odabrati između praznog projekta (koji nas zanima trenutno), *Django* projekta, *Web2Py* te mnogo drugih projekata.



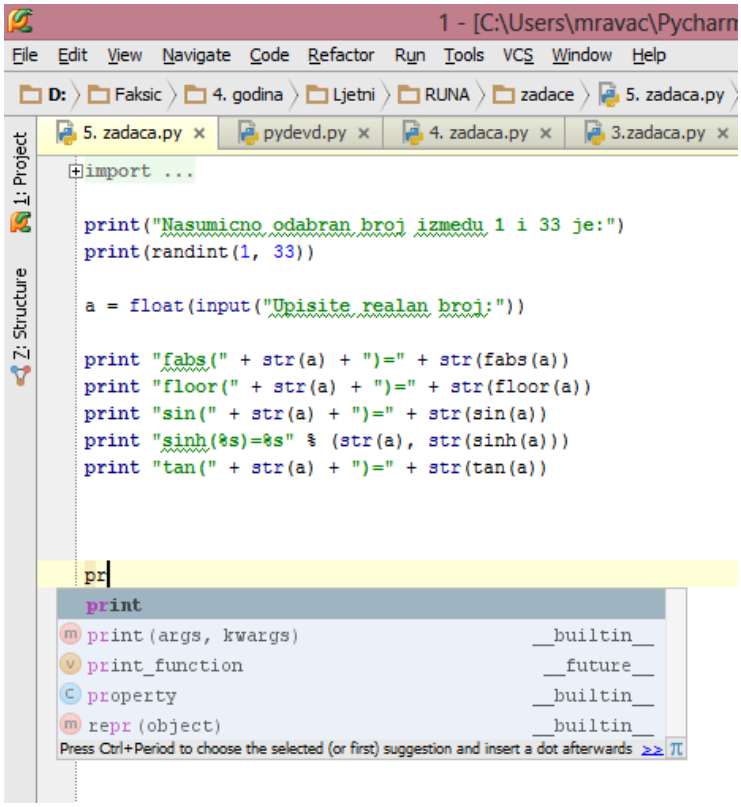
Slika 2: Novi projekt

Sad kad smo otvorili novi projekt, možemo otvoriti neke Python datoteke u našem projektu (sl.3). Ja sam otvorio zadaće koje sam radio. Vidimo da se svaka zadaća otvorila u svom tabu i mi jednostavno možemo prelaziti s jedne zadaće na drugu. Također vidimo i u alatnoj traci velik izbor funkcija koje nam nudi paket. Od jednostavnog dodavanja datoteka, editiranja do nekih naprednijih funkcija tipa *refactor*, izvršavanje programa i ostalih alata.

