1. Numpy – Introduction

Table of Contents

1. Numpy – Introduction

1. Numpy

- ✓ NumPy is the fundamental package for scientific computing in Python.
- ✓ It is a Python library that provides multidimensional array object.
- ✓ The full form of **numpy** is 'Numerical Python'.
- ✓ Numpy was created by *Travis Oliphant*

2. What is an array?

- ✓ Array is an object which stores a group of values.
- ✓ Also called as, ordered collection of values.
- ✓ Array can store same type of values.

3. Why to use numpy array?

- ✓ Python lists are a bit slow in process.
- ✓ Numpy arrays are faster than python list.
- ✓ The array object in numpy is called as ndarray.

4. Open source

- ✓ Numpy is an open source means it’s free.

5. numpy – Installation

- ✓ By default numpy will not be available with python installation.
- ✓ Explicitly we need to install numpy package.
- ✓ Run below command in command prompt.

```
pip install numpy
```

6. pip command in python

- ✓ pip stands for **p**ython **i**nstaller **p**ackage
- ✓ Pip is a package management system.
- ✓ It is used to install and manage software packages.
    - pip install package_name
    - pip install numpy

7. Check installed python library.

- ✓ We can check installed python library by using below command.

```
pip show numpy
```

8. import numpy package

- ✓ We can import numpy package by using import.

Program Name: importing numpy package
demo1.py

```
import numpy
print('numpy imported successfully')
```

Output

```
numpy imported successfully
```

9. ModuleNotFoundError

- ✓ If numpy is not installed then we will get below error.

Program Name: checking numpy installed or not
demo2.py

```
import numpy
print('numpy imported successfully')
```

Output

```
Traceback (most recent call last):
File "demo2.py", line 1, in <module>
import numpy
ModuleNotFoundError: No module named 'numpy'
```

10. Famous Alias name to numpy

- ✓ We can give alias name to numpy.
- ✓ Note this name can be any name but the famous alias name is np

Program Name: alias name to numpy
demo3.py

```
import numpy as np
print('numpy imported successfully')
print('Alias name given to numpy as np')
```

Output

```
numpy imported successfully
Alias name given to numpy as np
```

2. Numpy – Fundamentals

Contents

2. Numpy – Fundamentals

1. Creating numpy array

- ✓ We can create numpy array by using array(p) function.
- ✓ Internally it creates object to ndarray.
- ✓ We can pass list, tuple etc as a parameter to the array(p) function.
- ✓ Having same type of values is recommended.

2. numpy.ndim

- ✓ Ndim is predefined variable in numpy
- ✓ By using this we can check the array dimensions.

Program Name: Creating numpy array with single value
demo1.py

```
import numpy as np

age = 44
value = np.array(age)

print(value)
print(type(value))
print(value.ndim)
```

Output

```
44
<class 'numpy.ndarray'>
0
```

Program Name: Creating numpy array with group of values
demo2.py

```
import numpy as np

details = [10, 20, 30, 40, 50]
sales = np.array(details)

print(sales)
print(type(sales))
print(sales.ndim)
```

Output

```
[10 20 30 40 50]
<class 'numpy.ndarray'>
1
```

Program Name	Creating numpy array with group of values
demo3.py

```
import numpy as np

details = [[10, 20], [30, 40]]
sales = np.array(details)

print(sales)
print(type(sales))
print(sales.ndim)
```

Output

```
 [[10 20]
 [30 40]]
<class 'numpy.ndarray'>
2
```

Program Name

Creating numpy array with group of values
demo4.py

```
import numpy as np

details = [[10, 20], [30, 40], [50, 60]]
sales = np.array(details)
print(sales)
print(type(sales))
print(sales.ndim)
```

Output

```
 [[10 20]
 [30 40]
 [50 60]]
<class 'numpy.ndarray'>
2
```

3. Indexing and Slicing

- ✓ We can access numpy array values by using indexing and slicing
- ✓ Numpy array having indexing nature.
- ✓ Numpy array index start with 0.
    - First element stores in 0th index
    - Second element stores in 1st index etc
- ✓ By using slicing we can access piece of array from the main array.

