



CS324 - SKRIPTING JEZICI

Python i numeričko programiranje

Lekcija 08

PRIRUČNIK ZA STUDENTE

CS324 - SKRIPTING JEZICI

Lekcija 08

PYTHON I NUMERIČKO PROGRAMIRANJE

- ✓ Python i numeričko programiranje
- ✓ Poglavlje 1: Uvod u numpy i pandas
- ✓ Poglavlje 2: Nizovi i objekti nizova
- ✓ Poglavlje 3: Rad sa nizovima, matricama
- ✓ Poglavlje 4: Uvod u pandas paket i instalacija
- ✓ Poglavlje 5: pandas: učitavanje i pisanje datoteka
- ✓ Poglavlje 6: Pokazne vežbe #8
- ✓ Poglavlje 7: Individualne vežbe #8
- ✓ Poglavlje 8: Domaći zadatak #8
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

Uvod u 8. lekciju - paket numpy i pandas

Numpy je jedan od osnovnih paketa za naučno programiranje u Python programskom jeziku. Predstavlja biblioteku koja pruža podršku za višedimenzionalne nizove, matrice i grupu rutina za brzu operaciju nad nizovima. Ove rutine su matematičke, logičke, ali i za sortiranje, menjanje oblika, selekciju, kao i za ulaz i izlaz.

Takođe sadrži rutine za Furijeovu transformaciju koja se često koristi u obradi signala, osnovne rutine linearne algebre i statističkih operacija.

Numpy kao biblioteka predstavlja jednu od osnovnih, ako ne i najosnovniju biblioteku koja treba da se savlada ukoliko želite da se bavite razvojem bilo kakvih aplikacija koje imaju veze sa hardverom, embedded sistema ili sa Internet of Things uređajima.

Zbog toga je bitno da se savladaju osnove numpy paketa, a to je objekat *ndarray*. Ovaj objekat enkapsulira n-dimenzionalne nizove homogenih tipova podataka, uz operacije koje su brže od standardnih Python operacija za rad sa nizovima.

Pored numpy-a, u ovoj lekciji obradiće se i **pandas** biblioteka. Pandas predstavlja paket za obradu podataka i za analizu podataka. Pandas pruža strukture podataka i operacije za manipulaciju numeričkih tabela i vremenskih nizova. Ime pandas je skraćenica za **panel data**, što je izraz za skupove podataka koji se posmatraju u nekom vremenskom periodu.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 1

Uvod u numpy i pandas

PAKET NUMPY

Paket Numpy predstavlja osnovni paket za naučno programiranje u Python programskom jeziku.



Slika 1.1 NumPy logo. [Izvor: <https://numpy.org>]

Paket Numpy predstavlja osnovni paket za naučno programiranje u Python programskom jeziku. Najosnovniji objekat koji paket NumPy pruža jeste višedimenzionalni niz, različite izvedene objekte (maskirani nizovi i matrice), kao i asortiman rutina za brz rad sa nizovima.

Ove rutine uključuju matematičke i logičke operacije, operacije promene oblika niza, sortiranje, odabir, ulaz-izlaz (en. **input-output**, **I/O**), diskretne Furijeove transformacije, osnovne operacije linearne algebre, osnovne statističke operacije, alate za simulaciju i mnoge druge.

Više informacija o samom paketu na: <https://numpy.org>

Programeri koji su upoznati sa **MATLAB** (**M**atrix **L**aboratory) programskim jezikom, prepoznaće NumPy paket kao alternativu za rad sa višedimenzionalnim nizovima. Može se reći da je NumPy besplatna zamena za osnovne funkcionalnosti MATLAB programskog jezika, naravno unutar Python jezika.

NumPy predstavlja osnovu za bilo koju vrstu programiranja pri kojoj je potreban rad sa višedimenzionalnim nizovima. Paketi za naučno programiranje (**SciPy**), vizuelizaciju podataka (**matplotlib**), mašinsko učenje (**scikit-learn**), backend programiranje pri razboju web aplikacija, i mnogi drugi paketi koriste upravo NumPy.

NumPy paket u potpunosti podržava objektno-orijentisani pristup. Osnova NumPy paketa jeste klasa `ndarray`, koja podržava brojne attribute i metode. Mnoge metode jesu preslikane funkcijama u *spoljašnjem prostoru imena* (en. *outer-most namespace*), dozvoljavajući programerima da pišu programe u paradigmi koja im odgovara.

Fleksibilnost koja pruža paket NumPy niz dovela je do toga da je klasa `ndarray` *de-facto* način za rad sa višedimenzionalnim podacima u Python jeziku.

INSTALACIJA NUMPY PAKETA

Nepisano je pravilo da se numpy paket preimenuje u `np` prilikom uvoza u program.

Instalaciju NumPy paketa je jednostavna kroz paketni menadžer *pip*. Dovoljno je u konzoli ukucati sledeću komandu:

```
pip install numpy
```



Slika 1.2 Instalacija NumPy paketa kroz pip. [Izvor: Autor]

Provera (sa verzijom koja je instalirana) se može izvršiti pokretanjem *pip list* komande.

Slika 1.3 Provera instalacije i verzije paketa. [Izvor: Autor]

Uvoz NumPy paketa u imenski prostor programa

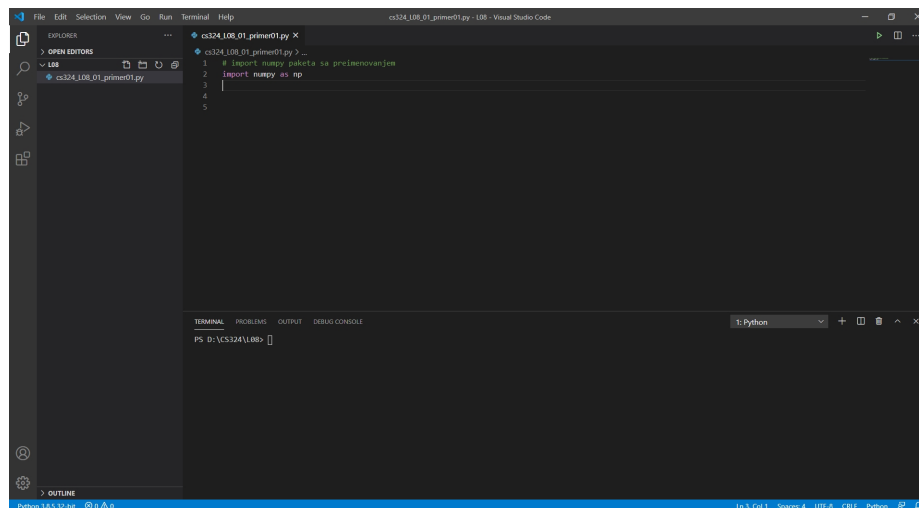
Nakon instalacije paketa NumPy, dovoljno je uvesti paket u glavni program.

```
import numpy
```

Preporučuje se uvoz sa preimenovanjem:

```
# import numpy paketa sa preimenovanjem  
import numpy as np
```

Prilikom korišćenja NumPy paketa postalo je nepisano pravilo da se preimenuje u *np*.



Slika 1.4 Uvoz numpy paketa sa preimenovanjem u np paket. [Izvor: Autor]

INSTALACIJA NUMPY PAKETA - VIDEO

Sledi autorski video o instalaciji numpy paketa korišćenjem menadžera paketa PIP

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VEKTORIZACIJA IZVRŠENJA OPERACIJA

Vektorizacija izvršenja operacija predstavlja odsustvo eksplicitnih petlji i indeksiranja unutar koda.

Vektorizacija izvršenja operacija predstavlja odsustvo eksplicitnih petlji i indeksiranja unutar koda. Petlje i indeksiranje i dalje postoje, ali se izvršavaju iza kulisa, i to u optimizovanom kodu koji je *pre-kompajlovan* (en. *pre-compiled code*) i napisanom u C jeziku.

Vektorizovan kod ima mnogo prednosti, a ističu se sledeće:

- koncizniji kod koji je lakši za čitanje,
- manji broj linija koda u opštem slučaju znači manji broj grešaka,
- kod više liči na standardnu matematičku notaciju, što omogućava da se pravilnije kodiraju matematički konstrukti,
- Vektorizacija kao rezultat ima kod koji sam po sebi deluje "Python-ovski". Ovo znači da kod ne poseduje brojne neefikasne *for petlje* koje se teško čitaju i tumače.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 2

Nizovi i objekti nizova

KLASA I OBJEKAT NDARRAY

"Srž" NumPy paketa jeste klasa ndarray i njen objekat ndarray.

