UDŽBENIK ELEKTROTEHNIČKOG FAKULTETA U BEOGRADU

Milan Bjelica

Programski jezik Python

skripta za studente telekomunikacija

Beograd, 2018.

1 Uvod

1.1 Šta je Python

S Wikipedije:

> Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.
>
> Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.
>
> Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems.

Tvorac Pythona je Guido van Rossum (1990).

Ime mu potiče od TV serije *Monty Python's Flying Circus*.

1.2 Zašto ga učimo?

- slobodni softver
- lako se uči
- **prvi programski jezik** na MIT
- bogate biblioteke:
  - matematički proračuni
  - crtanje grafika funkcija
  - obrada TK signala
  - simulacija
  - upravljanje laboratorijskim instrumentima
  - programiranje mrežnih aplikacija (Google, Youtube)

Radimo verziju 2 (tj. 2.7). Postoji i verzija 3, u pitanju je tzv. *forking*. Na današnjem stepenu razvoja, razlike se tiču sintakse naredbe/funkcije `print`, deljenja celih brojeva, funkcije `xrange` i *unicode* stringova.

1.3 Kako se instalira?

1.3.1 Linux

Sve što treba je prevući iz baze python 2.7 i potrebne module – npr. numpy, scipy, matplotlib, pylab (obuhvata tri prethodna), python-serial... Neki moduli, kao npr. vxi11, nisu u bazi, te ih treba preuzeti s njihovog sajta i slediti uputstvo za instalaciju (najčešće jedna komanda u terminalu).

1.3.2 Win

Preporuka je da se instalira distribucija *Canopy* sa sajta `www.enthought.com`, koja sadrži najveći broj standardnih modula. One ostale, poput pySeriala, treba preuzeti s njihovog sajta i slediti uputstvo za instalaciju.

1.4 Dokumentacija

Python je jako dobro i detaljno dokumentovan. Za uvodni kurs, dovoljne su *online* reference sa sajta `www.python.org/doc` i sa sajtova pripadajućih modula. Veoma dobri materijali za samostalno učenje mogu se preuzeti sa sajta predmeta Praktikum iz softverskih alata u elektronici (prof. Pejović).

2 Prvi koraci u Pythonu

2.1 Opšte

Za sada radimo terminalski – izvršavamo komandu po komandu u prozoru IDLE, dok ćemo pisanje programa probati nešto kasnije.

Python je *case sensitive* – razlikuje velika i mala slova. Razmake ignoriše. !

Komentar počinje simbolom `#`. Ukoliko komanda (ili izraz) ne može da stane u jedan red, za prelazak u novi treba kucati simbol \.

Pomoć za neku naredbu dobijamo kucanjem `help(`*`naredba`*`)`.

Dodeljivanje vrednosti je standardno – npr. `a = 1.17` ili `X = 8`.

2.2 Vrste podataka od interesa za nas

- brojevi
  - celi (*integers*)
    - obični (*plain integers*) – celi brojevi u opsegu od -2147483648 do 2147483647, ili širem
    - dugi (*long integers*) – celi brojevi u neograničenom rasponu
    - Booleovi (*booleans*) – dve vrednosti koje se predstavljaju kao `False` i `True` (s velikim početnim slovom). !
  - realni (*floating point numbers*) – npr. `7.0` ili `3.185`
  - kompleksni (*complex numbers*) – npr. `3 + 4j`
- nizovi (*sequences*) – konačni uređeni skupovi čiji su indeksi nenegativni celi brojevi
  - niske/stringovi (*strings*) – npr. `'13E033TM'`
  - n-torke (*tuples*)
  - liste (*lists*) – npr. `[1, 9, 10, 78]`
- preslikavanja (*mappings*)
- izvršivi tipovi (*callable types*)
  - funkcije koje definišu korisnici (*user-defined functions*)
  - metode koje definišu korisnici (*user-defined methods*)

Funkcija `type(`*`argument`*`)` vraća kojoj vrsti podataka pripada argument.

2.3 Zapisi celih brojeva

Standardni zapis je u decimalnom brojnom sistemu, npr. `-3`. Pored njega, mogu se koristiti i oktalni (npr. `-0o27` ili `-0O27`), heksadecimalni (npr. `-0x2F` ili `-0X2F`) i binarni sistem (npr. `-0b110` ili `-0B110`).

Za konverziju zapisa u ove sisteme, koriste se redom funkcije `oct`, `hex` i `bin`, npr. `hex(0o472)`.

2.4 Kompleksni brojevi

Imaginarna jedinica označava se simbolom `j` ili `J`. Kompleksni broj se može formirati funkcijom `complex`, npr. `a = complex(1, 2)`, ili se može zadati eksplicitno, npr. `b = 3 + 4j`.

Osnovne funkcije i metode nad kompleksnim brojevima navedene su u tabeli. U primeru je pretpostavljeno da je kompleksni broj smešten u promenljivu `z`.

Oznaka	Opis
`z.real`	realni deo
`z.imag`	imaginarni deo
`abs(z)`	moduo
`z.conjugate()`	konjugovano-kompleksni broj

2.5 Python kao kalkulator – računske operacije

Simbol	Značenje
$+$	sabiranje
$-$	oduzimanje
$*$	množenje
$/$	deljenje*
$**$	stepenovanje
$//$	celobrojno deljenje
$\%$	ostatak pri cel. deljenju

* – Ekvivalentno celobrojnom deljenju ukoliko su oba operanda celi brojevi.

Ukoliko se rezultat operacije dodeljuje jednom od operanada, tada se može koristiti sažeti zapis, npr. `a **= 3` umesto `a = a**3`. !

2.6 Logičke operacije

Operandi su tipa `bool` (`True` ili `False`; u mnogim slučajevima takođe i `1` ili `0`).

