Pokretanje raznih primjera iz postavke laboratorijske vježbe

```
1 print('Jupyter Notebook - primjer')
2 x = 42
```

```
Jupyter Notebook - primjer
```

Tekst

```
1 # Kodne celije dijele isti prostor za varijable
2 print(x)
```

```
42
```

```
1 import numpy as np
```

```
1 a = np.arange(15).reshape(3, 5)
2 print(a)
3 a = np.array([[ 1, 1, 2, 3, 4],[ 5, 6, 7, 8, 9],[10, 11, 12, 13, 14]])
4 print(a)
5 print(a.shape)
6 print(a.ndim)
7 print(a.dtype)
8 print(a.itemsize)
9 print(a.size)
10 b = np.array([42.0, 42.1, 42.2])
11 print(b.dtype)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
[[ 1  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
(3, 5)
2
int64
8
15
float64
```

```
1 a = np.zeros((3,4))
2 print(a)
3 a = np.ones((2,3,4)) # niz matrica
4 print(a)
```

```
b = np.empty((3,3))
print(b)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]

 [[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]]
[[0. 0. 0.]
 [0. 0. 0.]
```

```
a = np.array( [20, 30, 40, 50] )
b = np.arange(4)
print(b)
c = a - b
print(c)
print(b ** 2)
print(10 * np.sin(a))
print(a < 35)

A = np.array([[1,1], [0,1]])
B = np.array([[2,0], [3,4]])
print(A * B)
print(A @ B)
```

```
[0 1 2 3]
[20 29 38 47]
[0 1 4 9]
[ 9.12945251 -9.88031624  7.4511316  -2.62374854]
[ True  True False False]
[[2 0]
 [0 4]]
[[5 4]
 [3 4]]
```

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

x = np.arange(0, np.pi*2, 0.05)
y = np.sin(x)
plt.plot(x, y)
plt.xlabel("Ugao")
plt.ylabel("Sinus")
plt.title("Sinusoida")
plt.show()
```

Zadaci za rad u laboratoriji

Zadatak 1 - osnovno izračunavanje

```
1 import math
2 print(13 * 2.5 ** (124/9.3) + math.sqrt(3))
3 print(math.cos(math.pi / 3) + math.tan(math.pi) + (math.e)**(math.pi))
4 print((2 + 3 ** 8) * 10)
5 print(79 // 18)
```

```
2629120.071450335
23.640692632779263
65630
4
```

Zadatak 2

a) Napisati Python program koji prvo od korisnika traži unos 10 elemenata liste. Nakon toga, na ekranu ispisati drugi najveći element u listi.

```
1 import numpy as np
2 lista = []
3 n=10
4 for i in range(0,n):
5   element = int(input("Unesite "+ str(i+1)+". element: "))
6   lista.append(element)
7
8 print("Lista glasi: ", lista)
9 print("Second largest element is:", sorted(lista)[-2])
```

```
Unesite 1. element: 0
Unesite 2. element: 0
Unesite 3. element: 0
Unesite 4. element: 0
Unesite 5. element: 121
Unesite 6. element: 90
Unesite 7. element: 0
Unesite 8. element: 0
Unesite 9. element: 0
Unesite 10. element: 0
Lista glasi:  [0, 0, 0, 0, 121, 90, 0, 0, 0, 0]
Second largest element is: 90
Lista glasi:  [0, 0, 0, 0, 121, 90, 0, 0, 0, 0]
```

b) Napisati Python program koji za početnu vrijednost n računa n-ti Fibonacci broj. Obavezno koristiti for petlju.

```
def fibonaccijevBroj(n):
  fibonacijeviBrojevi = [0, 1]
  if n > 2:
    for i in range (2, n):
      fibonacijeviBrojevi.append(fibonacijeviBrojevi[i-1] + fibonacijeviBrojevi[i-2])
  return fibonacijeviBrojevi[n-1];

n = int(input("Unesite broj n: "))
if n<= 0:
  print("Unijeli ste negativan broj")
else:
  print(str(n)+". Fibonacijev broj je: "+str(fibonaccijevBroj(n)))
```

```
Unesite broj n: 5
5. Fibonacijev broj je: 3
```

c) Napisati Python program koji za zadane vrijednosti a, b, i c računa vrijednosti rješenja kvadratne jednačine, tj.

x1,2 = −b ± √ b 2 − 4ac 2a

U slučaju da je diskriminanta negativna, napisati "Rješenja su kompleksna!" i ne računati rješenja. U slučaju da je diskriminanta nula, ispisati "Rješenja su ista!" i ispisati rješenje. Obavezno koristiti if strutkuru.

