#Table of Contents

In [ ]:
```
1.what is python Numpy?
2.Functions to Create Arrays
3.Attributes and Functions on Array
4.Indexing Array
5.Slicing Array
6.Operating on 1D Array
7.Operating on 2D Array
8.
9.Concatenation Arrays
10.Stacking of Array
11.Splitting of Array
12.Iterating Over Numpy Array
```

- What is Python Numpy?

1.The Numpy is a Python packages that Stands for Numerical Python

2.Numpy is also a core Library in Python for **Scientific Computation**

3.Numpy contains an n - dimensional array Objects

In [1]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in /home/student/my_project_en
v/lib/python3.6/site-packages (1.19.5)
```

In [3]:
```
# Things to import Numpy Lib
import numpy as np
```

In [ ]:
```
```

Difference B/W Arrays v/s List

1.Array Occupy less memory

2.It works Faster than List

3.It is very Convenient to use

The N-Dimensional Array

In [4]:
```
# Create a 2 list of height and weight
person_height = [5.2,5.4,8.9,5.6,6]
person_weight = [81,55,65,75,44,54]

# How to convert the following list to Array
person_height = np.array(person_height)
person_weight = np.array(person_weight)
print(type(person_height))
```

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

Advantages of Numpy Array Over List

In [ ]:
```
-Python Arrays are more compact as comapared to list
```

In [7]:
```
# Consider the list of numbers
num_list = [4,5,6,1,2,3]
# Add 1 to each Elelment
num_modified_list = [i + i for i in num_list]
num_modified_list
```

Out[7]:
```
[8, 10, 12, 2, 4, 6]
```

In [6]:
```
# Create an Array from a list:
num_array = np.array([4,5,6,1,2,3])
num_modified_array = num_array + 1
num_modified_array
```

Out[6]:
```
array([5, 6, 7, 2, 3, 4])
```

In [5]:
```
# Create a Numpy Array Using a Arange Functions
my_array = np.arange(10)
#arrays comsume less space comapred to list
# Create a Python list using Range:
my_list = list(range(10))
print("Range function by Arrays - ",my_array)
print("Range function of list - ",my_list)
```

```
Range function by Arrays -  [0 1 2 3 4 5 6 7 8 9]
Range function of list -  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

2 days pending 14 and 15

In [11]:
```
import numpy as np
num_array = np.array([1,2,3,4])
print(num_array**2)
print(num_array**4)
print(num_array*2)
price_pens = np.array([34,56,76,89,100])
output_array = price_pens >= 40
print(output_array)
print(price_pens[output_array])
```

```
[ 1  4  9 16]
[  1  16  81 256]
[2 4 6 8]
[False  True  True  True  True]
[ 56  76  89 100]
```

In [12]:
```
# another method
price_pens = np.array([34,56,76,89,100])
price_pens[price_pens >= 40]
```

Out[12]:
```
array([ 56,  76,  89, 100])
```

In [19]:
```
# Operations an 2D Array:

quantity_A = np.array([[33,12],[70,45]])
print('The First Array is : ',quantity_A)
print('\n')
quantity_B = np.array([[43,56],[78,90]])
```

```
print('The second Array is : ',quantity_B)
print(quantity_A + quantity_B)
quantity_A +=2
print(quantity_A )
quantity_A -=2
print(quantity_A )
```

```
The First Array is :  [[33 12]
 [70 45]]


The second Array is :  [[43 56]
 [78 90]]
[[ 76  68]
 [148 135]]
[[35 14]
 [72 47]]
[[33 12]
 [70 45]]
```

In [20]:
```
# matix multiplication
# method1
quantity_A @ quantity_B
```

Out[20]:
```
array([[2355, 2928],
       [6520, 7970]])
```

In [21]:
```
#method 2
quantity_A.dot(quantity_B)
```

Out[21]:
```
array([[2355, 2928],
       [6520, 7970]])
```

In [28]:
```
a1 = np.array([5,6,7,8])
print(a1)
print(a1.sum())
print(a1.min())
print(a1.max())
# get the cube of all the elements
print(a1**3)
# power -
np.power(a1,3)
```

```
[5 6 7 8]
26
5
8
[125 216 343 512]
```

Out[28]:
```
array([125, 216, 343, 512])
```

In [3]:
```
import numpy as np
#Assignments
#write a numpy program to create a 2D array with 1 on the border and
a = np.ones([4,4])
print(a)
a[1:-1,1:-1] = 0
print(a)
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]
```

In [5]:
```
# Create a
import numpy as np

matrix_empty = np.empty((2,2),dtype = float)
print(matrix_empty)
matrix_empty = np.empty((2,2),dtype = int)
print(matrix_empty)
```

```
[[ 1.29157455e-316  0.00000000e+000]
 [ 2.25313711e-301 -3.13176097e-294]]
[[           26141760                    0]
 [  109019212209084522 -9007198898218598399]]
```

