1) dot product

import numpy as np

def create\_matrix(mc):

print("\n ARRAY "+str(mc)+"Elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("\n ARRAY"+str(mc)+"ROW COLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\n Row and column size not match with total elements!!retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

print("\n ARRAY"+str(mc))

print(array\_1)

return array\_1

arr1=create\_matrix(1)

arr2=create\_matrix(2)

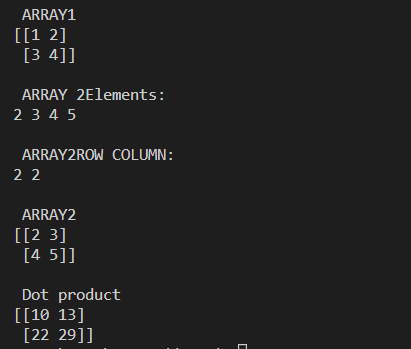
if(arr1.shape == arr2.shape):

print("\n Dot product")

print(np.dot(arr1,arr2))

else:

print("\n dimensions not matching!")



2) Transpose

import numpy as np

def create\_matrix(mc):

print("\n ARRAY "+str(mc)+"elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("\n array "+str(mc)+",ROWCOLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\n row and column size not match with total elements!! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

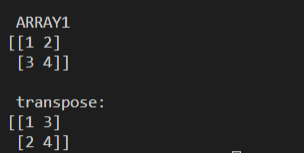
print("\n ARRAY"+str(mc))

print(array\_1)

print("\n transpose:")

return (array\_1)

print(create\_matrix(1).transpose())



3) feelings

l1=['good','fine','happy','nice','positive']

l2=['bad','frustrated','not','sad','negative']

str1=input('enter your response')

flag=0

ncount=0

pcount=0

t=str1.split()

for i in range(len(t)):

for j in range(len(l1)):

if t[i]==l1[j]:

flag=1

pcount+=1

for k in range(len(l2)):

if t[i]==l2[k]:

flag=1

ncount+=1

if flag==0:

print('you are in another mood')

elif ncount%2==0:

print('positive response')

else:

print('negative response')







4)Rank

import numpy as np

def create\_matrix(mc):

print("\n ARRAY "+str(mc)+"elements:")

array\_1=map(int,input().split())

array\_1=np.array(list(array\_1))

print("\n array "+str(mc)+",ROWCOLUMN:")

row,column=map(int,input().split())

if(len(array\_1)!=(row\*column)):

print("\n row and column size not match with total elements!! retry")

return create\_matrix(mc)

array\_1=array\_1.reshape(row,column)

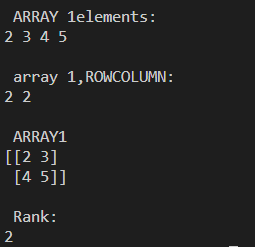
print("\n ARRAY"+str(mc))

print(array\_1)

print("\n Rank:")

return (array\_1)

print(np.linalg.matrix\_rank(create\_matrix(1)))



5)KNN

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import load\_iris

irisdata=load\_iris()

x=irisdata.data

y=irisdata.target

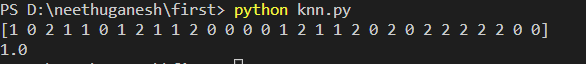
x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

knn=KNeighborsClassifier(n\_neighbors=3)

knn.fit(x\_train,y\_train)

print(knn.predict(x\_test))

print(knn.score(x\_test,y\_test))



6)KNN 2

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

x=([1],[2],[3],[4],[5],[6],[7],[8],[9],[10])

y=[1,4,9,16,25,36,49,64,81,100]

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

knn=KNeighborsClassifier(n\_neighbors=1)

knn.fit(x\_train,y\_train)

print(x\_test)

print(knn.predict(x\_test))

print(knn.score(x\_test,y\_test))