```
a, b, c = input("Unesite koeficijente a,b i c: ").split()
print("Jednacina glasi: {}x1+{}x2+{}x3".format(a, b, c))
a = int(a)
b = int(b)
c = int(c)
diskriminanta = b ** 2 - 4 * a * c
```

```
print("Diskriminanta iznosi: ", diskriminanta)
if diskriminanta > 0:
  x1 = (-1 * b + diskriminanta) / (2 * a)
  x2 = (-1 * b - diskriminanta) / (2 * a)
  print("Rješenja kvadratne jednačine su: ",str(x1)+" i "+ str(x2))
elif diskriminanta == 0:
  x = (-1 * b) / (2 * a)
  print("Rješenja su ista!")
  print("X = ",x)
else:
  print ("Rješenja su kompleksna!")
```

```
Unesite koeficijente a,b i c: 1 5 6
Jednacina glasi: 1x1+5x2+6x3
Diskriminanta iznosi:  1
Rješenja kvadratne jednačine su:  -2.0 i -3.0
```

d) Definisati funkciju fact(n) koja računa faktorijel broja n. Zatim definisati funkciju choose(n, k) koja računa n k

. Demonstrirati rad ovih funkcija na primjeru.

```
def fact(n):
  return 1 if (n==1 or n==0) else n * fact(n - 1);

def choose(n,k):
  return fact(n)/(fact(k)*fact(n-k))

n, k = input("Unesite brojeve n i k: ").split()
n = int(n)
k = int(k)
print("Primjer racunanja faktorijela: {}!={}".format(n,fact(n)))
print("Primjer racunanja binomnog koeficijenta: ({} {})={}".format(n,k,choose(n,k)))
```

```
Unesite brojeve n i k: 5 3
Primjer racunanja faktorijela: 5!=120
Primjer racunanja binomnog koeficijenta: (5 3)=10.0
```

Zadatak 3 - Rad sa matricama

Pomoću NumPy, generisati sljedeće matrice: a) Matricu dimenzija 4 × 3 čiji su svi elementi nule;

b) Matricu dimenzija 2 × 2 čiji su svi elementi jedinice;

c) Vektor-kolonu dužine 10 čiji su elementi brojevi od 1 do 10;

d) Matricu datu u nastavku:

e) Za prethodno kreiranu matricu, izračunati:

```
(a) Transponovanu matricu;
(b) Matricu dobivenu sabiranjem transponovane matrice i originalne matrice;
(c) Proizvod matrice sa sumom njene prve kolone.
```

f) Za matricu M prikazati:

```
(a) Treći red matrice;
(b) Drugu kolonu matrice;
(c) Element na lokaciji (0, 2).
```

g) Definisati funkciju kvadratna(M) koja prima NumPy 1D niz, te isti preoblikuje u kvadratnu matricu, ukoliko je to moguće. Ukoliko nije moguće, funkcija ne treba da uradi ništa.

```
import numpy as np
print("a)--------------------------")
matrica = np.zeros((4,3))
print(matrica)

print("b)-----------------------------")
matrica = np.ones((2,2))
print(matrica)

print("c)-----------------------------")
matrica = np.arange(10).reshape(10,1)+1
print(matrica)

print("d)-----------------------------")
matrica = np.arange(9).reshape(3,3)+1
#matrica = matrica.astype(np.float64)
matrica[1][0] = 1/3.6
matrica[1][2] = 23
matrica[2][0] = 2 ** 10.5
matrica[2][1] = 42
matrica[2][2] = np.cos(80.841)
print(matrica)

print("e)--------------------------")
transponovana = matrica.transpose()
zbir_matrica = matrica + transponovana
suma_kolone = sum(matrica[:,0])
proizvod = matrica * suma_kolone
print("Transponovana matrica: \n",transponovana)
```

```
print("Zbir pocetne i njene transponovane matrice: \n",zbir_matrica)
print("Proizvod matrice i njene sume prve kolone \n", proizvod)

print("f)--------------------------")
print("Treći red matrice: \n",matrica[2,:])
print("Druga kolona matrice: \n", matrica[:,1])
print("Element na lokaciji (0,2): \n",matrica[0][2])

print("g)-------------------------")
def kvadratna(M):
  velicina = len(M)
  korijen_velicine = velicina ** 0.5
  provjera_ostatka = (korijen_velicine % 1) == 0
  if provjera_ostatka:
    korijen_velicine = int(korijen_velicine)
    M = np.reshape(M,(korijen_velicine,korijen_velicine))
    return M
M = np.arange(9)
M = kvadratna(M)
print(M)
```

```
a)--------------------------
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
b)------------------------------
[[1. 1.]
 [1. 1.]]
c)------------------------------
[[ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]]
d)------------------------------
[[   1    2    3]
 [   0    5   23]
 [1448   42    0]]
e)--------------------------
Transponovana matrica:
 [[   1    0 1448]
 [   2    5   42]
 [   3   23    0]]
Zbir pocetne i njene transponovane matrice:
 [[   2    2 1451]
 [   2   10   65]
```

