NUMPY

Introduction

- NumPy is a Python library.
- NumPy is used for working with arrays.
- NumPy is short for "Numerical Python".

Creating a NumPy array

```
import numpy as np

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

print(arr)

print(type(arr))
```


Creating a NumPy array

```
import numpy as np

arr = np.array((1, 2, 3, 4, 5))

print(arr)
```


0-D Arrays

```
import numpy as np

arr = np.array(42)

print(arr)
```


1-D Arrays

```
import numpy as np

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

print(arr)
```


2-D Arrays

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr)
```


3-D Arrays

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3],
[4, 5, 6]]])

print(arr)
```


Check Number of Dimensions?

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3],
[4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```


Higher Dimensional Arrays

```
import numpy as np

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

print(arr)
print('number of dimensions :', arr.ndim)
```

Access Array Elements

```
import numpy as np

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

print(arr[0])
print(arr[1])
print(arr[2] + arr[3])
```


Access 2-D Arrays

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])
print('5th element on 2nd row: ', arr[1, 4])
```


Access 3-D Arrays

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9],
[10, 11, 12]]])

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


Access 3-D Arrays

```
import numpy as np
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9],
[10, 11, 12]]])
print(arr[0, 1, 2])
```


Negative Indexing

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('Last element from 2nd dim: ', arr[1, -1])
```


Slicing arrays

```
import numpy as np

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

print(arr[1:5])
print(arr[4:])
print(arr[:4])
print(arr[-3:-1])
print(arr[1:5:2])
print(arr[::2])
```


Slicing 2-D Arrays

```
import numpy as np

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])
print(arr[0:2, 2])
print(arr[0:2, 1:4])
```


Data Types in NumPy

NumPy has some extra data types, and refer to data types with one character, like `i` for integers, `u` for unsigned integers etc.

Below is a list of all data types in NumPy and the characters used to represent them.

- `i` - integer
- `b` - boolean
- `u` - unsigned integer
- `f` - float
- `c` - complex float
- `M` - datetime
- `O` - object
- `S` - string
- `U` - unicode string
- `V` - fixed chunk of memory for other type (void)

Checking the Data Type of an Array

```
import numpy as np

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

arr2 = np.array(['apple', 'banana', 'cherry'])
print(arr2.dtype)
```


Creating Arrays With a Defined Data Type

```
import numpy as np

arr = np.array([1, 2, 3, 4], dtype='S')

print(arr)
print(arr.dtype)
```

Creating Arrays With a Defined Data Type

```
import numpy as np

arr = np.array([1, 2, 3, 4], dtype='i4')

print(arr)
print(arr.dtype)
```


Creating Arrays With a Defined Data Type

```
import numpy as np

arr = np.array(['a', '2', '3'], dtype='i')
```


Converting Data Type on Existing Arrays

```
import numpy as np

arr = np.array([1.1, 2.1, 3.1])

newarr = arr.astype('i')

print(newarr)
print(newarr.dtype)
```


Converting Data Type on Existing Arrays

```
import numpy as np

arr = np.array([1.1, 2.1, 3.1])

newarr = arr.astype(int)

print(newarr)
print(newarr.dtype)
```


Converting Data Type on Existing Arrays

```
import numpy as np

arr = np.array([1, 0, 3])

newarr = arr.astype(bool)

print(newarr)
print(newarr.dtype)
```


The Difference Between Copy and View

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42

print(arr)
print(x)
```


The Difference Between Copy and View

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42

print(arr)
print(x)
```


Make Changes in the VIEW

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
x[0] = 31

print(arr)
print(x)
```


Check if Array Owns its Data

```
import numpy as np

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

x = arr.copy()
y = arr.view()

print(x.base)
print(y.base)
```


Get the Shape of an Array

```
import numpy as np

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

print(arr.shape)
```

Get the Shape of an Array

```
import numpy as np

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

print(arr)
print('shape of array :', arr.shape)
```


Reshaping arrays

```
import numpy as np

arr=np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

x = arr.reshape(4, 3)
y = arr.reshape(2, 3, 2)

print(x)
print(y)
```


Reshaping arrays

```
# Can not Reshape Into any Shape

import numpy as np

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

newarr = arr.reshape(3, 3)

print(newarr)
```


Reshaping arrays

```
# The returned array is a copy or a view?