Najosnovniji deo NumPy paketa jeste klasa ndarray i njen objekat ndarray. Ovaj objekat enkapsulira n-dimenzionalne nizove sa homogenim tipovima podataka.

Pitanje:

Šta je homogeni, a šta heterogeni niz?

Mnoge operacije koje se mogu izvršiti nad `ndarray` objektom izvršavaju se jako brzo jer su napisane u kompajlovanom kodu (en. `compiled code operations`)

Objekat `ndarray` jeste višedimenzionalni kontejner elemenata istog tipa podataka i veličine, i sam objekat je najčešće fiksne veličine.

Broj dimenzija i elemenata unutar niza jeste definisan pomoću oblika (en. `array shape`), koji predstavlja `tuple` ne-negativnih celih brojeva koji specificiraju veličine svake dimenzije.

Tip podataka elemenata niza je specificiran posebnim `data-type` objektom `dtype`.

Kao i kod ostalih iterabilnih objekata u Python jeziku, sadržaju `ndarray` objekta se može pristupiti indeksiranjem, ali i metodama i atributima samog objekta.

Različiti `ndarray` objekti mogu deliti iste podatke, tako da promene u jednom objektu budu vidljive u drugom.

Jedan objekat može služiti da se "pregleda" sadržaj drugog objekta, i podaci na koji taj objekat ukazuje jesu podaci u "osnovnom" (en. `base`) objektu.

KREIRANJE NIZA KLASSE NDARRAY

Kreiranje niza klase ndarray je jednostavno, ali treba obratiti pažnju da je parametar koji se prosleđuje lista.

Kreiranje niza klase ndarray je jednostavno gotovo koliko i kreiranje liste.

```
arr = np.array(<list>)
```

Parametar <list> ukazuje da se treba proslediti lista. Ukoliko je u pitanju niz sa više dimenzija, unosi se lista listi, gde je svaka lista jedna dimenzija niza.

Primer: Jednodimenzioni i dvodimenzioni niz (2 minuta)

Pomoću numpy napraviti tri niza. Prti je niz od 5 elemenata, dok je drugi 2x3 matrica. Treći niz jeste niz stringova različitog broja karaktera.

```
import numpy as np

a = np.array([1, 2, 3, 4, 5])
b = np.array([[1, 2, 3], [4, 5, 6]])
c = np.array(['Ovo', 'je', 'test', 'string', 'za', 'rad', 'sa', 'numpy',
'paketom....'])
```

Broj elemenata se može dobiti pozivom metode `.size`

```
print(a.size)
>>> 5
print(b.size)
>>> 6
print(c.size)
>>> 9
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

DIMENZIJE, TIP PODATAKA I VELIČINE ELEMENATA U MEMORIJI

Pozivanjem različitih atributa numpy niza moguće je dobiti dodatne informacije o samom nizu.

Dimenzije niza se mogu vratiti pozivanjem atributa `.ndim`

```
print(a.ndim)
>>> 1
print(b.ndim)
>>> 2
print(c.ndim)
>>> 1
```

Oblik niza predstavlja tuple koji vraća broj elemenata po dimenziji niza, i dobija se pozivanjem atributa `.shape`

```
print(a.shape)
>>> (5, )
print(b.shape)
>>> (2, 3)
```



```
print(c.shape)
>>> (9, )
```

Tip niza se može vratiti pozivanjem atributa `.dtype`

```
print(a.dtype)
>>>int32
```

Prilikom kreacije niza moguće je kao dodatni argument napisati i **eksplicitno** koji je tip podataka.

```
a = np.array([1, ,2 ,3 ,4 ,5], dtype='int64')
```

```
# tipovi podataka
bool_ - Boolean tip podataka, 1B po elementu
int8 - ceo broj, 1B po elementu (vrednosti -128 do 127)
int32 - ceo broj, 4B po elementu
int64 - ceo broj, 8B po elementu
uint8/16/32/64 - nenegativan ceo broj, 1B/2B/4B/8B po elementu
float16/32/64 - razlomljeni broj, 2B/4B/8B po elementu
complex64/128 - kompleksni broj, 8B/16B po elementu
U_ - string, gde _ oznacava broj karaktera najveceg stringa
```

Veličina elemenata niza se može dobiti pozivom metode `.itemsize`, koja vraća veličinu u bajtovima.

```
print(a.itemsize)
>>> 8
print(b.itemsize)
>>> 4
print(c.itemsize)
>>> 48
```

Ukupna veličina niza u memoriji može se dobiti množenjem broja elemenata i veličine elemenata

```
print(a.size * a.itemsize)
>>> 40
```

RAZLIKE IZMEĐU NDARRAY, LISTA I UGRAĐENIH NIZOVA

Većina softvera za matematičko i naučno izračunavanje koja je bazirana na Python programskom jeziku koristi NumPy nizove.

Postoje nekoliko bitnih razlika između Numpy nizova (`ndarray` objekata) i standardnih Python iterativnih tipova, kao što su liste i "obični", ugrađeni `array` nizovi.

- Numpy nizovi imaju fiksnu veličinu pri pravljenju niza. Python liste je dinamički tip podataka (može proizvoljno da raste broj elemenata). Promena veličine ndarray niza napraviće novi niz, a stari će obrisati.
- Elementi ndarray niza moraju biti istog tipa, i zbog toga svaki element zauzima istu količinu memorije.
- Nad ndarray nizovima moguće je izvršiti veliki broj matematičkih operacija nad velikom količinom podataka istovremeno. Ovakav pristup izvršenja operacija je efikasniji i brži u odnosu na ugrađene operacije u Python interpreteru.
- Sve veći broj naučnih i matematičkih paketa za Python programski jezik koriste ndarray nizove. Iako se unos podataka obavlja preko standardnih tipova podataka, oni se konvertuju u NumPy nizove pre procesiranja. Ovo znači da većina softvera za matematičko i naučno izračunavanje koja je bazirana na Python programskom jeziku koristi NumPy nizove.

Napomena:

Moguće je imati niz objekata (uključujući i NumPy nizove), što omogućuje da niz sadrži elemente različitih veličina.

PRIMER: MNOŽENJE ELEMENATA DVA NIZA ISTE DUŽINE

Korišćenjem Numpy paketa za rad sa nizovima pojednostavljuje pisanje koda.

Primer: Proizvod elemenata lista (5 minuta)

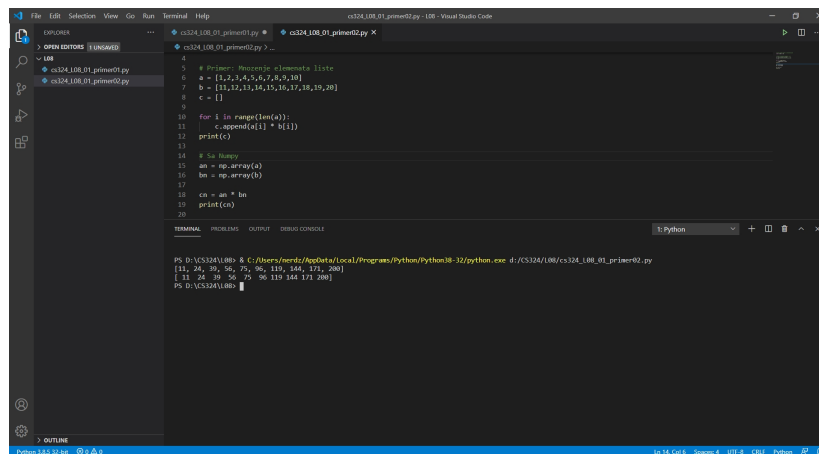
Data je lista **a** od 10 elemenata (brojevi od 1 do 10), i lista **b** od 10 elemenata (brojevi od 11 do 20). Napraviti listu **c** u kojoj je svaki element proizvod elemenata **a** i **b** sa istim indeksom.