Simbol	Značenje
`and`	logičko *i*
`or`	logičko *ili*
`not`	logičko *ne*

2.7 Operatori poređenja

Simbol	Značenje
`==`	je jednako
`!=`	nije jednako
`<>`	nije jednako
`>`	veće od
`<`	manje od
`>=`	veće ili jednako
`<=`	manje ili jednako

3 Korišćenje modula

Mnoge funkcije su sadržane u specijalizovanim bibliotekama, tzv. modulima. Da bi se ovakve funkcije mogle koristiti, odgovarajući modul treba učitati naredbom `import`. Učitani modul se briše naredbom `del`.

Kao primer, razmotrimo korišćenje trigonometrijskih funkcija i konstante π:

```
import math
math.sin(math.pi/4)
```

Očigledno, ovakvo pozivanje funkcija (i konstanti) ne samo da je naporno, već je i programski kôd nepregledan. Obrišimo stoga modul `math` (`del math`) i kucajmo

```
import math as m
m.sin(m.pi/4)
del m
```

Umesto oznake `m`, mogli smo koristiti i bilo koju drugu; ako nam je cilj ušteda vremena (i izbegavanje grešaka u kucanju), bilo bi razumno da oznaka bude kraća od naziva modula. !

Moduo možemo učitati i tako da bude podrazumevani deo radnog prostora:

```
from math import *
sin(pi/4)
```

4 Liste

Listu čine elementi uzajamno odvojeni zapetama, koji se nalaze unutar uglastih zagrada:

```
A = [2, 8, 9.4, -13, 25]
```

4.1 Indeksiranje elemenata liste

Prvi element liste koja ima N elemenata ima indeks 0, drugi indeks 1, treći 2... dok poslednji element ima indeks $N-1$. Elementi se mogu indeksirati i unatrag, tako da poslednji ima indeks -1, pretposlednji -2... i prvi $-N$. Izdvajanje elemenata liste ilustrovano je u narednoj tabeli. !

Oznaka	Značenje
`A[0]`	prvi element
`A[1]`	drugi element
`A[-1]`	poslednji element
`A[-2]`	pretposlednji element
`A[i:j]`	elementi od indeksa i do indeksa $j-1$
`A[0:3]`	elementi od prvog do trećeg
`A[:3]`	elementi od prvog do trećeg
`A[1:3]`	elementi od drugog do trećeg
`A[2:-1]`	elementi od trećeg do pretposlednjeg
`A[1:]`	elementi od drugog do poslednjeg

4.2 Operacije nad listama

Operacija	Opis	Primer upotrebe
`len`	dužina (broj elemenata) liste	`len(A)`
`del`	brisanje liste ili elementa	`del A[2]`
`append`	dopisivanje novog elementa	`A.append([5, 7])`
`extend`	konkateniranje novih elemenata na listu	`A.extend([5, 7])`
`reverse`	obrtanje redosleda elemenata	`A.reverse()`
`sort`	sortiranje elemenata	`A.sort()`
`index`	indeks prvog javljanja elementa	`A.index(5)`
`count`	broj ponavljanja elementa u listi	`A.count(5)`
`remove`	uklanjanje elementa iz liste	`A.remove(5)`
`in`	nalazi li se element u listi	`5 in A`

Lista koja sadrži redom cele brojeve u zadatom opsegu generiše se naredbom `range`. Na primer, `range(7)` vraća listu dužine 7, čiji je prvi element 0, a poslednji 6.

4.3 Nizovi i matrice

Nizovi se mogu shvatiti kao liste. Prazan niz A inicijalizuje se naredbom `A = []` i potom popunjava metodom `append`. Operatorom dodeljivanja vrednosti (`=`) ne može se upisati podatak u nepostojeći član niza. Niz koji ima npr. deset nula generiše se naredbom `[0]*10`.

!

Matrice su višedimenzionalne liste, npr. `A = [[1, 2], [3, 4], [5, 6]]`. Elementima matrica pristupa se analogno kao i elementima lista, npr. `A[1][0]`.

5 Kontrola toka

Za razliku od programskih jezika koji programske blokove razdvajaju pomoću vitičastih zagrada ili ključnih reči (npr. `end`), Python koristi uvlačenje teksta. !

```
# if - elif - else
if a > b:
   print "a je vece od b"
elif b > a:
   print "b je vece od a"
else:
   print "a i b su jednaki"

# for petlja, indeksiranje po listi
W = range(100)
for i in W:
   print 'korak broj', i+1

# for petlja, indeksiranje po stringu
Q = 'programski jezik Python'
for j in Q:
   print j
```

6 Funkcije

Sledeći primer ilustruje definisanje i pozivanje korisničke funkcije.

```
def pdv(osnovica, stopa):
  '''Funkcija koja racuna porez na dodatnu vrednost.
  Argumenti su poreska osnovica i stopa (u %).'''
  return osnovica * (1 + stopa/100.)

pdv(100, 8)       # pozivanje funkcije
help(pdv)         # pomoc
del pdv           # brisanje funkcije
```

7 Stringovi

String (niska) je grupa simbola unutar apostrofa (') ili navodnika ("), npr. `'13E033tm'`. Ukoliko string zauzima više od jednog reda, kao delimiter se koriste tri apostrofa (`'''`), kao što je to bio slučaj u primeru iz odeljka 6.

!

Podatak drugog tipa prevodi se u string naredbom `str`.

Simboli unutar stringa indeksiraju se analogno elementima lista (odeljak 4.1).

