

Skript jezici V06 - numpy, pandas, matplotlib

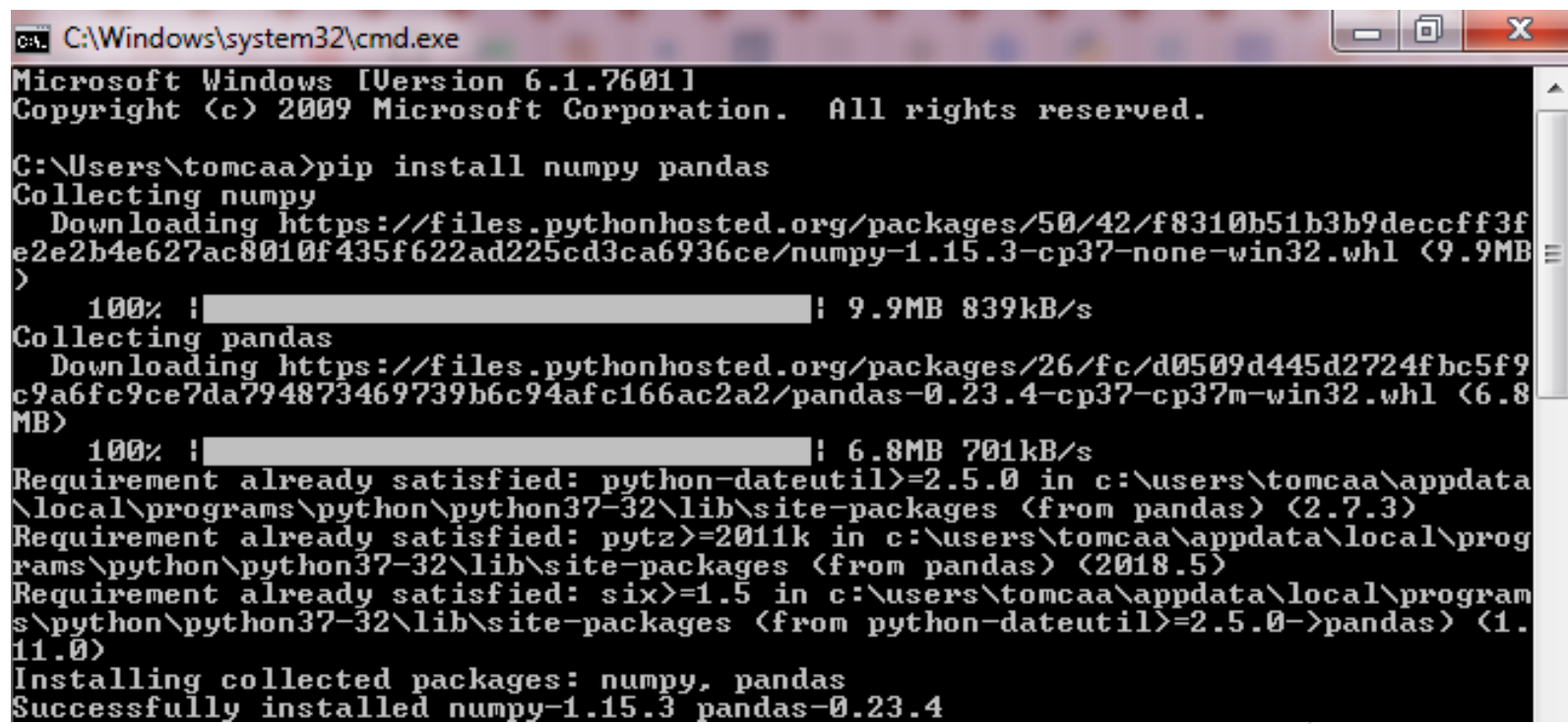
Milan Tomić

Sadržaj

- ▶ Manipulacija podacima služeći se bibliotekama numpy i pandas
- ▶ Crtanje grafika pomoću matplotlib

Instalacija

- ▶ Za trenutnog korisnika (na računarima na fakultetu):
- ▶ `pip install --user numpy pandas matplotlib`
- ▶ (generalno, na svojim računarima možete bez `--user` flaga)



