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 0^{th} index
  - Second element stores in 1^{st} 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 Name: Creating numpy array with group of values
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'>
```