NumPy:- Prepared by : Vithusan

- **Numerical Computing:** NumPy is a Python library for numerical computations.
- **Multidimensional Arrays:** Provides `ndarray` , allowing operations on multi-dimensional arrays.
- **Efficient Operations:** Offers mathematical functions for array operations, enhancing performance.
- **Broadcasting:** Performs implicit element-wise operations on arrays of different shapes.
- **Linear Algebra:** Includes tools for matrix operations and linear algebra tasks.
- **Integration:** Often used with Pandas, Matplotlib, and other data science libraries.
- **Used in:** Data analysis, machine learning, scientific computing, and more.

In [1]:

```python
#initialy Lets import numpy
import numpy as np
```

In [2]:

```python
my_list = [1, 2, 3, 4, 5]
arr = np.array(my_list)
```

In [3]:

```python
type(arr)
```

Out[3]:

```
numpy.ndarray
```

In [4]:

```python
print(arr)
```

```
[1 2 3 4 5]
```

In [5]:

```python
arr
```

Out[5]:

```
array([1, 2, 3, 4, 5])
```

In [6]:

```python
arr.shape #to find the size of the array
```

Out[6]:

```
(5,)
```

In [7]:

```python
#Multinested array

my_list1 = [1, 2, 3, 4, 5]
my_list2 = [2, 3, 4, 5, 6]
my_list3 = [9, 7, 6, 8, 9]

arr = np.array([my_list1, my_list2, my_list3]) #Converting to multidimensional array
```

In [8]:

```python
arr
```

Out[8]:

```
array([[1, 2, 3, 4, 5],
       [2, 3, 4, 5, 6],
       [9, 7, 6, 8, 9]])
```

```
In [9]:  #see the size of the array
         arr.shape

Out[9]:  (3, 5)
```

```
In [10]:  arr.reshape(3, 5)

Out[10]:  array([[1, 2, 3, 4, 5],
                 [2, 3, 4, 5, 6],
                 [9, 7, 6, 8, 9]])
```

```
In [11]:  arr.reshape(1, 15)

Out[11]:  array([[1, 2, 3, 4, 5, 2, 3, 4, 5, 6, 9, 7, 6, 8, 9]])
```

arr.reshape(3, 5) takes existing array arr and reshapes it into 3 rows, 5 columns.

Keeps all elements from original array.

Total number of elements must be the same for both shapes.

No new data copy: just reinterprets existing data.

indexing

```
In [12]:  arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [13]:  arr[0]

Out[13]:  1
```

```
In [14]:  arr.shape

Out[14]:  (9,)
```

```
In [15]:  multiArr = np.array([[1, 2, 3, 4, 5],
                     [2, 3, 4, 5, 6],
                     [9, 7, 6, 8, 9]])
```

```
In [16]:  multiArr[0][0]

Out[16]:  1
```

```
In [17]:  multiArr[0:2, 0:2]

Out[17]:  array([[1, 2],
                 [2, 3]])
```

```
In [18]:  multiArr[1:4, 3:5]

Out[18]:  array([[5, 6],
                 [8, 9]])
```

In [19]:
```
multiArr[1:]
```
Out[19]:
```
array([[2, 3, 4, 5, 6],
       [9, 7, 6, 8, 9]])
```
In [20]:
```
multiArr[1:3, 2:4]
```
Out[20]:
```
array([[4, 5],
       [6, 8]])
```
In [21]:
```
multiArr[1:3, 1:4]
```
Out[21]:
```
array([[3, 4, 5],
       [7, 6, 8]])
```
In [22]:
```
multiArr = np.arange(0, 10, 2) #np.arange generates evenly spaced numbers. (start, e
```
In [23]:
```
multiArr
```
Out[23]:
```
array([0, 2, 4, 6, 8])
```
In [24]:
```
"""
np.linspace: generates even spacing between numbers.
1, 10, 50: Starting value, ending value (not included), number of elements.
Creates an array of 50 numbers between 1 and 10, evenly spaced.

linspace offers more precise control over spacing compared to arange.
"""
np.linspace(1, 10, 50)
```
Out[24]:
```
array([ 1.        ,  1.18367347,  1.36734694,  1.55102041,  1.73469388,
        1.91836735,  2.10204082,  2.28571429,  2.46938776,  2.65306122,
        2.83673469,  3.02040816,  3.20408163,  3.3877551 ,  3.57142857,
        3.75510204,  3.93877551,  4.12244898,  4.30612245,  4.48979592,
        4.67346939,  4.85714286,  5.04081633,  5.2244898 ,  5.40816327,
        5.59183673,  5.7755102 ,  5.95918367,  6.14285714,  6.32653061,
        6.51020408,  6.69387755,  6.87755102,  7.06122449,  7.24489796,
        7.42857143,  7.6122449 ,  7.79591837,  7.97959184,  8.16326531,
        8.34693878,  8.53061224,  8.71428571,  8.89795918,  9.08163265,
        9.26530612,  9.44897959,  9.63265306,  9.81632653, 10.        ])
```
In [25]:
```
#copy function and broadcasting
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
arr[3:]
```
Out[25]:
```
array([4, 5, 6, 7, 8, 9])
```
In [26]:
```
arr[3:] = 100
arr
```
Out[26]:
```
array([  1,   2,   3, 100, 100, 100, 100, 100, 100])
```