import numpy as np

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

print(arr.reshape(2, 4).base)
```


Unknown Dimension

```
import numpy as np

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

newarr = arr.reshape(2, 2, -1)

print(newarr)
```


Flattening the arrays

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

newarr = arr.reshape(-1)

print(newarr)
```


Iterating 1-D Arrays

```
import numpy as np

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

for x in arr:
  print(x)
```


Iterating 2-D Arrays

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

for x in arr:
  print(x)
```


Iterating 2-D Arrays

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

for x in arr:
  for y in x:
    print(y)
```


Iterating 3-D Arrays

```
import numpy as np

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

for x in arr:
  for y in x:
    for z in y:
      print(z)
```

Using nditer()

```
import numpy as np

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

for x in np.nditer(arr):
  print(x)
```


Using nditer()

```
import numpy as np

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

for x in np.nditer(arr[:,::2]):
  print(x)
```


Using ndenumerate()

```
import numpy as np

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

for idx, x in np.ndenumerate(arr):
  print(idx, x)
```


Using ndenumerate()

```
import numpy as np

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

for idx, x in np.ndenumerate(arr):
  print(idx, x)
```


- Điểm thi =
- Điểm bài trắc nghiệm (tuần 15, 70 %)
- Điểm bài lập trình (10 đề) (tuần 16, 30%)

Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
```


Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([[1, 2], [3, 4]])

arr2 = np.array([[5, 6], [7, 8]])

arr = np.concatenate((arr1, arr2), axis=0)
print(arr)
arr = np.concatenate((arr1, arr2))
print(arr)
```


Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([[1, 2], [3, 4]])

arr2 = np.array([[5, 6], [7, 8]])

arr = np.concatenate((arr1, arr2), axis=1)

print(arr)
```


Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.stack((arr1, arr2), axis=0)

print(arr)
```


Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.stack((arr1, arr2), axis=1)

print(arr)
```

Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.hstack((arr1, arr2))

print(arr)
```


Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.vstack((arr1, arr2))

print(arr)
```


Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.dstack((arr1, arr2))

print(arr)
```


Joining NumPy Arrays

```
import numpy as np

arr1 = np.array([[1], [2], [3]])

arr2 = np.array([[4], [5], [6]])

arr = np.dstack((arr1, arr2))

print(arr)
```


NumPy Splitting Array

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6])

newarr = np.array_split(arr, 3)

print(newarr)
```


NumPy Splitting Array

```
import numpy as np

arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8],
[9, 10], [11, 12]])

newarr = np.array_split(arr, 3)

print(newarr)
```


NumPy Splitting Array

```
import numpy as np

arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8],
[9, 10], [11, 12]])

newarr = np.array_split(arr, 3)

print(newarr)
```


NumPy Splitting Array

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9],
[10, 11, 12], [13, 14, 15], [16, 17, 18]])

newarr = np.array_split(arr, 3)

print(newarr)
```


NumPy Splitting Array

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9],
[10, 11, 12], [13, 14, 15], [16, 17, 18]])

newarr = np.array_split(arr, 3, axis=1)

print(newarr)
```


NumPy Splitting Array

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6],
[7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])

newarr = np.hsplit(arr, 3)

print(newarr)
```

Searching Arrays

```
import numpy as np

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

x = np.where(arr == 4)

print(x)
```


Searching Arrays

```
import numpy as np

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

x = np.where(arr%2 == 0)

print(x)
```


Search Sorted

```
import numpy as np

arr = np.array([6, 7, 8, 9])

x = np.searchsorted(arr, 7)

print(x)
```


Search Sorted

```
import numpy as np

arr = np.array([6, 7, 8, 9])

x = np.searchsorted(arr, 7, side='right')

print(x)
```


Search Sorted

```
import numpy as np

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

x = np.searchsorted(arr, [2, 4, 6])

print(x)
```