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\tomcaa>pip install numpy pandas
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/50/42/f8310b51b3b9deccff3f
e2e2b4e627ac8010f435f622ad225cd3ca6936ce/numpy-1.15.3-cp37-none-win32.whl (9.9MB
)
  100% |#####| 9.9MB 839kB/s
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/26/fc/d0509d445d2724fbc5f9
c9a6fc9ce7da794873469739b6c94afc166ac2a2/pandas-0.23.4-cp37-cp37m-win32.whl (6.8
MB)
  100% |#####| 6.8MB 701kB/s
Requirement already satisfied: python-dateutil>=2.5.0 in c:\users\tomcaa\appdata
\local\programs\python\python37-32\lib\site-packages (from pandas) (2.7.3)
Requirement already satisfied: pytz>=2011k in c:\users\tomcaa\appdata\local\prog
rams\python\python37-32\lib\site-packages (from pandas) (2018.5)
Requirement already satisfied: six>=1.5 in c:\users\tomcaa\appdata\local\program
s\python\python37-32\lib\site-packages (from python-dateutil>=2.5.0->pandas) (1
.11.0)
Installing collected packages: numpy, pandas
Successfully installed numpy-1.15.3 pandas-0.23.4
```

Upoznavanje

- ▶ Numpy je biblioteka za simultani rad sa numeričkim podacima
 - ▶ Podržava rad sa nizovima i matricama brojevni tipova i u biti po funkcionalnostima podseća na matlab
- ▶ Pandas je biblioteka zasnovana na Numpy za rad sa tabličnim podacima tj. setovima podataka
 - ▶ Od učitavanja, preko pronalaženja ili raspakivanja svojstava do ispisivanja
- ▶ Obično se numpy uvozi kao np, a pandas kao pd
 - ▶ Iz razloga što se često koriste u kodu, dvoslovne skraćenice za ove i još neke module su ustaljene

Numpy - kreiranje nizova

- ▶ Osnovni tip podatka u np je višedimenzioni niz - ndarray
- ▶ Bazični konstruktor se obično ne poziva, umesto toga poziva se neki od sledećih:
 - ▶ `np.array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)`
 - ▶ `np.zeros(shape, dtype=float, order='C')`
 - ▶ `np.ones(shape, dtype=None, order='C')`
 - ▶ `np.eye(N, M=None, k=0, dtype=float, order='C')`
 - ▶ `np.diag(v, k=0)`
 - ▶ `np.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)`
 - ▶ `np.arange(start, stop, step, dtype=None)`

Primeri kreiranja nizova

```
### kreiranje niza pomoću ugrađenih tipova
dva_sa_dva = np.array([[1, 2], [3, 4]])
print(repr(dva_sa_dva))
# array([[1, 2],
#        [3, 4]])
```

```
### kreiranje niza nula
sedam_nula = np.zeros(7, dtype='int')
print(sedam_nula)          # [0 0 0 0 0 0 0]
print(repr(sedam_nula))    #array([0, 0, 0, 0, 0, 0, 0])
```

```
### kreiranje matrice (dvodimenzionalnog niza) 2 x 4 popunjenog jedinicama
osam_jedan = np.ones((2, 4), dtype='float')
print(osam_jedan)
# [[1. 1. 1. 1.]
#   [1. 1. 1. 1.]]
print(repr(osam_jedan))
# array([[1., 1., 1., 1.],
#        [1., 1., 1., 1.]])
```

Primeri kreiranja nizova

```
##% Kreiranje niza elemenata kao u range
print(np.arange(0., 2., 0.2)) # [0.  0.2 0.4 0.6 0.8 1.  1.2 1.4 1.6 1.8]

##% Kreiranje niza elemenata na jednakim razmacima
print(np.linspace(0, 2, 11)) # [0.  0.2 0.4 0.6 0.8 1.  1.2 1.4 1.6 1.8 2. ]

##% Kreiranje jedinica na dijagonali matrice
# red jedinica
print(np.ones(5)) # [1.  1.  1.  1.  1.]
# po dijagonali matrice
print(np.diag(np.ones(5)))
# [[1.  0.  0.  0.  0.]
# [0.  1.  0.  0.  0.]
# [0.  0.  1.  0.  0.]
# [0.  0.  0.  1.  0.]
# [0.  0.  0.  0.  1.]]
# i opet
print(np.eye(5))

