

Lab Assignment 16

Student Name : Aryan Verma

Student Id: AF0417100

Topic: Numpy

KEY POINTS OF NUMPY:

- ****NUMPY** IS A PYTHON LIBRARY FOR NUMERICAL COMPUTING, OFFERING SUPPORT FOR LARGE, MULTI-DIMENSIONAL ARRAYS AND MATRICES.**
- **IT INCLUDES A WIDE RANGE OF MATHEMATICAL FUNCTIONS TO EFFICIENTLY PERFORM OPERATIONS ON ARRAYS.**
- **ARRAYS IN NUMPY ARE STORED IN CONTIGUOUS MEMORY, MAKING OPERATIONS FASTER THAN PYTHON LISTS.**
- **SUPPORTS **BROADCASTING**, ALLOWING OPERATIONS ON ARRAYS OF DIFFERENT SHAPES WITHOUT EXPLICIT LOOPING.**
- **PROVIDES **VECTORIZED OPERATIONS**, ELIMINATING THE NEED FOR EXPLICIT LOOPS IN ELEMENT-WISE CALCULATIONS.**

- IT OFFERS TOOLS FOR ****LINEAR ALGEBRA****, ****RANDOM NUMBER GENERATION****, AND MATRIX MANIPULATION.
- INTEGRATES SEAMLESSLY WITH LIBRARIES LIKE ****PANDAS****, ****MATPLOTLIB****, AND ****SCIKIT-LEARN**** FOR DATA ANALYSIS AND SCIENTIFIC COMPUTING.
- NUMPY ALLOWS ****SHAPE MANIPULATION**** (RESHAPE, FLATTEN, TRANSPOSE) FOR FLEXIBLE ARRAY HANDLING.
- ESSENTIAL FOR NUMERICAL TASKS IN DATA SCIENCE, MACHINE LEARNING, AND ENGINEERING APPLICATIONS.

SOME FUNCTIONS USED IN NUMPY:

1. ****`NUMPY.ARRAY()`****: CREATES AN ARRAY FROM LISTS OR TUPLES FOR NUMERICAL OPERATIONS.
2. ****`NUMPY.ZEROS()`****: GENERATES AN ARRAY FILLED WITH ZEROS, USEFUL FOR INITIALIZING DATA STRUCTURES.

3. ``NUMPY.ONES()`**`: PRODUCES AN ARRAY FILLED WITH ONES, OFTEN USED IN MATHEMATICAL COMPUTATIONS.**

4. ``NUMPY.ARANGE()`**`: GENERATES AN ARRAY WITH A SPECIFIED RANGE OF VALUES, AIDING IN ITERATION.**

5. ``NUMPY.Linspace()`**`: CREATES EVENLY SPACED VALUES OVER A SPECIFIED RANGE, USEFUL FOR PLOTTING FUNCTIONS.**

6. ``NUMPY.RESHAPE()`**`: CHANGES THE SHAPE OF AN ARRAY WITHOUT ALTERING ITS DATA, AIDING DATA MANIPULATION.**

7. ``NUMPY.TRANSPOSE()`**`: SWITCHES THE AXES OF AN ARRAY, COMMONLY USED IN LINEAR ALGEBRA.**

8. ``NUMPY.FLATTEN()`**`: CONVERTS MULTI-DIMENSIONAL ARRAYS INTO 1D, SIMPLIFYING DATA HANDLING.**

9. ``NUMPY.CONCATENATE()`**`: JOINS MULTIPLE ARRAYS ALONG A SPECIFIED AXIS FOR COMBINED ANALYSIS.**

10. ``NUMPY.ADD()`**`: PERFORMS ELEMENT-WISE ADDITION OF ARRAYS, ESSENTIAL FOR MATHEMATICAL OPERATIONS.**

11. ``NUMPY.SUBTRACT()`**`: EXECUTES ELEMENT-WISE SUBTRACTION, AIDING IN DATA TRANSFORMATIONS.**

12. ``NUMPY.MULTIPLY()`**`: MULTIPLIES TWO ARRAYS ELEMENT-WISE, CRUCIAL FOR SCALING DATA.**

13. ``NUMPY.DIVIDE()`**`: DIVIDES ARRAYS ELEMENT-WISE, USEFUL FOR NORMALIZATION TASKS.**

14. `numpy.mean()`: CALCULATES THE AVERAGE VALUE OF AN ARRAY, IMPORTANT IN STATISTICAL ANALYSIS.

15. `numpy.sum()`: COMPUTES THE SUM OF ARRAY ELEMENTS, OFTEN USED IN AGGREGATING DATA.

16. `numpy.min()`: FINDS THE MINIMUM VALUE IN AN ARRAY, USEFUL IN DATA COMPARISON.

17. `numpy.max()`: DETERMINES THE MAXIMUM VALUE IN AN ARRAY FOR RANGE CALCULATIONS.