```
# Primer: Mnozenje elemenata liste
a = [1,2,3,4,5,6,7,8,9,10]
b = [11,12,13,14,15,16,17,18,19,20]
c = []

for i in range(len(a)):
    c.append(a[i] * b[i])
print(c)
```

Korišćenjem paketa NumPy ista operacija se može mnogo lakše izvršiti:

```
# Preko Numpy
an = np.array(a)
bn = np.array(b)
cn = an * bn
print(cn)
```



```

1 # Primer: Množenje 1D nizova
2
3 a = [1,2,3,4,5,6,7,8,9,10]
4 b = [11,12,13,14,15,16,17,18,19,20]
5 c = []
6
7 for i in range(len(a)):
8     c.append(a[i] * b[i])
9     print(c)
10
11 # Sa numpy
12
13 an = np.array(a)
14 bn = np.array(b)
15
16 cn = an * bn
17 print(cn)
18
19
20

```

Terminal output:

```

PS D:\CS324\108> & C:\Users\nerdz\AppData\Local\Programs\Python\Python38-32\python.exe d:/CS324/108/cs324_108_01_primer02.py
[11, 24, 39, 56, 75, 96, 119, 144, 171, 200]
[11, 24, 39, 56, 75, 96, 119, 144, 171, 200]
PS D:\CS324\108>

```

Slika 2.1 Množenje elemenata jednodimenzionog niza. [Izvor: Autor]

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 3

Rad sa nizovima, matricama

INICIJALIZACIJA NIZOVA SA NULAMA I JEDINICAMA

U okviru numpy paketa moguće je izvršiti inicijalizaciju nizova koji imaju sve nule ili sve jedinice.

Specijalni nizovi i specijalne matrice predstavljaju nizove i matrice kojima su (početne) vrednosti inicijalizovane pozivom funkcije koja nije [nd.array](#).

Nizovi i matrice sa nulama

Pozivom funkcije [zeros\(\)](#) vraća se numpy objekat koji ima sve elemente jednake nulama.

```
import numpy as np

n = zeros((shape), dtype='float')
```

Ukoliko se u parametar shape unese samo jedna vrednost, vratiće niz sa toliko elemenata. Ukoliko se naznači oblik (kao kod shape atributa - tuple sa brojem elemenata po dimenziji) napraviće se takav niz ili matrica.

Drugi argument je tip podataka i podrazumevano je [float](#), dok se može eksplicitno namestiti da je u pitanju drugi tip podataka.

Primer: Nula matrica (2 minuta)

Napraviti niz od 5 elemenata i matricu 2x10 sa svim nulama.

```
import numpy as np

# nizovi i matrice sa sve nulama
za = np.zeros(5)
zb = np.zeros((2,10))
print(za)
print(zb)
```

Nizovi i matrice sa jedinicama

Pozivom funkcije [ones\(\)](#) vraća se numpy objekat koji ima sve elemente jednake jedinicama.

```
import numpy as np

n = ones((shape), dtype='float')
```

Kao i kod nizova sa nulama, prvi parametar označava oblik niza, a drugo označava tip podataka.

Primer: Matrice sa jedinicama (2 minuta)

Napraviti matricu 3x2 sa sve jedinicama. Pomnožiti celu matricu sa brojem 3. Štampati rezultat.

```
import numpy as np

o32 = np.ones((3,2))
o32 *= 3
print(o32)
```

Kada se nad svim elementima numpy nizova vrše aritmetičke operacije, moguće je direktno interagovati sa samim nizom, tj. nije neophodno praviti numpy niz sa jednim elementom koji bi bio drugi sabirak/množilac.

FULL NIZOVI, NIZOVI SA OBLIKOM DRUGOG NIZA, JEDINIČNA MATRICA

Takođe je moguće je izvršiti inicijalizaciju nizova koji imaju sve iste vrednosti, ili oblik drugog niza.

Nizovi sa podešenim početnim vrednostima - full nizovi

Pozivom funkcije `full()` pravi se numpy niz koji ima željeni oblik (prvi parametar), određenu vrednost (drugi parametar) i tip podataka.

```
import numpy as np
n = np.full((shape), value, dtype)
```

Primer: Full niz stringova (2 minuta)

Napraviti numpy niz 2x2 kojem su sve vrednosti 'CS324 - FIT' a tip podataka je '<U5'

```
import numpy as np

fa = np.full((2,2), 'CS324 - FIT', dtype='<U5')
print(fa)
```

Nizovi sa oblikom drugog niza

Nizovi sa nulama, jedinicama i puni nizovi se mogu napraviti da imaju oblik već postojećeg niza tako što se pozivaju funkcije sa dodatkom `_like` na kraju:

```
import numpy as np

#postojeci niz a = np.array(...)
```

```
a_zeros = np.zeros_like(a)
a_ones = np.ones_like(a)
a_full = np.full_like(a, value)
```

Primer: Nizovi sa oblicima postojećih nizova (4 minuta)

Napraviti niz 2x5 (dva reda, 5 kolona) koji sadrži cele brojeve. Napraviti nizove sa jedinicama, nulama, i pun niz sa jednom od vrednosti prvobitnog niza.

```
import numpy as np

niz = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(niz)

niz_zeros = np.zeros_like(niz)
print(niz_zeros)

niz_ones = np.ones_like(niz)
print(niz_ones)

niz_full = np.full_like(niz, 10)
print(niz_full)
```

Jedinična matrica

Moguće je napraviti jediničnu matricu (en.[identity matrix](#)) pozivom funkcije [identity\(\)](#), gde je parametar size veličina ove kvadratne matrice.

```
import numpy as np
n = np.identity(size)
```

PRIMERI SA NIZOVIMA SA SPECIJALNIM VREDNOSTIMA

Slede autorski video klipovi sa primerima sa specijalnim nizovima.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

NASUMIČNI ELEMENTI NIZA

Paket numpy poseduje svoj modul random koji je specijalizovan za rad sa nizovima i nasumičnim vrednostima.

Paket numpy poseduje svoj modul random koji je specijalizovan za rad sa nizovima, i može se iskoristiti kada je potrebno da elementi niza budu nasumični brojevi.

Nasumični razlomljeni brojevi

Nasumični razlomljeni brojevi se mogu pozvati funkcijom `random.rand()`

```
import numpy as np
n = np.random.rand(d0,d1,...)
```

Kao parametre ova funkcija uzima samo cele brojeve, gde je prvi broj elemenata u prvoj dimenziji, drugi broj za drugu dimenziju i sl. Interval koji se koristi jeste [0,1).

Ukoliko se kao parametar prosleđuje oblik postojećeg niza, onda treba koristiti funkciju `random.random_sample()`

```
import numpy as np
n = np.random.random_sample(size)
```

Nasumični razlomljeni brojevi u datom intervalu

Niz sa nasumičnim brojevima u datom intervalu (sa uniformnom raspodelom) mogu se dobiti pozivom funkcije `random.uniform()`

```
import numpy as np
n = np.random.uniform(low, high, shape)
```

Prvi parametar je donja granica, drugi parametar je gornja granica.

Nasumični celi brojevi u datom intervalu

Niz sa nasumičnim celim brojevima (sa diskretnom uniformnom raspodelom) mogu se dobiti pozivom funkcije `random.randint()`

```
import numpy as np
n = np.random.randint(low, high, shape)
```

Slede primeri poziva nizova sa različitim nasumičnim brojevima.

```
import numpy as np

# nasumicni brojevi u intervalu [0,1) sa uniformnom raspodelom
a = np.random.rand(2,4)
print(a)

# nasumicni brojevi u intervalu [0,1) sa uniformnom raspodelom,
# prima array.shape kao parametar
b = np.random.random_sample(a.shape)
print(b)

# nasumicni celi brojevi u datom intervalu sa diskretnom uniformnom raspodelom
c = np.random.randint(-10,10,b.shape)
print(c)
```

```
# nasumicni brojevi sa uniformnom raspodelom u odredjenom intervalu
d = np.random.uniform(-5,5,c.shape)
print(d)
```

NIZOVI I LINEARNA ALGEBRA

Numpy paket sadrži brojne funkcije za rad sa nizovima i matricama iz oblasti linearne algebre.

Numpy paket sadrži brojne funkcije za rad sa nizovima i matricama iz oblasti linearne algebre, kroz modul [linalg](#)

Inverzna matrica

Nalaženje inverzne matrice moguće je pozivom funkcije [linalg.inv\(\)](#)

```
import numpy as np
n_inv = np.linalg.inv(n)
```

Determinanta matrice

Nalaženje determinante kvadratne matrice moguće je pozivom funkcije [linalg.det\(\)](#)

```
import numpy as np
n_det = np.linalg.det(n)
```

Sopstveni vektori i sopstvene vrednosti

Nalaženje sopstvenih vektora kvadratne matrice i sopstvenih vrednosti matrice moguće je pozivom funkcija [linalg.eig\(\)](#) i [linalg.eigvals\(\)](#).