##% Slučajna matrica
print(np.random.random(size=(2, 4)))
# [[0.52794895 0.58879849 0.25818468 0.72661907]
# [0.57670114 0.712767  0.58038972 0.39108583]]
```

Oblik, dimenzije, indeksiranje i slajs

- ▶ Oblik niza (matrice) definisan je brojem njenih dimenzija i veličinom po svakoj dimenziji
 - ▶ Recimo ako je oblik (*shape*) 3x2, to se zapisuje kao (3, 2) i tada matrica ima dve dimenzije, u prvoj ima 3 reda, u drugoj 2 kolone (obratno ako je order 'F' - fortranovski način zapisivanja podrazumeva da su prva dimenzija kolone a druga redovi matrice)
- ▶ Pristup elementima može biti direktan pomoću indeksiranja (počevši od 0)
 - ▶ U jednodimenzionalnom nizu klasično, npr. `a[2]` je treći element niza
 - ▶ U višedimenzionim nizovima indeksi razdvojeni zapetama, npr. `a[2,3]` je element treće vrste, četvrte kolone
- ▶ Može se koristiti i slajs, pri čemu se proporcionalno smanjuje broj dimenzija u zavisnosti od toga da li su neke vrste i kolone celokupno uključene ili ne

Primeri pristupa elementima

```
### Dimenzije matrice
m = np.array(((2, 3), (5, 6), (9, 10)))
print(f'Broj dimenzija: {m.ndim}, Oblik: {m.shape}, Velicina: {m.size}')
# Broj dimenzija: 2, Oblik: (3, 2), Velicina: 6
```

```
### pristup elementima (indeks i slajs)
print(dva_sa_dva[0,0])    # 1
print(dva_sa_dva[:,1])    # [2 4] - druga kolona
print(dva_sa_dva[0,:])    # [1 2] - prva vrsta
```

Transformacije

- ▶ `np.concatenate((a1, a2...), axis=0, out=None)`
- ▶ `np.stack(arrays, axis=0, out=None)`
 - ▶ `np.hstack(tup)`
 - ▶ `np.vstack(tup)`
 - ▶ `np.dstack(tup)`
- ▶ `np.block(arrays)`
- ▶ `np.split(ary, indices_or_sections, axis=0)`
 - ▶ `np.vsplit(ary, indices_or_sections) (axis=0)`
 - ▶ `np.hsplit(ary, indices_or_sections) (axis=1)`
 - ▶ `np.dsplit(ary, indices_or_sections) (axis=2)`

Operacije

- ▶ `+`, `-`, `*`, `/` - sabiranje, oduzimanje, množenje i deljenje - po elementima
- ▶ `@` - matrično množenje (takođe i `.dot` metoda na nizu)
- ▶ `np.sum`, `np.divide`, `np.multiply`, `np.dot`, `np.abs`, `np.power`, `np.mod`, `np.sin`, `np.cos`, `np.tan`, `np.log`, `np.var`, `np.mean`, `np.min`, `np.max`...
- ▶ Standardne matematičke operacije, ali mogu se primenjivati direktno na nizove ili u kombinaciji sa skalarima

Pandas

- ▶ Biblioteka za rad sa tabelarnim podacima
 - ▶ Operacije zasnovane na numpy
- ▶ Uvod u strukture podataka
- ▶ Manipulacija podacima u Pandasu

pandas.Series

- ▶ Jednodimenzionalni labelirani (označeni) niz koji podržava bilo koji tip podataka (cele i realne brojeve, stringove, objekte...)
- ▶ Oznake / labela se nazivaju **index**
- ▶ Ima nekoliko načina za kreiranje Series objekta:
 - ▶ Poziv `pd.Series(data, index=index)` / dict, ndarray, skalar
 - ▶ Index predstavljaju oznake za odgovarajuće podatke
 - ▶ Ako je data ndarray (isl.), index ako se prosledi mora biti iste dužine (ako se ne prosledi, biće numerički 0-(n-1))
 - ▶ Ako je data dict i prosledi se index, podaci će biti uređeni po indexu (ako neki ključ u dictu ne postoji, biće NaN)
 - ▶ Ako je data dict i ne prosledi se index, u zavisnosti od verzije Pythona i pandasa biće ili po abecedi ili po redosledu dodavanja u dict
 - ▶ Ako je data skalar, mora da se prosledi index (i svakom podatku iz indexa biće pridružena ista vrednost)

pandas.Series - inicijalizacija

```
### Pandas Series - od niza
```

```
import pandas as pd  
import numpy as np
```

```
s = pd.Series(np.random.rand(3), index=['A', 'B', 'C'])
```

```
print(s)
```

```
# A    0.750326  
# B    0.379455  
# C    0.472512  
#dtype: float64
```

```
### Pandas Series - od dicta
```

```
s = pd.Series({'a': 1, 'b': 2, 'c': 3})
```

```
print(s)
```

```
# a    1  
# b    2  
# c    3  
# dtype: int64
```

```
### Pandas Series - od skalara
```

```
print(pd.Series(5, ('a', 'b', 'c')))
```

```
# a    5  
# b    5  
# c    5  
# dtype: int64
```

Operacije nad pd.Series - pristup elementima, slice, filter, np funkcije