7.1 Operacije nad stringovina

Simbol	Opis	Primer upotrebe
`len`	dužina (broj elemenata) stringa	`len(s)`
`+`	konkateniranje	`s + 'abc'`
`*`	ponavljanje	`s * 5`
`[]`	indeksiranje elementa	`s[5]`
`[:]`	izdvajanje opsega	`s[2:7]`
`in`	nalazi li se element u stringu	`'a' in s`

7.2 Metode nad stringovima

Primer primene	Opis
`s.capitalize()`	Kopija stringa `s` u kome je prvi simbol – ukoliko se radi o slovu – napisan kao veliko slovo.
`s.count(`*`target`*`)`	Broj pojavljivanja podstringa *`target`* u stringu `s`.
`s.find(`*`target`*`)`	Indeks prvog pojavljivanja podstringa *`target`* u stringu `s`.
`s.lower()`	Kopija stringa `s` napisana malim slovima.
`s.replace(`*`old`*`, `*`new`*`)`	Kopija stringa `s` u kojoj su – gledano sleva udesno – sva pojavljivanja podstringa *`old`* zamenjena podstringom *`new`*.
`s.split(`*`sep`*`)`	Lista podstringova stringa `s`, za koje je delimiter podstring *`sep`*.
`s.upper()`	Kopija stringa `s` napisana velikim slovima.

7.3 Formatirani izlaz

7.3.1 Funkcija `print()`

S funkcijom `print()` susreli smo se u primerima iz odeljka 5. Ona ispisuje svoj argument na monitoru:

```
n = 5
print(n)
print n      # moze i bez zagrada
```

Funkciji se kao argument može predati više objekata, koji ne moraju biti istog tipa; njihove vrednosti će se ispisati u istom redu, razdvojene razmacima:

```
print(n, 18/5, 'abc?def')
```

7.3.2 Metoda `format`

Format ispisa podataka može se zadati metodom `format`. Metoda se poziva nad stringom koji predstavlja željeni format ispisa, dok joj se kao argument predaju podaci koje treba ispisati: *'format_ispisa'*`.format(`*podaci*`)`.

Format ispisa zadaje se unutar vitičastih zagrada. Na primer, `'{0}:{2};{1}'` znači da se prvi podatak (s indeksom 0) ispisuje na početku, potom sledi dvotačka, pa treći podatak (indeks 2), iza njega tačka-zarez i, na kraju, drugi podatak (indeks 1). Ukoliko se podaci ispisuju redosledom kojim su navedeni – prvi, drugi, treći... – vitičaste zagrade se mogu ostaviti prazne (`'{}:{};{}'`).

Dodatnim parametrima unutar vitičastih zagrada mogu se zadati širina polja za ispis podatka, poravnanje, preciznost i vrsta zapisa broja.

Minimalna širina polja zadaje se prirodnim brojem koji sledi posle simbola : – na primer, `'{0:3},{1:5}'` znači da su za prvi podatak rezervisana tri mesta, a za drugi pet; podaci su pri tome razdvojeni zarezom. Ukoliko je podatak veći, koristiće se onoliko mesta koliko je potrebno da bi se u potpunosti prikazao. Ukoliko se podaci ispisuju redom kojim su navedeni u argumentu metode `format`, vodeća cifra u vitičastim zagradama može se izostaviti, npr. `'{:3},{:5}'`.

Kada je širina polja veća od broja cifara, brojevi se poravnavaju uz desnu ivicu; nasuprot tome, stringovi se podrazumevano poravnavaju iz levu.

Preciznost označava koliko se cifara realnog broja ispisuje pre i posle decimalne tačke; zadaje se prirodnim brojem koji sledi posle širine polja i simbola . – na primer, `'{:8.4}'` znači da je za ispis rezervisano osam mesta, a da se prikazuju četiri cifre.

Vrsta zapisa broja zadaje se slovnom oznakom, prema tabeli. Podrazumevani ispis je u decimalnoj notaciji.

Simbol	Notacija ispisa broja
`b`	binarna
`c`	Unicode simbol
`d`	decimalna
`o`	oktalna
`x`	heksadecimalna, mala slova `a-f`
`X`	heksadecimalna, velika slova `A-F`

Primeri:

```
>>> n = 10
>>> '{:b}'.format(n)
'1010'
>>> '{:c}'.format(n)
'\n'
>>> '{:d}'.format(n)
'10'
>>> '{:o}'.format(n)
'12'
>>> '{:x}'.format(n)
'a'
>>> '{:X}'.format(n)
'A'
```

8 Unos podataka preko komandne linije

Podatak koji se unese preko komandne linije (prompta) čita se u vidu stringa naredbom `raw_input`:

```
x = raw_input('Koji je danas dan?  ')
print('Danas je ' + x + '.')
```

Ako je ulazni podatak druge prirode (npr. broj), potrebno je konvertovati mu format: !

```
y = raw_input('Unesite broj:  ')
z = float(y)**2
print 'Njegov kvadrat je', z
```

9 Rad s datotekama

Naziv trenutnog radnog direktorijuma možemo videti na sledeći način:

```
import os
print os.getcwd()
```

Datoteka se otvara komandom `open`, čiji je prvi argument – naziv datoteke u vidu stringa – obavezan; drugi, opcioni argument, pokazuje otvara li se datoteka samo za pisanje (`'w'` ili `'a'`) ili za čitanje (`'r'`). Ukoliko datoteka već postoji, kada se izabere opcija `'w'`, novi sadržaj se prepisuje preko postojećeg, dok se za opciju `'a'` (*append*) novi sadržaj upisuje na kraj datoteke.

Metode za rad s datotekama navedene su u tabeli.