```
import numpy as np
n_eigvec = np.linalg.eig(n)
n_eigval = np.linalg.eigvals(n)
```

```
import numpy as np

# linearna algebra - mnozenje matrica
a = np.random.randint(-10, 10, (5, 2))
print(a)

b = np.random.randint(-10, 10, (2, 5))
print(b)

print(np.matmul(a,b))

# inverzna matrica
c = np.random.randint(-10, 10, (5, 5))
print(c)
```



```
c = np.linalg.inv(c)
print(c)

# determinanta
d = np.random.randint(-5, 5, (4, 4))
print(d)
d = np.linalg.det(d)
print(d)
```

NUMPY NIZOVI I RAD SA DATOTEKAMA

Kroz numpy moguće je raditi i sa datotekama.

Učitavanje iz datoteke

Učitavanje iz datoteke je moguće pozivom funkcije `genfromtxt()`

```
import numpy as np
filedata = np.genfromtxt('path_to_file', delimiter, dtype)
```

Prvi argument jeste ime (i putanja) datoteke, drugi parametar jeste `delimiter`, a opciono se može staviti koji je tip podataka u pitanju.

Primer: (4 minuta)

Zapamtiti tekst u nastavku zadatka u datoteku "podaci.txt". Zatim, kroz numpy funkciju učitati tekst. Proveriti koji su elementi učitano niza veći od 15.

Test podaci:

```
1,3,5,7,9,11,13,15,17,19
2,4,6,8,10,12,14,16,18,20
3,7,11,15,19,23,27,31,35,39
```

Rešenje:

```
import numpy as np

filedata = np.genfromtxt('podaci.txt', delimiter=',', dtype='int32')
print(filedata)

print(filedata > 15)
```

Upis u datoteku

Upis u datoteku moguće je pozivom funkcije `savetxt()`

```
import numpy as np
savetxt('path_to_file', source_array, fmt, delimiter, dtype)
```

Prvi argument jeste ime (i putanja) datoteke, drugi parametar jeste niz koji se upisuje u datoteku, parametar *fmt* jeste format svakog elementa, sledi delimiter, a opciono se može staviti koji je tip podataka u pitanju.

Primer: (4 minuta)

Uvesti tekst iz "podaci.txt". Zatim, napraviti novi niz istog oblika, a svaki element ima vrednost 100. Sabrati ta dva niza u sačuvati kao datoteku "novi_podaci.txt" sa zapetom kao *delimiter*.

```
import numpy as np

filedata = np.genfromtxt('podaci.txt', delimiter=',', dtype='int32')

a = np.full_like(filedata, 100)
a += filedata

np.savetxt('novi_podaci.txt', a, fmt='%d', delimiter=',')
```

▼ Poglavlje 4

Uvod u pandas paket i instalacija

PAKET PANDAS

Pandas je osnovni paket za rad sa manipulacijom i analizom velike količine podataka



Slika 4.1 Paket pandas. Izvor: [<https://pandas.pydata.org>]

Paket pandas predstavlja osnovni paket za rad sa manipulacijom podataka i analizom podataka. Pandas nudi rad sa strukturama podataka i operacijama nad numeričkim i alfa-numeričkim tabelama, i vremenskim nizovima (en. *time series*).

Ime pandas je od "panel data", što predstavlja termin za skupove podataka koji se pregledaju u različitim vremenskim periodima.

Osnovni deo pandas paketa jeste objekat DataFrame koji služi za manipulaciju podataka. Pandas takođe sadrži:

- Alate za čitanje i pisanje podataka u različitim formatima,
- Alate za podešavanje podataka i obradu podataka koji nedostaju,
- Alate za promenu oblika predstavljanja podataka,
- Mogućnost naprednog indeksiranja i pravljenja podskupova podataka ukoliko se radi o velikom skupu podataka,
- Mogućnost dodavanje redova i kolona unutar strukture podataka,
- Grupisanje i sjedinjavanje struktura podataka,
- Filtriranje podataka

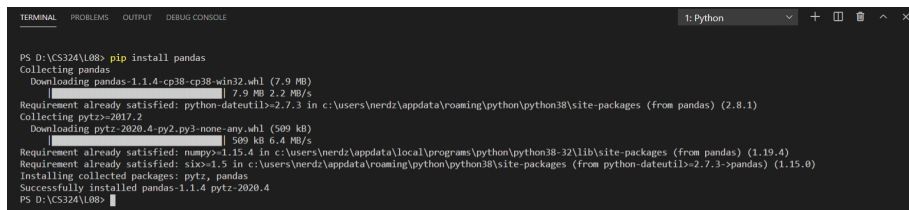
Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

INSTALACIJA PANDAS PAKETA I JUPYTER SVEZAKA

Za bolji pregled datoteka, preporučuje se korišćenje Jupyter svezaka.

Instalaciju pandas paketa je jednostavna kroz paketni menadžer [pip](#). Dovoljno je u konzoli ukucati sledeću komandu:

```
pip install pandas
```



Slika 4.2 Instalacija pandas paketa kroz pip. [Izvor: Autor]

Nakon instalacije, potrebno je uvesti paket u trenutni imenski prostor komandom:

```
# import pandas paketa sa preimenovanjem  
import pandas as pd
```

Napisano je pravilo da se pandas paket preimenuje u [pd](#).

Nakon instalacije pandas paketa, poželjno je instalirati i [jupyter](#) ekstenziju, zbog boljeg pregleda tabelarnih podataka.

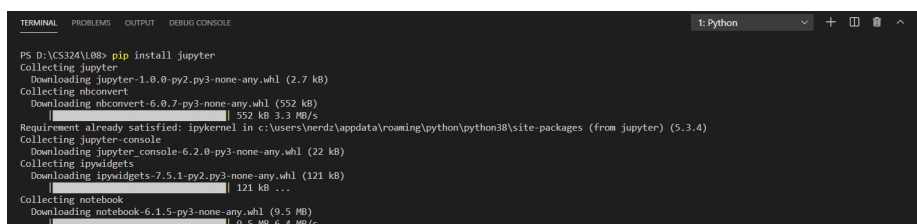
Pored konzolnog i klasičnog IDE prikaza, Python poseduje mogućnost da se kod izvršava kroz interaktivni IDE - [Jupyter](#).

Korišćenjem tzv. jupyter svezaka, moguće je izvršavati deo koda, postavljati tekst koji nije deo koda, izvršiti vizuelizaciju podataka, ali i izvršiti integraciju sa drugim programskim jezicima kao što je R programski jezik.

Jupyter se uglavnom koristi za rad sa velikim podacima, numeričku simulaciju, statističkom modelovanje, i mašinsko učenje.

Jupyter se može instalirati kao ekstenzija unutar Visual Studio Code IDE-a:

```
pip install jupyter
```



Slika 4.3 Instalacija jupyter kroz pip. [Izvor: Autor]

Nakon instalacije, potrebno je pokrenuti jupyter serverski proces pozivom sa Ctrl+Shift+P

Jupyter: Create New Jupyter Notebook

DODATNI PAKETI PRI RADU SA PANDAS PAKETOM

Nekada je potrebno koristiti dodatni paket pri radu sa nekim od formata koje pandas podržava.

Pri radu sa dodatnim formatima nekada je neophodno instalirati dodatne pakete zbog zavisnosti.

Na primer, prilikom rada sa tabelama koje se mogu pokrenuti u Microsoft Excel-u ili nekom sličnom programu (datoteke sa ekstenzijom `.xls` ili `.xlsx`), potrebno je ubaciti dodatni paket ili se javlja sledeća greška:

```
Traceback (most recent call last):
  File "d:/CS324/108/Pandas/cs324_108_pandas01.py", line 6, in <module>
    df2 = pd.read_excel('FIT_presmeti.xlsx')
  File "C:\Users\nerdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\io\excel\_base.py", line 304, in read_excel
    io = ExcelFile(io, engine=engine)
  File "C:\Users\nerdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\io\excel\_base.py", line 867, in __init__
    self._reader = self._engine[self._engine](self._io)
  File "C:\Users\nerdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\io\excel\_xlrd.py", line 21, in __init__
    import_optional_dependency("xlrd", extra_err_msg)
  File "C:\Users\nerdz\AppData\Local\Programs\Python\Python38-32\lib\site-packages\pandas\compat\optional.py", line 110, in import_optional_dependency
    raise ImportError(msg) from None
ImportError: Missing optional dependency 'xlrd'. Install xlrd >= 1.0.0 for Excel support Use pip or conda to install xlrd.
```

Slika 4.4 Greška pri radu sa `.xls` datotekama. [Izvor: Autor]

Neophodno je instalirati paket **xlrd**, koji daje dodatnu podršku za rad sa ovakvim tabelama.

