Name: KARTHIK.M.

1) Numpy Assignment.

1). Eigenvalues and Eigenvectors of an array.

→ Function: The function returns a tuple consisting of a vector and an array. The vector and contains the eigen values. The array contains the corresponding eigenvectors, one eigen vector eio per column.

→ Syntax: numpy.linalg. eig()

→ Code:
```
# Importing numpy library.
import numpy as np
# Create numpy 2d-array
m = np. array ([[1,2],
                [2,3]])
Print ("Printing the original squar array:\n",
        m)
# finding eigenvalues and eigenvectors
w, v = np.linalg. eig (m).
# printing eigen walues
Print ("printing the Eigen values of the given
        Square array :\n", w)
# printing eigen vectors.
Print ("printing Right eigenvectors of the
        given square array :\n. v).
```

→ application: "Allow us to "reduce" a linear operation to separate, simpler problems..

Essentially the firest Step

2) digitize()

→ function: digitize. Return the indices of the bins to which each value in input array belongs. if values in X are beyond the bounds of bins, 0 or len (bins) is returned as approropriate.

→ Syntax: np. digitize(array, Bin, Right).

→ code:

```
# import numpy
import numpy as np
a = np. array ([1. 2, 2.4, 3.6, 4.8])
bins = np. array ([1.0, 1.3, 2.5, 4.0,
# using np. digitize ( ) method
gfg = np. digitize (a, bins)
print (gfg).
```

→ application: Converting data to a digital format. Digitization can reap efficiency benefits when the digitized data is used to ~~ato~~ automate processes and enable better accessibility.

Name:- KARTHIK. M

3. repeat():

→ function: The repeat() method Constructs and returns a new string which contains the specified number of copies of the string on which it was called, contatenated together.

→ Syntax: numpy. repeat (arr, repetitions, axis = None).

→ Code:
```
# python Program illustrating
# numpy. repeat ()
import numpy as geek.
# working on 1D
arr = geek. arange (5)
Print (" arr :\n", arr)
repetitions = 2
a = geek. repeat (arr, repetitions)
Print ("\n Repeating arr 2 times :\n", a)
Print (" Shape : ", a. Shape)
repetitions = 3
a = geek. repeat (arr , repetitions)
Print ("\n Repeating arr 3 times :\n", a)
#[0 0 0..., 4 4 4] means [0 0 0 1 1 1 2 2 2 3 3 3
                        4 4 4]
# since it was long output, so it uses [...]
Print (" Shape : ", a. Shape).
```

→ Application: A repeat loop is used to iterate over a block of Code multiple number of times.

4). Squeeze():

→ functions: is to remove single - dimensional entries from the shape of an array.

→ Syntax: numpy. squeeze (arr, axis = None).

→ code:
```
# python program explaining.
# numpy. squeeze function.
import numpy as geek.
in_arr = geek. array ([[[2, 2, 2], [2, 2, 2]]])
Point (" Input array : ", in_arr)
Point ("Shape of input array : ", in_arr. Shape)
out_arr = geek. squeeze (in_arr)
Point ("output squeezed array: ", out_arr)
Point ("shape of output array: ", out_arr. Shape)
```

5) linspace():

→ functions: returns evenly spaced numbers over a specified interval [start, stop]. The endpoint of the interval can optionally be excluded.

→ Syntax: numpy. linspace (start,
```
                stop,
                num = 50,
                endpoint = True,
                retstep = False,
                dtype = None).
```

Name: KARTHIK.M

→ Code:
```
# Python Programming illustrating
# numpy. linspace method.
import numpy as geek.
# restep set to True.
Print ("B\n", geek.linspace (2.0, 3.0, num = 5,
                 retstep = True), "\n")
 # To evaluate sin() in long range.
 X = geek. linspace (0, 2, 10)
 Print ("A\n", geek. Sin (x)).
```

6. Clip()

→ function: to clip the values in an array. Given an interval, values outside the interval are clipped to the interval edges.

→ Syntax: numpy.clip (a, a_min, a_max, out = None).