Metoda	Opis
`f.read(n)`	iz datoteke se čita `n` simbola (ili manje, ukoliko se dođe do kraja), koji se vraćaju u vidu stringa
`f.read()`	datoteka se čita do kraja i pročitani sadržaj se vraća u vidu stringa
`f.readline()`	iz datoteke se čita jedan red i vraća u vidu stringa
`f.readlines()`	datoteka se čita do kraja i sadržaj se vraća u vidu liste redova
`f.write(s)`	string `s` se upisuje u datoteku
`f.close()`	datoteka se zatvara

Primer:

```
f = open('dat.txt', 'w')    # datoteka dat.txt se otvara za pisanje
f.write('novi tekst')       # string 'novi tekst' se upisuje u datoteku
f.close()                   # datoteka se zatvara
```

10 Izvršavanje programa

Neka je Python program sačuvan u vidu datoteke `prog.py`. Iz *komandne linije* ćemo ga pokrenuti na sledeći način:

```
> python prog.py
```

Pod Linuxom može biti neophodno da se u prvoj liniji programa označi putanja do Python interpretera: !

```
#!/usr/bin/python
```

Ako program treba da prihvata ulazne argumente, npr. dva numerička parametra, realni broj `m` i ceo broj `k`, tada treba postupiti na sledeći način:

```
import sys
m = float(sys.argv[1])
k = int(sys.argv[2])
```

Program se sad iz komandne linije poziva uz numeričke vrednosti argumenata, npr.

```
> python prog.py 127.0 -5
```

11 Matematičke funkcije

U nastavku ovoga odeljka, podrazumevaćmo da je učitan moduo `pylab`

```
from pylab import *
```

ili moduli `numpy`, `scipy` i `matplotlib`, koje on sadrži.

11.1 Tabeliranje funkcija

Sve važnije matematičke funkcije – a takođe i konstante – dostupne su u modulu `numpy`. Iako kao argument prihvataju i liste, zbog mogućnosti kasnijeg crtanja njihovih grafika, bilo bi poželjno da argumenti budu nizovi (*arrays*). Postojeću listu možemo konvertovati u niz naredbom `array`, ili možemo napraviti niz nekom od naredbi `zeros`, `ones` ili `arange` (ima značenje kao `array(range)`).

Funkcija `linspace` generiše niz čije su vrednosti ekvidistantno raspodeljene u zadatom opsegu, dok funkcija `logspace` generiše logaritamski raspodeljene vrednosti.

```
x = arange(0, 1, .01)*2*pi      # moze i preko linspace
y = cos(x)
z = sin(x)
```

11.2 Snimanje i učitavanje rezultata

Rezultate matematičkih izračunavanja često je potrebno sačuvati na disku i odatle kasnije ponovo učitati. Preporučeni format je u vidu binarne `.npy` datoteke:

```
np.save('vrednosti_y.npy', y)
w = np.load('vrednosti_y.npy')
```

11.3 Crtanje grafika funkcija

Osnovna naredba je `plot`, čiji su argumenti redom niz vrednosti po apscisi, niz vrednosti po ordinati i, opciono, parametri linije, npr. `plot(x, y, 'b-')`.

Za logaritamsku razmeru po osama, umesto `plot` koriste se naredbe `semilogx`, `semilogy` ili `loglog`.

Grafik se prikazuje naredbom `show()`. Ponavljanjem naredbi za crtanje pre `show`, u istom prozoru može se iscrtati više grafika.

Simboli za opis vrste linije i oznake boja dati su u tabelama na narednoj strani.

Pored koordinata tačaka koje treba prikazati na grafiku, te simbola za specificiranje vrste i boje linije ili markera, naredbe `plot`, `semilogx`, `semilogy` i `loglog` uzimaju i neke opcione argumente.

Simbol	Značenje
`'-'`	puna linija
`'--'`	isprekidana linija
`'-.'`	„tačka-crta”
`':'`	tačkasta linija
`'.'`	tačkasti marker
`'o'`	kružni marker

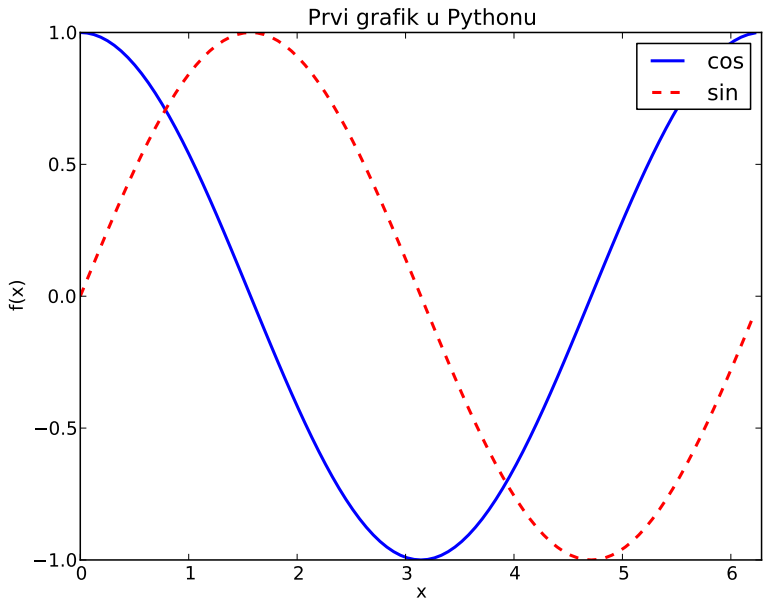
Oznaka	Značenje
`'b'`	plava
`'g'`	zelena
`'r'`	crvena
`'c'`	svetloplava
`'m'`	ljubičasta
`'k'`	crna
`'w'`	bela

Kada se u istom prozoru crta više grafika, poželjno je postaviti legendu u kojoj će biti objašnjeno na koju se funkciju (ili skup podataka) odnosi svaki od njih. To se postiže stringovskim argumentom `label`.

