Lab Assignment 17

Student Name : Aryan Verma

Student Id: AF0417100

Topic: Numpy Slicing

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.

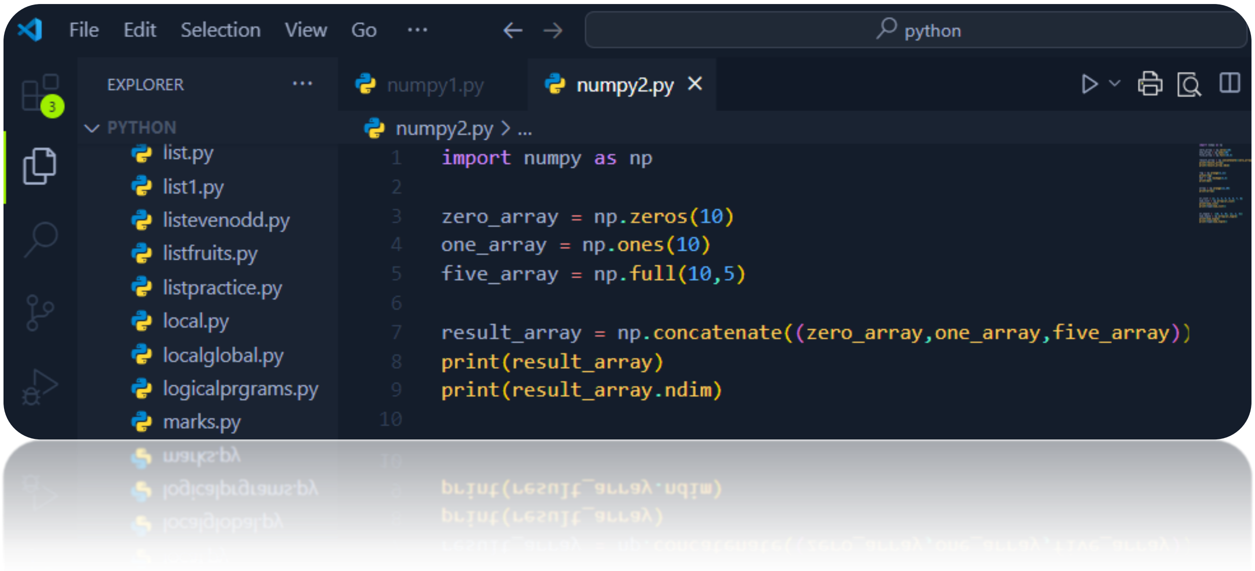
18. **`NUMPY.STD()`**: COMPUTES THE STANDARD DEVIATION, INDICATING DATA VARIABILITY.

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. Write a Numpy program to create an array of 10 zeros, 10 ones, and 10 fives

Ans:

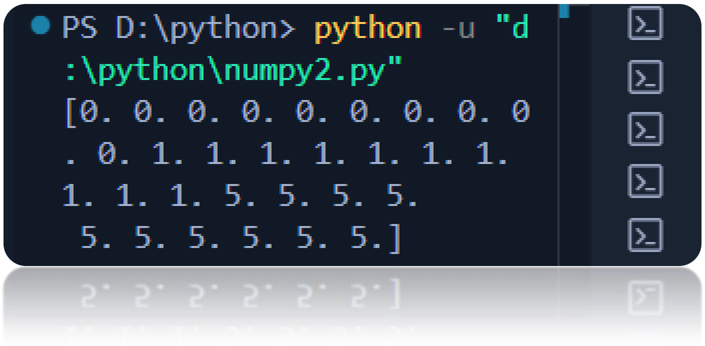


```python
import numpy as np

zero_array = np.zeros(10)
one_array = np.ones(10)
five_array = np.full(10,5)

result_array = np.concatenate((zero_array,one_array,five_array))
print(result_array)
print(result_array.ndim)
```