18. `numpy.std()`: COMPUTES THE STANDARD DEVIATION, INDICATING DATA VARIABILITY.

19. `numpy.var()`: CALCULATES THE VARIANCE OF AN ARRAY, ASSESSING DATA DISPERSION.

20. **`**`NUMPY.WHERE()``**: RETURNS INDICES OF ELEMENTS SATISFYING A CONDITION, AIDING DATA FILTERING.

1. Convert the below list into numpy array then display the array.

My_list = [1, 2, 3, 4, 5]

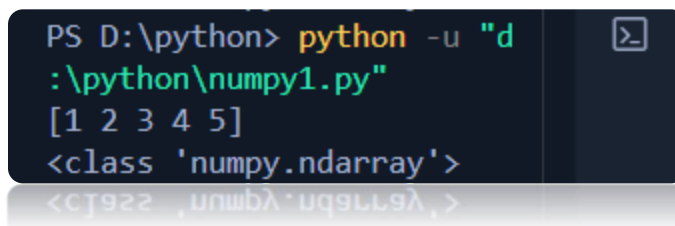
Ans:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'PYTHON' containing several files: local.py, localglobal.py, logicalprgrams.py, marks.py, matchday.py, numpy1.py, and numpy5.py. The code editor shows the following code in a file named 'numpy1.py':

```
1 import numpy as np
2 list = [1,2,3,4,5]
3 np_list = np.array(list)
4 print(np_list)
5 print(type(np_list))
```

Output:



The screenshot shows a terminal window with the following output:

```
PS D:\python> python -u "d
:\python\numpy1.py"
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

2. Convert the below list into a numpy array then display the array then display the first and last index and then multiply each element by 2 and display the result.

Input: my_list = [1, 2, 3, 4, 5]

Ans:

```
8
9 list1 = [1,2,3,4,5]
10 numpy_list = np.array(list1)
11 print(list1)
12 print(list1[0])
13 print(list1[4])
14
15 multiply_arr = numpy_list*2
16 print(multiply_arr)
```

output:

```
[1, 2, 3, 4, 5]
1
5
[ 2  4  6  8 10]
```

Sample program with output :

The image shows a VS Code editor window with a Python file named `numpy1.py`. The script performs various NumPy operations and prints the results. The Explorer sidebar on the left lists several Python files, including `list.py`, `list1.py`, `listevenodd.py`, `listfruits.py`, `listpractice.py`, `local.py`, `localglobal.py`, `logicalprgrams.py`, `marks.py`, `matchday.py`, `numpy1.py`, `numpy2.py`, `numpy3.py`, `numpy4.py`, `numpy5.py`, `oddeven.py`, `palindromenum.py`, `patternpractice.py`, `posneg.py`, `practiceforloop.py`, and `prime.nv`. The main editor displays the following code:

```
40 import numpy as np # Import numpy for numerical operations
41 # Create a 1D array with a single element
42 x = np.array(5)
43 print(x.ndim) # Print the number of dimensions of 'x' (should be 1)
44 # Create a 1D array with elements [1, 2, 3]
45 y = np.array([1, 2, 3])
46 print(y.ndim) # Print the number of dimensions of 'y' (should be 1)
47 # Create a 2D array with 2 rows and 3 columns
48 z = np.array([[1, 2, 3], [4, 5, 6]])
49 print(z.ndim) # Print the number of dimensions of 'z' (should be 2)
50 m = np.array([[1, 2], [3, 4], [5, 6]])
51 print(m.ndim) # Print the number of dimensions of 'm' (should be 2)
52 print(m) # Print the array 'm'
53 print(m.dtype) # Print the data type of elements in 'm'
54 a = np.arange(10)
55 print(a) # Print the array 'a'
56 # Create a 2D array with 2 rows and 3 columns, filled with zeros
57 b = np.zeros((2, 3))
58 print(b) # Print the array 'b'
59 c = np.ones((3, 2))
60 print(c) # Print the array 'c'
61 d = np.eye(3)
62 print(d) # Print the identity matrix 'd'
63 # Create a 2x2 array of random values between 0 and 1
64 e = np.random.rand(2, 2)
65 print(e) # Print the array 'e'
```

The output of the script is displayed on the right side of the editor:

```
[[0. 0. 0.]
 [0. 0. 0.]]
3
int64
[[[1 2 3]
 [2 4 5]
 [6 8 9]]]
[[0.00802556 0.9069945 ]
 [0.60961288 0.44259707]]
0
1
2
2
[[1 2]
 [3 4]
 [5 6]]
int64
[0 1 2 3 4 5 6 7 8 9]
[[0. 0. 0.]
 [0. 0. 0.]]
[[1. 1.]
 [1. 1.]
 [1. 1.]]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[0.59741337 0.45891039]
 [0.78126367 0.95263777]]
PS D:\python>
```

This block is a blurred version of the VS Code editor window shown above, displaying the same Python script and its output.