https://scipy.org/ (https://scipy.org/) search in google

https://www.numpy.org (https://www.numpy.org)

https://www.numpy.org/devdocs/user/quickstart.ht (https://www.numpy.org/devdocs/user/quickstart.h

Numpy Library

- Processing N-Dimensional arrays

In [2]:
```
import numpy as np

li = [1,2,3]

a=np.array(li)
type(a)
```

Out[2]:
```
numpy.ndarray
```

In [3]:
```
import numpy as np

li = [1,2,3,'z']

a=np.array(li)
a
```

Out[3]:
```
array(['1', '2', '3', 'z'], dtype='<U11')
```

In [4]:
```
import numpy as np

li = [1,2,3,'z']

a=np.array(li)

b=np.arange(15)
b
```

Out[4]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [6]:
```
rn= np.random.randint(0,100,size=10)
rn

```

Out[6]:
```
array([22,  6, 15, 19, 47, 30, 26, 31, 15, 41])
```

In [7]:
```
# For 3-Dimensional matrix

m3 = np.random.randint(0,1,size=(3,3))
m3
```

Out[7]:
```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

In [8]:
```
m3 = np.random.randint(0,2,size=(3,3))
m3
```

Out[8]:
```
array([[1, 1, 1],
       [1, 0, 0],
       [0, 1, 0]])
```

In [9]:
```
m3 = np.random.randint(0,2,size=(10,10))
m3
```

Out[9]:
```
array([[1, 1, 1, 0, 1, 0, 1, 0, 0, 1],
       [1, 1, 1, 1, 0, 1, 0, 1, 1, 1],
       [1, 1, 1, 1, 1, 1, 0, 1, 1, 1],
       [1, 0, 1, 1, 1, 1, 0, 0, 1, 1],
       [0, 0, 1, 1, 1, 0, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 1, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 1, 1, 0, 1, 1],
       [1, 1, 1, 0, 0, 1, 0, 0, 0, 0],
       [0, 1, 1, 1, 0, 1, 0, 1, 0, 1],
       [0, 0, 0, 0, 1, 1, 0, 1, 1, 0]])
```

In [10]:
```
# 3-Dimensional data
m3 = np.random.randint(0,2,size=(3,3,3))
m3
```

Out[10]:
```
array([[[0, 0, 0],
        [1, 1, 1],
        [1, 1, 1]],

       [[1, 0, 1],
        [1, 0, 0],
        [1, 0, 0]],

       [[1, 1, 0],
        [0, 1, 1],
        [1, 1, 1]]])
```

In [12]:
```
m3 = np.random.randint(0,2,size=(3,3,3,3))
m3
```

Out[12]:
```
array([[[[0, 0, 1],
         [0, 1, 0],
         [1, 1, 1]],

        [[1, 0, 0],
         [0, 1, 1],
         [0, 0, 1]],

        [[1, 1, 1],
         [1, 1, 1],
         [0, 1, 1]]],


       [[[1, 1, 0],
         [1, 0, 0],
         [0, 0, 0]],

        [[1, 0, 1],
         [0, 1, 0],
         [0, 1, 0]],

        [[1, 0, 1],
         [1, 1, 0],
         [1, 1, 0]]],


       [[[1, 0, 1],
         [0, 0, 0],
         [0, 0, 0]],

        [[1, 0, 0],
         [0, 0, 1],
         [1, 0, 0]],

        [[0, 1, 1],
         [0, 0, 1],
         [0, 0, 0]]]])
```

In [16]:
```
m3 = np.random.randint(0,2,size=(3,3,3,3))
m3[2][2][2][1]

```

Out[16]:
```
1
```

In [17]:
```
m3.ndim
```

Out[17]:
```
4
```

In [18]:
```
m3.size
```

Out[18]:
```
81
```

In [19]:
```
1 m3.shape
```

Out[19]:
```
(3, 3, 3, 3)
```

In [20]:
```
1 m3.dtype
```

Out[20]:
```
dtype('int32')
```

In [21]:
```
1 m3.itemsize
```

Out[21]:
```
4
```

In [22]:
```
1 m3.nbytes
```

Out[22]:
```
324
```

GitHub repository of Akash sir

- PythonDataScienceHandbook

In [23]:
```
1 print(b)   # print will remove all comas(,)
```

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

In [25]:
```
1 b.reshape(10,3)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-151988767fc1> in <module>
----> 1 b.reshape(10,3)