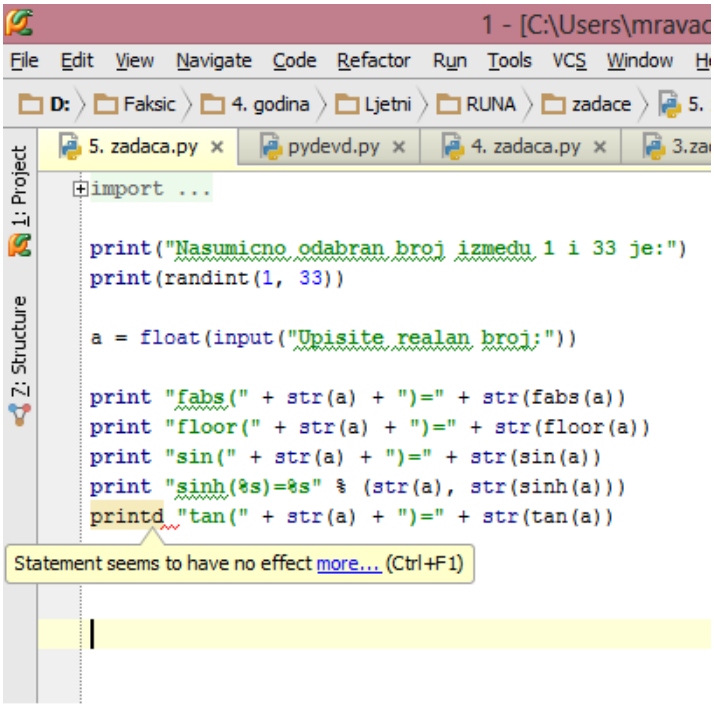


Slika 3: Strukturiran izgled

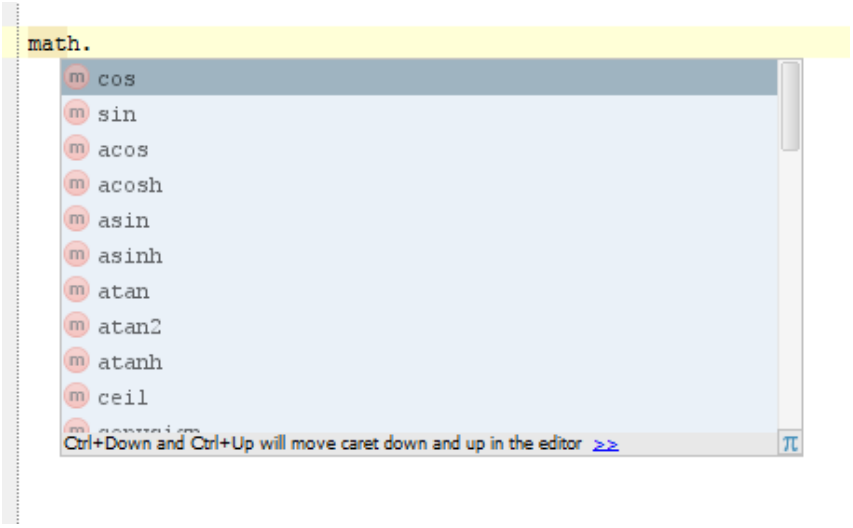
Započinjemo pisati kod i vidimo opširan izbor u dopunjavanju koda (sl.4) te podcrtavanja greške (sl.5). Napisali smo prva dva slova i paket nam već daje izbor da dopuni kod. Kod greške nam nudi opširnije obrazloženje udaljeno jednim klikom miša. Isto tako ako upišemo neki modul, ispisuje nam sve funkcije kojima raspolaže (sl.6). Ne trebamo tražit po internetu ili zezat kolegu. Automatski zatvara zagrade i navodnike. Pišete kod i otvorite zagradu, a paket vam sam zatvara zagradu. Isto tako radi i sa navodnicima. Ne gubi se vrijeme na duplo tiskanje po tipkovnici.