Program Name: Accessing numpy array by using indexing
demo5.py

```
import numpy as np

details = [10, 20, 30, 40, 50]
sales = np.array(details)
print(sales)
print(sales[0])
print(sales[1])
print(sales[2])
```

Output

```
10
20
30
```

Program Name

Accessing numpy array by using indexing
demo6.py

```
import numpy as np

details = [10, 20, 30, 40, 50]
sales = np.array(details)

print(sales)
print(sales[2:])
```

Output

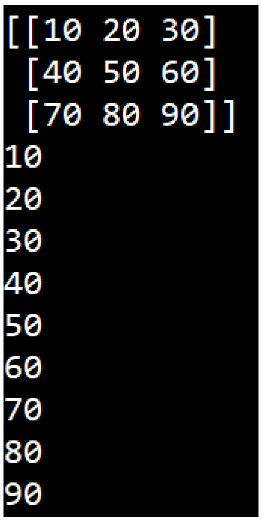
```
[30, 40, 50]
```

Program Name

Creating matrix and selecting elements
demo7.py

```
import numpy as np

matrix = np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]])

print(matrix)

print(matrix[0,0])
print(matrix[0,1])
print(matrix[0,2])

print(matrix[1,0])
print(matrix[1,1])
print(matrix[1,2])

print(matrix[2,0])
print(matrix[2,1])
print(matrix[2,2])
```

Output



```
[[10 20 30]
 [40 50 60]
 [70 80 90]]
10
20
30
40
50
60
70
80
90
```

IndexError

- ✓ If we try to access value with out of bounds of index then we will get IndexError.

Program Name	Accessing numpy array value demo8.py

```
import numpy as np

details = [10, 20, 30, 40, 50]
sales = np.array(details)

print(sales)
print(sales[22])
```

Output

```
IndexError: index 22 is out of bounds for axis 0 with size 5
```

4. Creating a array with all zeros

- ✓ We can create array with all zeros by using numpy.zeros() function

Program	Creating numpy array with group of values
Name	demo9.py

```
import numpy as np

sales = np.zeros(5)

print(sales)
print(type(sales))
```

Output

```
 [0. 0. 0. 0. 0.]
<class 'numpy.ndarray'>
```

5. Creating a array with all ones

- ✓ We can create array with all ones by using numpy.ones() function

Program Name: Creating numpy array with group of values
demo10.py

```
import numpy as np

sales = np.ones(5)

print(sales)
print(type(sales))
```

Output

```
[1. 1. 1. 1. 1.]
<class 'numpy.ndarray'>
```

3. NUMPY – ATTRIBUTES

Contents

3. NUMPY – ATTRIBUTES

1. Numpy Array Attributes

- ✓ Numpy array having predefined attributes to helps to understand the essentials functionality.

2. shape attribute

- ✓ shape is a predefined attribute in numpy array.
- ✓ We should access this shape attribute by using numpy array object
- ✓ By using this we can check number of rows and columns in an array.
- ✓ Shape attribute returns the tuple as number of rows and columns.

Program Name: Creating numpy array with group of values
demo2.py

```
import numpy as np

details = [10, 20, 30], [40, 50, 60]
sales = np.array(details)
print(sales)
print(sales.shape)
```

Output

```
[[10 20 30]
[40 50 60]]
(2, 3)
```

3. ndim attribute

- ✓ ndim is a predefined attribute in numpy array.
- ✓ We should access this ndim attribute by using numpy array object
- ✓ By using this we can check the dimensions of an array

Program	Creating numpy array, check with ndim attribute
Name	demo2.py

```
import numpy as np

details = [10, 20, 30, 40, 50]
sales = np.array(details)
print(sales)
print(sales.ndim)
```

Output

```
[10 20 30 40 50]
1
```

Program Name	Creating numpy array, check with ndim attribute
demo3.py

```
import numpy as np

details = [[10, 20], [30, 40]]
sales = np.array(details)
print(sales)
print(sales.ndim)
```

Output

```
 [[10 20]
 [30 40]]
2
```

Program Name	Creating numpy array with group of values
demo3.py

```
import numpy as np

details = [[10, 20], [30, 40], [50, 60]]
sales = np.array(details)
print(sales)
print(type(sales))
print(sales.ndim)
```