Ukoliko se grafik priprema za štampu, poželjno je podebljati linije zadavanjem veće vrednosti parametra `linewidth`.

Oznake na koordinatne ose postavljaju se naredbama `xlabel` i `ylabel`, a naslov grafika naredbom `title`.

```
plot(x, y, 'b-', label = 'cos', linewidth = 2)
plot(x, z, 'r-', label = 'sin', linewidth = 2)
legend()
xlabel('x')
ylabel('f(x)')
title('Prvi grafik u Pythonu')
xlim([0, 2*pi])    # opseg vrednosti po apscisi
ylim([-1, 1])      # opseg vrednosti po ordinati
show()
```



12 Upravljanje instrumentima

12.1 Uvod

Šta je cilj:

- poslati instrumentu komandu preko računara, kao ekvivalent fizičkom pritiskanju „dugmića" na prednjoj ploči,
- preuzeti rezultate merenja na računar, kao ekvivalent njihovom zapisivanju na papir,
- (kasnije:) obraditi ove rezultate

Kako se to u principu radi:

1. otvorimo i konfigurišemo Python objekat pridružen instrumentu,
2. (opciono) adresiramo instrument,
3. objektu pošaljemo string koji sadrži naredbu/upit,
4. ako je u pitanju upit, pročitamo odgovor, ponovo u vidu stringa,
5. na kraju zatvorimo objekat.

Potrebni moduli:

- pyserial (za RS-232)
- pyusb + usbtmc (USB)
- Python VXI-11 (Ethernet LAN)

Jako važno (iz iskustva):

Pre samog povezivanja, treba detaljno proučiti dokumentaciju instrumenta i razjasniti detalje o potrebnom kablu, parametrima komunikacije, skupu komandi, terminatoru linije i formatu odgovora. !

12.2 Pyserial

12.2.1 Otvaranje i konfigurisanje porta

Pre početka, proveriti koji je kabl potreban, *null-modem* ili *straight-through*. Standardna podešavanja serijskog porta u Pythonu su 9600 bps, 8-N-1. !

Poželjno je zadati razuman `timeout`, da se u slučaju greške program ne bi vreteo u mrtvoj petlji. !

Win, npr. COM2

```
import serial
instr = serial.Serial(1)      # indeksiranje krece od mule!
instr.timeout = 5             # na odgovor se ceka max. pet sekundi
```

Ubuntu, prvi port

```
import serial
par = serial.Serial(0, timeout = 2)      # moze i ovako
```

12.2.2 Slanje i čitanje podatka

Pri radu s modulom `pyserial`, neophodno je završiti svaku naredbu propisanim terminatorom (a to piše u uputstvu za programiranje instrumenta). !

Primer: želimo da na osciloskopu podesimo dvaput veći vertikalni razmer od trenutnog. Za upit trenutnog razmera, koristi se naredba `CHannel<n>:SCAle?`, a za podešavanje razmera `CHannel<n>:SCAle` *`value`*. Terminator je *line feed* (<LF>, \n).

```
# pitamo instrument koliki je tekuci vertikalni razmer kanala 1
instr.write('CH1:SCA?\n')
# procitamo odgovor kao string i konvertujemo ga u realni broj
scale = float(instr.readline())
# novi razmer je dvaput veci
newscale = scale * 2
# razmer se zadaje naredbom CH1:SCA numericka_vrednost
# pravimo string ovog formata i saljemo ga instrumentu
# ne zaboravljamo terminator!
instr.write('CH1:SCA ' + str(newscale) + '\n')
```

12.2.3 Zatvaranje porta

```
instr.close()
```

12.3 USB

Identifikatori instrumenta su *Vendor ID*, *Product ID* i, opciono, *Serial Number*.

Nije potreban terminator. Metoda `ask` šalje upit i čita odgovor. Nema eksplicitnog zatvaranja objekta. !

```
import usbtmc
fg = usbtmc.Instrument(0x0957, 0x1755)
fg.ask('*idn?')
```

12.4 Ethernet LAN

Identifikator instrumenta je njegova IP adresa. Potrebno je na instrumentu – i, eventualno računaru – podesiti kompatibilne maske podmreže i adrese mrežnog prolaza. Ukoliko se instrument priključuje direktno na računar, koristi se *crossover* kabl, u suprotnom *straight-through*.

!

Nije potreban terminator. Metoda `ask` šalje upit i čita odgovor.

!

```
import vxi11
sa = vxi11.Instrument('147.91.10.54')
sa.ask('*idn?')
sa.close()      # zatvaranje objekta
```

13 Obrada rezultata merenja

Neka je `X` niz s N rezultata merenja.

```
from pylab import *
```

13.1 Deskriptivna statistika

Funkcija	Opis
`min(X)`	minimalna vrednost
`max(X)`	maksimalna vrednost
`mean(X)`	matematičko očekivanje
`median(X)`	medijana
`var(X)`	varijansa, u imeniocu N
`var(X, ddof = 1)`	varijansa, u imeniocu $N-1$
`std(X)`	standardna devijacija, u imeniocu N
`std(X, ddof = 1)`	standardna devijacija, u imeniocu $N-1$
`histogram(X, bins = k)`	histogram s k „stubića”

13.2 Vizuelizacija raspodele

Funkcija	Opis
`boxplot(X)`	boks-dijagram
`hist(X, bins = k)`	histogram s k „stubića”

13.3 Polinomska regresija

Neka je `X` niz s vrednostima nezavisne i `Y` niz s vrednostima zavisne promenljive. Koeficijenti polinoma `p(x) = p[0] * x**deg + ... + p[deg]`, koji u smislu minimalne srednje kvadratne greške opisuje zavisnost `Y = p(X)` računaju se kao

```
p = polyfit(X, Y, deg)
```