Slika 4: Dopuna koda

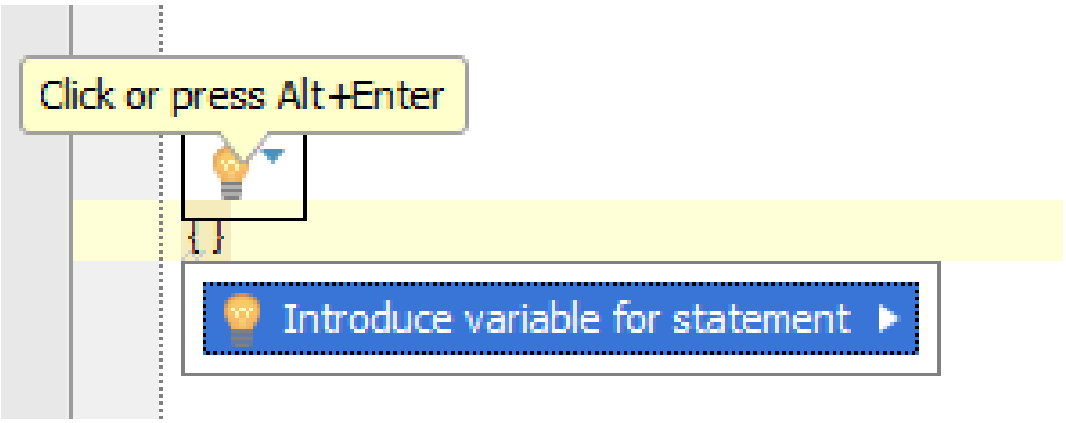


Slika 5: Podcrtavanje pogrešaka

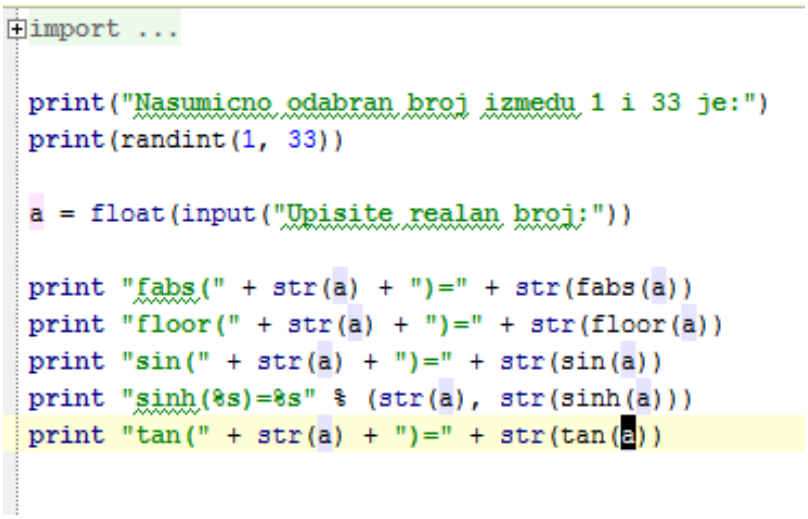


Slika 6: Ispis svih funkcija modula

Otvorimo vitičastu zagradu i paket je automatski zatvara te ponudi opciju da pritiskom tipke na tipkovnici ili pritiskom miša uvedemo varijablu (sl.7). podcrtava iste zagrade, navodnike, varijable (sl.8) što može biti jako korisno kod pisanja koda. Recimo da imamo kod od tridesetak redova i neku varijablu u samo tri reda. Nađemo je u jednom redu i paket nam označuje istog trena u ostala dva reda tu varijablu.