Filter

```
import numpy as np

arr = np.array([41, 42, 43, 44])

x = [True, False, True, False]

newarr = arr[x]

print(newarr)
```


Filter

```
import numpy as np
arr = np.array([41, 42, 43, 44])
# Create an empty list
filter_arr = []
for element in arr:
  if element > 42:
    filter_arr.append(True)
  else:
    filter_arr.append(False)
newarr = arr[filter_arr]
print(filter_arr)
print(newarr)
```


Filter

```
import numpy as np

arr = np.array([41, 42, 43, 44])

filter_arr = arr > 42

newarr = arr[filter_arr]

print(filter_arr)
print(newarr)
```


Filter

```
import numpy as np

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

filter_arr = arr % 2 == 0

newarr = arr[filter_arr]

print(filter_arr)
print(newarr)
```


NUMPY UFUNC

What are ufuncs?

ufuncs stands for "Universal Functions" and they are NumPy functions that operate on the ndarray object.

What are ufuncs?

- ufuncs stands for "Universal Functions" and they are NumPy functions that operate on the ndarray object.
- ufuncs are used to implement *vectorization* in NumPy which is way faster than iterating over elements.
- They also provide broadcasting and additional methods like reduce, accumulate etc. that are very helpful for computation.

Add the Elements of Two Lists

```
x = [1, 2, 3, 4]
y = [4, 5, 6, 7]
z = []

for i, j in zip(x, y):
  z.append(i + j)
print(z)
```


Create Your Own ufunc

```
import numpy as np

def myadd(x, y):
  return x+y

myadd = np.frompyfunc(myadd, 2, 1)

print(myadd([1, 2, 3, 4], [5, 6, 7, 8]))
```


Check if a Function is a ufunc

```
import numpy as np

print(type(np.add))
```

Note: A ufunc should return <class 'numpy.ufunc'>

Check if a Function is a ufunc

```
import numpy as np

print(type(np.concatenate))
```

Output:
<class 'builtin_function_or_method'>

Check if a Function is a ufunc

```
import numpy as np

if type(np.add) == np.ufunc:
  print('add is ufunc')
else:
  print('add is not ufunc')
```


Simple Arithmetic

```
import numpy as np

arr1 = np.array([10, 11, 12, 13, 14, 15])
arr2 = np.array([20, 21, 22, 23, 24, 25])

newarr = np.add(arr1, arr2)
print(newarr)
newarr = np.subtract(arr1, arr2)
print(newarr)
newarr = np.multiply(arr1, arr2)
print(newarr)
```


Divide

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 5, 10, 8, 2, 33])

newarr = np.divide(arr1, arr2)

print(newarr)
```


Power

```
import numpy as np

arr1 = np.array([10, 20, 30])
arr2 = np.array([3, 5, 6])

newarr = np.power(arr1, arr2)

print(newarr)
```

Remainder

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 7, 9, 8, 2, 33])

newarr = np.mod(arr1, arr2)
# newarr = np.remainder(arr1, arr2)

print(newarr)
```


Quotient and Mod

```
import numpy as np

arr1 = np.array([10, 20, 30, 40, 50, 60])
arr2 = np.array([3, 7, 9, 8, 2, 33])

newarr = np.divmod(arr1, arr2)
print(newarr)


Output:
(array([3, 2, 3, 5, 25, 1]), array([1, 6, 3, 0, 0, 27]))
```


Absolute Values

```
import numpy as np

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

newarr = np.absolute(arr)    # np.abs(arr)
print(newarr)


Output:
[1 2 1 2 3 4]
```


Rounding Decimals

```
import numpy as np

arr = np.trunc([-3.1666, 3.6667])  # [-3. 3.]
arr = np.fix([-3.1666, 3.6667])    # [-3. 3.]
arr = np.around(3.1666, 2)         # 3.17
arr = np.floor([-3.1666, 3.6667])  # [-4. 3.]
arr = np.ceil([-3.1666, 3.6667])   # [-3. 4.]
```


Logs

```
import numpy as np
arr = np.arange(1, 10)

print(np.log2(arr))
print(np.log10(arr))
print(np.log(arr))
```