Za potrebe grafičkog prikaza, regresioni polinom možemo jednostavno tabelirati na sledeći način:

```
fit_fn = poly1d(p)
Y_fit = fit_fn(X)
```

14 Obrada telekomunikacionih signala

Uzorci signala grafički se predstavljaju naredbom `stem`. Specijalizovane funkcije vezane za DSP dostupne su u modulu `scipy.signal`:

Funkcija	Opis
`convolve(x, y)`	izračunavanje konvolucije signala `x` i `y`
`correlate(x, y)`	izračunavanje korelacije signala `x` i `y`
`hilbert(x)`	izračunavanje analitičkog signala pomoću Hilbertove transformacije
`decimate(x, q)`	decimiranje signala `x` za faktor `q`
`resample(x, n)`	interpoliranje signala `x` do `n` uzoraka

14.1 Spektralna analiza

Jednodimenzionalna diskretna Fourierova transformacija računa se naredbom `fft` iz modula `pylab`.

U modulu `scipy.signal` dostupne su i sledeće funkcije:

`periodogram` – ocena spektralne gustine snage metodom periodigrama,

`spectrogram` – izračunavanje spektrograma metodom uzastopnih Fourierovih transformacija.

14.2 Projektovanje filtara

Sledeće funkcije dostupne su u modulu `scipy.signal`:

`y = lfilter(b, a, x)` – filtriranje signala `x` kroz filtar čiji su koeficijenti polinoma u brojniku funkcije prenosa `b`, a polinoma u nazivniku `a`;

`w, h = freqz(b, a=1, worN=None, whole=0, plot=None)` – izračunavanje frekvencijskog odziva filtra. Koeficijenti polinoma u brojniku su `b`, a u nazivniku `a` (podrazumevana vrednost je `a = 1`). Treći argument je opcioni i označava broj tačaka u kojima se računa frekvencijski odziv (podrazumevana vrednost je 512). četvrti argument označava računaju li se vrednosti ferkvencija na gornjoj polovini jediničnog kruga (`0`), ili na celom krugu (`True`). Poslednji argument označava poziva li se odmah crtanje grafika frekvencijskog odziva, ili ne. Izlazi funkcije su vektor normalizovanih frekvencija, `w` (u radijanima po uzorku) i frekvencijski odziv, `h`;

`w, gd = group_delay((b, a), w=None, whole=False)` – izračunavanje grupnog kašnjenja digitalnog filtra. Značenja argumenata: `b`, `a` - koeficijenti polinoma u brojniku i nazivniku funkcije prenosa; `w` - opcioni argument, ako je `None`, frekvencije se uzimaju u

512 ekvidistantno raspoređenih tačaka, ako je prirodan broj, frekvencije se računaju u toliko tačaka, ako je niz, kašnjenje se računa na tim vrednostima; `whole` - opcioni argument, kao za funkciju `frequz`. Izlazi: vektor normalizovanih frekvencija, `w` (u radijanima po uzorku) i grupno kašnjenje, `gd`.

14.2.1 FIR filtri

Osnovna funkcija za projektovanje FIR filtra je `firwin` iz modula `scipy.signal`. Njen prvi argument je broj koeficijenata filtra (red filtra uvećan za jedan), a drugi su normalizovane frekvencije koje odgovaraju granicama propusnih/nepropusnih opsega. Funkcija vraća koeficijente filtra. Ako je red filtra paran, filtar je tipa I; u suprotnom je tipa II.

Funkcija	Opis
`firwin(N, f)`	NF filtar, Hammingov prozor
`firwin(N, f, window = 'blackman')`	NF filtar, Blackmanov prozor
`firwin(N, f, pass_zero = False)`	VF filtar, Hamming
`firwin(N, [f1, f2], pass_zero = False)`	propusnik opsega, Hamming
`firwin(N, [f1, f2], window = 'bartlett')`	nepropusnik opsega, Bartlett

Funkcijom `remez` projektuje se minimax optimalni filtar, prema Remez algoritmu.

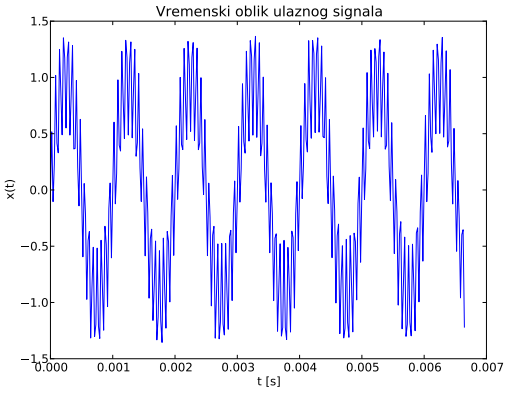
Primer: Generisaćemo $n = 320$ uzoraka signala

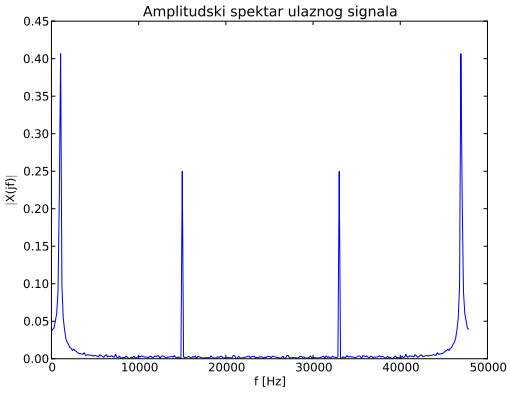
$$x(t) = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + n_G(t),$$