In [20]:
```
#create 2 x 2 full matrix
matrix_full = np.full((2,2),fill_value = 10)
print(matrix_full)
print('\n')
#identity matrix
print(np.identity(3))
print('\n')
#create a 4 x 3 matrix
print(np.eye(N = 4,M = 3, k = 0))
# k = 0 represents main diagonal
# k > 0 represents upper diagonal
# k < 0 represents lower diagonal
print('\n')
print(np.eye(N = 4,M = 3, k = 2))
```

```
[[10 10]
 [10 10]]


[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]


[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 0.]]


[[0. 0. 1.]
 [0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [22]:
```
# string using numpy array we can able to store
array_string = np.array(['python','sql','C'])
print(array_string)
```

```
['python' 'sql' 'C']
```

In [37]:
```
import numpy as np
array = np.array(["I am jose and I am okay to sleep all nights and i
print(np.char.capitalize(array)[0][5:9])
array
```
```
jose
```
Out[37]:
```
array(['I am jose and I am okay to sleep all nights and i work at d
ays'],
      dtype='<U62')
```

In [32]:
```
step1 = np.char.add(['*'],array)
step2 = np.char.add(step1,['*'])
print(step2)
```
```
['*I am jose and I am okay to sleep all nights and i work at days*
']
```

In [ ]:
```
# Assignments
#Write a numpy program to create a vector with value ranging from 15
#Write a numpy program to create a vector of length 5 filled with arb
#Write a numpy program to create a 3 x 4 matrix filled with values 10
#write a numpy program to create a 3 x 3 x 3 array filled with arbitr
#write a numpy program to compute the inner product of two given vect
```

In [50]:
```
import numpy as np
my_list = np.arange(15,55)
print(my_list[1:-1])
```
```
[16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39
 40 41 42 43 44 45 46 47 48 49 50 51 52 53]
```

In [4]:
```
import numpy as np
print(np.char.center('hello',10,fillchar = '*'))
print(np.char.center('hello',3,fillchar = '*'))
```
```
**hello***
hel
```

In [6]:
```
print('concatenate two string')
print(np.char.add(['Hello'],['XYZ']))
print('\n')
print('concatenation example')
print(np.char.add(['Hello','hiii'],['XYZ','abc']))
```
```
concatenate two string
['HelloXYZ']



concatenation example
['HelloXYZ' 'hiiabc']
```

In [7]:
```
#using Upper and lower:
print(np.char.upper('hello'))
print(np.char.upper(['Hello World']))
```
```
HELLO
['HELLO WORLD']
```

In [12]:
```
# Using Multiply:
print(np.char.multiply('hello',3))
```

```
# Split:
print(np.char.split('hello how are you'))
#strip
print(np.char.strip(['I am good','aerkan','java'],'a'))
```

```
hellohellohello
['hello', 'how', 'are', 'you']
['I am good' 'erkan' 'jav']
```

In [16]:
```
# join
print(np.char.join(':','dmy'))
print('\n')
print(np.char.join([':','-'],['dmy','dmm']))
print(np.char.replace('He is a good boy','is','was'))
```

```
d:m:y


['d:m:y' 'd-m-m']
He was a good boy
```

numpy.median() ~ ** median is defined as the value seperating the higher half of the data from the lower half

In [20]:
```
a = np.array([[30,65,70],
             [80,97,90],
             [45,89,60],
             [50,60,70]])
print(np.median(a))
```

```
67.5
```

np.mean() ~**Arithmatic mean is the sum of elements along the axis divided by the number of elements ~**it returns the Arithmetic mean of all elements in the array

In [22]:
```
a = np.array([[30,65,70],
             [80,97,90],
             [45,89,60],
             [50,60,70]])
print(np.mean(a))
```

```
67.16666666666667
```

**the weight Average is calculated by adding the product of the corresponding elemnts and dividing by the sum of weights

a =[1,2,3,4] weight = [4,3,2,1] weighted_average = (1*4+2*3+3*2+4*1*)/(4+3+2+1)

In [25]:
```
a = np.array([1,2,3,4])
print(a)
print('\n')
print('Applying the averge() functiomn')
print(np.average(a))
wts = np.array([4,3,2,1])
print('\n')
print('Applying Average () Function again')
print(np.average(a,weights = wts))
print('sum of weights :')
```

```
#if the return value is true-it gives the sum of weights
print(np.average([1,2,3,4],weights = [4,3,2,1],returned = True))
```

```
[1 2 3 4]


Applying the averge() functiomn
2.5


Applying Average () Function again
2.0
sum of weights :
(2.0, 10.0)
```

In [ ]:
```
### numpy linear algebra:
```

##Assignments 19/04/23

Given a 2D binary matrix A of dimensions NXM,determine the row that contains a minimum number of 1's note-The matrix contains only 1s and 0s.Also,if two or more rows contain the minimum number of 1's, the answer is the lowest of those indices input: n=4,m=5 A=[[1,1,1,1],[1,1,0,0],[0,0,1,1],[1,1,1,1]] output: 2 Explanation: Rows 2 and 3 contain the minimum numberof 1's(2 each).Since,2 is less than 3. thus the answer is 2. your Task: you donot need to read input or print anything.your task is to complete the function minrow() which takes the two integers N,M as well as the 2D matrix as input parameter and returns the minimum index of the row which contains the least number of 1's

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: