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.

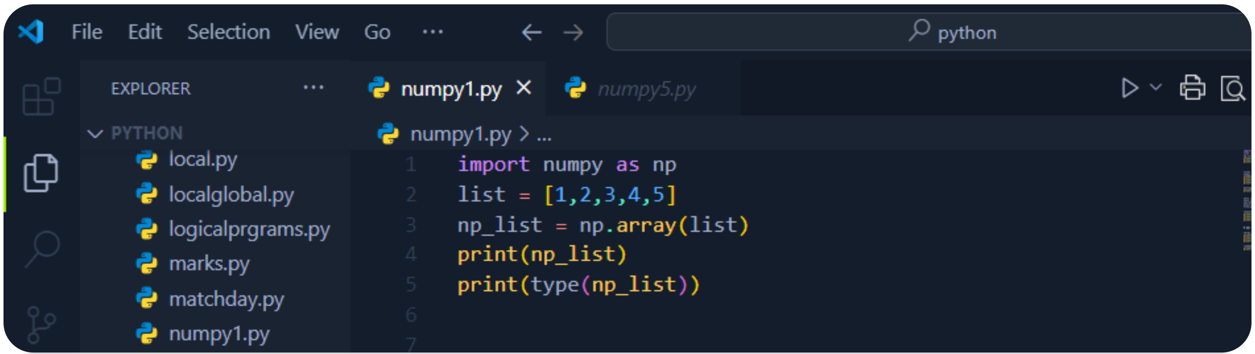
19. **`NUMPY.VAR()`**: CALCULATES THE VARIANCE OF AN ARRAY, ASSESSING DATA DISPERSION.

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.

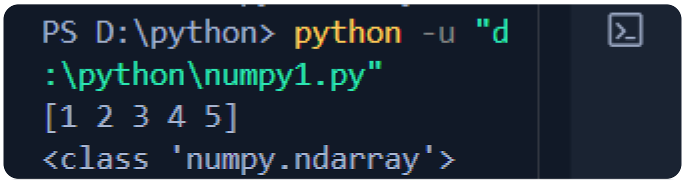
My_list = [1, 2, 3, 4, 5]

Ans:



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

Output:

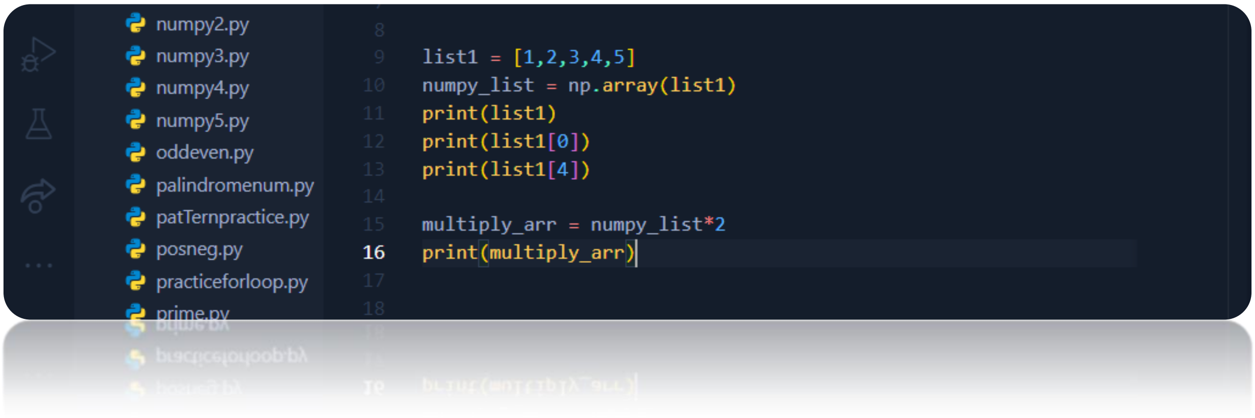


```
PS D:\python> python -u "d
:\python\numpy1.py"
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

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.

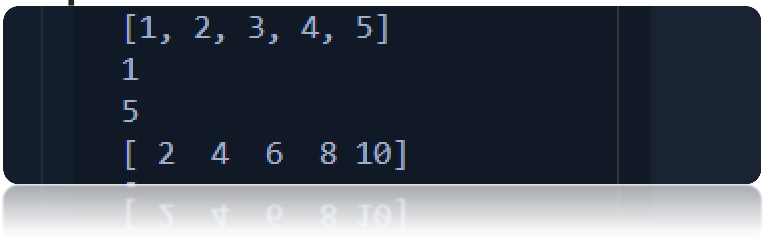
Input: my_list = [1, 2, 3, 4, 5]

Ans:



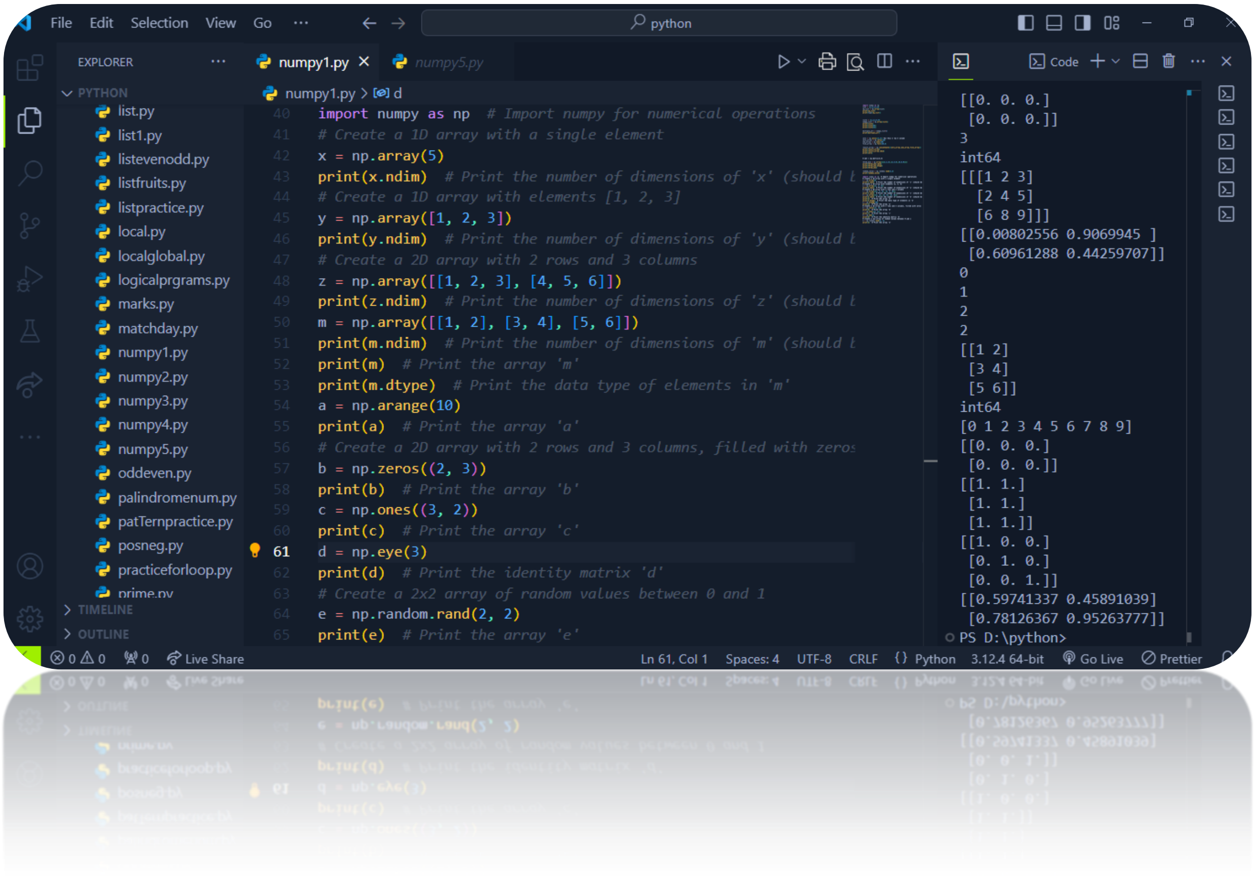
```
list1 = [1,2,3,4,5]
numpy_list = np.array(list1)
print(list1)
print(list1[0])
print(list1[4])

multiply_arr = numpy_list*2
print(multiply_arr)
```

output:



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

Sample program with output :



```python
import numpy as np  # Import numpy for numerical operations
# Create a 1D array with a single element
x = np.array(5)
print(x.ndim)  # Print the number of dimensions of 'x' (should b
# Create a 1D array with elements [1, 2, 3]
y = np.array([1, 2, 3])
print(y.ndim)  # Print the number of dimensions of 'y' (should b
# Create a 2D array with 2 rows and 3 columns
z = np.array([[1, 2, 3], [4, 5, 6]])
print(z.ndim)  # Print the number of dimensions of 'z' (should b
m = np.array([[1, 2], [3, 4], [5, 6]])
print(m.ndim)  # Print the number of dimensions of 'm' (should b
print(m)  # Print the array 'm'
print(m.dtype)  # Print the data type of elements in 'm'
a = np.arange(10)
print(a)  # Print the array 'a'
# Create a 2D array with 2 rows and 3 columns, filled with zeros
b = np.zeros((2, 3))
print(b)  # Print the array 'b'
c = np.ones((3, 2))
print(c)  # Print the array 'c'
d = np.eye(3)
print(d)  # Print the identity matrix 'd'
# Create a 2x2 array of random values between 0 and 1
e = np.random.rand(2, 2)
print(e)  # Print the array 'e'
```

```
[[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>
```