**1. Numpy:**

a. Using NumPy create random vector of size 15 having only Integers in the range 1-20.

1. Reshape the array to 3 by 5
2. Print array shape.
3. Replace the max in each row by 0
4. Create a 2-dimensional array of size 4 x 3 (composed of 4-byte integer elements), also print the shape, type and data type of the array.

#Numpy Creation of 3x5

importnumpyasnp

m=np.random.randint(1,20,15)

print(m)

[ 4 5 19 4 9 15 4 8 3 16 12 3 13 6 19]

#reshape the array

n=m.reshape((3,5))

print(n)

[[ 4 5 19 4 9]

[15 4 8 3 16]

[12 3 13 6 19]]

#print shape of array

print(n.shape)

(3, 5)

#replace max in each row by 0

np.max(n,axis=1)

#print(len(n))

For i in n:

  #print(i)

  o=i.max()

  #print(o)

  for j in range(len(i)):

    if i[j]==0:

      #print(i[j])

      i[j]=0

      #print(i[j])

print(n)

[[ 4 5 19 4 9]

[15 4 8 3 16]

[12 3 13 6 19]]

# creation of two-dimentional array of size 4x3

a=np.array([[1,7,9],[9,5,7],[10,6,5],[14,23,3]],np.int32)

print(type(a),a.shape,a.dtype)

<class 'numpy.ndarray'> (4, 3) int32

b. Write a program to compute the eigenvalues and right eigenvectors of a given square array given below: [[ 3 -2] [ 1 0]]

#calculate eigen values and Right eigen vector

ar=np.array([[3,-2],[1,0]])

print(ar)

e,v=np.linalg.eig(ar)

print("Eigen value:\n", e)

print("Right eigen vector :\n", v)

[[ 3 -2]

[ 1 0]]

Eigen value:

[2. 1.]

Right eigen vector :

[[0.89442719 0.70710678]

[0.4472136 0.70710678]]

c. Compute the sum of the diagonal element of a given array. [[0 1 2] [3 4 5]]

#sum of diagonal elements

arr=np.array([[0,1,2],[3,4,5]])

print(arr)

sum=np.trace(arr)

print(sum)

[[0 1 2]

[3 4 5]]

4

d. Write a NumPy program to create a new shape to an array without changing its data.

Reshape 3x2:

[[1 2]

[3 4]

[5 6]]

Reshape 2x3:

[[1 2 3]

[4 5 6]]

#reshape the given data

arr1=np.arange(1,7)

print(arr1)

arr1=arr1.reshape((3,2))

print(arr1)

arr1=arr1.reshape((2,3))

print(arr1)

[1 2 3 4 5 6]

[[1 2]

[3 4]

[5 6]]

[[1 2 3]

[4 5 6]]

**2. Matplotlib**

1. Write a Python programming to create a below chart of the popularity of programming Languages.

2. Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++

Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

#matplotlib

%matplotlib inline

from matplotlib importpyplotas pt

# available styles

pt.style.available

['Solarize\_Light2',

'\_classic\_test\_patch',

'\_mpl-gallery',

'\_mpl-gallery-nogrid',

'bmh', 'classic',

'dark\_background',

'fast',

'fivethirtyeight',

'ggplot',

'grayscale',

'seaborn-v0\_8',

'seaborn-v0\_8-bright',

'seaborn-v0\_8-colorblind',

'seaborn-v0\_8-dark',

'seaborn-v0\_8-dark-palette',

'seaborn-v0\_8-darkgrid',

'seaborn-v0\_8-deep',

'seaborn-v0\_8-muted',

'seaborn-v0\_8-notebook',

'seaborn-v0\_8-paper',

'seaborn-v0\_8-pastel',

'seaborn-v0\_8-poster',

'seaborn-v0\_8-talk',

'seaborn-v0\_8-ticks',

'seaborn-v0\_8-white',

'seaborn-v0\_8-whitegrid',

'tableau-colorblind10']

pt.style.use("grayscale")

#sample data

#programming languages

prog=['Java', ' Python', 'PHP', 'JavaScript', 'C#','C++']

#popularity

po=[22.2, 17.6, 8.8, 8, 7.7, 6.7]

#color

colour=['orange','green','blue','red','cyan','indigo']

explode=[0.2,0,0,0,0,0]

pt.pie(x=po,labels=prog,colours=colours,explode=explode,wedgeprops={'edgecolor':'black'},shadow=True,autopct='%1.1f%%')

pt.tight\_layout()

pt.title("Popularity of Programming Languages")

pt.show()

A picture containing text, diagram, circle, screenshot

Description automatically generated

**GitHub Link:**

https://github.com/Mouni5/PrograAssignmML.git

**Video Presentation Link:**

**https://drive.google.com/file/d/1-lpUZGZO5h9KR4TmanOO\_X-0xhS8cPAr/view?usp=drive\_link**