

Osnovi računarstva II

Časovi 12 i 13

Miloš Daković

Elektrotehnički fakultet – Podgorica

13. maj 2024.

Programski jezik Python

- Python je programski jezik visokog nivoa, opšte namjene.
- Python programi se interpretiraju, slično kao Octave/MATLAB skript fajlovi.
- Python programi su koncizni i pregledni.
- Python je open-source, lako dostupan na raznim hardverskim platformama.
- Postoji veliki broj modula za obavljanje standardnih ili specifičnih zadataka.

Okruženja za rad sa Pythonom:

- Komandna linija
- IDLE, PyCharm, Jupyter, Anaconda
- Google Colab, XCode, Visual Studio Code, Eclipse (PyDev), ...

Objekti

Python je objektno orjentisani jezik. Osnovni objekti su:

- Cijeli brojevi `135` `-658000`
- Decimalni brojevi `25.384` `6.02e+19`
- Kompleksni brojevi `1+3j` `2.17-0.15j`
- Stringovi `'abcd'` `"Slova 2"`
- Logičke vrijednosti `True` `False`

Dodatno imamo na raspolaganju:

- n-torke `(15, 2, 8)` `('ponedjeljak', 25, 3.14)`
- liste `[7, 5, 18]` `['utorak', 25, "a", 3.14]`
- skupove `{5, 55, 105}`
- rječnike `{"K1":15, "K2": 25, "Ispit":42}`
- funkcije `def našaFunkcija(x, y):`
- klase `class našaKlasa :`

Imena – varijable

Imena u Pythonu su oznake (pokazivači) na objekte.

- Izvršavanjem naredbe `a=3` kreira se novi objekat (broj 3) i dodjeljuje mu se ime `a` koje pokazuje na njega.
- Komandom `print(a)` prikazujemo sadržaj objekta na koji pokazuje ime `a`.
- Ako nakon toga izvršimo komandu `a=7`, kreira se novi objekat (broj 7) i ime `a` pokazuje na njega. Prethodni objekat (broj 3) ostaje u memoriji i sistem vodi računa o tome da, nakon nekog vremena oslobodi memorijske lokacije na koje ne pokazuje nijedno ime.
- Komanda `a=a+1` kreiraće novi objekat (broj 8) i postavi da ime `a` pokazuje na novokreirani objekat.
- U python-u je tip podataka vezan za objekat a ne za ime (varijablu).

Imena i objekti

- Ovo ponašanje je različito od situacije koju smo imali u Octave/MATLAB okruženju gdje se varijabla odnosi na konkretnu memorijsku lokaciju. Octave komande: **$a=3$** ; **$a=7$** ; **$a=a+1$** jednom zauzmu memoriju za varijablu a i sve operacije (promjenu sa 3 na 7, uvećanje za 1 rade u okviru te memorijske lokacije).
- Objekti mogu biti promjenljivi (*mutable*) ili nepromjenljivi (*immutable*).
- Osnovni objekti tipa: cijeli broj, decimalni broj, kompleksan broj, string, Logička vrijednost i n-torka su nepromjenljivi tipovi podataka.
- Liste, skupovi i rječnici su primjer promjenljivih tipova podataka.
- Iako komandom **$a=314$** kreiramo nepromjenljivi objekat (broj 314) komandom **$a=a+10$** kreiramo novi nepromjenljivi objekat (broj 324) i njemu dodjeljujemo ime a .

Operatori

- Operator za dodjelu imena objektu je =
- Operacije sa cijelim i decimalnim brojevima:

- Osnovne aritmetičke operacije + - * /

$$2+3*7-9/3 \rightarrow 20$$

- Stepenovanje **

$$2**10 \rightarrow 1024 = 2^{10}$$

- Cjelobrojno dijeljenje //

$$20//3 \rightarrow 6$$

- Ostatak pri dijeljenju %

$$20\%3 \rightarrow 2$$

- Operacije sa kompleksnim brojevima:

- + - * / **

- Operacije sa stringovima:

- Spajanje stringova +

$$\text{"abcd"}+\text{"FG"} \rightarrow \text{abcdFG}$$

- Ponavljanje stringa *

$$3*\text{"Abc"} \rightarrow \text{AbcAbcAbc}$$

- Moguće je koristiti operatore dodjele imena kombinovane sa aritmetičkim operatorima: $a+=7 \rightarrow a=a+7$, $a*=3 \rightarrow a=a*3$, kao i višestruko imenovanje $b=c=d=8$.

n-torke (eng. *tuples*)

- Definišu se sa običnim zagradama, elementi su odvojeni zarezom.

$$a = (1, 2, 4)$$

uređena trojka

$$b = 1, 2, 4$$

zagrade nijesu neophodne

$$c = (a, b)$$

objekti kao elementi

$$d = (4,)$$

n-torka sa jednim elementom

Iz primjera vidimo da zagrade nijesu striktno neophodne, ali su poželjne. Zapeta je u poslednjem primjeru neophodna da bi se n-torka od jednog elementa razlikovala od objekta: cijeli broj 4.

