# **Documentation On Numpy And Pandas Library (Assignment1)**

This documentation is intended to give a general overview of how to install the Numpy and Pandas python library to your system and their basic or everyday used API's, Functions and Methods.

## **Installation Using PIP**

Once python and python package manager (pip) are installed in your system, you can then go ahead and use the package manager to install the modules on your system.

**Installing Numpy:** Open your terminal and type the command **pip install numpy**

**Installing Pandas:** Open your terminal and type the command **pip install pandas**

To check if installation went successfully, import the modules from your terminal.

### **Example**

*python -c "import numpy"* and *python -c "import pandas"*

If the two commands ran without any error then congratulations you have installed the libraries successfully else refer to the installation guide.

## **Numpy Basic Attributes and Functions**

Numpy is a library with N-dimensional array objects used in scientific computing, NumPy can also be used as an efficient multi-dimensional container of generic data.Arbitrary data-types can be defined.

Importing Numpy to your worksheet/notebook/script

In [4]:

**import** numpy **as** np

NOTE: the "as" after the numpy is a python keyword that allows you to use a shortened form of the library name.

In [5]:

*# Creating a python list that we are going to use throughout the examples*​

​

list\_one **=** [1,2,4,7]

list\_two **=** [7,1,3,5]

**array()** | The array function is used to create an array of any dimension you wish

In [6]:

*# Example*

np.array(list\_one) *# One dimensional array*

Out[6]:

array([1, 2, 4, 7])

In [7]:

np.array([list\_one,list\_two]) *# Two dimensional array*

​

*#NOTE: Here we had to use squre brackets to indicate that we joining the two list to form a two dimensional array*

Out[7]:

array([[1, 2, 4, 7],

[7, 1, 3, 5]])

### **Basic Attributes**

In this section, we will be talking about the basic Numpy attributes and how to use them.

**shape,** This attribute is used to get the dimensionality of any Numpy array. i.e to know the number of rows and columns the array has

In [8]:

*#Example*

np\_array **=** np.array(list\_one) *# Creating a Numpy array*

np\_array.shape

Out[8]:

(4,)

**dtype** The dtype attribute is used to determine the data type of a Numpy array

In [9]:

*#Example*

np\_array.dtype

Out[9]:

dtype('int64')

### **Basic Functions**

The below functions can be used to create a default Numpy array.

**zeros()** The function is used to create an array of zeros

In [10]:

*#Examples*

​

arr\_zeros **=** np.zeros(5)

arr\_zeros

Out[10]:

array([0., 0., 0., 0., 0.])

**ones()** This function is used to create an array filled with ones's

In [11]:

arr\_ones **=** np.ones(5)

arr\_ones

Out[11]:

array([1., 1., 1., 1., 1.])

**eye()** The eye function is used to create two or more dimensional arrays filled with zeros and ones but the ones are placed diagonal across the arrays

In [17]:

arr\_eye **=** np.eye(2)

arr\_eye

Out[17]:

array([[1., 0.],

[0., 1.]])

Notice how the values are placed diagonal, the **eye** function helps us create dimensional array with diagonal placement

In [18]:

arr\_eye **=** np.eye(5)

arr\_eye

Out[18]:

array([[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.]])

**arange()** This function is used to create an array from a range of numbers

In [22]:

arr\_arange **=** np.arange(1,10,2)

arr\_arange

Out[22]:

array([1, 3, 5, 7, 9])

### **Operators**

| These functions are used to add,multiple and divide two or more arguments(list/array) together by pairing their elements

**add()** The add function is used to add two or more list/arrays together element wise

In [13]:

*# Example*

np.add(list\_one,list\_two)

Out[13]:

array([ 8, 3, 7, 12])

**multiply()** The multiply function is used to multiply two or more arrays across each value in the array

In [14]:

np.multiply(list\_one,list\_two)

Out[14]:

array([ 7, 2, 12, 35])

**subtract()** The subtract function is used to subtract two or more arrays across each value in the array

In [23]:

np.subtract(list\_one,list\_two)

Out[23]:

array([-6, 1, 1, 2])

**true\_divide()** The divide function is used to divide two or more arrays across each value in the array

In [25]:

np.true\_divide(list\_one,list\_two)

Out[25]:

array([0.14285714, 2. , 1.33333333, 1.4 ])

**save()** The save function is used to save Numpy arrays for later use

In [26]:

np.save('saved array',arr\_arange)

**load()** The load function is used to load any Numpy save array back to the memory or to a new variable

In [29]:

new\_array **=** np.load('saved array.npy')

new\_array

Out[29]:

array([1, 3, 5, 7, 9])

## **Pandas Basic Attributes and Functions**

Pandas is an open source python library providing high-performance,easy-to-use data structures and data analysis tools.