ValueError: cannot reshape array of size 15 into shape (10,3)
```

In [26]:
```
1 b.reshape(5,3)
```

Out[26]:
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11],
       [12, 13, 14]])
```

In [27]:
```
1 import numpy as np
2 M=np.ones((3,3))
3 print("Matrix M: \n",M)
```

```
Matrix M:
 [[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

Pandas

- Data Cleaning
- Data Transformation
- Data Analysis

Notations

- series --> 1D it is a row
- DataFrames --> table

In [2]:
```
import pandas as pd

internal1 = {'s1':21, 's2':18, 's3':24}

internal1 = pd.Series(internal1)

internal2 = {'s1':15,'s2':18,'s3':12}
internal2 = pd.Series(internal2)
internal2

```

Out[2]:
```
s1    15
s2    18
s3    12
dtype: int64
```

In [4]:
```
final = {'internal1':internal1, 'internal2':internal2}

final = pd.DataFrame(final)
final
```

Out[4]:

	internal1	internal2
s1	21	15
s2	18	18
s3	24	12

In [10]:
```
final = {'internal1':internal1, 'internal2':internal2}

final = pd.DataFrame(final)
final['internal1']
```

Out[10]:
```
s1    21
s2    18
s3    24
Name: internal1, dtype: int64
```

In [8]:
```
final.columns  # Name of all columns
```

Out[8]:
```
Index(['internal1', 'internal2'], dtype='object')
```

In [9]:
```
final.values  # Lists of all rows

```

Out[9]:
```
array([[21, 15],
       [18, 18],
       [24, 12]], dtype=int64)
```

In [11]:
```
final.values[2]   # accessing the 2nd row i.e 3 rd row i.e 0-1-2
```

Out[11]:
```
array([24, 12], dtype=int64)
```

In [12]:
```
final.values[2,0]  # 3 row 1st column
```

Out[12]:
```
24
```

In [13]:
```
final.values[2][0]
```

Out[13]:
```
24
```

In [16]:
```
for row in final.values:
    print('internal1 -',row[0],', internal2 -',row[1])
```

```
internal1 - 21 , internal2 - 15
internal1 - 18 , internal2 - 18
internal1 - 24 , internal2 - 12
```

In [17]:
```
final
```

Out[17]:

	internal1	internal2
s1	21	15
s2	18	18
s3	24	12

In [20]:
```
final.loc['3'] = [11, 10]
final
```

Out[20]:

	internal1	internal2
s1	21	15
s2	18	18
s3	24	12
3	11	10
s4	11	10
3	11	10

In [21]:
```
1 final.drop(3)
```

Out[21]:

	internal1	internal2
s1	21	15
s2	18	18
s3	24	12
s4	11	10
3	11	10

In [22]:
```
1 final.drop(3)
```

Out[22]:

	internal1	internal2
s1	21	15
s2	18	18
s3	24	12
s4	11	10
3	11	10

In [23]:
```
1 final.values[2,0] = 25
2 final
```

Out[23]:

	internal1	internal2
s1	21	15
s2	18	18
s3	25	12
3	11	10
s4	11	10
3	11	10

In [24]:
```
1 final
```

Out[24]:

	internal1	internal2
s1	21	15
s2	18	18
s3	25	12
3	11	10
s4	11	10
3	11	10

In [25]:
```
final.values[2] = [12,25]
```

In [26]:
```
final
```

Out[26]:

	internal1	internal2
s1	21	15
s2	18	18
s3	12	25
3	11	10
s4	11	10
3	11	10

In [27]:
```
final.drop(3)
```

Out[27]:

	internal1	internal2
s1	21	15
s2	18	18
s3	12	25
s4	11	10
3	11	10

In [2]:
```
# Reading CSV file data
import pandas as pd
filepath = 'DataFiles/Income.csv'
incomedf = pd.read_csv(filepath)
incomedf
```

Out[2]:

	GEOID	State	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	04000US01	Alabama	37150	37952	42212	44476	39980	40933	42590	43464	41381
1	04000US02	Alaska	55891	56418	62993	63989	61604	57848	57431	63648	61137
2	04000US04	Arizona	45245	46657	62993	46914	45739	46896	48621	47044	50602
3	04000US05	Arkansas	36658	37057	40795	39586	36538	38587	41302	39018	39919
4	04000US06	California	51755	55319	55734	57014	56134	54283	53367	57020	57528

In [4]:
```
# Extracting all The states income in the year 2013
for row in incomedf.values:
    print('state -',row[1],':','yearIncome-',row[-1])
```