- n-torke možemo koristiti i za dodjelu imena:

$$a, b = 5, 7$$

$$(a, b, c) = (1, 3, 5)$$

- Operacije sa n-torkama:

- dodavanje (nadovezivanje) $(2, 3) + (5, 7, 9) \rightarrow (2, 3, 5, 7, 9)$
uočite da će $2, 3+5, 7, 9$ dati drugačiji rezultat $\rightarrow (2, 8, 7, 9)$
- ponavljanje $3 * (4, 5) \rightarrow (4, 5, 4, 5, 4, 5)$

Liste

- Definišemo ih korišćenjem uglastih zagrada
`m = [1, 2, 5]`
- Operacije sa listama su `+` i `*`. One funkcionišu na isti način kao i operacije sa n-torkama
- Osnovna razlika je da liste sadrže objekte koji mogu da se mijenjaju dok su objekti u n-torkama nepromjenljivi.
- Funkcije za rad sa listama (i n-torkama):

```
len(m)           # dužina liste
min(m)           # najmanji element liste
max(m)           # najveći element liste
```

- Operatori `in` i `not in`

```
5 in m           # Tačno
3 not in m       # Tačno
7 in m           # Netačno
```


Indeksiranje

- Nekom elementu liste ili n-torke možemo pristupiti preko njegovog indeksa:

$a = (3, 5, 7)$ # n-torka
 $b = [2, 4, 5, 8]$ # lista
 $a[2]$ # daje odgovor 7
 $b[1]$ # daje odgovor 4

- Indeksiranje kreće od indeksa 0. Prvi element je $a[0]$
- Indeks može biti negativan. Tada se računa „od kraja“, odnosno $a[-1]$ je poslednji element n-torke a , dok je $b[-2]$ pretposlednji element liste b .
- Ako je dužina n-torke ili liste n tada se indeks može kretati u granicama $-n, -n + 1, \dots, 0, 1, \dots, n - 1$ Pokušaj indeksiranja van ovih granica će uzrokovati grešku.

Operator : i indeksiranje

Neka je data lista (ili n-torka) **a=[5, 2, 8, 7, 3, 4, 8]**

- **a[2:5]** vraća listu (n-torku) sa indeksima 2,3, i 4. Uočite da element sa indeksom pet nije uključen u rezultat.
- **a[3:]** vraća elemente sa indeksima 3,4,... do kraja liste.
- **a[:4]** daje elemente sa indeksima 0,1,2 i 3.
- **a[:]** daje sve elemente liste.
- **a[1:7:2]** daje elemente sa indeksima 1,3 i 5 (korak u indeksiranju je 2)

Indeksiranje se može koristiti i u stringovima:

```
s = "abcdABCDmnpq"  
s[0]           # a  
s[-2]         # p  
s[3:7]        # dABC  
s[7:]         # Dmnpq
```

Metodi za rad sa listama

Primjeri metoda koje možemo primijeniti na liste:

```
a = [1, 9, 4, 7, 3, 9]
a.sort()
print(a)
a.reverse()
a.index(3)
a.count(9)
a.insert(2, 17)
a.extend([2, 5, 8])
c = a
b = a.copy()
b.append(22)
a.clear()
print(c)
print(b)
```

sortira listu
prikazuje sadržaj liste
okreće listu
pozicija elementa 3 u listi (prva od početka)
koliko se puta element 9 pojavljuje
na poziciji 2 umeće element 17
proširuje listu novim elementima
c je samo drugo ime za listu a
kopira sadržaj liste u novu listu b
dodaje element na kraj liste
briše sve elemente iz liste
šta će biti prikazano?
šta će biti prikazano?

