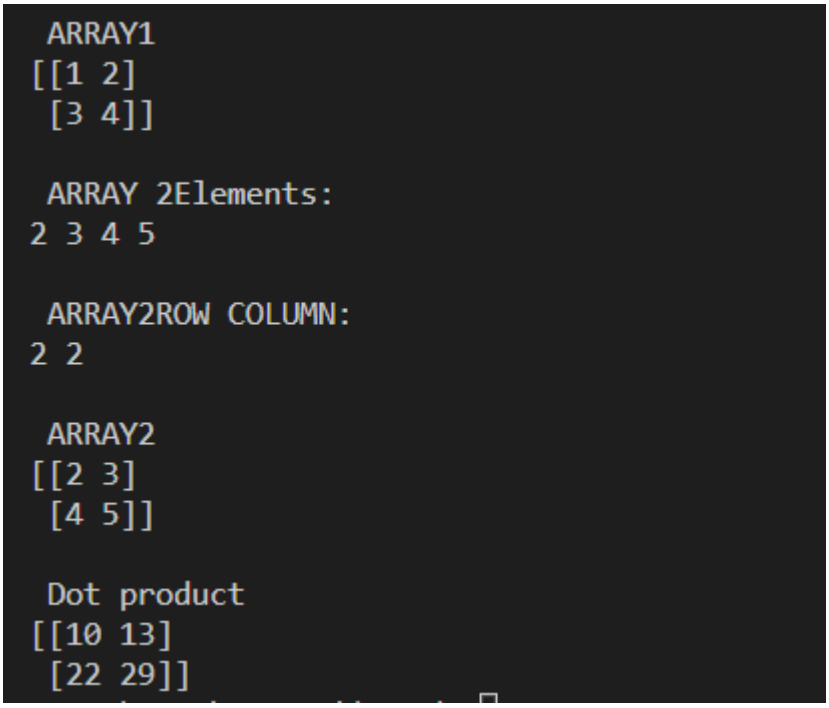
1) dot product

```
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')
```

```
enter your responsebad
negative response
```

```
enter your responsegood
positive response
```

```
enter your responsenot bad
positive 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)))
```

```
 ARRAY 1elements:
2 3 4 5

 array 1,ROWCOLUMN:
2 2

 ARRAY1
[[2 3]
 [4 5]]

 Rank:
2
```

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))
```

```
PS D:\neethuganesh\first> python knn.py
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
1.0
```

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))
```

```
[[9], [2]]
[64  1]
0.0
```