Output

```
 [[10 20]
 [30 40]
 [50 60]]
<class 'numpy.ndarray'>
2
```

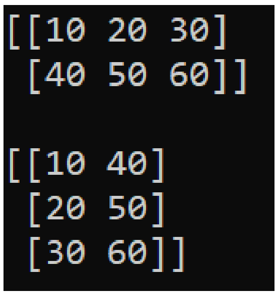
4. arrayobject.T

- ✓ T is a predefined attribute in numpy array.
- ✓ We should access this T attribute by using numpy array object
- ✓ By using this we can transpose the array means it convers rows as columns and columns as rows.

Program Name: T attribute
demo2.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(sales.T)
```

Output



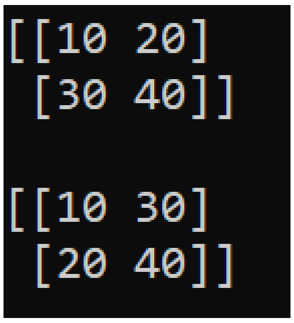
```
[[10 20 30]
 [40 50 60]]

[[10 40]
 [20 50]
 [30 60]]
```

Program Name

T attribute
demo3.py

```
import numpy as np

details = [[10, 20], [30, 40]]
sales = np.array(details)
print(sales)
print()
print(sales.T)
```

Output



```
[[10 20]
 [30 40]]

[[10 30]
 [20 40]]
```

4. NUMPY – IMPORTANT METHODS

Contents

4. NUMPY – IMPORTANT METHODS

1. Numpy Array Methods

- ✓ Numpy array having predefined methods to perform different operations over array.

2. min() method

- ✓ min() is a predefined method in numpy array.
- ✓ We should access this min() method by using numpy array object
- ✓ By using this we can check minimum value from the array.

Program	min() method
Name	demo1.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print(sales.min())
```

Output

```
10
```

3. max() method

- ✓ max() is a predefined method in numpy array.
- ✓ We should access this max() method by using numpy array object
- ✓ By using this we can check maximum value from the array.

Program Name	max() method demo2.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print(sales.max())
```

Output

```
60
```

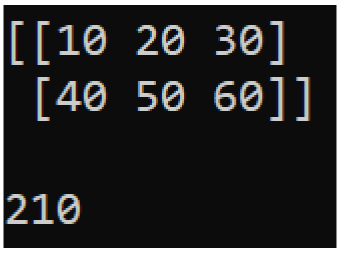
4. sum() method

- ✓ sum() is a predefined method in numpy array.
- ✓ We should access this method by using numpy array object
- ✓ By using this we can get sum of all values from array.

Program Name: sum() method
demo3.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(sales.sum())
```

Output



```
[[10 20 30]
 [40 50 60]]

210
```

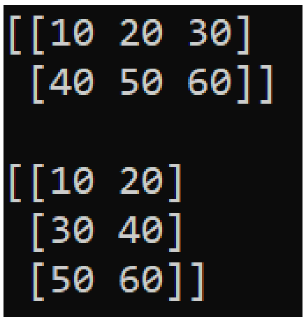
5. reshape() method

- ✓ reshape() is a predefined method in numpy array.
- ✓ We should access this method by using numpy array object
- ✓ By using this we can change the shape of an array.

Program Name: reshape() method
demo4.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(sales.reshape(3, 2))
```

Output

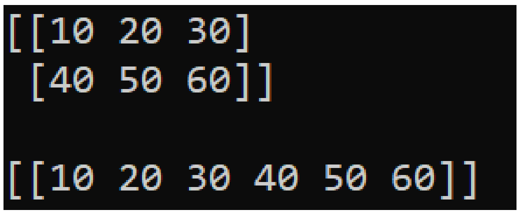


```
[[10 20 30]
 [40 50 60]]

[[10 20]
 [30 40]
 [50 60]]
```

Program Name	reshape() method demo5.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(sales.reshape(1, 6))
```