Skupovi

- Kreiramo ih korišćenjem velikih zagrada `{ }`.
- Skup je neuređena kolekcija elemenata.
- Unutar skupa, elementi se ne mogu ponavljati.
- Sa skupovima možemo obavljati operacije unije (operator `|`), presjeka (operator `&`), skupovne razlike (operator `-`) ili simetrične razlike (operator `^`).
- Utvrđivanje da li se neki element nalazi ili ne nalazi u skupu realizujemo operatorima `in` i `not in`.

```
A = {1, 2, 5, 7, 11}
B = {"a", 5, 7, "b", 9}
print(A | B)
print(A & B)
print(B - A)
```

Rječnici

- Kreiramo ih korišćenjem velikih zagrada `{}`.
- Elementi rječnika su parovi oblika ključ:vrijednost. Odvajamo ih zarezima.
- Ključ mora biti nepromjenljivi objekat.
- Ključ je u rječniku jedinstven (ne možemo imati više vrijednosti za isti ključ).
- Rječnik je neuređen skup parova ključ:vrijednost.

```
r = {6:"šest", 8:"osam", "A":"odličan", "C":"dobar"}
r["A"]           # odličan
r[8]             # osam
r.get("C")       # dobar
r.get("D")       # vraća vrijednost None – ništa
r["D"]          # greška: nema ključa "D"
```

Python programi (skript fajlovi)

- Python program je niz naredbi zapisanih u tekstualnom fajlu (podrazumijevana ekstenzija fajla je **.py**), na primjer **test.py**
- Programe možemo pisati u bilo kom editoru. Jednostavnije je koristiti editore sa ugrađenom podrškom za python sintaksu. U windows okruženju možemo u komandnoj liniji otkucati:
notepad test.py
- Jedan od načina startovanja programa je da iz komandne linije startujemo python okruženje i kao argument navedemo ime programa koji želimo izvršiti, na primjer:
python test.py
- IDLE okruženje ima ugrađeni editor gdje možemo pisati programe i izvršavati ih direktno iz editora

Ulazne i izlazne naredbe

- Osnovna naredba za izlaz podataka od programa ka korisniku je funkcija **print**. Kao argument joj se prosljeđuje jedan (ili više objekata) koje želimo poslati na izlaz:

```
a=15; b=[3, 5, 8]
print("Rezultat:", a, b, 7*a, 2*b)
```

- Ulaz podataka obezbjeđujemo funkcijom **input**. Njen argument je tekst koji je potrebno ispisati korisniku (objašnjenje), a njena povratna vrijednost je string koji je korisnik unio kao odgovor.

```
ime = input("unesite ime: ")
ocjena = int(input("unesite ocjenu: "))
print(ime, ocjena)
```

- Ako unosimo numeričku vrijednost (broj), tada string sa ulaza treba konvertovati u broj nekom od funkcija: **int()**, **float()**, **complex()**, **eval()**, ...

Kontrola toka programa - if naredba

Ovom naredbom se implementira algoritamski korak selekcije. Imamo blok naredbi koje treba izvršiti ako je uslov tačan i eventualno blok naredbi koje treba izvršiti ako je uslov netačan.

Blokove u Pythonu definišemo uvlačenjem redova koji čine blok.

Preporučeno uvlačenje je 4 razmaka. Na početku bloka je naredba koja definiše zaglavlje bloka. Ona se obavezno završava simbolom `:`.

```
if x>10:           # ako je x veće od 10
    x -= 10       # umanji x za 10
    y = 1         # postavi y na 1

if z<21:          # ako je z manje od 21
    z = z+1       # tada uvećaj z za 1
else:
    z = z-1       # u suprotnom, umanji z za 1
```


If naredba sa više uslova

```
if x<0:                # prvi uslov
    y = 0
elif x<10:            # ako prvi uslov nije ispunjen provjeravamo drugi uslov
    y = x**2
elif x<20:            # samo ako prethodna dva uslova nijesu ispunjeni
    y = 110-x
else:                  # ako nijedan od uslova nije ispunjen
    y = 90
```

Operatori poređenja su `==`, `<`, `>`, `<=`, `>=` i `!=`. Logičke uslove možemo kombinovati korišćenjem logičkih operatora **and**, **or** i **not**.

```
if x<0 or x>=100 or x==10:
    y = 0
elif 25<x<50 and x!=30:
    y = 1
```

For petlja

Koristi se za prolazak kroz objekte tipa n-torke, liste ili rječnika.

Varijabla uzima sve moguće vrijednosti iz zadatog opsega i blok koda nakon for naredbe se izvršava za svaku od vrijednosti varijable.

```
for x in (5,10,15):  
    print(x)
```

```
for y in [1,2,4,8,16]:  
    print(y,2**y)
```

```
for c in "ABCDE":  
    print(c, ord(c))
```

```
r = {6:"E",7:"D",8:"C",9:"B",10:"A"}  
for m in r:  
    print(m, r[m])
```

While petlja

Ova petlja se izvršava sve dok je uslov naveden u petlji tačan.

```
x = 10; y = 0      # više naredbi možemo staviti u jednu liniju odvojene sa ;
while x>0:        # sve dok je x veće od nule
    y += x        # uvećaj y za x
    x -= 1        # smanji x za 1
print(y)
```

Prijevremeni izlaz iz petlje (for ili while) može se postići komandom **break**. Obično je ona unutar neke **if** naredbe.