→ code:
```
# Python 3 code demonstrate clip () function
# importing the numpy
 import numpy as np
 in_array = [1, 2, 3, 4, 5, 6, 7, 8]
 Print ("Input array : ", in array).
 out_array = np. clip (in-array, a-min
   = 2, a_max = 6)
 Print ("output array : ", out-array).
```

7. extract():

→ function:- It is an inbuilt function. The extract() function does array to an Variable Conversion. That is it converts array keys into variable names and array values into variable value., imports array to the Symbol table.

→ Syntax: numpy. extract (Condition, arr)

→ Code:-
```
import numpy as np
array = np. arange(10). reshape (5, 2)
Print ("original array : \n", array)
Condition = np. mod (array, 3) ==0
# ~~The~~ print ("\n Array Condition : \n",
            (condition).
 # Print ("\n Elements that statisfies the
     Condition : \n", np. extract (condition,
                                   array)).
```

8. argpartition():

→ function: to create a indirect partitioned copy of input array with its elements rearranged in such a way that the value of the element in k-th position is in the position it would be in a sorted array.

→ Syntax: numpy. argartition.(arr, kth, axis =1, kind =' introselect', order =None).

Name: KARTHIK.M.

→ Code:

```
# Python program explaining
# argpartition() function
import numpy as geek.
# input array
in_arr = geek.array([[2,0,1],[5,4,9]])
Print ("Input array :\n", in_arr)
out_arr = geek.argpartition(in_arr, 1,
                axis=1)
 Print ("Output partitioned array indices :
                  \n", out_arr).
```

9. Setdiff1d():

→ functions: Find the set difference of two arrays and return the unique values in arr1 that are not in arr2.

→ Syntax: numpy.setdiff1d(arr1, arr2, assume_unique = False).

→ Code:

```
# python program explaining
# numpy.setdiff1d() function
# importing numpy as geek.
import numpy as geek.

arr1 = [5, 6, 2, 3, 4]
arr2 = [1, 2, 3]
gfg = geek.setdiff1d(arr1, arr2)
Print(gfg)
```

10) **itemsize** : → function :- returns the size of each element. of a NumPy array.

→ Syntax :- numpy.ndarray.itemsize (arr).

→ Code :-
```
# python program explaining
# numpy.ndarray.itemsize () function
# importing numpy as geek.
import numpy as geek.
arr = geek.array ([1, 2, 3, 4], dtype = geek.
                                        float64).
gfg = arr.itemsize.
Print (gfg)
```

11) **hstack () :-**

→ function :- to stack the sequence of input arrays. horizontally to make a single array.

→ Syntax :- numpy.hstack (tup).

→ Code :-
```
# python program explaining
# hstack () function.
import numpy as geek
# Input array.
in-arr1 = geeks.array ([1,2,3])
Print (" 1st Input array :\n", in-arr1)
in-arr2 = geek.array ([4,5,6])
Print ("2nd Input array :\n", in-arr2)
# stacking the two arrays horizontally.
out-arr = geek.hstack ((in-arr1, inarr2))
Print ("output horizontally stacked array :\n",
    out-arr).
```

Name:- KARTHIK. M.

→ function :

12) Vstack() : Stack arrays in sequence vertically (row wise). This is equivalent to Contatenation along the first axis after 1-D arrays of shape(N,) have been reshaped to (1,N). This function makes most sense for arrays with up to 3 dimensions.

→ Syntax: numpy.Vstack (tup).

→ code:

```
# python program explaining
# Vstack ( ) function.
import numpy as geek
# Input array
in-arr1 = geek.array ([1,2,3])
Print (" 1st Input array :\n", in-arr1)
in-arr2 = geek.array ([4,5,6])
Print ("2nd Input array:\n", in-arr2)
# stacking the two arrays vertically
out-arr = geek.Vstack ((in-arr1, in-arr2))
Print ("output vertically stacked array:\n",
        out-arr)
```

13) hsplit():

→ function: to split an array into multiple sub-arrays horizontally (column-wise). hsplit is equivalent to split with axis = 1, the array is always splint along the second axis regardless of the array dimension.

→ Syntax: numpy.hsplit (arr, indices_or_sections).