```
pip install xlrd
```

Slika 4.5 Paket xlrd. [Izvor: Autor]

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 5

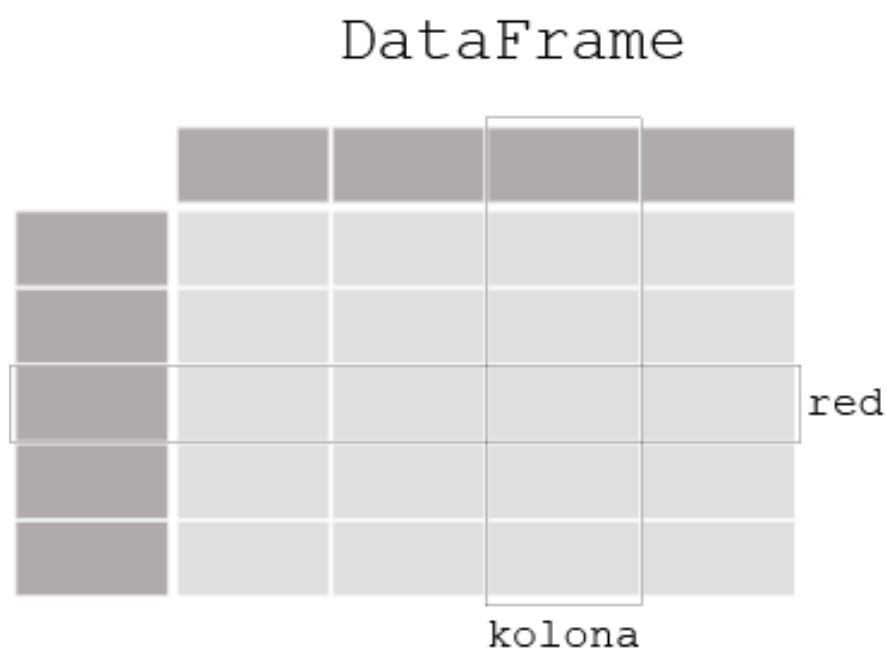
pandas: učitavanje i pisanje datoteka

PANDAS I DATAFRAME OBJEKAT

U pandas paketu, tabela sa podacima se naziva DataFrame (okvir podataka).

Kada se radi sa tabelarnim podacima, kao što su podaci koji su smešteni u tabelama ili bazama podataka, pandas paket olakšava rad sa ovakvim tipovima podataka.

U pandas paketu, tabela sa podacima se naziva DataFrame (okvir podataka).



Slika 5.1 DataFrame objekat u pandas paketu. [Izvor: Autor]

[DataFrame](#) predstavlja dvodimenzionalnu strukturu podataka koja može smestiti podatke različitih tipova (uključujući karaktere, cele brojeve, razlomljene brojeve, kategorisane podatke) u kolonama. Vrlo je slična običnoj tabeli.

Svaka kolona unutar [DataFrame](#) objekta jeste [Series](#) (red) objekat.

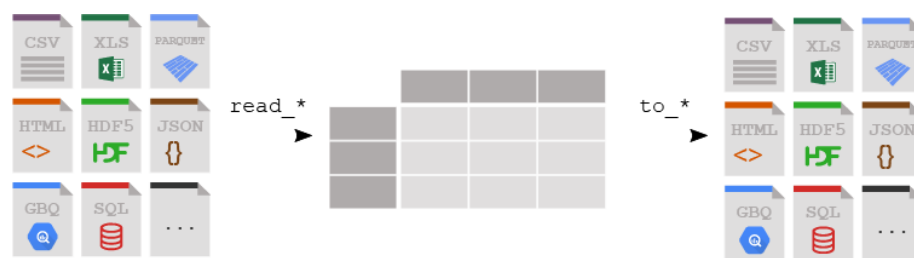
Mnoge operacije pri korišćenju pandas paketa vratiće kao tip podataka [DataFrame](#) ili [Series](#) objekte.

PANDAS I PODRŽATI FORMATI

Pandas podržava integraciju sa velikim brojem tipova datoteka ili sa izvorima podataka

Pandas podržava integraciju sa velikim brojem tipova datoteka ili sa izvorima podataka (*excel* tabele, *csv* datoteke, *SQL*, *JSON* i mnogi drugi formati).

Uvoz podataka iz svakog od ovih izvora podataka uvek kreće sa funkcijom *read_** gde zvezdica opisuje koji je tip podataka u pitanju. Na sličan način, metode *to_** upisuju podatke u datoteke.



Slika 5.2 Upis i čitanje preko pandas paketa. [Izvor: <https://pandas.pydata.org>]

Podaci za uvoz u excel:

```
Fakultet,Smer,Godina,Semestar,Sifra,PunoIme
FIT,IT,1,1,CS101,Uvod u objektno-orjentisano programiranje
FIT,IT,1,1,IT101,Osnove informacionih tehnologija
FIT,IT,1,1,MA101,Matematika 1
FIT,IT,1,1,NT111,Engleski 1
FIT,IT,1,2,CS102,Objekti i apstrakcija podataka
FIT,IT,1,2,IT210,Sistemi informacionih tehnologija
FIT,IT,1,2,CS115,Diskretne strukture
FIT,IT,1,2,NT112,Engleski 2
FIT,IT,2,3,IT331,Racunarske mreze i komunikacije
FIT,IT,2,3,SE201,Uvod u softversko inzenjerstvo
FIT,IT,2,3,IT350,Baze podataka
FIT,IT,2,3,NT213,Engleski za informaticare
FIT,IT,2,4,CS230,Distribuirani sistemi
FIT,IT,2,4,IT370,Interakcija covek-racunar
FIT,IT,2,4,CS220,Arhitektura racunara
FIT,IT,2,4,CS323,C/C++ programski jezik
FIT,IT,3,5,IT225,Veb sistemi 1
FIT,IT,3,5,CS225,Operativni sistemi
FIT,IT,3,5,IT335,Administracija racunarskih sistema i mreza
FIT,IT,3,5,CS324,Skripting jezici
FIT,IT,3,6,SE325,Upravljenje projektima razvoja softvera
FIT,IT,3,6,IT355,Veb sistemi 2
FIT,IT,3,6,CS330,Razvoj mobilnih aplikacija
FIT,IT,3,6,MA273,Verovatnoca i statistika
FIT,IT,3,6,IT375,Racunarsko upravljanje sistemima
FIT,IT,3,6,CS450,Klaud kompjuting
```

```
FIT,IT,3,7,IT381,Zastita i bezbednost informacija
FIT,IT,3,7,CS322,C# programski jezik
FIT,IT,4,7,CS103,Algoritmi i strukture podataka
FIT,IT,4,7,IT333,Bezicne I mobilne komunikacije
FIT,IT,4,7,IT376,Robotika
FIT,IT,4,7,OM350,Preduzetnistvo
FIT,IT,4,7,SE311,Projektovanje i arhitektura softvera
FIT,IT,4,7,SE321,"Obezbedjivanje kvaliteta, testiranje i odrzavanje softvera"
FIT,IT,4,8,IT390,Prefesionalna praksa i etika
FIT,IT,4,8,NT310,Profesionalna komunikacija
FIT,IT,4,8,SE490,Strucna praksa (4 meseca)
FIT,IT,4,8,SE495,Zavrsni rad
```

Primer: Uvoz iz Excel tabela (6 minuta)

Tekst sa strane ubaciti u "FIT_IT_predmeti.xlsx" datoteku. Zatim, pročitati datoteku kroz [pandas](#) i prikazati u [jupyter](#) svesci.

Rešenje:

```
import pandas as pd
df2 = pd.read_excel('FIT_IT_predmeti.xlsx')
df2
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ATRIBUTI I METODE DATAFRAME OBJEKTA

Pri radu sa DataFrame objektima potrebno je znati najčešće korišćene atribute i metode

Konstrukcija [DataFrame](#) objekta iz drugih tipova podataka

Imenik

Ključ imenika jeste ime kolone, a lista vrednosti jesu sadržaj te kolone.

```
import pandas as pd

d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
```

Lista

Pri unosu liste, potrebno je dodatno naznačiti ime kolone.

```
import pandas as pd

lst = [1994, 1996, 1998, 1999, 2002, 2003, 2005, 2007, 2010, 2011, 2013, 2016, 2019]
df = pd.DataFrame(lst, columns=['Album Issued'])
```


Ukoliko je unutar liste još listi, onda je svaka lista jedna kolona.

