##Obrnuta lista

lista = [ 1, 3, 5, 9, 11, 27]

obrnutaLista = lista[::-1]

print("Originalna lista: ", lista)

print("Obrnuta lista: ", obrnutaLista)

##Element na "n" poziciji ( n = 0,1,2,3...)

odabraniElement = lista[3]

##zadnja 2 elementa

zadnjaDva = lista[-2:]

##svako drugi element

svakoDrugi = lista[::2]

##Obrnuti niz znakova

nizZnakova = "abcdefghijkl"

obrnutiNizZnakova = nizZnakova[::-1]

print("Originalni niz znakova:", nizZnakova)

print("Obrnuti niz znakova:", obrnutiNizZnakova)

##Zamjena vrijednosti varijabli

a = 10

b = 15

print(a,b)

a,b = b,a

print(a,b)

##Spremanje vrijednosti u varijable iz liste

#Inicijalizirati listu proizvoljnog sadrzaja

lista = [ 4, 15, 23]

a,b,c = lista

print(a, b, c)

##Ulancana usporedba

a = 10

print(3<a<20)

print(10<a<20)

##else se nece izvrsiti ( break )

lista = [1, 2, 3, 4, 5, 6]

for element in lista:

if element == 5:

break

else:

print("Nije pronađeno!")

##Ternarni operator ( na osnovu logickog operatora - true/false odlucuje koji ce se izraz izvrsiti)

x = 10 if (True) else 20

print(x)

x = 10 if (False) else 20

print(x)

##Prva f-ija se izvrsava ako je True, druga ako je False

def funPrva(vrijednost):

print("funPrva(), vrijednost:", vrijednost)

def funDruga(vrijednost):

print("funDruga(), vrijednost:", vrijednost)

x = True

(funPrva if x else funDruga) (5)

##"pass" koristimo za uvjet gdje ne zelimo da se bilo sta izvrsava ( jer ne mozemo ostaviti prazno )

#pass ne prekida izvrsavanje linija koda koje se u tijelu nalaze ispod nje

rijec = "Srxce"

for e in rijec:

if e == 'x':

pass

print("Ključna riječ pass")

else:

print(e)

##Vracanje vise vrijednosti

def fun():

return 1, 2, 3, 4, 5

a, b, c, d, e = fun()

print(a, b, c, d, e)

##Zamjena vrijednosti varijabli

a = 5

b = 13

print(a,b)

a,b = b,a

print(a,b)

##S tipkovnice ucitajte broj proizvoljne vrijednosti. Za tako ucitan broj provjerite zadovoljava li on uvjet

# 10<broj<30, ako je uvjet zadovoljen, ispisite na ekran "Zadovoljava!", ako ne, ispisite "Ne zadovoljava!".

# Ovaj zadatak potrebno je rijesiti koristenjem operatora usporedbe, bez koristenja logickih operatora i

# logickih izraza

x = int(input("Unesite broj: "))

if 10 < x < 30:

print("Zadovoljava!")

else:

print("Ne zadovoljava!")

##Kreirajte listu te ju popunite s 5 proizvoljnih brojeva. Nakon toga ucitajte proizvoljni broj s tipkovnice.

#Unutar petlje for provjerite nalazi li se vrijednost koju ste ucitali sa tipkovnice unutar liste. Ako

#trazena vrijednost postoji u listi, ispisite na ekran "Postoji!", u suprotnom ispisite "Ne postoji!".

lista =[ 5, 7, 13, 21, 27]

broj = int(input("Unesite broj:"))

for x in lista:

if x == broj:

print("Postoji!")

break

else:

print("Ne postoji!")

#S tipkovnice ucitajte broj proizvoljne vrijednosti. Pomocu ternarnog operatora detektirajte je li

#ucitani broj paran ili neparan. Ako je paran, ispisite "Paran", inace "Neparan".

broj = int(input("Unesite broj:"))

print("Paran!" if broj%2==0 else "Neparan!")