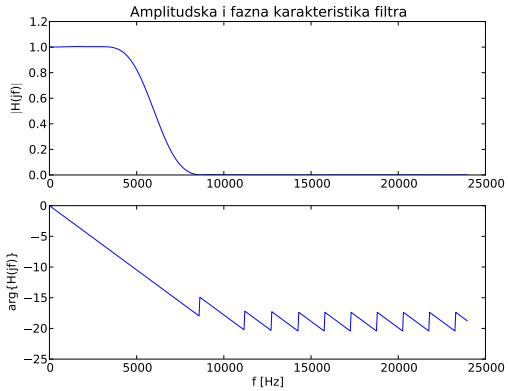
gde je $A_1 = 1$, $f_1 = 1$ kHz, $A_2 = 0.5$ i $f_2 = 15$ kHz i $n_G(t)$ Gaußov šum srednje snage $P_n = 1$ mW, uzetih s frekvencijom $f_s = 48$ kHz. Projektovaćemo Hammingov NF FIR filtar 32. reda, granične frekvencije propusnog opsega $f_c = 6$ kHz i propustiti signal $x(t)$ kroz njega.

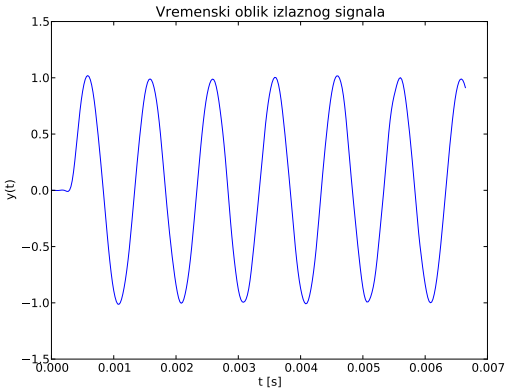
```
from pylab import *
from scipy.signal import firwin, lfilter, freqz

# vrednosti parametara
fs = 48000.
n = 320
f1 = 1000.
A1 = 1.
f2 = 15000.
A2 = 0.5
Pn = 1.e-3
fc = 6000.
N = 32

t = arange(n)/fs
```

```
x = A1*sin(2*pi*f1*t) + A2*sin(2*pi*f2*t) + sqrt(Pn)*randn(len(t))
figure(1)
plot(t, x)
xlabel('t [s]')
ylabel('x(t)')
title('Vremenski oblik ulaznog signala')
figure(2)
plot(arange(n)*fs/n, abs(fft(x)/n))
xlabel('f [Hz]')
ylabel('|X(jf)|')
title('Amplitudski spektar ulaznog signala')
show()

b = firwin(N+1, fc/(fs/2))
w, h = freqz(b, 1)
subplot(211)
title('Amplitudska i fazna karakteristika filtra')
plot(w*.5*fs/pi, abs(h))
ylabel('|H(jf)|')
subplot(212)
plot(w*.5*fs/pi, unwrap(arctan2(imag(h),real(h))))
xlabel('f [Hz]')
ylabel('argH(jf)')
show()

y = lfilter(b, 1., x)
plot(t, y)
xlabel('t [s]')
ylabel('y(t)')
title('Vremenski oblik izlaznog signala')
show()
```









14.2.2 IIR filtri

Osnovna funkcija za projektovanje IIR filtara je `iirdesign` iz modula `scipy.signal`:

```
iirdesign(wp, ws, gpass, gstop, analog=False, ftype='ellip', output='ba')
```

Njeni argumenti su, redom:

- granične kružne učestanosti propusnih (`wp`) i nepropusnih opsega (`ws`), normalizovane na Nyquistovu kružnu učestanost;
- maksimalno slabljenje (u decibelima) u propusnom (`gpass`) i nepropusnom opsegu (`gstop`);
- vrsta filtra – analogni (`analog = True`), ili digitalni (u suprotnom);
- aproksimacija funkcije prenosa – Butterworthova (`'butter'`), čebiševljeva I vrste (`'cheby1'`), čebiševljeva II vrste (`'cheby2'`), Cauerova (`'ellip'`), ili Bessel-Thomsonova (`'bessel'`);
- vrsta izlaza – brojnik i nazivnik (`'ba'`), nule, polovi i dobitak (`'zpk'`) ili sekcije drugog reda (`'sos'`).

Na drugi način, IIR filtri se projektuju u dva koraka, tako što se u prvom odredi red filtra kojim se postiže željeni gabarit, dok se u drugom koraku određuje funkcija prenosa:

1.	2.	Tip filtra
`buttord`	`butter`	Butterworthov
`cheb1ord`	`cheby1`	čebiševljev I vrste
`cheb2ord`	`cheby2`	čebiševljev II vrste
`ellipord`	`ellip`	Cauerov (eliptički)

Izuzetak su Bessel-Thomsonovi filtri, koji se projektuju u jednom koraku, naredbom `bessel`.

15 Programiranje mrežnih aplikacija

Moduo `socket` pruža funkcionalnosti za pristup mrežnom interfejsu.

Osnovna funkcija je `socket.socket([family[, type[, proto]]])`, kojom se pravi novi *socket* objekt. Argumenti su joj, redom:

- adresna familija – AF_INET (podrazumevana), AF_INET6 ili AF_UNIX; na Linux platformama i AF_PACKET i PF_PACKET;
- vrsta *socketa* – SOCK_STREAM (podrazumevana), SOCK_DGRAM ili neka od raspoloživih SOCK_ konstanti, prema dokumentaciji (npr. SOCK_RAW);
- numerička oznaka protokola (najčešće nula i može se izostaviti).

15.1 Metode za rad nad objektima

`socket.accept()` – Prihvata konekciju. Potrebno je da *socket* bude povezan s adresom i da osluškuje konekcije. Vraća se par `(conn, address)`, gde je `conn` novi *socket* objekt koji može da šalje i prima podatke po konekciji, a `address` je adresa povezana sa *socketom* na drugoj strani konekcije.