Numpy Niz

Unos numpy niza je sličan kao kod unosa listi.

```
import numpy as np
import pandas as pd

n = np.array([1, 2, 3], [4, 5, 6], [7, 8, 9])
df = pd.DataFrame(n, columns=['a', 'b', 'c'])
```

Tipovi podataka unutar *DataFrame* objekta

atribut *.dypes* vraća tipove podataka po koloni

```
import pandas as pd
df = pd.DataFrame(...)

print(df.dtypes)
```

Broj redova i kolona

Atribut *.shape* vratiće tuple koji sadrži broj redova i kolona u *DataFrame* objektu

```
import pandas as pd
df = pd.DataFrame(...)

print(df.shape)
```

Broj redova i kolona i tipovi podataka

Metoda *.info()* vratiće informaciju o broju redova i kolona, ali i o tipovima podataka po koloni

```
import pandas as pd

df = pd.DataFrame(...)
print(df.info())
```

ATRIBUTI ZA PRIKAZ DATAFRAME OBJEKATA

Pri radu sa Jupyter sveskama, inicijalno se neće moći videti sve kolone i svi redovi, pa se moraju podesiti opcije prikaza.

Opcije prikaza

Pri radu sa Jupyter sveskama, inicijalno se neće moći videti sve kolone i svi redovi.

Ukoliko je potrebno da se vidi određeni broj redova i/ili kolona, potrebno je podesiti opcije prikaza

```
import pandas as pd

# DataFrame objekat sa puno redova i kolona
df = pd.DataFrame(...)

# podesavanje max kolona i redova
pd.set_option('display.max_columns', max_kolone)
pd.set_option('display.max_rows', max_redovi)
```

Pregled samo prvih *n* ili poslednjih *n* redova

Metodama `.head()` i `.tail()` moguće je imati pregled samo prvih ili samo poslednjih redova objekta

```
import pandas as pd

df = pd.DataFrame(...)

print(df.head(no_rows_from_first))
print(df.tail(no_rows_from_last))
```

Lociranje određenih elemenata

Pristup grupi redova i kolona po labelama moguće je korišćenjem metode `.loc[]`.

Ukoliko se koristi lociranje, a labele su celi brojevi, onda se koristi metoda `.iloc[]`

Brisanje kolone

Moguće je brisanje kolone komandom `del`.

```
import pandas as pd

df = pd.DataFrame(...)
del df['column_name']
```

```
import pandas as pd

df = pd.DataFrame(...)

df.loc['column_label']
df.iloc[integer]
```

PRIMER PRIKAZA KROZ KONZOLU I KROZ JUPYTER SVESKU

Kod pri normalnom radu i u Jupyter sveskama je (gotovo) identičan, ali je prikaz drugačiji.

Primer: Prikaz tabela kroz izlaz konzole i kroz izlaz Jupyter sveske (10 minuta)

Tekst u nastavku sačuvati kao "**FIT_SE_predmeti.csv**" datoteku, i postaviti je u radni direktorijum. Otvoriti preko pandas paketa i prikazati sadržaj.

Ponoviti isto kroz Jupyter svesku i uočiti razliku.

```
Fakultet,Smer,Godina,Semestar,Sifra,PunoIme
FIT,SE,1,1,CS101,Uvod u objektno-orjentisano programiranje
FIT,SE,1,1,IT101,Osnove informacionih tehnologija
FIT,SE,1,1,MA101,Matematika 1
FIT,SE,1,1,NT111,Engleski 1
FIT,SE,1,2,CS102,Objekti i apstrakcija podataka
FIT,SE,1,2,IT210,Sistemi informacionih tehnologija
FIT,SE,1,2,CS115,Diskretne strukture
FIT,SE,1,2,NT112,Engleski 2
FIT,SE,2,3,CS103,Algoritmi i strukture podataka
FIT,SE,2,3,SE201,Uvod u softversko inženjerstvo
FIT,SE,2,3,IT350,Baze podataka
FIT,SE,2,3,NT213,Engleski za informaticare
FIT,SE,2,4,CS230,Distribuirani sistemi
FIT,SE,2,4,IT370,Interakcija covek-racunar
FIT,SE,2,4,SE211,Konstruisanje softvera
FIT,SE,2,4,MA202,Matematika 2
FIT,SE,3,5,IT225,Veb sistemi 1
FIT,SE,3,5,SE311,Projektovanje i arhitektura softvera
FIT,SE,3,5,SE322,Inženjerstvo zahteva
FIT,SE,3,5,SE321,"Obezbedjivanje kvaliteta, testiranje i održavanje softvera"
FIT,SE,3,6,SE325,Upravljenje projektima razvoja softvera
FIT,SE,3,6,IT355,Veb sistemi 2
FIT,SE,3,6,CS220,Arhitektura racunara
FIT,SE,3,6,CS323,C/C++ programski jezik
FIT,SE,3,6,MA273,Verovatnoca i statistika
FIT,SE,3,6,IT375,Racunarsko upravljanje sistemima
FIT,SE,3,6,CS450,Klaud kompjuting
FIT,SE,3,6,CS330,Razvoj mobilnih aplikacija
FIT,SE,4,7,IT381,Zastita i bezbednost informacija
FIT,SE,4,7,IT376,Robotika
FIT,SE,4,7,CS322,C# programski jezik
FIT,SE,4,7,SE401,Timski razvoj softvera
FIT,SE,4,7,CS225,Operativni sistemi
FIT,SE,4,7,CS324,Skripting jezici
FIT,SE,4,7,OM350,Preduzetnistvo
FIT,SE,4,8,IT390,Prefesionalna praksa i etika
FIT,SE,4,8,NT310,Profesionalna komunikacija
FIT,SE,4,8,SE490,Strucna praksa (4 meseca)
FIT,SE,4,8,SE495,Zavrsni rad
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Rešenje:

Kod je isti, ali prikaz biće drugačiji.