```
 [1451   65    0]]
Proizvod matrice i njene sume prve kolone
 [[   1449    2898    4347]
 [      0    7245   33327]
 [2098152   60858       0]]
f)--------------------------
Treći red matrice:
 [1448   42    0]
Druga kolona matrice:
 [ 2  5 42]
Element na lokaciji (0,2):
 3
g)-------------------------
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

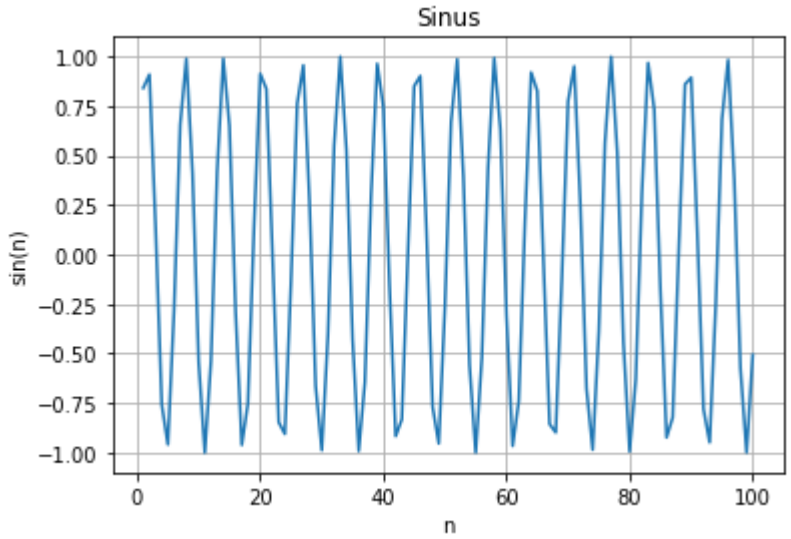
Zadatak 4 - Rad sa grafikom

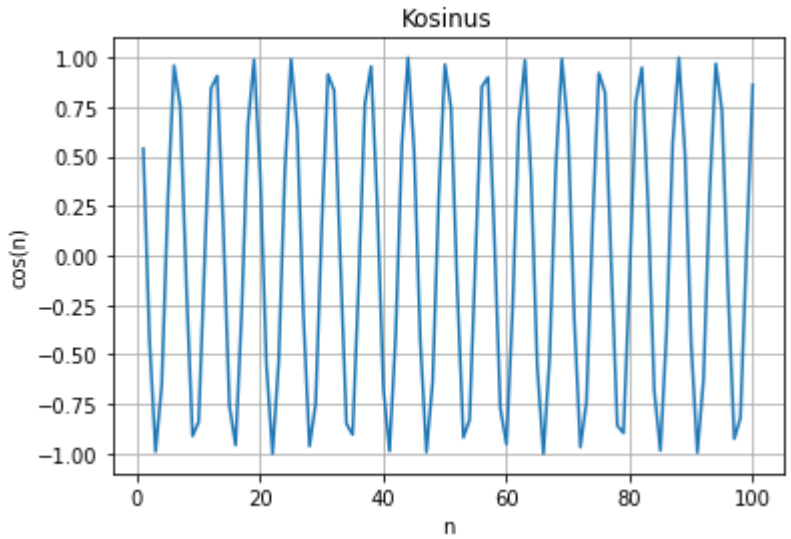
a) Neka je data funkcija x = sin(n) na 100 tačaka (n = 1:100). Iscrtati linijski dijagram ove funkcije. Postaviti odgovarajuće labele, te naslov. Zatim preko tog dijagrama (na isti graf) nacrtati funkciju y = cos(n), također na 100 uzoraka. Uključiti mrežu (grid);

```
 1 import matplotlib.pyplot as plt
 2 import numpy as np
 3
 4 n = np.arange(1,101)
 5 x = np.sin(n)
 6 plt.plot(n,x, label = 'sin(n)')
 7 plt.xlabel("n")
 8 plt.ylabel("sin(n) & cos(n)")
 9 plt.title("Sinus i kosinus")
10
11 y = np.cos(n)
12 plt.plot(n,y, label = 'cos(n)')
13 plt.grid()
14 plt.legend()
15 plt.show()
16
```

b) Ponoviti prethodni zadatak, ali sada iscrtati u istom prozoru (ali na različitim graficima) signale x i y. Postaviti odgovarajuće labele i naslove za oba grafika;

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

n = np.arange(1, 101, 1)
x = np.sin(n)
y = np.cos(n)

plt.figure(1)
plt.plot(n, x)
plt.xlabel("n")
plt.ylabel("sin(n)")
plt.title("Sinus")
plt.grid()

plt.figure(2)
plt.plot(n,y)
plt.xlabel("n")
plt.ylabel("cos(n)")
plt.title("Kosinus")
plt.grid()

plt.show()
```





✓ 0s completed at 5:47 PM