In [27]:

```python
arr1 = arr
arr1[3:] = 500
print(arr1)
```

```
[  1   2   3 500 500 500 500 500 500]
```

In [28]:

```python
arr
```

Out[28]:

```
array([  1,   2,   3, 500, 500, 500, 500, 500, 500])
```

In [29]:

```python
"""
arr1 = arr.copy(): creates a new, independent copy of arr in arr1.
Useful for preventing changes to arr from affecting arr1.
Different from assignment (arr1 = arr), which only creates a reference.
"""

arr1 = arr.copy()
```

In [30]:

```python
print(arr)
arr1[3:] = 1000
print(arr1)
```

```
[  1   2   3 500 500 500 500 500 500]
[   1    2    3 1000 1000 1000 1000 1000 1000]
```

In [31]:

```python
### Some conditions very useful in Explonataty Data Analysis

val = 2
arr < 2
```

Out[31]:

```
array([ True, False, False, False, False, False, False, False, False])
```

In [32]:

```python
arr * 2
```

Out[32]:

```
array([   2,    4,    6, 1000, 1000, 1000, 1000, 1000, 1000])
```

In [33]:

```python
arr / 2
```

Out[33]:

```
array([  0.5,   1. ,   1.5, 250. , 250. , 250. , 250. , 250. , 250. ])
```

In [34]:

```python
arr % 2
```

Out[34]:

```
array([1, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int32)
```

In [35]:

```python
arr[arr < 3]
```

Out[35]:

```
array([1, 2])
```

In [42]:

```
"""
np.ones: Creates arrays filled with 1s.
4: Number of elements in the array.
dtype=int: Ensures integer data type for elements.
"""

np.ones(20, dtype = int)
```

Out[42]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [45]:

```
np.random.rand(3, 3)
```

Out[45]:

```
array([[0.10637853, 0.72503824, 0.05965371],
       [0.22502411, 0.4938258 , 0.04048343],
       [0.42567472, 0.54601295, 0.12857453]])
```

Explanation: random.rand(3, 3)

- **Purpose:** Creates a 3x3 NumPy array filled with random values between 0 (inclusive) and 1 (exclusive).
- **Breakdown:**
  - `np.` : Refers to the NumPy library.
  - `random` : A submodule of NumPy for generating random numbers.
  - `rand` : A function within `random` that creates arrays of random floats.
  - `(3, 3)` : Specifies the desired shape of the array, 3 rows and 3 columns.

Short Notes:

- `np.random.rand` : Generates arrays of random floats.
- `(3, 3)` : 3 rows, 3 columns.
- Values between 0 (inclusive) and 1 (exclusive).

Key Points:

- Useful for initializing arrays with random values for simulations, testing, or machine learning.
- Produces a different array each time it's called, ensuring randomness.

Additional Information:

- For other random distributions (e.g., standard normal), use `np.random.randn` .
- To control the randomness, consider setting a seed using `np.random.seed` .

In [38]:

```
np.random.randint(0, 100, 8)
```

Out[38]:

```
array([13, 35, 32,  7, 95, 25, 20, 57])
```

In [39]:

```
np.random.randint(0, 100, 8).reshape(2, 4)
```

Out[39]:

```
array([[27, 23, 29, 87],
       [37, 76, 88, 34]])
```

```
In [49]: np.random.random_sample((1, 5))
```

```
Out[49]: array([[0.1415306 , 0.49941298, 0.61931647, 0.35958998, 0.70495338]])
```

Explanation: `np.random.random_sample((1, 5))` :

- **Purpose:** Creates a 1x5 NumPy array filled with random floats between 0 (inclusive) and 1 (exclusive).
- **Breakdown:**
  - `np.` : Refers to the NumPy library.
  - `random` : A submodule of NumPy for generating random numbers.
  - `random_sample` : A function within `random` that creates arrays of random floats.
  - `(1, 5)` : Specifies the desired shape of the array, 1 row and 5 columns.

Short Notes:

- `np.random.random_sample` : Generates arrays of random floats.
- `(1, 5)` : 1 row, 5 columns.
- Values between 0 (inclusive) and 1 (exclusive).

Key Points:

- Similar to `np.random.rand` , but with a broader range of possible values.
- Also produces a different array each time it's called.

Additional Information:

- For integer-valued random arrays, use `np.random.randint` .
- To control the randomness, consider setting a seed using `np.random.seed` .

```
In [ ]: 
```

2024.01.10

[Find Me on Linkedin (https://bit.ly/3U2fXs6)](https://bit.ly/3U2fXs6)

[Github NumPy Repository (https://bit.ly/3tONGuz)](https://bit.ly/3tONGuz)