```
### Pristup elementima, slice
print(s['a'])
# 1
print(s[:1])
# a    1
# dtype: int64
print(s[s >= s.mean()])
# b    2
# c    3
# dtype: int64
print(np.power(s, 2))
# a    1
# b    4
# c    9
# dtype: int64
```

Operacije

- ▶ Operacije podržane nad numpy tipovima su takođe operacije podržane nad `pd.Series`
- ▶ `pd.Series` operacije automatski poravnavaju podatke u odnosu na oznake
- ▶ Ako neke oznake nedostaju u nekom od operanada, u rezultatu će za te oznake biti vraćen NaN (not a number)
 - ▶ Oznake u rezultatu su unija oznaka u svim operandima
- ▶ Pomoću funkcije `dropna` mogu se izostaviti podaci koji imaju vrednost NaN
 - ▶ `dropna` može da primi `inplace` argument, koji ako je `True` radi operaciju na istom objektu, a ako je `False` (podrazumevano) vraća novi objekat

Primer operacija i rada sa NaN

```
### Operacije i NaN
a = pd.Series([1, 2, 3], index=['a', 'b', 'c'])
b = pd.Series([5, 6, 7], index=['b', 'c', 'd'])
c = a + b
print(c)
# a      NaN
# b      7.0
# c      9.0
# d      NaN
# dtype: float64
print(c.dropna())
# b      7.0
# c      9.0
# dtype: float64
print(c.fillna('x'))
# a      x
# b      7
# c      9
# d      x
# dtype: object
```

pd.DataFrame

- ▶ Struktura nalik na dvodimenzionalni niz sa oznakama
 - ▶ Podseća na Excel tabelu
- ▶ Svaki red predstavlja jedan unos - objekat ili entitet
- ▶ Svaka kolona predstavlja jednu oznaku, tj. atribut objekta
- ▶ Ima oznake: index koji označava redove i columns koji označava kolone

```
### Primeri inicijalizacije DataFrame-a - dict pd.Series  
d = {'a': pd.Series([1,2,3]), 'b': pd.Series([4, 5, 6])}  
print(pd.DataFrame(d))  
#      a  b  
# 0    1  4  
# 1    2  5  
# 2    3  6
```

pd.DataFrame

```
#%% Primeri inicijalizacije DataFrame-a - dict list-a
print(pd.DataFrame({'a': [1, 2, 3], 'b': [4, 5, 6]}, index=['X', 'Y', 'Z']))
#      a  b
# X    1  4
# Y    2  5
# Z    3  6

#%% Primeri inicijalizacije DataFrame-a - lista dict-ova (kao objekata)
print(pd.DataFrame([{'a': 1, 'b': 2}, {'a': 14, 'b': 12}], index=['X', 'Y']))
#      a  b
# X     1  2
# Y    14 12
```

pd.DataFrame - indeksiranje

- ▶ Može se indeksirati i slajsovati po kolonama direktno
- ▶ Mogu se dodavati nove kolone (dodaju se ili na kraj direktnom dodelom ili pomoću insert metode na određeno mesto)
- ▶ Slično, mogu se brisati kolone
- ▶ Takođe, podaci mogu da se slajsuju
- ▶ Slično je kao dvodimenzionalni niz

pd.DataFrame - indeksiranje

```
### Pristup kolonama direktno preko indeksiranja
df = pd.DataFrame(d)
print(df['a'])
# 0      1
# 1      2
# 2      3
# Name: a, dtype: int64