##Implementirajte dvije funkcije. Prva neka se zove paran(), i ona neka ispisuje "Paran", druga neparan()

# i neka ispisuje "Neparan". S tipkovnice ucitajte broj proizvoljne vrijednosti, te pomocu ternarnog

# operatora detektirajte je li ucitana vrijednost parna ili neparna, te ovisno o parnosti pozovite jednu

#od prethodno kreiranih funkcija.

def paran():

print("Paran")

def neparan():

print("Neparan!")

broj = int(input("Unesite broj:"))

paran() if broj%2==0 else neparan()

#moze i (paran if broj % 2 == 0 else neparan)()

##Implementirajte funkciju prototipa rezultat(var1,var2), funkcija moze primiti 2 vrijednosti. Tako kreirana

# funkcija neka u pozivajuci dio programa vraca tri vrijednosti dobivene sljedecim aritmetickim operacijama

#var1\*var2, var1+var2, 10\*var1+var2. U glavnom dijelu programa pozovite prethodno implementiranu f-ju

# te njene povratne vrijednosti spremite u tri zasebne varijable proizvoljnog imena i ispisite ih

def rezultat(var1, var2):

a = var1\*var2

b = var1+var2

c = 10\*var1+var2

return a,b,c

var1,var2, var3 = rezultat(3, 5)

print(var1, var2, var3)

##Napisite program unutar kojeg ce biti inicijalizirane dvije varijable proizvoljnog imena. U prvu varijablu

#spremite niz znakova: "Hello World!, a u drugu varijablu spremite listu [1,2,3,4]. Ispisite na ekran niz

#znakova i listu u obrnutom poretku na dva nacina. Najprije vrijednosti ispisite pomocu petlje, a nakon

#toga vrijednosti ispisite koristenjem "reverse()"

niz = "Hello WOrld!"

lista = [1,2,3,4]

i = len(niz) - 1

print("Niz znakova (petlja): ", end="")

while i>=0:

print(niz[i], end="")

i -= 1

print("\nLista (petlja): ", end="")

i = len(lista) - 1

while i>=0:

print(lista[i], end="")

i -=1

print("\nNiz znakova", niz[::-1])

print("Lista: ", lista[::-1])

##Rekurzija

def faktorijel(x):

if x <= 1:

return 1

else:

return x \* faktorijel(x - 1)

rez = faktorijel(4)

print(rez)

##Bezimena funkcija

zbroji = lambda a,b,c: a + b + c

print(zbroji(1,2,3))

##Ugnijezdjene funkcije

def vanjska(x):

print("Vanjska funkcija, x =", x)

def unutarnja(y):

print("Unutarnja funkcija, y =", y)

unutarnja(x \* 2)

vanjska(10)

#numpy insert

#Definition : insert(arr, obj, values, axis=None)

#B = np.insert(B, 3, np.zeros((2 ,3)), axis=0)

#3 = gdje radimo insert, tj na indexu = 3 ( 0,1,2,3...)

#np.zeros((2,3)) = formiramo matricu nula sa 2 retka i 3 kolone ( kad je axis ?= 0)

#axis = 0, osa x; axis = 1, osa y

##Za mnozenje matrica koristimo dot() metodu. \* se koristi za mnozenje nizova.

import numpy as np

A = np.array([[3, 6, 7], [5, -3, 0]])

B = np.array([[1, 1], [2, 1], [3, -3]])

C = A.dot(B)

print(C)

#Kvadratna matrica - ako je broj redaka m jednak broju stupaca n

#Dijagonalna - matrica ciji su elementi izvan glavne dijagonale jednaki 0, odnosno Aij=0 za sve i != j

#Jedinicna - dijagonalna matrica ciji su elementi na glavnoj dijagonali jednaki 1. Standardna oznaka je I

#Nulmatrica - svi elementi jednaki 0, oznacava se sa O.

#Gornjotrokutasta - kvadratna matrica ciji su elementi ispod glavne dijagonale jednaki 0

#Donjotrokutasta - kvadratna matrica ciji su elementi iznad glavne dijagonale jednaki 0

#Matrice koje imaju samo jedan redak ili stupac, nazivamo vektorima. Matrica tipa 1,n, je jednoredna

#matrica, ili vektor-redak, a tipa m,1, onda jednostupcana matrica ili vektor-stupac.