```
import pandas as pd

df = pd.read_csv('FIT_SE_predmeti.csv')
print(df)
```

Broj predmeta	Naziv predmeta	Broj predmeta	Naziv predmeta	Broj predmeta	Naziv predmeta
1	Matematika	1	Matematika	1	Matematika
2	Fizika	2	Fizika	2	Fizika
3	Hemija	3	Hemija	3	Hemija
4	Biologija	4	Biologija	4	Biologija
5	Geografija	5	Geografija	5	Geografija
6	Istorija	6	Istorija	6	Istorija
7	Književnost	7	Književnost	7	Književnost
8	Musika	8	Musika	8	Musika
9	Umjetnost	9	Umjetnost	9	Umjetnost
10	Šport	10	Šport	10	Šport
11	Pravna nauka	11	Pravna nauka	11	Pravna nauka
12	Medicina	12	Medicina	12	Medicina
13	Poljoprivreda	13	Poljoprivreda	13	Poljoprivreda
14	Arhitektura	14	Arhitektura	14	Arhitektura
15	Informacione tehnologije	15	Informacione tehnologije	15	Informacione tehnologije
16	Ekologija	16	Ekologija	16	Ekologija
17	Antropologija	17	Antropologija	17	Antropologija
18	Sociologija	18	Sociologija	18	Sociologija
19	Psihologija	19	Psihologija	19	Psihologija
20	Statistika	20	Statistika	20	Statistika
21	Ekonomika	21	Ekonomika	21	Ekonomika
22	Pravna nauka	22	Pravna nauka	22	Pravna nauka
23	Medicina	23	Medicina	23	Medicina
24	Poljoprivreda	24	Poljoprivreda	24	Poljoprivreda
25	Arhitektura	25	Arhitektura	25	Arhitektura
26	Informacione tehnologije	26	Informacione tehnologije	26	Informacione tehnologije
27	Ekologija	27	Ekologija	27	Ekologija
28	Antropologija	28	Antropologija	28	Antropologija
29	Sociologija	29	Sociologija	29	Sociologija
30	Psihologija	30	Psihologija	30	Psihologija
31	Statistika	31	Statistika	31	Statistika
32	Ekonomika	32	Ekonomika	32	Ekonomika
33	Pravna nauka	33	Pravna nauka	33	Pravna nauka
34	Medicina	34	Medicina	34	Medicina
35	Poljoprivreda	35	Poljoprivreda	35	Poljoprivreda
36	Arhitektura	36	Arhitektura	36	Arhitektura
37	Informacione tehnologije	37	Informacione tehnologije	37	Informacione tehnologije
38	Ekologija	38	Ekologija	38	Ekologija
39	Antropologija	39	Antropologija	39	Antropologija
40	Sociologija	40	Sociologija	40	Sociologija
41	Psihologija	41	Psihologija	41	Psihologija
42	Statistika	42	Statistika	42	Statistika
43	Ekonomika	43	Ekonomika	43	Ekonomika
44	Pravna nauka	44	Pravna nauka	44	Pravna nauka
45	Medicina	45	Medicina	45	Medicina
46	Poljoprivreda	46	Poljoprivreda	46	Poljoprivreda
47	Arhitektura	47	Arhitektura	47	Arhitektura
48	Informacione tehnologije	48	Informacione tehnologije	48	Informacione tehnologije
49	Ekologija	49	Ekologija	49	Ekologija
50	Antropologija	50	Antropologija	50	Antropologija

Slika 5.3 Prikaz datoteke "FIT_SE_predmeti.csv" kroz konzolu. [Izvor: Autor]

Slika 5.4 Prikaz datoteke "FIT_SE_predmeti.csv" kroz Jupyter svesku. [Izvor: Autor]

▼ Poglavlje 6

Pokazne vežbe #8

UVOD U POKAZNE VEŽBE

Pokazne vežbe rade se 45 minuta.

Pokazna vežba #8 odnosi se na rad sa numpy i pandas paketima.

U pokaznoj vežbi prelaze se 4 zadatka, po dva za svaki paket.

Zadatak #1 odnosi se na rad sa transponovanim vektorima.

Zadatak #2 odnosi se na ubacivanje matrica u matricu.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADATAK #1 (15 MINUTA)

Zadatak #1 odnosi se na rad sa transponovanim nizovima

Zadatak #1 (15 minuta)

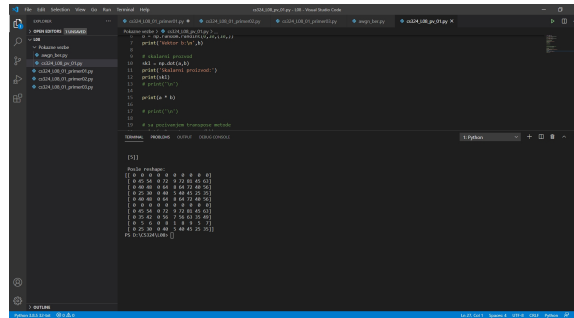
Napraviti dva niza od po sto elemenata. Svaki element je nasumično izabran broj od 0 do 10.

- 1. Pronaći skalarni proizvod vektora (en. **dot product**) koji se računa:

$$a \cdot b = \sum_{i=1}^n a_i b_i$$

- 2. Izračunati proizvod:

$$\begin{aligned} a \times b^T \\ b^T \times a \end{aligned}$$



Slika 6.1 Izlaz poslenjeg proizvoda kod pokazne vežbe #1. [Izvor: Autor]

```
import numpy as np

a = np.random.randint(0,10,(10,))
print('Vektor a:\n',a)

b = np.random.randint(0,10,(10,))
print('Vektor b:\n',b)

# skalarni proizvod
skl = np.dot(a,b)
print('Skalarni proizvod:')
print(skl)

# # poziv reshape petode
print(b)
print('posle pravog trnasponovanja')
b = b.reshape(-1,1)
print(b)

print('\n Posle reshape:')
print(b * a)
```

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADATAK #2 (5 MINUTA)

Zadatak #2 odnosi se na rad sa kreiranje željene matrice.

Zadatak #2 (5 minuta)

Napisati program koji će napraviti sledeću matricu:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 9 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

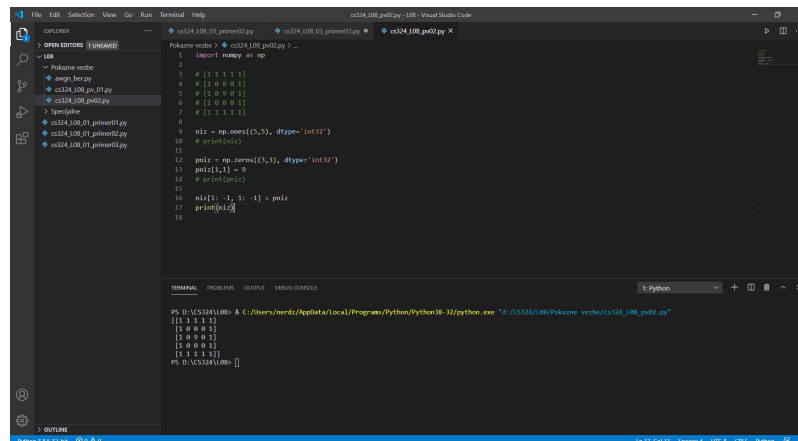
Moguće je direktno ubaciti samo cifru 9, a za ostale elemente koristiti numpy funkcije.

```
import numpy as np

# inicijalizovanje prvog niza
niz = np.ones((5,5), dtype='int32')
print(niz)

# pomocni, unutrasni niz
pomocni_niz = np.zeros((3,3), dtype='int32')
pomocni_niz[1,1] = 9
print(pomocni_niz)

# ubacivanje jednog niza u drugi
niz[1:-1, 1:-1] = pomocni_niz
print(niz)
```



Slika 6.2 Konstrukcija željene matrice. [Izvor: Autor]

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADATAK #3 (10 MINUTA)

Zadatak #3 odnosi se na preveru vrednosti DataFrame objekta i maskiranje

Zadatak #3 (10 minuta)

Napisati program koji će kroz pandas napraviti sledeću tabelu:

AAA	BBB	CCC
4	10	100
5	20	50
6	30	-30
7	40	-50

Zatim, uraditi sledeće:

- Ukoliko su vrednosti kolone AAA veće ili jednake broju 5, onda će u tim redovima, vrednosti kolone BBB biti jednake -1
- Ukoliko su vrednosti kolone AAA veće ili jednake broju 5, onda će u tim redovima, vrednosti kolona BBB i CCC biti jednake 555.
- Postaviti masku df_mask. i primeniti na tabelu, sa vrednošću -1000. Maska je data:

```
df_mask = pd.DataFrame({'AAA': [True] * 4,
                        'BBB': [False] * 4,
                        'CCC': [True, False] * 2})
```

Rešenje:

```
import pandas as pd

df = pd.DataFrame({'AAA': [4, 5, 6, 7],
                  'BBB': [10, 20, 30, 40],
                  'CCC': [100, 50, -30, -50]})

print(df)

#1
df.loc[df.AAA >= 5, 'BBB'] = -1
print(df)

#2
df.loc[df.AAA >= 5, ['BBB', 'CCC']] = 555
print(df)