df['c'] = df['b'] + df['a']
print(df['c'][1]) # 7
print(df)
#      a  b  c
# 0  1  4  5
# 1  2  5  7
# 2  3  6  9
```

Pristup redovima - `df.loc` i `df.iloc`

- ▶ Pristup određenom redu iz tabele može se izvršiti preko `loc` ili `iloc`
 - ▶ `loc` može da se indeksira po `index-u`
 - ▶ `iloc` može da se indeksira po celobrojnoj poziciji (tj. rednom broju reda)
- ▶ Pristup određenom elementu po koordinatama može se izvršiti preko `at` ili `iat`
 - ▶ `at` može da indeksira po `index-u` i `column-u`
 - ▶ `iat` može da indeksira po celobrojnoj poziciji (rednom broju reda i kolone)
- ▶ Ovi pristupi su brži nego pristup preko uglastih zagrada
- ▶ Slajs redova radi se direktno na `df` objektu
 - ▶ Zadaju se celi brojevi ili boolean vektor koji u sebi sadrži filter onih redova koje treba izabrati

Operacije

- ▶ Operacije nad DataFrame-ovima su podržane kao i operacije nad višedimenzionim nizovima
- ▶ I index i column oznake se poravnavaju pri ovim operacijama
 - ▶ U rezultatima su ponovo odgovarajuće unije
- ▶ Za sve nedefinisanе rezultate podrazumevana vrednost je NaN
- ▶ Kada se radi operacija između DataFrame-a i Series objekta, index u Series objektu će biti upoređivan sa column u DataFrame-u
 - ▶ Osim u slučaju kada je DateTimeIndex u pitanju, gde se podrazumeva suprotno ponašanje
- ▶ DataFrame se može transponovati pomoću .T atributa
- ▶ Operacije nad skalarima ponašaju se uobičajeno
- ▶ Metoda .dot na DF može da se koristi za matrično množenje (sa drugim DF); slično Series.dot radi skalarni proizvod vektora

Učitavanje podataka

- ▶ Pandas podržava učitavanje csv fajlova iz tabelarnog zapisa u DataFrame (funkcija `pd.read_csv`)
- ▶ Podrazumevano je da kada se ispisuje df, ako je previše veliki, biva skraćen
- ▶ Metoda `df.info()` daje informacije o sadržaju DataFrame-a
- ▶ Metoda `df.to_string()` vraća string reprezentaciju df-a u tabelarnom obliku
- ▶ `df.index`, `df.columns`, `df.values` vraćaju index, columns i vrednosti
- ▶ `df.head()` i `df.tail()` vraćaju početne i završne redove
- ▶ `df.describe()` može da prikaže brzu statistiku
- ▶ `df.sort_index(axis, ascending)` može da sortira df po nekoj osi
- ▶ `df.sort_values(by)` može da sortira df po vrednostima nekih atributa
- ▶ Još mnogo podržanih operacija potražite u dokumentaciji:
<https://pandas.pydata.org/pandas-docs/stable/10min.html#min>

Primer - df.describe

```
#%% Opis  
print(df.describe())
```

#	<i>a</i>	<i>b</i>	<i>c</i>
# <i>count</i>	3.0	3.0	3.0
# <i>mean</i>	2.0	5.0	7.0
# <i>std</i>	1.0	1.0	2.0
# <i>min</i>	1.0	4.0	5.0
# 25%	1.5	4.5	6.0
# 50%	2.0	5.0	7.0
# 75%	2.5	5.5	8.0
# <i>max</i>	3.0	6.0	9.0

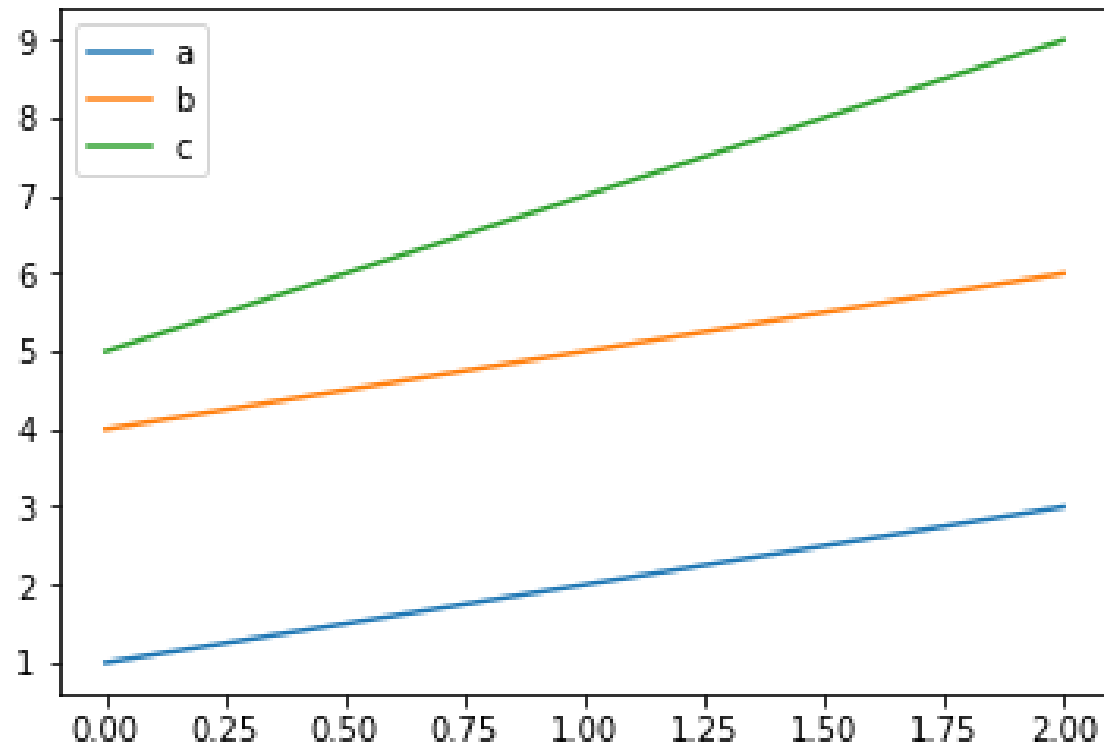
Matplotlib

- ▶ Matplotlib je biblioteka za iscrtavanje (vizuelizaciju) podataka
- ▶ Veoma je složena što se tiče podržanih funkcionalnosti i njeno detaljno opisivanje bi uzelo mnogo vremena
- ▶ Kompletna dokumentacija može se naći na <https://matplotlib.org/contents.html>
- ▶ Pandas ima podršku za neka osnovna iscrtavanja
- ▶ Vrste plotova: line, bar/barh, hist, box, kde/density, area, scatter, hexbin, pie
- ▶ pd.plotting biblioteka sadrži još funkcija za iscrtavanje grafika

df.plot()