The first thing to do is to import pandas to your working environment

In [57]:

**import** pandas **as** pd

**Note:** the alias used here is just my convention, you could choose to alias the module with any abbreviation of your choice.

### **Basic Functions**

These are the basic functions that we can use in Pandas to meet our computational needs

**read\_csv()** The read\_csv function is used to load csv files into our work environment as a dataframe

In [71]:

*#Example*

pd\_csv **=** pd.read\_csv("artifacts\_collections.csv")

pd\_csv

Out[71]:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **1.1** | **215** |
| 0 | 2 | 2 | 215.0 |
| 1 | 3 | 3 | 215.0 |
| 2 | 4 | 4 | 215.0 |
| 3 | 5 | 5 | 215.0 |
| 4 | 6 | 6 | 215.0 |
| ... | ... | ... | ... |
| 335317 | 335319 | 503157 | 868.0 |
| 335318 | 335320 | 503158 | 868.0 |
| 335319 | 335321 | 503159 | 868.0 |
| 335320 | 335322 | 503160 | 868.0 |
| 335321 | 335323 | 503161 | 994.0 |

335322 rows × 3 columns

**series()** The series function is used to create a column wise frame. i.e it is a one dimensional array in the form of an object.

In [ ]:

*#Example*

pd\_series **=** pd.Series(arr\_arange)

pd\_series

**isnull()** This is used to check if a series or dataframe has a null value which can be returned as a new series or dataframe with boolean values (True when a null value is meant and False when not null value is meant)

In [65]:

*#Example*

pd.isnull(pd\_series)

Out[65]:

0 False

1 False

2 False

3 False

4 False

dtype: bool

**notnull()** This is used to check if a series or dataframe has no null value which can be returned as a new series or dataframe with boolean values (True when a not null value is meant and False when a null value is meant)

In [66]:

*#Example*

pd.notnull(pd\_series)

Out[66]:

0 True

1 True

2 True

3 True

4 True

dtype: bool

**read\_clipboard()** This function is used to read data save in the computer clipboard into a variable as a dataframe

In [67]:

*#Example*

pd\_clipboard **=** pd.read\_clipboard()

pd\_clipboard

Out[67]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **2** | **anonymous, India** | **NULL** | **NULL.1** | **1** |
| 0 | 3 | Seidl-Geuthner, F., Paris, France | NaN | NaN | 1 |
| 1 | 5 | Franklin & Marshall College, Lancaster, PA, USA | NaN | NaN | 0 |
| 2 | 6 | Yale Babylonian Collection, New Haven, Connect... | NaN | NaN | 0 |
| 3 | 7 | Bruce, Mrs. James | NaN | NaN | 1 |
| 4 | 10 | Museum of Anthropology, University of British ... | NaN | NaN | 0 |
| ... | ... | ... | ... | ... | ... |
| 1082 | 1308 | Osaka Prefectural Chikatsu-Asuka Museum, Osaka... | NaN | NaN | 0 |
| 1083 | 1309 | anonymous, Los Angeles, California, USA | NaN | NaN | 1 |
| 1084 | 1310 | R.F. Harper collection | NaN | NaN | 1 |
| 1085 | 1311 | Idaho State Archives, Idaho State Historical S... | NaN | NaN | 0 |
| 1086 | 1312 | Couvent Saint-Étienne, Jerusalem | NaN | NaN | 0 |

1087 rows × 5 columns

**drop()** The drop function is used to drop or delete any series from a dataframe

In [72]:

*#Example*

pd\_series.drop(1)

pd\_series

Out[72]:

0 1

1 3

2 5

3 7

4 9

dtype: int64

**sort\_index** and **sort\_values** are used to sort a dataframe index wise or value wise respectively.

In [75]:

*#Example sort by index*

pd\_csv.sort\_index()