Output

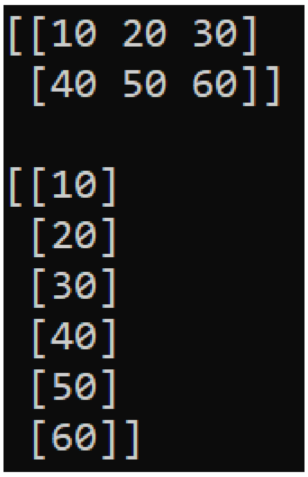


```
[[10 20 30]
 [40 50 60]]

[[10 20 30 40 50 60]]
```

Program Name reshape() method
demo6.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(sales.reshape(6, 1))
```

Output



```
[[10 20 30]
 [40 50 60]]

[[10]
 [20]
 [30]
 [40]
 [50]
 [60]]
```

6. count_nonzero(p) function

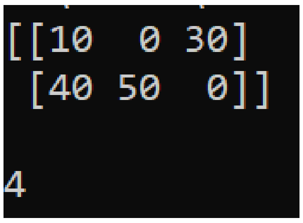
- ✓ count_nonzero(p) is a predefined function in numpy array.
- ✓ We should access this function by using numpy.
- ✓ By using this we can get non zero values from numpy

Program Name

count_nonzero(p) function
demo7.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(np.count_nonzero(sales))
```

Output



```
[[10  0 30]
 [40 50  0]]

4
```

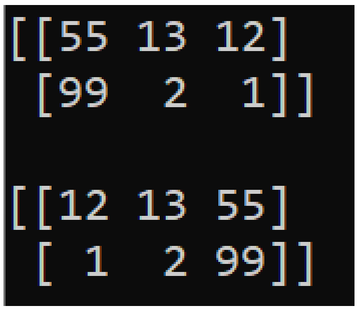
7. sort() method

- ✓ sort() is a predefined method in numpy array.
- ✓ We should access this method by using numpy array object
- ✓ By using this we can sort values in array.

Program Name	sort() method demo8.py

```
import numpy as np

details = [[55, 13, 12], [99, 2, 1]]
sales = np.array(details)
print(sales)
sales.sort()

print()
print(sales)
```

Output



```
[[55 13 12]
 [99  2  1]]

[[12 13 55]
 [ 1  2 99]]
```

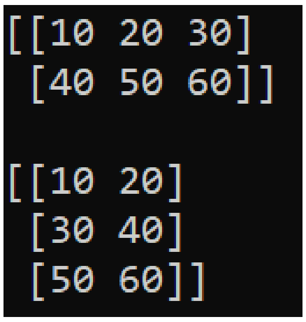
8. flatten() method

- ✓ flatten() is a predefined method in numpy array.
- ✓ We should access this method by using numpy array object
- ✓ This method keeps all values in one dimension array.

Program Name	flatten() method demo9.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(sales.flatten())
```

Output



```
[[10 20 30]
 [40 50 60]]

[[10 20]
 [30 40]
 [50 60]]
```

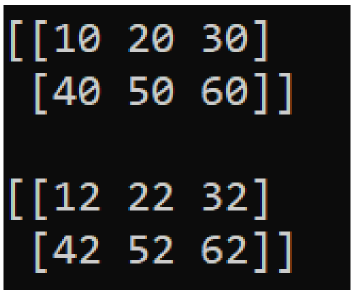
9. adding value to array of values

- ✓ Based on requirement we can add value to array of values.

Program Name: Adding value to array of values
demo10.py

```
import numpy as np

details = [[10, 20, 30], [40, 50, 60]]
sales = np.array(details)
print(sales)
print()
print(sales + 2)
```

Output



```
[[10 20 30]
 [40 50 60]]

[[12 22 32]
 [42 52 62]]
```

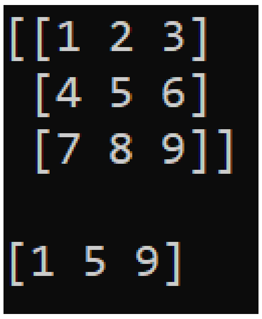
10. Diagonal of a Matrix

- ✓ Diagonal elements of a matrix.

Program Name	Diagonal matrix demo11.py