Funkcija **range** vraća listu cijelih brojeva. Može biti korisna u for petljama kada želimo napraviti klasičnu brojačku petlju.

```
for n in range(10):
    print(n)

for n in range(10, 30):          # probajte: range(10,30,2)
    print(n)
```

Funkcije

Započinju ključnom riječju **def**, nakon toga slijedi ime funkcije, u zagradama navedeni argumenti funkcije i simbol **:**. Poslije toga ide blok koji definiše tijelo funkcije. Naredbom **return** vraćamo rezultat.

```
def ff(x, y=10):  
    a = x + y*5  
    return a
```

Funkciju možemo pozivati: **ff(7)**, **ff(7, 5)**, **ff(y=2, x=4)**

```
def cif(x):  
    c = []  
    while x>0:  
        c.append(x%10)  
        x = x // 10  
    return c
```

Ispitajte šta je rezultat: **cif(721558)**

Praktični savjeti za korišćenje pythona na ETF-u

- Postoji velika šansa da je python već instaliran na računaru.
- Otvorite komandnu liniju (**cmd**, **Terminal**, ...) i unesite komandu: **python** ili **python3**.
- Ako izvršenjem ove komande ne dobijete python okruženje, daće vam se uputstvo kako da izvršite instalaciju.
- Alternativno, možete u windows okruženju pokušati da startujete IDLE aplikaciju. Ako je python instaliran, dobićete komandnu liniju u suprotnom, mogućnost da instalirate IDLE (i python).
- Ako na računaru nemate instaliran neki moduo (biće obrađeni u nastavku), odnosno ako dobijete grešku unosom komande tipa **import numpy as np**, potrebno je instalirati moduo.
- To možemo postići komandom: **python -m pip install numpy** ili **pip install numpy** (u komandnoj liniji operativnog sistema)

Zadaci za vježbu I

- 1 Napišite python program koji računa zbir cifara svih prirodnih brojeva od 1 do 100.
- 2 Napišite python funkciju *newtonsqrt* koja računa kvadratni korijen zadatog broja Njutnovim algoritmom (algoritam je dat u prezentaciji sa II termina nastave).
- 3 Napišite python funkciju *pon* koja kao argument uzima listu i vraća listu u kojoj nema ponavljanja elemenata, a redoslijed odgovara pojavljivanju elementa u originalnoj listi. Na primjer:
pon([1, 3, 7, 5, 2, 3, 1, 8, 7]) vraća *[1, 3, 7, 5, 2, 8]*
- 4 Koristeći se python-om izračunajte:

$$S = \sum_{k=0}^{1000} \frac{k}{k^2 + 1}$$

Zadaci za vježbu II

- 5 Napišite python program koji odgovara pseudo-kodu:

```
START
N, k : INTEGER
S : FLOAT
INPUT N
S = 0
FOR k = 1, N
    S = S + (k+1) / (3*k+2)
NEXT
WHILE S > 10
    S = S - 10
ENDWHILE
OUTPUT S
END
```

Moduli – kreiranje

Moduo je skript fajl u kojem su definisani neki objekti (konstante, funkcije, klase...). Na primjer, možemo napisati jednostavan moduo i sačuvati ga pod nazivom **testmod.py**:

```
x = 42
s = "Primjer"
def sc(x):
    return sum(cif(x))
def cif(x):
    c = []
    while x>0:
        c.append(x%10)
        x = x // 10
    return c
```

U njemu su definisane dvije funkcije, *sc* i *cif* i dvije varijable *x* i *s*.

Moduli – uključivanje u druge programe

Modulo možemo uključiti u naše okruženje i koristiti se njegovim objektima:

```
import testmod
print(testmod.x)
testmod.sc(759)
print(testmod.cif(2024))
print(testmod.s*5)
```

Modulu možemo dodijeliti alternativno (kraće) ime:

```
import testmod as tm
print(tm.x)
tm.sc(759)
print(tm.cif(2024))
print(tm.s*5)
```

Moduli – uvoz određenih objekata iz modula

Specijalno ako nam trebaju samo neki objekti iz modula, možemo ih uvesti direktno u naše okruženje. Uočite da u ovom slučaju nije potrebno navoditi ime modula kad koristimo ove objekte.

```
from testmod import x, sc
from testmod import cif as cifre_broja
print(x)
sc(759)
print(s)           # dobijamo poruku greške jer nijesmo uvezli s
print(cifre_broja(2024))
```