Out[75]:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **1.1** | **215** |
| 0 | 2 | 2 | 215.0 |
| 1 | 3 | 3 | 215.0 |
| 2 | 4 | 4 | 215.0 |
| 3 | 5 | 5 | 215.0 |
| 4 | 6 | 6 | 215.0 |
| ... | ... | ... | ... |
| 335317 | 335319 | 503157 | 868.0 |
| 335318 | 335320 | 503158 | 868.0 |
| 335319 | 335321 | 503159 | 868.0 |
| 335320 | 335322 | 503160 | 868.0 |
| 335321 | 335323 | 503161 | 994.0 |

335322 rows × 3 columns

In [84]:

*#Example sort by values*

pd\_csv.sort\_values(by **=** '1')

Out[84]:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **1.1** | **215** |
| 0 | 2 | 2 | 215.0 |
| 1 | 3 | 3 | 215.0 |
| 2 | 4 | 4 | 215.0 |
| 3 | 5 | 5 | 215.0 |
| 4 | 6 | 6 | 215.0 |
| ... | ... | ... | ... |
| 335317 | 335319 | 503157 | 868.0 |
| 335318 | 335320 | 503158 | 868.0 |
| 335319 | 335321 | 503159 | 868.0 |
| 335320 | 335322 | 503160 | 868.0 |
| 335321 | 335323 | 503161 | 994.0 |

335322 rows × 3 columns

### **Basic Attributes**

In this section, we will be looking at some Pandas Object attribute which can be applied on any Pandas object.

The **values** attribute, when applied to a pandas object, it returns the values contained in the dataframe or series

In [59]:

*#Example*

​

pd\_series.values

Out[59]:

array([1, 3, 5, 7, 9])

The **index** attribute is used to know/get the range of the dataframe or series indexing.

In [60]:

*#Example*

​

pd\_series.index

Out[60]:

RangeIndex(start=0, stop=5, step=1)

**NOTE:** The **index** attribute, when passed to the **Series** function as an argument, can be used to define a default indexing for that particular series being created.

In [61]:

*#Example*

​

pd\_index **=** pd.Series(arr\_arange,index**=**['A','B','C','D','E'])

pd\_index

Out[61]:

A 1

B 3

C 5

D 7

E 9

dtype: int64

**to\_dict** This attribute is used to convert a series to a python dictionary

In [64]:

*#Example*

pd\_dict **=** pd\_index.to\_dict()

pd\_dict

Out[64]:

{'A': 1, 'B': 3, 'C': 5, 'D': 7, 'E': 9}

**head()** and **tail()** | The tail and head attributes are used to gain insight from our dataset by printing the first five rows and last five rows from our dataset

In [69]:

*#Example head*

pd\_clipboard.head()

Out[69]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **2** | **anonymous, India** | **NULL** | **NULL.1** | **1** |
| 0 | 3 | Seidl-Geuthner, F., Paris, France | NaN | NaN | 1 |
| 1 | 5 | Franklin & Marshall College, Lancaster, PA, USA | NaN | NaN | 0 |
| 2 | 6 | Yale Babylonian Collection, New Haven, Connect... | NaN | NaN | 0 |
| 3 | 7 | Bruce, Mrs. James | NaN | NaN | 1 |
| 4 | 10 | Museum of Anthropology, University of British ... | NaN | NaN | 0 |

In [70]:

*#Example tail*

pd\_clipboard.tail()

Out[70]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **2** | **anonymous, India** | **NULL** | **NULL.1** | **1** |
| 1082 | 1308 | Osaka Prefectural Chikatsu-Asuka Museum, Osaka... | NaN | NaN | 0 |
| 1083 | 1309 | anonymous, Los Angeles, California, USA | NaN | NaN | 1 |
| 1084 | 1310 | R.F. Harper collection | NaN | NaN | 1 |
| 1085 | 1311 | Idaho State Archives, Idaho State Historical S... | NaN | NaN | 0 |
| 1086 | 1312 | Couvent Saint-Étienne, Jerusalem | NaN | NaN | 0 |

**sum()**, **min()**, **max()** and **describe()** are used to get statistical insight on our dataset.

In [85]:

*#Example of sum*

pd\_series.sum()

Out[85]:

25

In [86]:

*#Example of min (Minimum)*

pd\_series.min()

Out[86]:

1

In [87]:

*#Example of max (Maximum)*

pd\_series.max()

Out[87]:

9

In [88]:

*#Example of describe*

pd\_series.describe()

Out[88]:

count 5.000000

mean 5.000000

std 3.162278

min 1.000000

25% 3.000000

50% 5.000000

75% 7.000000

max 9.000000

dtype: float64