#Kod transponiranja, retci matrice A postaju stpci matrice AT

#Zbrajati i oduzimati se mogu samo matrice istog tipa, a rezultat je opet matrica istog tipa

#Mozemo mnoziti samo ulancane matrice, tj ako je broj stupaca matrice A jednak broju redaka matrice B

##Generirati marticu A dimenzija 4x5 sa slucajno odabranim cijelim brojevima od 0 do 100

#Potencirati svaki element matrice potencijom 2

#Generirati matricu B dimenzija 3x3 slucajno odabranim brojevima od 1 do 10

#Pomnoziti matricu A i B ( dopuniti matricu B nulama kako bi dimenzije dviju matrica bile iste )

#Rezultat spremiti u varijablu "rezultat" i svaki element u dobivenoj matrici povecati za 2

#U prvi redak rezultata postaviti sve nule umjesto postojecih elemenata

#Transponirati dobivenu matricu rezultat

#U varijablu b spremiti vektor ciji su elementi sume elemenata stupca matrice

#Ispisati dobiveni vektor b

#Napomena: obavezno koristiti naredbe za rad s matricama i rjesenje spremiti kao zadatak2.py i pozivati

#iz naredbenog retka

import numpy as np

a = np.random.rand(4,5)\*100

A = np.fix(a)

print("A=", A)

POT\_A = A\*A

print(POT\_A)

b = np.random.rand(3,3)\*10

B = np.fix(b)

print("B=", B)

B = np.insert(B, 3, np.zeros((2 ,3)), axis=0)

B = np.insert(B, 3, np.zeros((1,5)), axis=1)

print("B=", B)

rezultat = np.dot(A,B)

print("rezultat=", rezultat)

rezultat = rezultat + 2

rezultat[0, :] = 0

print("rezultat=", rezultat)

rez\_transpose = rezultat.transpose()

print("Transponirana: ", rez\_transpose)

b = np.sum(rezultat, axis=1)

print("SUma=", b)

##Generirati matrice A = ( 8 po dijagonali i u zadnjem stupcu/koloni) i B = sve 3

#Dopuniti matricu B tricama kako bi dimenzije dviju matrica bile iste

#Zbrojiti matrice A i B, rezultat spremiti u varijablu "rezultat" i svaki element u dobivenoj povecati za 2

#U prvi redak rezultata postaviti sve nule umjesto postojecih elemenata

#U varijablu b spremiti vektor ciji su elementi sume elemenata redaka dobivene matrice

#Ispisati dobiveni vektor b

import numpy as np

A = 8\*np.eye(5)

A[:,4] = 8

print(A)

B = 3\*np.ones((4,4))

B = np.insert(B, 3, 3\*np.ones((1,4)), axis = 0)

B = np.insert(B, 3, 3\*np.ones((1,5)), axis = 1)

print(B)

rezultat = np.add(A,B)

rezultat = rezultat + 2

rezultat[0, :] = 0

print(rezultat)

b = np.sum(rezultat, axis = 1)

print(b)

##Generirati matrice A i B dimenzija kao na slici..

#U varijablu a spremiti vektor ciji su elementi sume elemenata redaka matrice A

#U varijablu b spremiti vektor ciji su elementi sume elemenata redaka matrice B

#Zbrojiti vektore a i b, rezultat spremiti u varijablu c

#Kreirati matricu C dimenzija 5x5 u kojoj su svi retci jednaki vektoru c

#U prvi stupac matrice C postaviti sve nule umjesto postojecih elemenata

#Ispisati dobivenu matricu C

import numpy as np

A = 4\*np.ones(5) + 6\*np.eye(5)

A[:,4] = 8

A[4,4] = 10

print(A)

B = 2\*np.eye(4)

B[:,3]=3

print(B)

a = np.sum(A, axis = 1)

print(a)

b = np.sum(B, axis = 1)

print(b)

b = np.insert(b, 4, 0, axis = 0)

print(b)

c = a + b

print(c)

C = np.ones((5,5))

C[:,:] = c

print(C)

C[:,0]=0

print(C)

##Definirati vektor a u intervalu [0,10] s korakom 0.1

#Definirati vektor b = a\*\*2. Na istom grafu ( u istom prostoru - figure ) nacrtati

#vektor b ( zelenom bojom, crtkana linija)

#sin(b) (crvenom bojom)

#dodati labele za svaki

#Naslov slike treba biti "Ispitni zadatak"

#naslov x-ose "x-osa" (interval[-1,10])

#naslov y-ose "y-osa" (interval[-2,9])

import numpy as np

import matplotlib.pyplot as plt

a = np.arange(0, 10, 0.1)

b = a\*\*2

sinb = np.sin(b)

fig = plt.figure()

fig.suptitle('Ispitni zadatak')

fig.subplots\_adjust(hspace=0.5)

fig.subplots\_adjust(wspace=0.5)

ax1 = fig.add\_subplot(211)

ax1.plot(a,b,'g--', label='b')

ax1.plot(a,sinb,'r', label='sin(b)')

ax1.set\_xlabel('x-osa')

ax1.set\_ylabel('y-osa')

ax1.legend()

limit = plt.gca()

limit.set\_xlim([-1,10])

limit.set\_ylim([-2,9])

plt.show()