#3
df_mask = pd.DataFrame({'AAA': [True] * 4,
                        'BBB': [False] * 4,
                        'CCC': [True, False] * 2})
print(df.where(df_mask, -1000))
```

ZADATAK #4 (15 MINUTA)

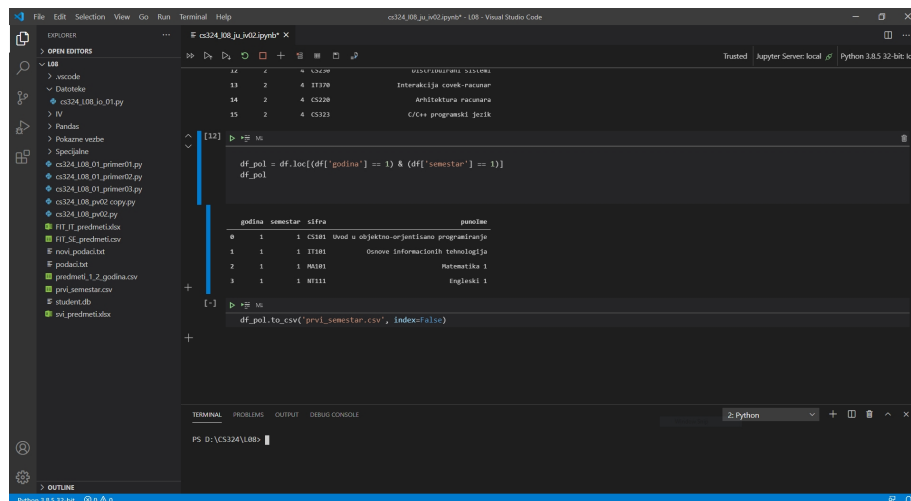
Zadatak #4 se odnosi na odabir elemenata DataFrame objekta po više kriterijuma.

Zadatak #4 (15 minuta)

Dat je CSV dokument "predmeti_1_2_godina.csv" sa podacima u nastavku.

Učitati i odštampati kao DataFrame objekat. Zatim, pronaći predmete iz prve godine i iz prvog semestra, i smestiti i novi DataFrame objekat.

Štampati novi DataFrame objekat u CSV datoteku bez indeksa.



Slika 6.3 Izlaz preko Jupyter svezaka. [Izvor: Autor]

```

godina,semestar,sifra,punoIme
1,1,CS101,Uvod u objektno-orijentisano programiranje
1,1,IT101,Osnove informacionih tehnologija
1,1,MA101,Matematika 1
1,1,NT111,Engleski 1
1,2,CS102,Objekti i apstrakcija podataka
1,2,IT210,Sistemi informacionih tehnologija
1,2,CS115,Diskretne strukture
1,2,NT112,Engleski 2
2,3,IT331,Racunarske mreze i komunikacije
2,3,SE201,Uvod u softversko inženjerstvo
2,3,IT350,Baze podataka
2,3,NT213,Engleski za informaticare
2,4,CS230,Distribuirani sistemi
2,4,IT370,Interakcija covek-racunar
2,4,CS220,Arhitektura racunara
2,4,CS323,C/C++ programski jezik

```

```

import pandas as pd

df = pd.read_csv('predmeti_1_2_godina.csv')
print(df)

df_pol = df.loc[(df['godina'] == 1) & (df['semestar'] == 1)]
df_pol
print(df_pol)

df_pol.to_csv('prvi_semestar.csv', index=False)

```

▼ Poglavlje 7

Individualne vežbe #8

INDIVIDUALNE VEŽBE

Zadaci individualnih vežbi se ukupno rade 90 minuta

Zadatak #1 (45 minuta)

Korišćenjem numpy generisati niz od 10000 elemenata, koji sadrži elemente 0 i 1. Prebrojati koliko ima vrednosti 0 i koliko ima vrednosti 1. Zatim, sve vrednosti koje su 0 prebaciti u -1.

Napraviti novi niz koji ima nasumične razlomljene elemente u vrednosti od -1.5 do +1.5. Sabrati elemente niza u novi niz.

Elemente novog niza preurediti tako da svi negativni elementi postanu nule, a svi pozitivni elementi postanu jedinice.

Prebrojati nule i jedinice, i uporediti sa brojem jedinice i nula prvobitnog niza. Štampati razlike u nulama i jedinicama podeljeno sa brojem elemenata (10000). Ovo je verovatnoća greške.

Ponoviti za 1000 elemenata i 100000 elemenata.

Uraditi isti zadatak bez numpy i uporediti brzinu izvršenja.

Zadatak #2 (45 minuta)

Pomoću sqlite3 paketa napraviti bazu podataka "moji_predmeti.db" sa kolonama:

```
Godina: Int # Godina na kome ste slusali predmet
Semestar: Int # Semestar u kome ste slusali predmet
Sifra: Text # Sifra predmeta
PunoIme: Text # Puno ime predmeta
Polozen: Bool # Status predmeta (True = polozen, False = nije polozen)
Ocena: Int # Nema ocenu ako nije polozen
```

Zatim, napraviti funkciju koja učitava predmete (unos se sa konzole) i ubacuje u bazu podataka.

Pomoću pandas paketa uzeti ceo sadržaj baze. Napraviti filter koji će ispisivati samo položene predmete. Sačuvati novu tabelu u datoteku "polozeni_predmeti.csv". Tabela ime sledeće kolone:

```
Sifra: Text # Sifra predmeta
PunoIme: Text # Puno ime predmeta
Ocena: Int # Ocena
```

Godina: Int # Godina na kome ste slusali predmet
Semestar: Int # Semestar u kome ste slusali predmet

▼ Poglavlje 8

Domaći zadatak #8

ZADACI ZA DOMAĆI ZADATAK #8

Osmi domaći zadatak se radi okvirno 3h.

Domaći zadatak #1

Korišćenjem numpy paketa napraviti sledeće matrice. Nije dozvoljen ručni unos vrednosti po indeksu, već treba uneti matricu u matricu.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 \\ 0 & 10 & 10 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Domaći zadatak #2

Napraviti bazu podataka "biblioteka.db", koja sadrži tabelu:

```
knjige:
(katBroj int, naslov text, izdavac text, godinaIzdavanja int, izdata bool)
```

Napraviti funkciju za ubacivanja novih knjiga sa konzole. Ubacuje se **katBroj**, **naslov**, **izdavac**, **godinalzdanja**, a atribut **izdat** se podrazumevano stavlja na **False**. Uхватiti izuzetak ukoliko je knjiga izdata u budućnosti (proveriti preko **datetime** paketa).

Napraviti funkciju **podesilzdat(katBroj)** koja podešava vrednost **izdat** i upisuje u bazu novu vrednost za datu knjigu.

Preko pandas paketa učitati sadržaj tabele knjige. Napraviti novi DataFrame objekat **savremene_knjige** koji će sadržati knjige kod kojih je godina izdavanja od 2000. godine.

Upisati u csv datoteku sve knjige koje su izdate u datoteku **izdate_knjige.csv** sa kolonama **Naslov, izdavac, godinalzdavanja**.

PREDAJA DOMAĆEG ZADATKA

Nakon 10 dana od lekcije, poeni za domaći zadatak za tradicionalne studente se umanjuju za 50%.

Tradicionalni studenti:

Domaći zadatak treba dostaviti najkasnije nedelju dana nakon predavanja za 100% poena. Nakon toga poeni se umanjuju za 50%.

Internet studenti:

Domaći zadatak treba dostaviti najkasnije 10 dana pred polaganja ispita. Domaći zadaci se brane!

Domaći zadatak poslati dr Nemanji Zdravkoviću: nemanja.zdravkovic@metropolitan.ac.rs

Obavezno koristiti uputstvo za izradu domaćeg zadatka.

Uz .doc dokument (koji treba sadržati i screenshot svakog urađenog zadatka kao i komentare za zadatak), poslati i izvorne i dodatne datoteke.

▼ Poglavlje 9

Zaključak

ZAKLJUČAK

Zaključak lekcije #8

Rezime:

U ovoj lekciji bilo je reči o paketima numpy i pandas za Python 3 programski jezik. Oba paketa se instaliraju preko pip upravljača paketa.

Paket NumPy radi sa homogenim nizovima i više je optimizovan nego pri radu sa listama, zbog vektorizacije izvršenja operacija. NumPy se može smatrati alternativom za MATLAB programski jezik.

Paket pandas služi za manipulaciju podacima koja se mogu predstaviti tabelarno. Pandas pruža mnoge mogućnosti sortiranja, filtracije nad podacima, ali i podržava veliki broj formata koji koriste razne aplikacije, pa i baze podataka.

Literatura:

- David Beazley, Brian Jones, Python Cookbook: Recipes for Mastering Python 3, 3rd edition, O'Reilly Press, 2013.
- Mark Lutz, Learning Python, 5th Edition, O'Reilly Press, 2013.
- Andrew Bird, Lau Cher Han, et. al, The Python Workshop, Packt Publishing, 2019.
- Al Sweigart, Automate the boring stuff with Python, 2nd Edition, No Starch Press, 2020.