```
state - Alabama : yearIncome- 41381
state - Alaska : yearIncome- 61137
state - Arizona : yearIncome- 50602
state - Arkansas : yearIncome- 39919
state - California : yearIncome- 57528
```

In [7]:
```
1 incomedf
```

Out[7]:

	GEOID	State	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	04000US01	Alabama	37150	37952	42212	44476	39980	40933	42590	43464	41381
1	04000US02	Alaska	55891	56418	62993	63989	61604	57848	57431	63648	61137
2	04000US04	Arizona	45245	46657	62993	46914	45739	46896	48621	47044	50602
3	04000US05	Arkansas	36658	37057	40795	39586	36538	38587	41302	39018	39919
4	04000US06	California	51755	55319	55734	57014	56134	54283	53367	57020	57528

In [6]:
```
1 # Extracting all The states income in the year 2013
2 for row in incomedf.values:
3     print(row[1],':',row[-1])
```

```
Alabama : 41381
Alaska : 61137
Arizona : 50602
Arkansas : 39919
California : 57528
```

In [9]:
```
1 # Extracting all income data in the year 2009
2 for row in incomedf.values:
3     print(row[1],':',row[-5])
```

```
Alabama : 39980
Alaska : 61604
Arizona : 45739
Arkansas : 36538
California : 56134
```

In [12]:
```
1 # Average income of the state Arizona
2 sum = 0
3 for i in range(2,11):
4     sum +=incomedf.values[2][i]
5
6
7 sum/len(incomedf.values[2][2:])
```

Out[12]: 48967.88888888889

In [14]:
```
1 # Average income of the year 2012
2
3 sum=0
4 for row in incomedf.values:
5     sum +=row[-2]
6 sum/len(incomedf.values)
```

Out[14]: 50038.8

In [15]:
```
1 incomedf
```

Out[15]:

	GEOID	State	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	04000US01	Alabama	37150	37952	42212	44476	39980	40933	42590	43464	41381
1	04000US02	Alaska	55891	56418	62993	63989	61604	57848	57431	63648	61137
2	04000US04	Arizona	45245	46657	62993	46914	45739	46896	48621	47044	50602
3	04000US05	Arkansas	36658	37057	40795	39586	36538	38587	41302	39018	39919
4	04000US06	California	51755	55319	55734	57014	56134	54283	53367	57020	57528

In [18]:
```
1 # each row is a value, here we have 5 values means 5 rows
2 # in that each row --->  row[0][-2] means 1st row last but one i.e 43464
3 incomedf.values
```

Out[18]:
```
array([['04000US01', 'Alabama', 37150, 37952, 42212, 44476, 39980, 40933,
        42590, 43464, 41381],
       ['04000US02', 'Alaska', 55891, 56418, 62993, 63989, 61604, 57848,
        57431, 63648, 61137],
       ['04000US04', 'Arizona', 45245, 46657, 62993, 46914, 45739, 46896,
        48621, 47044, 50602],
       ['04000US05', 'Arkansas', 36658, 37057, 40795, 39586, 36538,
        38587, 41302, 39018, 39919],
       ['04000US06', 'California', 51755, 55319, 55734, 57014, 56134,
        54283, 53367, 57020, 57528]], dtype=object)
```

In [ ]:
```
1
```