How to Import Pandas in Python?

The very first and the most important operation is to import Python Pandas library properly.

```
import pandas as pd
```

While importing Pandas library, we abbreviate it as ***pd***.

Panda Series Objects

A series object can contain any type of data, including mixed types. Now, let's have a look at how we can create series objects in Python Pandas with some examples.

Example1:

```
series1 = pd.Series([1, 2, 3, 4]
series1
```

```
Out[4]: 0    1
        1    2
        2    3
        3    4
        dtype: int64
```

Just to make sure that the object we have created just now is indeed a series object, we can use type() on the object.

```
type(series1)
```

```
Out[13]: pandas.core.series.Series
```

Example 2:

Further we can specify the index the of the series object as shown below:

```
series2 = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
series2
```

```
Out[12]: a    1
         b    2
         c    3
         d    4
         dtype: int64
```

Python Pandas DataFrames

DataFrames in Pandas are defined as 2-dimensional labeled data structures with columns of potentially different Python Data types

Creating a DataFrame from a list:

Let's take a list of integers and then create a DataFrame using that Python list:

```
list1 = [1,2,3,4,5]
list1
```

```
Out[18]: [1, 2, 3, 4, 5]
```

Creating a DataFrame from a list of lists:

Now, we will create a DataFrame using a list of lists.

```
list_of_lists = [['apple',10],['mango',12],['banana',13]]
df = pd.DataFrame(list_of_lists,columns=['fruit','count'],dtype=int)
df
```

Out[24]:

	fruit	count
0	apple	10
1	mango	12
2	banana	13

Creating a DataFrame from a dictionary:

We can also create DataFrames with the help of Python dictionaries

```
dict1 = {'fruit':['apple', 'mango', 'banana'],'count':[10,12,13]}
df = pd.DataFrame(dict1)
```

Out[26]:

	fruit	count
0	apple	10
1	mango	12
2	banana	13

Note: Since we are familiar with DataFrames and series objects, keep in mind that each column in a DataFrame is a series object.

Importing Data with Pandas in Python

Here, we will first read the data. The data is stored in a csv format, i.e., comma-separated values, where each record is separated by a comma ‘,’ and each row by a new line. Here are the first few rows of the Amazon_Products_Review.csv file:

```
,Product_Review_Phrase,Product_Title,Website_URL,Platform,Product_Rating,Product_Category,is_Amazon_Advantage_product,Product_Launch_Year,Product_Launch_Month,Product_Launch_Day
0,Amazing - Highest Selling,Ladela Bellies,http://www.amazon.com/ladela-bellies/p/itmeh4kmxght7tuc?pid=SHOEH4KM2W3Z6EH5,Mobile App,5,Footwear,FALSE,2016,7,12
1,Good - Average Selling,Bulaky vanity case Jewellery Vanity Case,http://www.amazon.com/bulaky-vanity-case-jewellery/p/itmdzy4ycfjhvctj?pid=VANDZY4YZFPEG85T,Website,3,Beauty and Personal Care ,FALSE,2016,7,12
2,Good - Average Selling,Roadster Men's Zipper Solid Cardigan,http://www.amazon.com/roadster-men-s-zipper-solid-cardigan/p/itmedfy7ueukSmfs?pid=CGNEDFY77SGZTEQ2,Website,3.6,Clothing ,FALSE,2016,7,12
3,Amazing - Highest Selling,"Camerii WM64 Elegance Analog Watch  - For Men, Boys",http://www.amazon.com/camerii-wm64-elegance-analog-watch-men-boys/p/itme6y6duhfcummh?pid=WATE6Y6D2MZHWGBZ,Mobile App,5,Others,FALSE,2016,7,12
4,Amazing - Highest Selling,"Colat COLAT_MW20 Sheen Analog Watch  - For Men, Women, Boys, Girls",http://www.amazon.com/colat-colat-mw20-sheen-analog-watch-men-women-boys-girls/p/itme2rx9jpyxcyqc?pid=WATE2RX9HHGBUQGA,Mobile App,5,Others,FALSE,2016,7,12
5,Amazing - Highest Selling,"Rorlig RR-028 Expedition Analog Watch  - For Men, Boys",http://www.amazon.com/rorlig-rr-028-expedition-analog-watch-men-boys/p/itmebyzgghrmgbnf?pid=WATEBYZGFCZPUJAR,Mobile App,5,Others,FALSE,2016,7,12
6,Amazing - Highest Selling,Lyc White Casual Boots,http://www.amazon.com/lyc-white-casual-boots/p/itme96sz8vzhacqv?pid=SHOE4UC3MJZF3VFJ,Mobile App,5,Footwear,FALSE,2016,7,12
7,Good - Average Selling,Fluid DMF-002-GR01 Digital Watch  - For Boys,http://www.amazon.com/fluid-dmf-002-gr01-digital-watch-boys/p/itme4cgcdzuhejcx?pid=WATE4CG2AQAGWGSF,Website,3.5,Others,TRUE,2016,7,12
8,Great - High Selling,Bruno Manetti Cannelita Boots,http://www.amazon.com/bruno-manetti-cannelita-boots/p/itmeygk5jzcbzeh7?pid=SHOEYGK5AYYBVXXH,Both,4,Footwear,FALSE,2016,7,12
```

As we can see, each row in the data represents a single product that was reviewed by Amazon. Here, we also have a leading column that contains row index values. Currently, we will not discuss about this column; later on, we’ll dive into what index values are. To work with data in Python, the first step is to import the file into a Pandas DataFrame. A DataFrame is nothing but a way to represent and work with tabular data, and tabular data has rows and columns.

Our file is of .csv format. So, **pd.read_csv()** function is going to help us read the data stored in that file. This function will take the input as a csv file and return the output as a DataFrame.

```
import pandas as pd
Product_Review=pd.read_csv(“Amazon_Products_Review.csv”)
```

Let’s inspect the type of **Product_Review** using the **type()** function.

```
type(Product_Review)
```

```
Out[31]: pandas.core.frame.DataFrame
```

For file types other than .csv, the importing conventions are mentioned below:

- pd.read_table("filename")
- pd.read_excel("filename")
- pd.read_sql(query, connection_object)
- pd.read_json(json_string)

Working with DataFrames:

Now that our DataFrame is ready, let's have a look at some of the operations in Pandas.

- head():This prints the first five rows of the DataFrame.
- tail():This prints the last five rows of the DataFrame.
- shape: This gives the number of rows and columns. In our DataFrame, we have 1,840 rows and 11 columns.
- info: This gives us the information of the index, data type, and memory of the DataFrame.
- describe: This gives summary statistics for numerical columns.

One of the big advantages of Python Pandas over Python NumPy is that Pandas in Python allows us to have columns with different data types. Here, in Product_Review, we have columns that store float values like Product_Rating, string values like Product_Review_Phrase, and integers like Product_Launch_Year.

Now, as the data is read properly, we will work on indexing Product_Review so that we can get the rows and columns as per our requirement.

Indexing DataFrames with Pandas in Python

Now, let's say, we want to select and have a look at a chunk of data from our DataFrame. There are two ways of achieving the same: First, selecting by position and, second, by label.

Selecting by Position: Using ***iloc***, we can retrieve rows and columns by position. Here, we need to specify the positions of rows and columns.

Example 1:
Suppose, we want only the first column out of the DataFrame. Then, we would use ***iloc*** on the DataFrame as shown below:

```
Product_Review.iloc[:,0]
```

This snippet of code shows that we want to have a look at all rows of the first column. Keep in mind that the position of the first column (or the first row) always starts with 0. As we needed all the rows, we specified just a colon (:) without mentioning any position.

Example 2:

Again, imagine that we want to have a look at the first five rows of the fourth column. We need to specify the position of the rows as **0:5**, which means that we want to view the first five rows from the position 0 to the position 4 (note that the position 5 is excluded here). Also, instead of writing 0:5, we can leave off the first position value and write like **:5** (but if we write **0:,** it means the '0^{th}' position to the last position).

```
Product_Review.iloc[0:5,4]
```

Also, in the example show above, instead of writing 0:5 we can leave off the first position value, like :5. This has the same meaning. But if we write **0:** this means indexing of 0th position to last position.
Similar examples:

- iloc[:,:] – To view the entire DataFrame
- iloc[6:,4:] – To view from Row 6 and Column 4 onward

Now, let's update our DataFrame by removing the first column, which contains no useful **information:**

```
Produt_Reviews= Product_Reviews.iloc[:,1]
Product_Reviews.head()
```

```
Out[108]: 0    Amazing - Highest Selling
          1       Good - Average Selling
          2       Good - Average Selling
          3    Amazing - Highest Selling
          4    Amazing - Highest Selling
          Name: Product _Review_Phrase, dtype: object
```

Now, since we are aware of the **NumPy** indexing methodologies, we can notice that they are quite similar to Pandas indexing by position. But unlike NumPy, each of the columns and rows in Pandas has a label. Yes, selecting by position is easy. But for large DataFrames, keeping track of columns and their corresponding positions becomes complicated. That's when our next method of indexing comes in handy.

Selecting by Label:

The second method is selecting by the label of columns. The ***.loc** method* allows us to index using labels instead of positions. Let's illustrate this method with some examples.

Example 1:

(Selecting some rows of one column)

Displaying the first five rows of the ***Product_Title*** using the ***.loc*** method:

```
Prodcut_Reviews.loc[:5,”Product_Title”]
```

Example 2:

(selecting some rows of more than one column)

Displaying the first five rows of the columns, ***Product_Title*** and ***Product_Rating***

```
Product_Reviews.loc[:5,”Product_Title”,”Product_Rating”]
```

Sorting DataFrames with Pandas in Python

Apart from indexing, another very simple yet useful feature offered by Pandas in Python is the sorting of DataFrames. To get a clear idea of the sorting feature, let’s look at the following examples.

Sorting the DataFrame based on the values of a column:

Say, we want to sort the Product_Rating column by its values.

```
Product_Review.sort_values(by=‘Product_Rating’)
```

Now, if we want to sort the Product_Rating column by its values in the descending order.

```
Product_Review.sort_values(‘Product_Rating’, ascending=False)
```

Pandas in Python DataFrame Methods

There are some special methods available in Pandas in Python which makes our calculation easier. Let’s apply those methods in our ***Product_Review*** DataFrame.

1) Mean of all columns in our DataFrame

```
Product_Review.mean()
```

```
Out[54]: Unnamed: 0                        919.500000
         Product_Rating                      3.802989
         is_Amazon_Advantage_product         0.164674
         Product_Launch_Year              2016.000000
         Product_Launch_Month                7.000000
         Product_Launch_Day                 12.000000
         dtype: float64
```

2) Median of each column in our DataFrame

```
Product_Review.median()
```

```
Out[57]: Unnamed: 0                          919.5
         Product_Rating                        4.0
         is_Amazon_Advantage_product           0.0
         Product_Launch_Year                2016.0
         Product_Launch_Month                  7.0
         Product_Launch_Day                   12.0
         dtype: float64
```

3) Standard deviation of each column in our DataFrame

```
Product_Review.std()
```

```
Unnamed: 0                          531.306566
Product_Rating                        1.263654
is_Amazon_Advantage_product           0.370987
Product_Launch_Year                   0.000000
Product_Launch_Month                  0.000000
Product_Launch_Day                    0.000000
dtype: float64
```

4) Maximum value of each column in our DataFrame

```
Product_Review.max()
```

```
Unnamed: 0                                                          1839
Product _Review_Phrase                                 Great - High Selling
Product_Title              zDelhi.com Car Washer Z1 Ultra High Pressure W...
Website_URL                http://www.amazon.com/zyxel-vmg1312-b10a-vdsl2...
Platform                                                          Website
Product_Rating                                                          5
Product_Category                                                   Others
is_Amazon_Advantage_product                                          True
Product_Launch_Year                                                  2016
Product_Launch_Month                                                    7
Product_Launch_Day                                                     12
dtype: object
```

5) Minimum of each column in our DataFrame

```
Product_Review.min()
```

```
Unnamed: 0                                                             0
Product _Review_Phrase                                 Amazing - Highest Selling
Product_Title                    3a Autocare 3D MAT Car Mat Suzuki New Swift
Website_URL                http://www.amazon.com/3a-autocare-3d-mat-car-s...
Platform                                                            Both
Product_Rating                                                         1
Product_Category                                                Automotive
is_Amazon_Advantage_product                                          False
Product_Launch_Year                                                  2016
Product_Launch_Month                                                    7
Product_Launch_Day                                                     12
dtype: object
```

6) Number of non-null values in each DataFrame column

```
Product_Review.count()
```

```
Unnamed: 0                      1840
Product _Review_Phrase          1840
Product_Title                   1840
Website_URL                     1840
Platform                        1840
Product_Rating                  1840
Product_Category                1840
is_Amazon_Advantage_product     1840
Product_Launch_Year             1840
Product_Launch_Month            1840
Product_Launch_Day              1840
dtype: int64
```