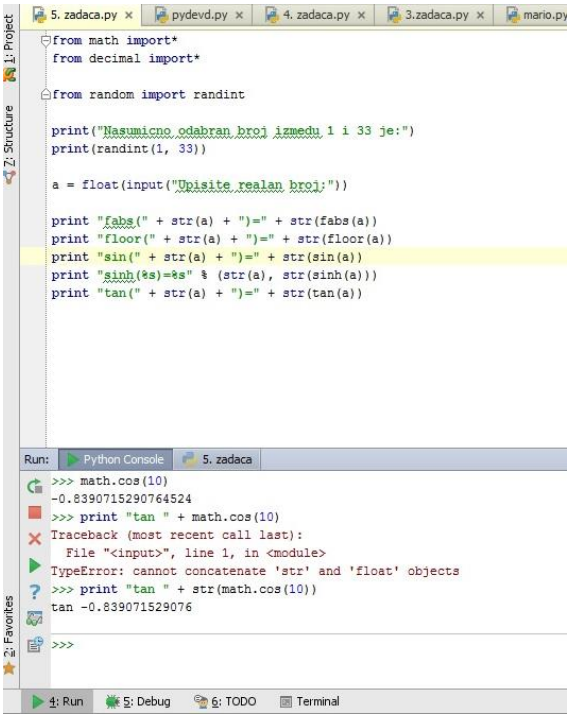
Slika 7: Uvođenje varijable



Slika 8: Podcrtavanje istih varijabli

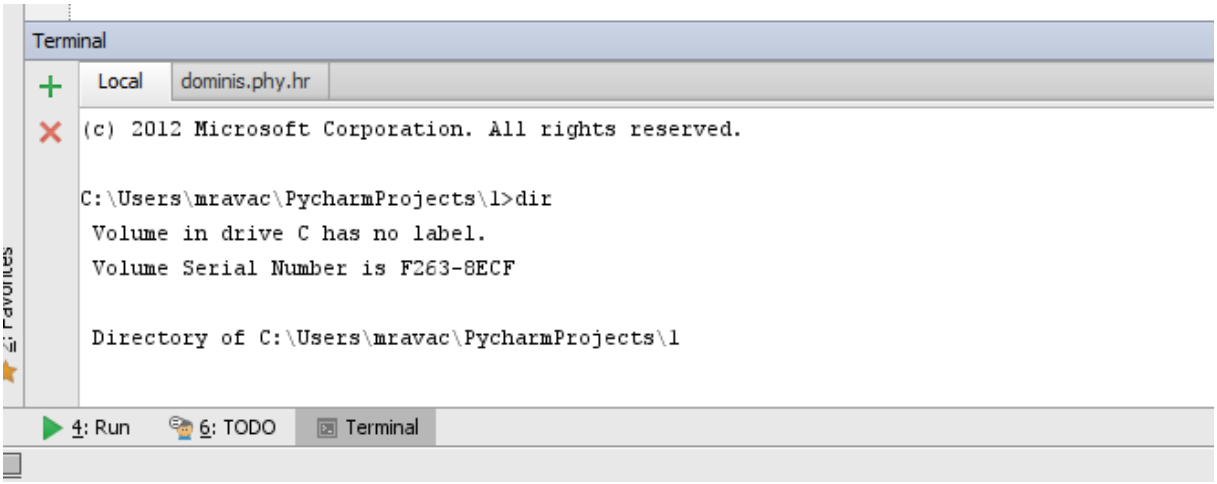
Ako se vratimo na sliku 3 i odemo na *tools* spustiti će se prozorčić u kojem vidimo da možemo pokrenuti Python konzolu, otvoriti terminal, pokrenuti SSH sesiju, Vagrant…

Python konzola nam je zgodna za testiranje koda. Npr. Prije nego što zapišemo išta u datoteku našeg programa, možemo testirati hoće li navedeno stvarno funkcionirati kako mi očekujemo (sl.9). Na slici vidimo da sam provjeravao kako da ispišem riječ i broj zajedno i u konzoli sam vidio da trebam dodat ispred broja *str* da ga prepozna kao riječ i ispiše.



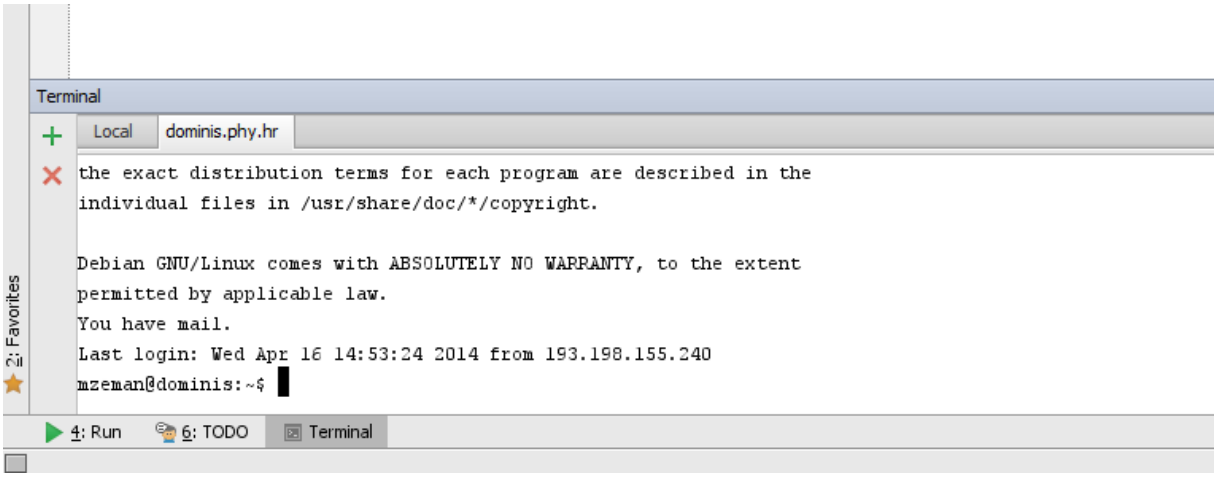
Slika 9: Python konzola

Open terminal nam otvara pristup konzoli našeg računala (sl.10). Omogućava nam direktan rad iz IDE-a s lokalnim računalom. To je zapravo *command prompt*, istovremeno gledamo u kod kad tražimo lokaciju određene datoteke.



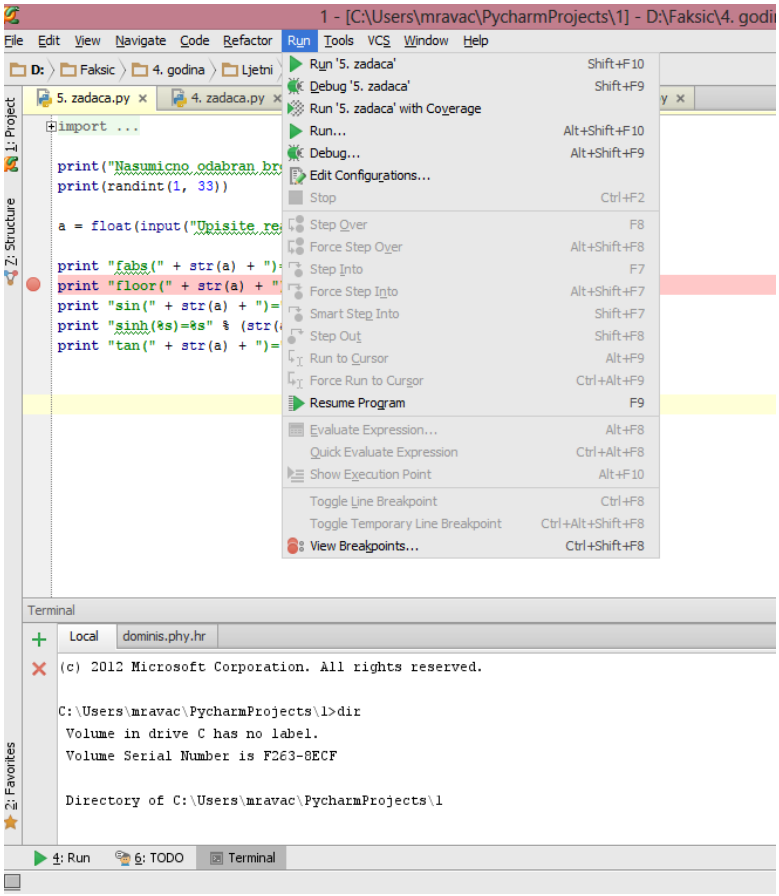
Slika 10: Open terminal

Start SSH session nam omogućava udaljen pristup serveru direktno iz IDE sučelja (sl.11). Ne izlazimo iz PyCharma i tražimo Putty (ako koristimo windows) nego se direktno klikom miša spajamo na udaljen server. Jako zgodno u slučaju da se radi na razvoju web-aplikacije pa želimo neku izmjenu vidjeti odmah na serveru.