```
import numpy as np

matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(matrix)
print()
print(matrix.diagonal())
```

Output



```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[1 5 9]
```

11. Trace of a Matrix

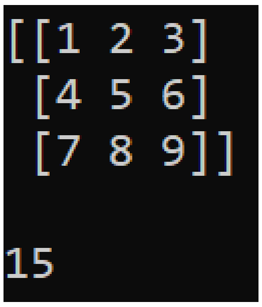
✓ The trace of a matrix is the sum of the diagonal elements.

Program Name

Trace of the matrix
demo12.py

```
import numpy as np

matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(matrix)
print()
print(matrix.trace())
```

Output



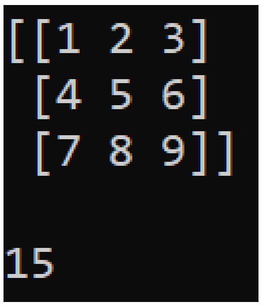
```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

15
```

Program Name

Trace of the matrix
demo13.py

```
import numpy as np

matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(matrix)
print()
print(sum(matrix.diagonal()))
```

Output



```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

15
```

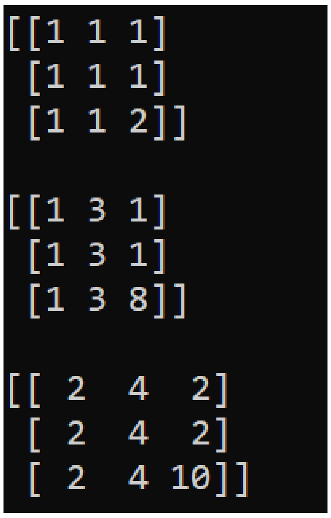
12. Adding and Subtracting Matrices

- ✓ We can add & subtract two matrices.
- ✓ We need to call add and subtract functions

Program Name: Adding two matrices
demo14.py

```
import numpy as np

matrix_a = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 2]])
matrix_b = np.array([[1, 3, 1], [1, 3, 1], [1, 3, 8]])

print(matrix_a)
print()
print(matrix_b)
print()
print(np.add(matrix_a, matrix_b))
```

Output



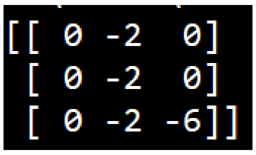
```
[[1 1 1]
 [1 1 1]
 [1 1 2]]

[[1 3 1]
 [1 3 1]
 [1 3 8]]

[[ 2  4  2]
 [ 2  4  2]
 [ 2  4 10]]
```

Program Name

Subtracting two matrices
demo15.py

```
import numpy as np

matrix_a = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 2]])
matrix_b = np.array([[1, 3, 1], [1, 3, 1], [1, 3, 8]])

print(matrix_a)
print()
print(matrix_b)
print()
print(np.subtract(matrix_a, matrix_b))
```

Output

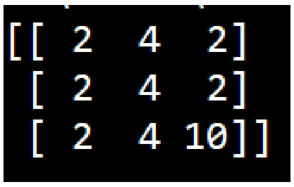


```
[[ 0 -2  0]
 [ 0 -2  0]
 [ 0 -2 -6]]
```

Program Name	Adding two matrices demo16.py

```
import numpy as np

matrix_a = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 2]])
matrix_b = np.array([[1, 3, 1], [1, 3, 1], [1, 3, 8]])

print(matrix_a + matrix_b)
```

Output

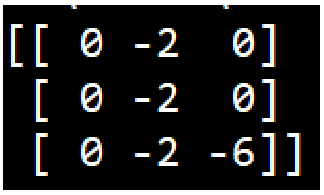


```
[[ 2  4  2]
 [ 2  4  2]
 [ 2  4 10]]
```

Program Name	Subtracting two matrices demo17.py

```
import numpy as np

matrix_a = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 2]])
matrix_b = np.array([[1, 3, 1], [1, 3, 1], [1, 3, 8]])

print(matrix_a - matrix_b)
```

Output



```
[[ 0 -2  0]
 [ 0 -2  0]
 [ 0 -2 -6]]
```