**Numpy and Pandas Documentation**

* Numpy stands for Numerical python. It supports large multi-dimensional arrays and matrices. It also contains large high level mathematical functions which makes computations on these arrays and matrices easier and efficient. Numpy data structure is called nd-array (n dimensional array). All the elements in the numpy arrays must be in same(homogeneous) data type.
* Pandas is a data manipulation library in python. It is built on top of the numpy library, and it extends the features further.

Pandas is giving us two types of data structures:

1. Series: - 1 dimensional and it can hold any type of data. Indexes can be numerical or character labels
2. DataFrame: - 2 dimensional with 3 principal components (rows, columns, values). Size of DataFrame can be changed anytime.

* Code links:
  + Github: <https://github.com/Sooriyakumar23/Taken-Mind-Global-Internship-Submissions/blob/main/Taken_Mind_Internship_Assignment_1.ipynb>
  + Google colab: <https://colab.research.google.com/drive/1JHNKUkEKOJhY4dkOy50w9nGIdmfZc8bp#scrollTo=sKtA-3QwSH8n>

**Appendix: Jupyter Notebook Code – work**

# -\*- coding: utf-8 -\*-

"""Taken Mind Internship - Assignment #1.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1JHNKUkEKOJhY4dkOy50w9nGIdmfZc8bp

### 0. Importing necessary libraries

"""

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from pandas import Series, DataFrame

"""# \*\*1. Numpy\*\*

### 1.1. Getting started

"""

list1 = [i\*\*1 for i in range(10)]

list2 = [i\*\*2 for i in range(10)]

list3 = [i\*\*3 for i in range(10)]

"""1.1.1. Creating 1D array"""

array\_1D = np.array(list1)

array\_1D

"""1.1.2. Creating 2D array"""

array\_2D = np.array([list1, list2])

array\_2D

"""1.1.3. Size"""

array\_1D.size

array\_2D.size

"""1.1.4. Shape"""

array\_1D.shape

array\_2D.shape

"""1.1.5. Reshape"""

array\_2D.reshape(5,4)

array\_2D.reshape(5,4).shape

"""1.1.6. Type"""

type(array\_2D)

"""1.1.7. Data Type"""

array\_2D.dtype

"""1.1.8. Zeros"""

zeros\_1D = np.zeros(10)

zeros\_1D

zeros\_2D = np.zeros([10,5])

zeros\_2D

"""1.1.9. Ones"""

ones\_1D = np.ones(10)

ones\_1D

ones\_4D = np.ones([6,5,4,3])

"""1.1.10. Empty"""

empty\_3D = np.empty([2,3,4])

empty\_3D

"""1.1.11. Identity matrix (EYE)"""

identityMatrix = np.eye(6)

identityMatrix

"""1.1.12. Arange"""

arangeArray = np.arange(2,50,2)

arangeArray

"""### 1.2. Scalar Operations"""

array1 = np.array([[i for i in range(5)], [i\*\*2 for i in range(5)], [i\*\*0.5 for i in range(5)]])

array1

array1 \* array1 == array1 \*\* 2

array1 \*\* 3

1/array1

array1 + 50

array1 \* 10

"""### 1.3. Array Indexes"""

array1

array1[0]

array1[0][0], array1[0][1], array1[0][2], array1[1][4], array1[2][3]

array1[1:3]

array1[:, 2:4]

array1[-1] = [1 for i in range(5)]

array1

array2 = array1.copy()

array2

"""### 1.4. Array Operations"""

array2

np.sqrt(array2)

np.max(array2)

np.min(array2)

np.argmax(array2)

np.argmin(array2)

np.exp(array2)

np.random.rand(5,3)

"""### 1.5. Loading & Saving Arrays to External memory"""

array1

np.save('SavedArray1', array1)

np.load("SavedArray1.npy")

np.savez('ArchivedArrays.npz', x=array1, y=array2)

archivedArrays = np.load('ArchivedArrays.npz')

archivedArrays['x']

archivedArrays['y']

np.savetxt('Array1.txt', array1, delimiter=',')

np.loadtxt('Array1.txt', delimiter=',')