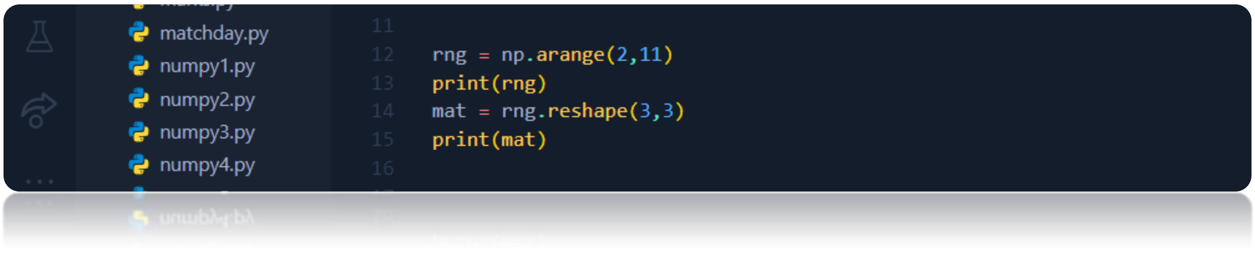
Output:



```
PS D:\python> python -u "d
:\python\numpy2.py"
[0. 0. 0. 0. 0. 0. 0. 0. 0
. 0. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 5. 5. 5. 5.
 5. 5. 5. 5. 5. 5.]
```

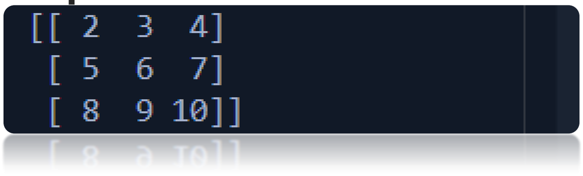
2. Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.

Ans:



```
rng = np.arange(2,11)
print(rng)
mat = rng.reshape(3,3)
print(mat)
```

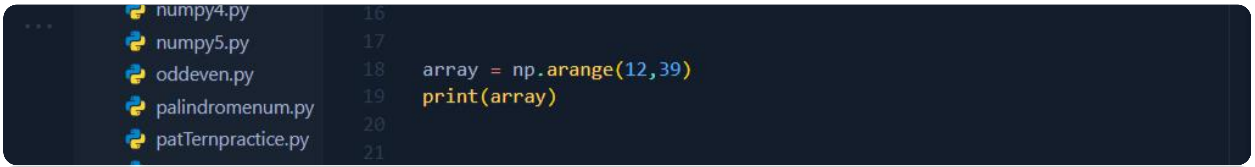
output:



```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

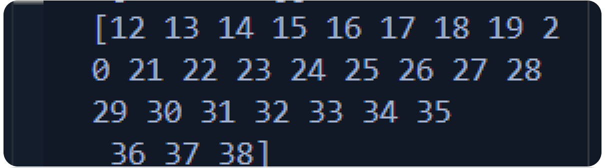
3. Write a NumPy program to create an array with values ranging from 12 to 38.

Ans:



