Good Morning Everyone

Day Objectives

1. Python map() and fiter()
2. Introduction NumPy Arrays
3. NumPy Basics
4. Math
5. Random
6. Indexing
7. Filtering
8. Statistics
9. Aggregation
10. Saving Data

map()

Syantax map(function, GroupOfElements)

strings,list,tuple

In [1]:

```
li = ['1','2','3',4,5,6,'5','88']

num = '44'

int_num = int(num)

print(int_num, type(int_num))
```

```
44 <class 'int'>
```

In [5]:

```
int_li = map(int, li)

print(int_li)
```

```
<map object at 0x000002E5EC22ECF8>
```

In [6]:

```
print(list(int_li))
print(tuple(int_li))
```

```
[1, 2, 3, 4, 5, 6, 5, 88]
()
```

In [7]:

```
li2 = []
for ele in li:
    li2.append(int(ele))
print(li2)
```

```
[1, 2, 3, 4, 5, 6, 5, 88]
```

filter()

filter(function, GroupOfElements)

In [10]:

```
numbers = [1,2,3,4,5,6,7,8,9,0,12,13,55,77,88,345]

def even(num):
    if num % 2 == 0:
        return True
    else:
        False

even_numbers = filter(even, numbers)
print(even_numbers)
```

```
<filter object at 0x000002E5EC235A90>
```

In [11]:

```
print(list(even_numbers))
```

```
[2, 4, 6, 8, 0, 12, 88]
```

- input(), Output()
- Data Types
- Type Conversions
- Operators
- Data Structures
- File Handlings
- Functions

- Data Science
- Data Analyst
- Data Engineer
- AI
- ML
- DL

- Data Manuplation -- Numpy
- Data Analysis, Data cleaning, Data Importing, Exporting - Pandas

- Data Visualization - Matplotlib

NumPy - Numericial Python

MultiDimensional Arrays

In [12]:

```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\jesus\anaconda3\lib\site-pa
ckages (1.16.2)
Note: you may need to restart the kernel to use updated packages.
```

In [13]:

```
import numpy as np
```

In [14]:

```
np.__version__
```

Out[14]:

```
'1.16.2'
```

Numpy_Official_Website (https://numpy.org/)

In [16]:

```
a1 = np.array([1,2,3,4])
a2 = np.array((1,2,3,4))

a1, a2
```

Out[16]:

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

In [17]:

```
print(type(a1))
```

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

In [19]:

```
print(a1.dtype)
```

```
int32
```

In [23]:

```
a3 = a1.astype(float)
a4 = a1.astype('float32')
print(a3, a3.dtype)
print(a4, a4.dtype)
```

```
[1. 2. 3. 4.] float64
[1. 2. 3. 4.] float32
```

In [28]:

```
arr = np.array([1,2,3,4,'5546545435435435435'])

print(arr, arr.dtype, type(arr))
```

```
['1' '2' '3' '4' '5546545435435435435'] <U19 <class 'numpy.ndarray'>
```

In [26]:

```
a = [343]

print(type(a))

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

print(type(arr), arr.dtype)
```

```
<class 'list'>
<class 'numpy.ndarray'> int32
```

In [29]:

```
# range(start = 0, stop = manditory - 1, inc/dec/step = 1)

print(np.arange(5))
print(np.arange(1,5))
print(np.arange(1, 100, 10))
```

```
[0 1 2 3 4]
[1 2 3 4]
[ 1 11 21 31 41 51 61 71 81 91]
```

In [30]:

```
arr = np.arange(1, 100, 10)

print(arr.shape)
```

```
(10,)
```

In [33]:

```
print(arr.reshape(2,5))
print(arr)
print(arr.reshape(5,5))
```

```
[[ 1 11 21 31 41]
 [51 61 71 81 91]]
[ 1 11 21 31 41 51 61 71 81 91]


---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-33-9568df82fe38> in <module>
      1 print(arr.reshape(2,5))
      2 print(arr)
----> 3 print(arr.reshape(5,5))

ValueError: cannot reshape array of size 10 into shape (5,5)
```