`socket.bind(address)` – Povezuje *socket* koji prethodno nije imao adresu s adresom. Format adrese zavisi od adresne familije.

`socket.close()` – Zatvara se *socket*.

`socket.connect(address)` – Uspostavlja se konekcija ka udaljenom *socketu* na specificiranoj adresi. Format adrese zavisi od adresne familije.

`socket.listen(backlog)` – „Osluškuje" se dolazna konekcija. Argumentom `backlog` zadaje se maksimalan broj simultanih konekcija.

`socket.recv(bufsize[, flags])` – Prima se podatak sa *socketa* i predstavlja stringom. Argumentom `bufsize` zadaje se maksimalna količina podataka koji se mogu odjednom pročitati. Preporuka je da to bude relativno mali broj stepena 2, npr. 4096. Argument `flags` je opcion i podrazumevana vrednost mu je nula.

`socket.recvfrom(bufsize[, flags])` – Kao za metodu `recv()`, s tim što se sada vraća par `(string, address)`. `String` je primljeni podatak, a `address` je adresa *socketa* koji ga je poslao. Format adrese zavisi od adresne familije.

`socket.send(string[, flags])` – Šalje se podatak na *socket*, koji prethodno mora biti povezan na udaljeni *socket*, a vraća se broj poslatih bajtova. Opcioni argument `flags` ima isto značenje kao za metodu `recv()`.

`socket.sendto(string, flags, address)` – Kao prethodna metoda, s tim što se sada zadaje adresa odredišta.

15.2 Primeri

15.2.1 Raw Ethernet

Direktan pristup mrežnom interfejsu moguć je iz Linuxa. Potrebno je da računari koji komuniciraju na ovaj način budu unutar istog kolizionog domena.

```
from socket import *
s = socket(AF_PACKET, SOCK_RAW)
s.bind(("eth0", 0))      # interfejs je eth0
# pretpostavka je da promenljiva packet sadrzi
# validan zapis Ethernet okvira
# okvir se moze napraviti npr. modulom dpkt
s.send(packet)     # slanje
s.recv(2048)     # prijem
s.close()
```

15.2.2 UDP

Klijentski program:

```
from socket import *
serverName = '78.30.142.18' # npr.
serverPort = 12000          # npr.
clientSocket = socket(AF_INET, SOCK_DGRAM)
# poruka za slanje je u promenljivoj message
clientSocket.sendto(message, (serverName, serverPort))
# prijem odgovora:
response, serverAddress = clientSocket.recvfrom(2048)
clientSocket.close()
```

Server:

```
from socket import *
serverPort = 12000    # npr, isto kao za klijent
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
while True:
  message, clientAddress = serverSocket.recvfrom(2048)
  # odgovor je u promenljivoj answer
  serverSocket.sendto(answer, clientAddress)
```

Pošto je UDP nekonektivni protokol, svejedno je hoćemo li pri testiranju prvo pokrenuti serversku ili klijentsku aplikaciju.

15.2.3 TCP

Klijent:

```
from socket import *
serverName = '78.30.142.18' # npr.
serverPort = 12000          # npr.
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
# poruka za slanje je u promenljivoj message
clientSocket.send(message)
# prijem odgovora:
response = clientSocket.recv(2048)
clientSocket.close()
```

Server:

```
from socket import *
serverPort = 12000 # npr, isto kao za klijent
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('', serverPort))
serverSocket.listen(1) # max. 1 konekcija
while True:
   # prihvata se dolazna konekcija
   connectionSocket, addr = serverSocket.accept()
   # prima se podatak
   message = connectionSocket.recv(2048)
   # odgovor je u promenljivoj answer
   connectionSocket.send(answer)
   connectionSocket.close()
```

TCP je konektivni protokol, pa je stoga neophodno prvo pokrenuti serversku aplikaciju.

Sadržaj

dr Milan Bjelica,
Univerzitet u Beogradu – Elektrotehnički fakultet
e–mail: milan@etf.rs

PROGRAMSKI JEZIK PYTHON:
SKRIPTA ZA STUDENTE TELEKOMUNIKACIJA
elektronski pomoćni udžbenik

Recenzenti:

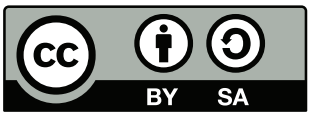
prof. dr Aleksandra Smiljanić,
doc. dr Mirjana Simić-Pejović

Nastavno-naučno veće Elektrotehničkog fakulteta odobrilo je objavljivanje ovoga nastavnog materijala odlukom broj 1313/4 od 12.7.2016. godine.

Izdaje i štampa:
Elektrotehnički fakultet
Beograd, 2018.

Tiraž: 50 primeraka

ISBN: 978-86-7225-058-9



Tekst ove knjige složen je u programskom paketu LaTeX 2_ε.

CIP - Каталогизација у публикацији - Народна библиотека Србије, Београд
004.43PYTHON(0.034.2)
БЈЕЛИЦА, Милан, 1977-
Programski jezik Python [Elektronski izvor] : skripta za studente telekomunikacija : [elektronski pomoćni udžbenik] / Milan Bjelica. - Beograd : Univerzitet, Elektrotehnički fakultet, 2018 (Beograd : Univerzitet, Elektrotehnički fakultet). - 1 elektronski optički disk (CD-ROM) ; 12 cm. - (Udžbenik Elektrotehničkog fakulteta u Beogradu)
Sistemski zahtevi: Nisu navedeni. - Nasl. sa naslovne strane dokumenta. - Tiraž 50.
ISBN 978-86-7225-058-9
a) Програмски језик "Python"
COBISS.SR-ID 258999820