```
array = np.arange(12,39)
print(array)
```

output:

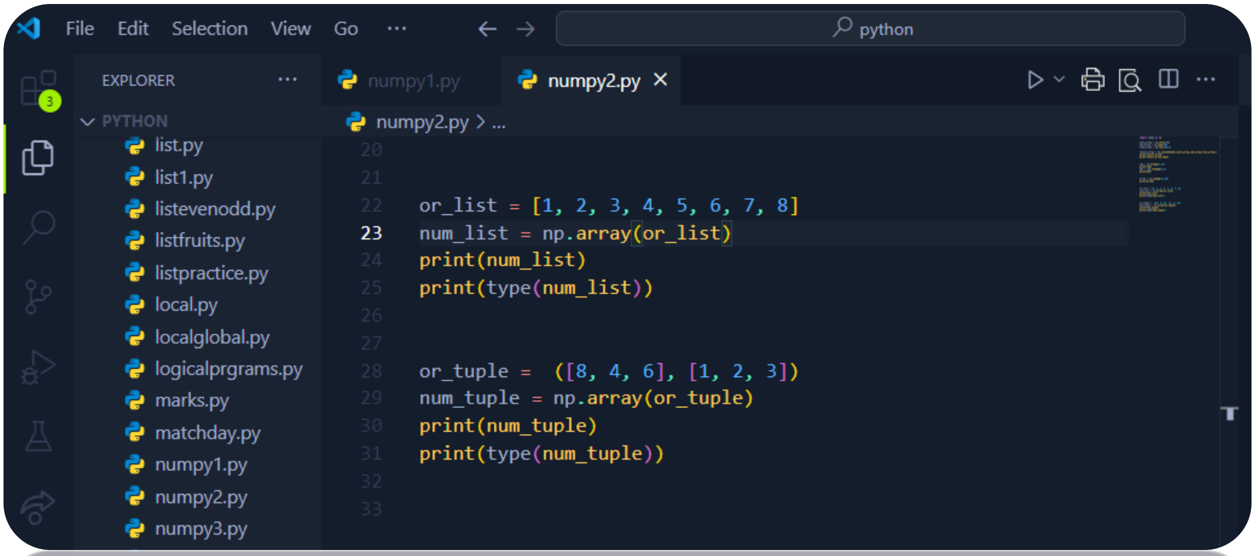


```
[12 13 14 15 16 17 18 19 2
0 21 22 23 24 25 26 27 28
29 30 31 32 33 34 35
 36 37 38]
```

4. Write a NumPy program to convert a list and tuple into arrays. Input: my_list = [1, 2, 3, 4, 5, 6, 7, 8]

Input: my_tuple = ([8, 4, 6], [1, 2, 3])

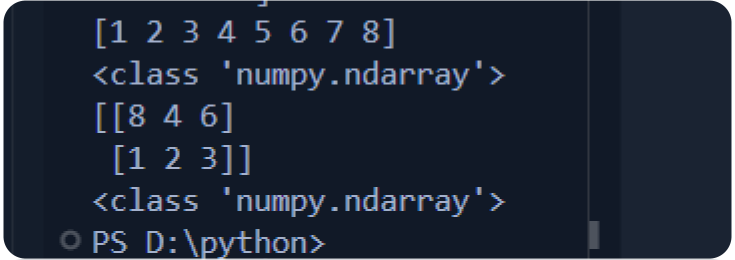
Ans:



```
or_list = [1, 2, 3, 4, 5, 6, 7, 8]
num_list = np.array(or_list)
print(num_list)
print(type(num_list))