In [41]:

```
arr2 = np.arange(50)
print(arr2.ndim)
arr3 = arr2.reshape(5,5,2)
print(arr3, arr3.ndim)
```

```
1
[[[ 0  1]
  [ 2  3]
  [ 4  5]
  [ 6  7]
  [ 8  9]]

 [[10 11]
  [12 13]
  [14 15]
  [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]]] 3
```

In [39]:

```
x = [1,2,3,4]
y = [[1,2], [3,4]]
```

In [44]:

```
ones = np.ones(5, dtype = int)
print(ones)
```

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

In [45]:

```
zeros = np.zeros(5, dtype = int)

print(zeros)
```

```
[0 0 0 0 0]
```

In [49]:

```
identity = np.eye(5,5, dtype = int)
print(identity)
```

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

In [51]:

```
print(identity.diagonal())
```

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

Mathematical Operations

In [53]:

```
marks = [2,3,4,5,7,7,5,3]

for i in marks:
    print(i + 1)
```

...

In [56]:

```
marks_arr = np.array(marks)

marks_arr += 1
```

In [57]:

```
marks_arr
```

Out[57]:

```
array([3, 4, 5, 6, 8, 8, 6, 4])
```

In [58]:

```
print(marks_arr + 1)
print(marks_arr - 1)
print(marks_arr * 1000)
print(marks_arr / 0.000001)
print(marks_arr ** 88)
```

...

indexing

In [60]:

```
arr = np.arange(50).reshape(10,5)
print(arr)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 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]]
```

In [64]:

```
arr[6][4]
```

Out[64]:

```
34
```

In [65]:

```
arr[6,4]
```

Out[65]:

```
34
```

In [66]:

```
arr[6:, 4]
```

Out[66]:

```
array([34, 39, 44, 49])
```

In [67]:

```
arr[6:, 2:]
```

Out[67]:

```
array([[32, 33, 34],
       [37, 38, 39],
       [42, 43, 44],
       [47, 48, 49]])
```

In [73]:

```
even_bool = arr % 2 == 0
even_bool
```

...

In [74]:

```
arr[even_bool]
```

Out[74]:

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32,
       34, 36, 38, 40, 42, 44, 46, 48])
```

Random Number Generation

In [80]:

```
print(np.random.randint(1, 100))
```

```
3
```

In [82]:

```
np.random.random(100)
```

. . .

Trignometric Functions

In [107]:

```
degrees = np.arange(0, 180, 15)
deginrad = np.radians(degrees)
print(np.sin(deginrad).astype(int))
print(np.cos(deginrad))
```

```
[0 0 0 0 0 0 1 0 0 0 0 0]
[ 1.00000000e+00  9.65925826e-01  8.66025404e-01  7.07106781e-01
  5.00000000e-01  2.58819045e-01  6.12323400e-17 -2.58819045e-01
 -5.00000000e-01 -7.07106781e-01 -8.66025404e-01 -9.65925826e-01]
```

In [92]:

```
cos = np.cos(deginrad)
print(np.ceil(cos))
print(np.floor(cos))
```

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

In [91]:

```
import math

print(math.ceil(100.5227))
print(math.floor(100.5227))
```

```
101
100
```

In [86]:

```
np.log(1)
```

Out[86]:

```
0.0
```

In [89]:

```

```

Out[89]:

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

In [87]:

```
print(np.nan)
```

```
nan
```

In [ ]:

```
print(np.inf)
```

In [98]:

```
np.ceil(55)
```

Out[98]:

```
55.0
```

In [103]:

```
sin = np.sin(deginrad)
np.save('sin',sin)
np.save('sin1.npy',sin)
```

In [104]:

```
ls
```

...

In [105]:

```
print(np.load('sin.npy'))
```

```
[0.         0.25881905 0.5        0.70710678 0.8660254  0.96592583
 1.         0.96592583 0.8660254  0.70710678 0.5        0.25881905]
```

In [106]:

```
len(dir(np))
```

Out[106]:

```
622
```