Ako želimo uvesti sve objekte iz modula, to možemo uraditi sa:

```
from testmod import *
```

Standardna biblioteka i dodatni paketi

To su moduli i paketi (kolekcija modula) koji su uključeni u standardnu python instalaciju. Neki moduli standardne biblioteke su:

- ***math*** – matematičke funkcije *sin*, *acos*, *log10*, ...
- ***cmath*** – matematičke funkcije za rad sa kompleksnim brojevima
- ***string*** – dodatne funkcionalnosti u radu sa stringovima
- ***datetime*** – rad sa datumima
- ***os*** – interakcija sa operativnim sistemom

Pored ovih modula često se u inženjerstvu koriste i moduli koje treba posebno instalirati. Na primjer:

- ***numpy*** – rad sa nizovima, matricama, ...
- ***scipy*** – optimizacije, obrada signala, statistika, ...
- ***sympy*** – rješavanje matematičkih problema u simboličkom obliku
- ***matplotlib*** – grafičko prikazivanje podataka

Primjer – grafik funkcije u python-u

Nacrtajmo grafik funkcije $y = f(x)$ na intervalu $-7 \leq x \leq 7$.

$$f(x) = \sqrt{\frac{1 + |x^2 - 1|}{1 + x^2}}$$

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-7, 7, 300)
y = np.sqrt((1 + np.abs(x**2 - 1)) / (1 + x**2))

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y=f(x)')
plt.title('Grafik funkcije')
plt.show()
```

Primjer – rješavanje sistema jednačina

Riješimo sistem jednačina,
odredimo inverznu matricu
i determinantu matrice sistema.

$$\begin{aligned}4x + y + z &= 5 \\ 2x - y &= -1 \\ 2y + 3z &= 7\end{aligned}$$

```
import numpy as np
A = np.array([[4, 1, 1], [2, -1, 0], [0, 2, 3]])
B = np.array([5, -1, 7])

r = np.linalg.solve(A, B)
print('Rješenje: ', r)
Ai = np.linalg.inv(A)
print('Inverzna matrica:\n', Ai)
d = np.linalg.det(A)
print('Determinanta: ', d)
```

Primjer – rješavanje u simboličkom obliku

Data je funkcija: $f(x) = (x^2 + 1) \sin^2(x)$.

Izračunajmo integral: $\int_0^{\pi} f(x) dx$ Odredimo izvod: $\frac{df(x)}{dx}$

```
import sympy as sp
x = sp.symbols('x')
f = (x**2+1) * sp.sin(x)**2

fint = sp.integrate(f, (x, 0, sp.pi))
print('Integral: ', fint)
sp.pprint(fint)

fdif = sp.diff(f, x)
print('Izvod: ', fdif)
print(sp.pretty(fdif))
```

Primjer – analiza broja riječi u stringu

Napišimo funkciju koja za zadati string (tekst) vraća rječnik sa svim riječima u stringu i broju pojavljivanja riječi. Iskoristimo funkciju da nađemo najčešću riječ u datom tekstu.

```
def statistika(s):  
    riječi = s.split()  
    r = {}  
    for riječ in riječi:  
        r[riječ] = (r.get(riječ, 0)) + 1  
    return r
```

```
st = "sir je sir miš je miš "  
st += "ako sir nije miš je li sir sir"  
r = statistika(st)  
print(r)  
print('Najčešća riječ:', max(r))
```

Još o listama u python-u

Pri kreiranju liste možemo koristiti for petlju unutar liste, kao i if naredbu da sugerišemo način kreiranja liste. Na primjer:

```
p = print
a = [1,3,5,7,11,15,19,8]
p('Lista a:',a)
b = [ x**3 for x in range(11) ]
p('Lista b:',b)
c = [ x*x for x in a ]
p('Lista c:',c)
d = [ x*x for x in a if x<10 ]
p('Lista d:',d)
e = [ [n*m for n in range(5)] for m in range(7) ]
p('Lista e:',e)
```

Uočite da smo u prvoj liniji kazali da je p ime za objekat (funkciju) print.

Šta nije uključeno u ovaj kratak uvod

- Rad sa klasama
- Iteratori
- Generatori
- Lambda funkcije
- Formatirani stringovi
- Dekoratori
- Paralelno izvršavanje koda
- Obrada izuzetaka
- Rad sa fajlovima
- Regularni izrazi
- ...

```
class Student:  
    a=iter([3,5,7]); next(a)  
        yield  
f = lambda x : x**2 + x+1  
    f' x={x}, y={y}'  
        @dekor  
import threading  
    try: except:  
open() read() write()  
        import re
```