→ Code :-

```
# python program explaining
# numpy. hsplit () function
# importing numpy as geek
import numpy as geek.
arr = geek. arange (16.0). reshape (4,4)
gfg = geek. hsplit (arr, 2)
Print (gfg)
```

14) Vsplit :

→ function : split an array into multiple sub-arrays vertically (row-wise). is equivalent to split with axis=0. the array is always split along the first axi's regardless of the array dimension.

→ Syntax :- numpy. Vsplit (arr, indices_or_sections)

→ Code :-

```
# python program explaining
# numpy. vsplit () function
# importing numpy as geek.
import numpy as geek
arr = geek. arange (9,0). reshape (3,3)
gfg = geek. vsplit (arr, 1)
Print (gfg).
```

Name:- KARTHIK. M

15) View Vs Shallow Copy:

→ function:- Contain A view of a numpy array is a shallow copy in sense A, i.e. it references the same data buffer as the original, so changes to the original data affect the view data and vice warsa.

→ Syntax:-

16) Deep Copy:-

→ function:- The ndarray. copy () function creates a deepcopy. It is a complete copy of the array and its data, and doesn't share with the original array.

→ Syntax:- numpy.copy (a, order ='K', subok = False).

→ Code:-
```
# importing copy module
import copy
# initializing list 1
li1 = [1, 2, [3, 5], 4]
# using copy for shallow copy
li2 = copy. copy (1i1)
# using copy for ~~Shallo~~ deepcopy
li3 = copy. deepcopy (1i1).
```

6) ... meshgrid (*xi, copy = True, sparse = False,

17) Copy():

→ function: returns a copy of the array.

→ Syntax:- numpy.ndarray.copy(order = 'C')

→ Code:-

```
# python program explaining
# numpy.ndarray.copy() function
Import numpy as geek.
X = geek.array([[0,1,2,3],[4,5,6,7]],
                                 order = 'F')
Print("X is:\n", X)
# Copying X to Y
Y = X.copy()
Print ("Y is :\n", Y)
Print ("\nX is Copied to Y")
```

18) meshgrid():

→ functions: is to Create a rectangular grid out of two given one-dimensional arrays. representing the Cartesian indexing (or) Matrix indexing. returns two 2-Dimensional arrays representing the X and Y coordinates of all the points.

→ Syntax:- numpy.meshgrid(*Xi, copy = True, sparse = False, indexing = 'xy').

→ Code:-

```
# Sample code for generation of firs.
import numpy as np
# from matplotlib import pyplot as plt
# pyplot imported for plotting graphs
X = np. linspace (-4 ,4, 9)
# numpy. linspace Creates an array of
# 9 linearly placed elements between.
# -4 and 4, both inclusive
Y = np. linspace ( -5 , 5, 11)

# The meshgrid function returns
# two 2-dimensional arrays.
X-1 , Y-1 = np. meshgrid (x , y)
Print (" X-1 = ")
Print (X-1)
Print ("Y-1 = ")
Print (Y-1).
```

19) Swapaxes():

→ function:- This function interchanges the two axes of an array. a view of the swapped array is returned.

→ Syntax: numpy. Swapaxes (a, axis1 , axis2)

→ Code:

```
# import the important module in python
import numpy as np
# make matrix with numpy
gfg = np. matrix ('[4, 1; 12, 3]')
```

```
# applying matrix. swapaxes ( ) method.
geek = gfg . swapaxes (0, 1)
Print (geek).
```

20) Column - stack ( )s

→ function: to stack 1 - D arrays as columns into a 2 - D array. It takes a sequence of 1 - D arrays and stack them as columns to make a single 2 - D array. 2 - Darrays are stacked as - is, just like with hstack function.

→ Syntax :- numpy. column - stack (tup)

→ Code :

```
# python program explaining
# column - stack () function
import numpy as geek.
# input array
in - arr 1 = geek . array (( 1, 2, 3 ))
Print ( " 1st Input array :\n", in - arr1)
in - arr2 = geek. array (( 4, 5, 6 ))
Print (" 2nd Input array :\n" , in - arr2)
# stacking the two arrays.
out_arr = geek. column - stack (( in - arr 1,
                                   in _ arr2 ))
Print ( "Output stacked array :\n ", out - arr).
```

Application :- used to stack 1 - D arrays as columns into a 2 - D array.