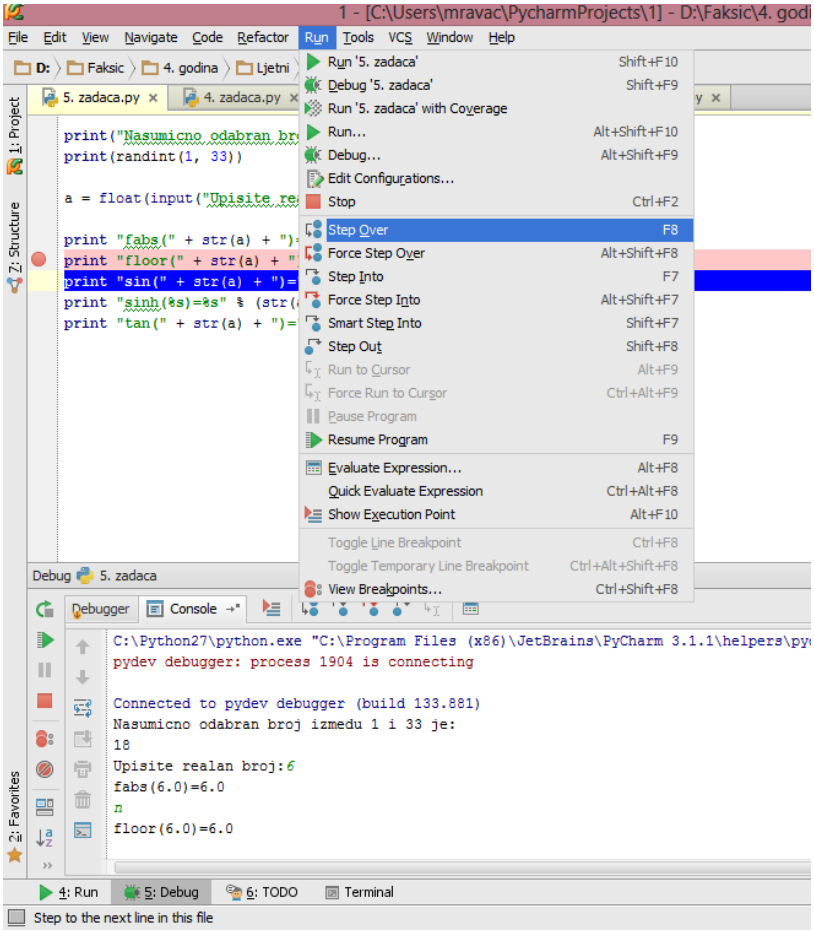


Slika 11: SSH sesija

Run opcija nam nudi pokretanje Python projekta, a također i napredne opcije debugiranja (sl.12). Npr. Odredimo *breakpoint* u dijelu koda, program se izvršava do tog dijela i stane. U *console* prozoru vidimo ispis programa od tog dijela u kodu. Kada program stigne do *breakpointa*, imamo nekoliko izbora kako da nastavimo s izvršavanjem (sl.13). Također u *debugger* prozoru imamo detaljan uvid u vrijednosti varijabli i pozvane module prilikom izvršavanja programa.