7) Summary statistics for numerical columns

```
Product_Review.describe()
```

	Unnamed: 0	Product_Rating	Product_Launch_Year	Product_Launch_Month	Product_Launch_Day
count	1840.000000	1840.000000	1840.0	1840.0	1840.0
mean	919.500000	3.802989	2016.0	7.0	12.0
std	531.306566	1.263654	0.0	0.0	0.0
min	0.000000	1.000000	2016.0	7.0	12.0
25%	459.750000	3.000000	2016.0	7.0	12.0
50%	919.500000	4.000000	2016.0	7.0	12.0
75%	1379.250000	5.000000	2016.0	7.0	12.0
max	1839.000000	5.000000	2016.0	7.0	12.0

Mathematical Operations in Pandas Python

We can also perform mathematical operations on series objects or DataFrame objects.

For example, for dividing every value in the Product_Rating column by 2, we use the following code:

```
Product_Review[“Product_Rating”] /2
```

```
Out[83]: 0    2.5
         1    1.5
         2    1.8
         3    2.5
         4    2.5
         Name: Product_Rating, dtype: float64
```

Note: All common mathematical operators that work in Python, like +, −, *, /, and ^, will also work in a DataFrame or a series object.

Filtering DataFrames in Python Panda

Now that we have learned about how to do mathematical operations in Pandas, let's have a look at the filtering methods available in Pandas Python and use them in our DataFrame.

Say, we want to find footwear that has a Product_Rating greater than 3.
First, let us generate a Boolean series with our filtering condition and see the first five results.

```
filter1 = Product_Review["Product_Rating"] > 3
filter1.head()
```

```
Out[84]: 0     True
         1    False
         2     True
         3     True
         4     True
         Name: Product_Rating, dtype: bool
```

Now that we have got the Boolean series, we use it to select only rows in a DataFrame where the series contains the value True so that we get the rows in Product_Review where Product_Rating is greater than 3.

```
filtered_new = Product_Review[filter1]
filtered_new.head()
```

	Unnamed: 0	Product _Review_Phrase	Product_Title	Website_URL	Platform	Product_Rating	Product_Category	is_Amazon_Advantage_product	I
0	0	Amazing - Highest Selling	Ladela Bellies	http://www.amazon.com/ladela-bellies/p/itmeh4k...	Mobile App	5.0	Footwear	False	
2	2	Good - Average Selling	Roadster Men's Zipper Solid Cardigan	http://www.amazon.com/roadster-men-s-zipper-so...	Website	3.6	Clothing	False	
3	3	Amazing - Highest Selling	Camerii WM64 Elegance Analog Watch - For Men,...	http://www.amazon.com/camerii-wm64-elegance-an...	Mobile App	5.0	Others	False	
4	4	Amazing - Highest Selling	Colat COLAT_MW20 Sheen Analog Watch - For Men...	http://www.amazon.com/colat-colat-mw20-sheen-a...	Mobile App	5.0	Others	False	
5	5	Amazing - Highest Selling	Rorlig RR-028 Expedition Analog Watch - For M...	http://www.amazon.com/rorlig-rr-028-expedition...	Mobile App	5.0	Others	False	

Let’s make it a bit complicated by adding more than one condition. Since we wanted to have a look at the footwear that has a Product_Rating greater than 3, we will now add our second condition in the Product_Category column of the DataFrame.

```
filter2 = (Product_Review["Product_Rating"] > 3) & (Product_Review["Product_Category"] == "Footwear")
filtered_review = Product_Review[filter2]
filtered_review.head(
```

	Unnamed: 0	Product _Review_Phrase	Product_Title	Website_URL	Platform	Product_Rating	Product_Category	is_Amazon_Advantage_product
0	0	Amazing - Highest Selling	Ladela Bellies	http://www.amazon.com/ladela-bellies/p/itmeh4k...	Mobile App	5.0	Footwear	False
6	6	Amazing - Highest Selling	Lyc White Casual Boots	http://www.amazon.com/lyc-white-casual-boots/p...	Mobile App	5.0	Footwear	False
8	8	Great - High Selling	Bruno Manetti Cannelita Boots	http://www.amazon.com/bruno-manetti-cannelita-...	Both	4.0	Footwear	False
11	11	Good - Average Selling	Kielz Ladies Boots	http://www.amazon.com/kielz-ladies-boots/p/itm...	Website	3.5	Footwear	False
13	13	Good - Average Selling	Kielz Ladies Boots	http://www.amazon.com/kielz-ladies-boots/p/itm...	Website	3.5	Footwear	False