##Tvonica proizvodi cetiri razlicita proizvoda, dnevna proizvedena kolicina prvog proizvoda je x1,

#kolicina drugog proizvoda x2 itd. Cilje je odrediti dnevni unos proizvodnje koji maksimizira profit

#za svaki proizvod, imajuci na umu sljedece uvjete:

#dobit po jedinici proizvoda je 20, 12, 40 i 25 $ za prvi, drugi, treci i cetvrti proizvod

#zbog ogranicenja u ljudstvu, ukupan broj proizvedenih jedinica dnevno ne moze biti veci od 50

#za svaku jedinicu prvog proizvoda trose se tri jedinice sirovine A. Svaka jedinica drugog proizvoda

#zahtjeva dvije jedinice sirovine A i jednu jedinicu sirovine B. Svaka jedinica treceg proizvoda treba

#jednu jedinicu A i dvije jedinice B. Konacno, svaka jedinica cetvrtog prozivoda zahtjeva tri jedinice B

#Zbog ogranicenja transporta i skladistenja, tvornica moze potrositi do stotinu jedinica sirovine A

#i devedeset jedinica B dnevno

#Matematicki model rjesenja

#max 20x1 + 12x2 + 40x3 + 25x4 ( profit )

#s.t.: x1 + x2 + x3 + x4 <= 50 ( manpower )

# 3x1 + 2x2 + x3 <= 100 ( material A)

# x2 + 2x3 + 3x4 <= 90 ( material B)

# x1,x2,x3,x4 >= 0

##Koristeci metodu Newton-Raphson optimizirati funkciju:

# f(x) = 2x\*\*4 - x\*\*3 + 4x x = [-3, 3]

import numpy as np

import matplotlib.pyplot as plt

def new\_rap(f, df1, df2, a, tol):

x = np.inf

n = 0

while abs(x-a)>tol:

x = a

a = x - df1(x)/df2(x)

n = n + 1

print("Ekstrem je u točci:", x)

print("Vrijednost funkcije u toj točci je:", f(x))

print("Ekstrem je pronadjen u %d iteracija" %n)

def f(x):

return 2\*x\*\*4-x\*\*3+4\*x

def df1(x):

return 8\*x\*\*3-3\*x\*\*2+4

def df2(x):

return 24\*x\*\*2-6\*x

#crtanje vrijednosti funkcije na intervalu

new\_rap(f, df1, df2, 100, 0.000001)

a = np.linspace(-3, 3, 100)

plt.plot(a, f(a))

#crtanje derivacije

plt.plot(a,df1(a))

osa = np.linspace(0, 0, 100)

plt.plot(a, osa)

##Koristeci metodu sekante optimizirati funkciju

# f(x) = 2x\*\*4 - x\*\*3 + 4x x = [-3, 3]

import matplotlib.pyplot as plt

import numpy as np

def f(x):

return 2\*x\*\*4-x\*\*3+4\*x

def secant(x0, x1, e, N):

print("\n\n\*\*\* SECANT METHOD IMPLEMENTATION\*\*\*")

step = 1

condition = True

while condition:

if f(x0) == f(x1):

print("Divide by zero error!")

break

x2 = x0 - (x1-x0)\*f(x0)/(f(x1) - f(x0))

print('Iteration-%d, x2=%0.6f and f(x2)=%0.6f' %(step, x2, f(x2)))

x0 = x1

x1 = x2

step = step + 1

if step > N:

print('Not convergent!')

break

condition = abs(f(x2)) > e

print('\n Required root is: %0.8f' %x2)

#Input section

x0 = input('Enter first guess:')

x1 = input('Enter second guess:')

e = input('Tolerable error:')

N = input('Maximum step:')

#Converting x0 and e to flot

x0 = float(x0)

x1 = float(x1)

e = float(e)

#converting N to integer

N = int(N)

#Starting secant method

secant(x0,x1,e,N)

a = np.linspace(-3,3,100)

plt.plot(a,f(a))

osa = np.linspace(0, 0, 100)

plt.plot(a, osa)

##