```
In [29]: df.plot()
```

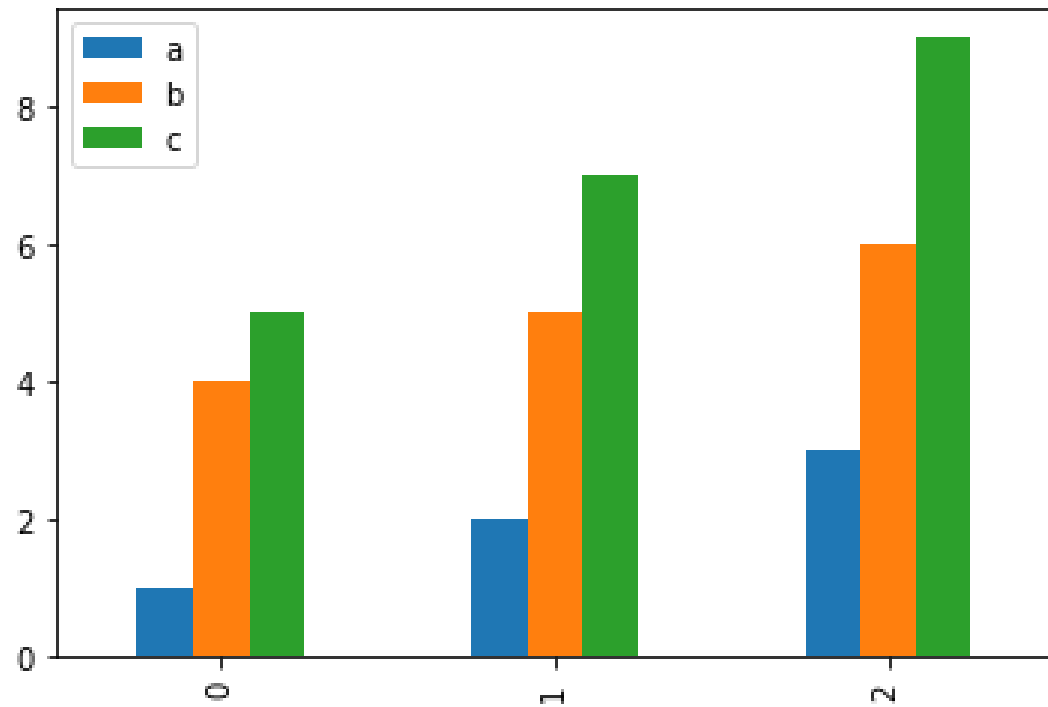
```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x9458b10>
```



df.plot(kind='bar')

```
In [31]: df.plot(kind='bar')
```

```
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0xa0359f0>
```



Zadatak

- ▶ Na <https://www.kaggle.com/> pronaći neki set podataka i izvršiti neku bazičnu analizu koristeći pandas
- ▶ Rezultate ilustrovati odgovarajućim graphicima