or_tuple =  ([8, 4, 6], [1, 2, 3])
num_tuple = np.array(or_tuple)
print(num_tuple)
print(type(num_tuple))
```

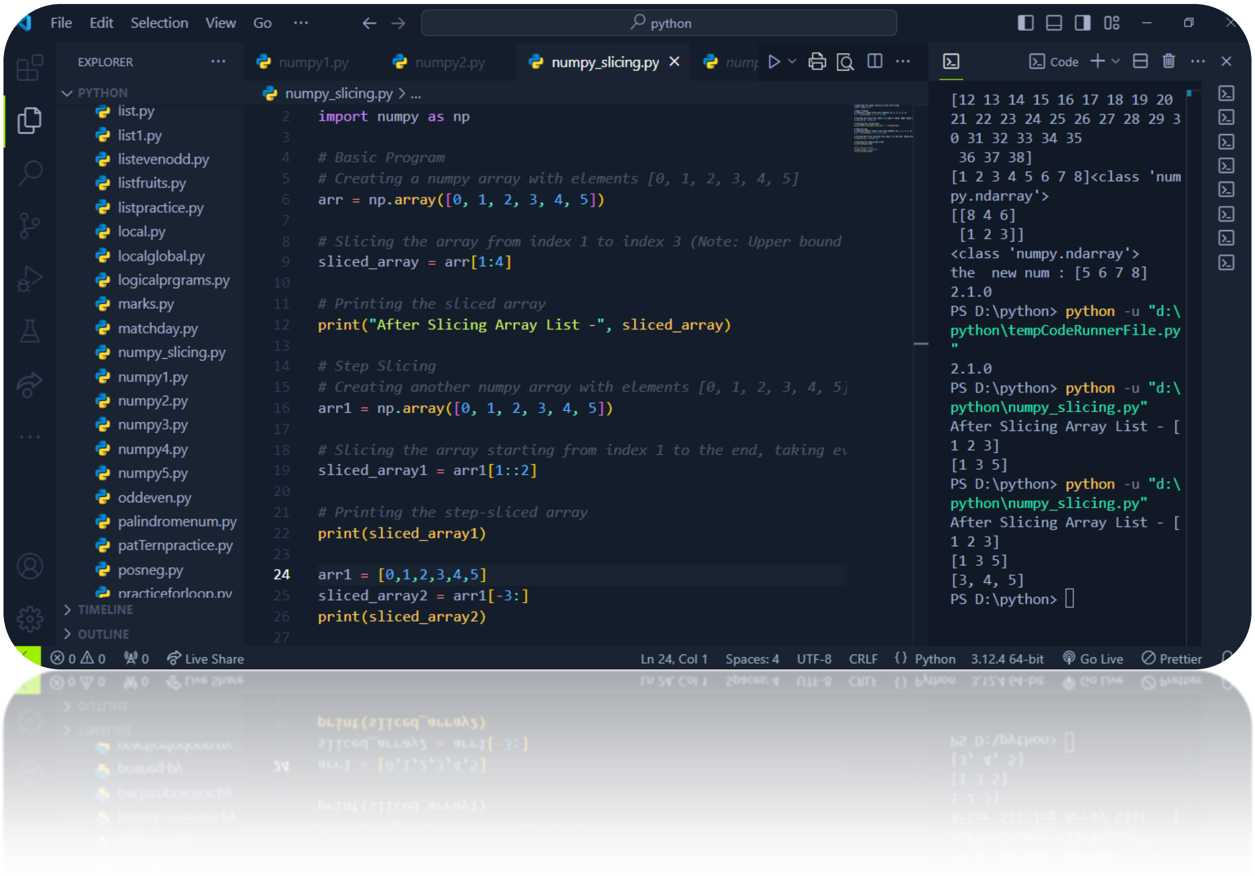
output:



```
[1 2 3 4 5 6 7 8]
<class 'numpy.ndarray'>
[[8 4 6]
 [1 2 3]]
<class 'numpy.ndarray'>
PS D:\python>
```

Sample programs with output:

1. **Basic program with slicing**
2. **Step slicing**



```python
import numpy as np

# Basic Program
# Creating a numpy array with elements [0, 1, 2, 3, 4, 5]
arr = np.array([0, 1, 2, 3, 4, 5])

# Slicing the array from index 1 to index 3 (Note: Upper bound
sliced_array = arr[1:4]

# Printing the sliced array
print("After Slicing Array List -", sliced_array)

# Step Slicing
# Creating another numpy array with elements [0, 1, 2, 3, 4, 5]
arr1 = np.array([0, 1, 2, 3, 4, 5])

# Slicing the array starting from index 1 to the end, taking ev
sliced_array1 = arr1[1::2]

# Printing the step-sliced array
print(sliced_array1)

arr1 = [0,1,2,3,4,5]
sliced_array2 = arr1[-3:]
print(sliced_array2)
```

```
[12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 3
0 31 32 33 34 35
 36 37 38]
[1 2 3 4 5 6 7 8]<class 'num
py.ndarray'>
[[8 4 6]
 [1 2 3]]
<class 'numpy.ndarray'>
the  new num : [5 6 7 8]
2.1.0
PS D:\python> python -u "d:\
python\tempCodeRunnerFile.py
"
2.1.0
PS D:\python> python -u "d:\
python\numpy_slicing.py"
After Slicing Array List - [
1 2 3]
[1 3 5]
PS D:\python> python -u "d:\
python\numpy_slicing.py"
After Slicing Array List - [
1 2 3]
[1 3 5]
[3, 4, 5]
PS D:\python> 
```