"""### 1.6. Statistical Processing & Sketching Graphs"""

x = np.arange(5)

y = np.arange(3,6)

x

y

x1, y1 = np.meshgrid(x, y)

x1

y1

z = 2\*x1 + 3\*y1

plt.imshow(z)

plt.title("Plot - 2x1 + 3y1")

plt.colorbar()

z1 = np.cos(x1) + np.sin(y1)

plt.imshow(z1)

plt.title("Plot - cos(x1) + sin(y1)")

plt.colorbar()

plt.savefig("cos\_sin\_plots.png")

"""### 1.7. Relational & Conditional Operations """

array3 = array2.reshape(1,array2.size)

array3

np.where(array3>5, 999, array3)

np.sum(array3)

np.mean(array3)

np.var(array3)

np.std(array3)

array4 = np.array([True, False, False, False, True, True])

array4.any()

array4.all()

array3.sort()

array3

np.unique(array3)

np.in1d([0,1,2,5,7,10,16], array3)

"""# \*\*2. Pandas\*\*

### 2.1. Series

"""

s1 = Series([i+100 for i in range(6)])

s1

s1.values

s1.index

s1.index.values

array4

s2 = Series(array4, index=["A", "B", "C", "X", "Y", "Z"])

s2

s2['X']

s2[s2==True]

'B' in s2, 'D' in s2

s2.to\_dict()

s2 + s2 + s2 + s2

s2.name = "USERs"

s2.index.name = "Logged Status"

s2

"""### 2.2. Introduction to DataFrame"""

data = {'Name': ['Tom', 'Joseph', 'Krish', 'John'], 'Age': [20, 21, 19, 18]}

df1 = pd.DataFrame(data)

df1

df1.columns

df1.values

df1['Name']

df1['salary'] = np.nan

df1

df1[['Name', 'salary']]

df1.head(2)

df1.tail(2)

df1['rank'] = np.arange(4)

df1

del(df1['salary'])

df1

"""### 2.3. Access with Indexes"""

s3 = Series(["Uber", "Microsoft", "Netflix", "Amazon", "Meta", "Google"], index=['Aa', 'Bb', 'Cc', 'Dd', 'Ee', 'Ff'])

s3

s3.index

s3[-3:]

s3.rename({'Aa':'aaa'})

"""### 2.4. Reindex"""

s1

s4 = s3.reindex(['a', 'b', 'c', 'd', 'e', 'f'])

s4

s1

s1.reindex(range(12), method='ffill')

"""### 2.5. Dropping Values"""

s3

s3.drop('Dd')

df1

df1.drop(2)

df1

df1.drop('Age', axis=1)

"""### 2.6. Handling NaN"""

s5 = Series([100, 110, 120, np.nan, 130, np.nan, 140, np.nan], index=['A', 'a', 'B', 'b', 'C', 'c', 'D', 'd'])

s5

s5.isnull()

s5.dropna()

df2 = DataFrame(np.random.randn(10,5))

df2

df2.loc[1,1] = np.nan

df2.loc[1,2] = np.nan

df2.loc[2,1] = np.nan

df2.loc[2,1] = np.nan

df2.loc[5,4] = np.nan

df2.loc[7,1] = np.nan

df2

df2.dropna(how='all')

df2.dropna(thresh=4) # index=1 row is deleted

df2.fillna(0)

"""### 2.7. Selecting & Modifying Entries"""

s2

s2[['X', 'B']]

s2[s2==True]

df2

df2 < 0

df2[df2<0]

df2.loc[6]

df2.loc[5: ,[0,2,4]]

"""### 2.8. Coordinate & Regulate Data"""

Series1 = Series([100, 200, 300, 400, 500])

Series2 = Series([110, 210, 310, 410, 510])

Series1 + Series2

df1

df1 + df1

"""### 2.9. Ranking & Sorting"""

ser1 = Series([100, 200, 300, 400, 500], index=['b', 'c', 'a', 'e', 'd'])

ser1

ser1.sort\_index()

ser1.sort\_values()

ser1.rank()

"""### 2.10. Statistical Data Analysis & Graphs"""

lastArray = [[10, 20, 30, np.nan, 50],

[60, 70, 80, np.nan, 100],

[110, 120, np.nan, 140, 150],

[np.nan, 170, 180, np.nan, 200]]

df3 = DataFrame(lastArray)

df3

df3.sum()

df3.sum(axis=1)

df3.mean()

df3.var()

df3.min()

df3.max()

df3.cumsum()

df3.describe()

plt.plot(df3)

plt.legend(df3.columns, loc='lower right')

plt.savefig('df3 image')

df3[1].value\_counts().unique()