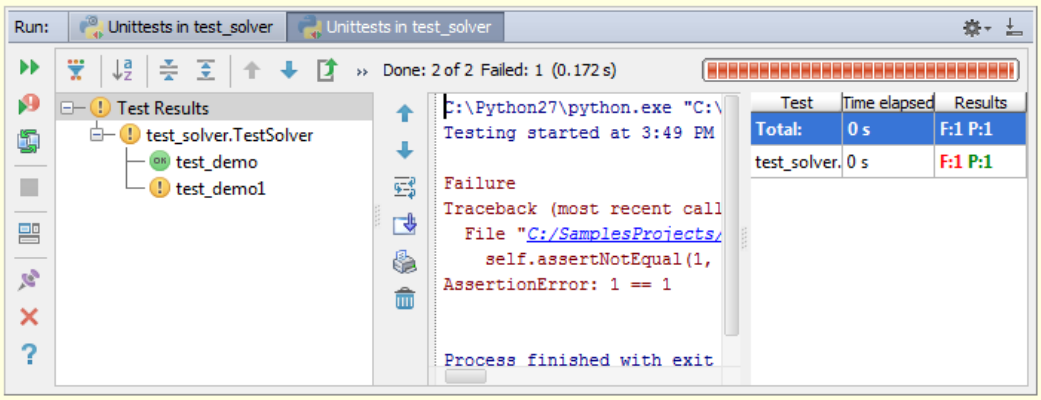


Slika 12: Run opcije



Slika 13: Breakpoint

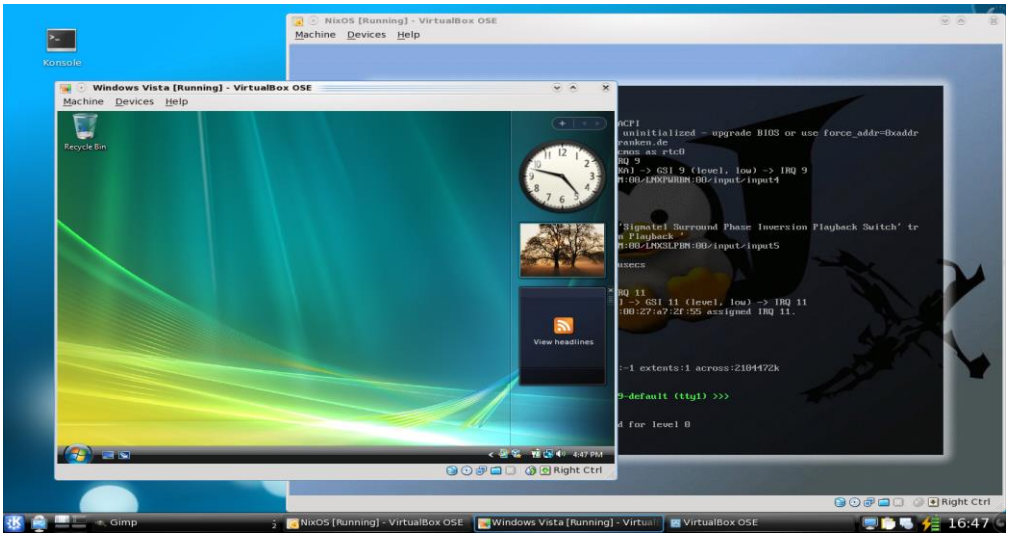
Testiranje koda podržava provjere koda do PEP8 standardu te je podržano i pokretanje vlastitih testova za kod (*unit* testovi, funkcionalni testovi) čiji su rezultati vidljivi u grafičkom *test runneru* (sl.14). Napisali smo kod i sada nas zanima radi li taj kod točno to što mi želimo da radi. Opišemo što neki dio koda mora raditi i onda taj *unit test* provjeri radi li kod upravo to. Recimo da pišemo kod za kalkulator. Mi napišemo kod za zbrajanje brojeva, ali nismo sigurni hoće li kod raditi. Tada ga testiramo i vidimo radi li ili ne.



Slika 14: Test runner

Integracija IDE sučelja sa sustavima za verzioniranje od kojih su najpopularniji Git, Mercurial i CVS olakšava programeru komunikaciju sa serverima. Nema potrebe za upisivanjem naredbi jer su one već integrirane u IDE. Omogućava da se na lak način prati koje promjene nad kodom radimo i po potrebi da se vratimo na neko od prethodnih stanja koda. Uvelike olakšava rad u timu.

Integracija s Vagrant virtualnom mašinom nam omogućava da se kod testira u OS-u drugačijem od onoga na kojem radimo razvoj (sl.15). Npr. Ako radimo na windowsu, a razvijamo kod za linux, tada možemo kreirati *vagrantbox* koji je u principu virtualna mašina, tek toliko da nema suvišnog grafičkog sučelja u PyCharmu no omogućava da se *run* komanda izvršava u tom *vagrant boxu*. Zgodno jer se iz IDE-a lako može testirati kod na nekom od drugih operativnih sustava bez *reboota* i popratnih muka već jednim klikom miša.



Slika 15: Virtualna mašina

Zaključak

Ideja PyCharma kao razvojnog sučelja (IDE-a) je da se maksimalno omogući programeru da se posveti programiranju, a ne tehnikalijama (primjerice da pazi na samo uvlačenje, zatvaranje zagrada). Omogućava nam suradnju s članovima tima kroz integraciju VCS. Prilagođen je za rad s *web framework*-cima, ne moramo napuštati sam IDE da bismo napravili promjenu na lokalnom računalu ili na udaljenom serveru te smo u mogućnosti razvijati kod za više platformi *Vagrantbox*-om. Kako god da bilo, PyCharm svakako olakšava učenje Pythona početniku jer mu ukazuje na banalne propuste u sintaksi, pokazuje mu sve funkcije koje neki modul ima i štedi mu vrijeme koje u konačnici može potrošiti na učenje jezika.

Literatura

- PyCharm web stranica, < raspoloživo na: http://www.jetbrains.com/pycharm/ >, [28.05.2014.].

- PyCharm wikipedia, < raspoloživo na: http://en.wikipedia.org/wiki/PyCharm >, [28.05.2014.].