

Numpy

python list :

```
lst = [1, 2, 3, 4, 5]
```

Create Numpy Array :

```
arr = np.array(lst)
```

properties of Numpy Array :

- Array : `arr`
- shape : `arr.shape`
- Data Type : `arr.dtype`
- Item size : `arr.itemsize`
- Dimensionality : `arr.ndim`

Functions for creating Numpy Array :

- `arr = np.arange(start, end, step)`
- `arr = np.ones((3, 3))`
- `arr = np.zeros((3, 3))`
- `arr = np.eye(3)`
(or)
- `arr = np.eye(3, 2)`

- create Numpy array by using Numpy Random Numbers :

- `arr = np.random.rand(rows, columns)`
- `arr = np.random.randn(rows, columns)`
- `arr = np.random.randint(start, stop, size=(rows, columns))`
- `arr = np.random.uniform(start, stop, size=(rows, columns))`

Data Types in Numpy :-

- i - Integer
- b - Boolean
- str - string
- f - float
- m - timedelta
- M - datetime
- O - Object
- u - unsigned integer
- c - complex float
- U - Unicode string
- v - fixed chunk of memory for other type (void)

→ `arr = np.array (lst, dtype = 'f')`

Indexing :

1-D Array

- Single Index :

`arr [idx]`

- Multiple Indexes :

`arr [[2, 4, 3]]`

2-D Array

- `arr [2]` # Returns 2nd row of an array

Syntax-1 - `arr [2][1]` # Both Returns

Syntax-2 - `arr [2, 1]` in 2nd row,
1st column index value

Accessing Multiple values :

`arr [[1, 3], [1, 2]]`



Indexing with Boolean Array:

```
arr = np.array([1, 2, 3, 4, 5])  
idx = [False, True, True, False, False]  
arr[idx] # [2 3]
```

Slicing:

1-D Array

```
arr[1:4:]
```

2-D Array

```
arr[:, 2, 1:3]
```

updating data in Numpy Array:

```
arr = np.array([[1, 2], [3, 4]])  
arr  
# [[1. 2.]  
   [3. 4.]]
```

```
arr[1, 1] = 7
```

```
arr  
# [[1 2]  
   [3 7]]
```

Flatten() & Ravel():

```
arr = np.array([[1, 2], [3, 4]])  
arr  
# [[1 2]  
   [3 4]]
```

```
flatten_array = arr.flatten() # [1 2 3 4]
```

```
ravel_array = arr.ravel() # [1 2 3 4]
```

Note: Flatten creates a copy of original array and flattens.
Ravel creates a view of original array as 1-D array.

Numpy Reshape :

```
arr = np.array([[3, 2], [4, 1], [5, 7]])
```

arr

```
# [[3 2]
   [4 1]
   [5 7]]
```

```
# [[3 2 4]
   [1 5 7]]
```

```
arr.reshape(2, 3)
```

Iterating over Numpy Array :

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

```
for i in arr.ravel():
```

```
    print(i, end='')
```

```
# 1 2 3 4
```

- nditer() :

```
for item in np.nditer(arr):
```

```
    print(item, end='')
```

```
# 1 2 3 4
```

Numpy Maths :

- Square root `np.sqrt()`

- Exponent `np.exp()`

- Sin `np.sin()`

- cos `np.cos()`

Numpy Statistics :

- Minimum `np.min()`

- Maximum `np.max()`

- Mean `np.mean()`

- Median `np.median()`

- Variance `np.var()`

- Std `np.std()`

Elementwise Operations :

- Addition : `np.add(x, y, axis=)`
- Subtraction : `np.subtract(x, y, axis=)`
- Multiply : `np.multiply(x, y, axis=)`
- division : `np.divide(x, y)`

Matrix Multiplication :

Way-1 : `np.matmul(x, y)`

Way-2 : `np.dot(x, y)`

Way-3 : `x@y`

- Diagonal :

`np.diag(x)`

- Sorting :

`np.sort(x)`

- TRANSPOSE :

`x.T`

- linspace :

`np.linspace(begin, end, No. of Elements)`

- Stacking :

- Vertical Stacking : `np.vstack()`

- Horizontal Stacking : `np.hstack()`

- concatenate :

`np.concatenate([arr-1, arr-2], axis=0)` # vertical stacking

`np.concatenate([arr-1, arr-2], axis=1)` # Horizontal Stacking

- Append :

`np.append(arr-1, arr-2, axis=0)` # vertical stacking

`np.append(arr-1, arr-2, axis=1)` # Horizontal stacking.

where :

`np.where (arr % == 2, 'Even', 'odd')`

argsort :

Indexes : `arr.argsort ()`

Sorted Array : `arr [arr.argsort ()]`

Reading CSV into Numpy Array :

`arr = np.loadtxt ('data/nyc-weather.csv', delimiter=',',
dtype='str')`