Logs

```
from math import log
import numpy as np

nplog = np.frompyfunc(log, 2, 1)

print(nplog(100, 15))
```


Summation

```
import numpy as np

arr1 = np.array([1, 2, 3])
arr2 = np.array([1, 2, 3])

newarr = np.sum([arr1, arr2])

print(newarr)

Output:
12
```


Summation Over an Axis

```
import numpy as np

arr1 = np.array([1, 4, 2])
arr2 = np.array([1, 2, 3])

newarr = np.sum([arr1, arr2], axis=1)

print(newarr)

Output:
[7 6]
```


Cummulative Sum

```
import numpy as np

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

newarr = np.cumsum(arr)

print(newarr)


Output:
[1 3 6]
```


Products

```
import numpy as np

arr1 = np.array([1, 2, 3, 4])
arr2 = np.array([5, 6, 7, 8])

x = np.prod(arr1)          # 24
x = np.prod([arr1, arr2]) # 40320
newarr = np.prod([arr1, arr2], axis=1) # [24 1680]
newarr = np.cumprod(arr2)  # [5 30 210 1680]
```

Differences

```
import numpy as np

arr = np.array([10, 15, 25, 5])

newarr = np.diff(arr)       # [ 5 10 -20]
newarr = np.diff(arr, n=2)  # [ 5 -30]
```


Finding LMC (Lowest Common Multiple)

```
import numpy as np

num1 = 4
num2 = 6

x = np.lcm(num1, num2)

print(x)
```


Finding LMC (Lowest Common Multiple)

```
import numpy as np

arr = np.array([3, 6, 9])

x = np.lcm.reduce(arr)

print(x)
```


Finding GCD (Greatest Common Denominator)

```
import numpy as np

num1 = 6
num2 = 9

x = np.gcd(num1, num2)

print(x)
```


Finding GCD (Greatest Common Denominator)

```
import numpy as np

arr = np.array([20, 8, 32, 36, 16])

x = np.gcd.reduce(arr)

print(x)
```


Trigonometric Functions

```
import numpy as np

x = np.sin(np.pi/2)

print(x)
```


Trigonometric Functions

```
import numpy as np

arr = np.array([np.pi/2, np.pi/3, np.pi/4, np.pi/5])

x = np.sin(arr)

print(x)


Output:
[1.         0.8660254  0.70710678 0.58778525]
```


Convert Degree Into Radians

```
import numpy as np

arr = np.array([90, 180, 270, 360])

x = np.deg2rad(arr)

print(x)


Output:
[1.57079633 3.14159265 4.71238898 6.28318531]
```


Radians to Degrees

```
import numpy as np

arr = np.array([np.pi/2, np.pi, 1.5*np.pi, 2*np.pi])

x = np.rad2deg(arr)

print(x)


Output:
[ 90. 180. 270. 360.]
```


Finding Angles

```
import numpy as np

x = np.arcsin(1.0)

print(x)


Output:
1.5707963267948966
```

Finding Angles

```
import numpy as np

arr = np.array([1, -1, 0.1])

x = np.arcsin(arr)

print(x)
```

```
Output:
[ 1.57079633 -1.57079633 0.10016742]
```


Hypotenues

```
import numpy as np

base = 3
perp = 4

x = np.hypot(base, perp)

print(x)
```

```
Output:
5.0
```


Hyperbolic Functions

```
import numpy as np

x = np.sinh(np.pi/2)

arr = np.array([np.pi/2, np.pi/3, np.pi/4, np.pi/5])
x = np.cosh(arr)

x = np.arcsinh(1.0)

arr = np.array([0.1, 0.2, 0.5])
x = np.arctanh(arr)
```


Set Operations

```
import numpy as np

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

x = np.unique(arr)

print(x)
```


Set Operations

```
import numpy as np

sét1 = np.array([1, 2, 3, 4])
set2 = np.array([3, 4, 5, 6])

newarr = np.union1d(set1, set2)
newarr=np.intersect1d(set1,set2, assume_unique=True)
newarr = np.setdiff1d(set1, set2, assume_unique=True)
newarr = np.setxor1d(set1, set2, assume_unique=True)
```