What is Pandas?
Pandas is a Python library. Pandas is used to analyze data.
Pandas is a Python library used for working with data sets.
It has functions for analyzing, cleaning, exploring, and manipulating data.
The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?
Pandas allows us to analyze big data and make conclusions based on statistical theories.
Pandas can clean messy data sets, and make them readable and relevant.
Relevant data is very important in data science.

What Can Pandas Do?
1 High performance data analysis tool
2 Working with large data sets
3 Supports different file formats
4 Working with missing data
5 Data represents in tabular format
6 Indexing-Slicing-Subsetting the large data sets
7.Easily merge and join two different data sets
8. Reshape the datasets.

Installation of Pandas
If you have Python and PIP already installed on a system, then installation of Pandas is very easy.

Install it using this command:
```
$pip install pandas
```

Syntax:
```
import pandas
```

Checking Pandas Version
The version string is stored under __version__ attribute.

Example
```
import pandas as pd
print(pd.__version__)
```

Show installed versions:
Need to know the versions of pandas' dependencies.

Example:
```
import pandas as pd
print(pd.show_versions())
```

Pandas as pd ==> Pandas is usually imported under the pd alias.
alias: In Python alias are an alternate name for referring to the same thing.

Syntax:
```
import pandas as pd
```

Pandas deals with the following three data structures:

The best way to think of these data structures is that the higher dimensional data structure is a container of its lower dimensional data structure.

Data Structure	Dimensions	Description
Series	1	1D homogeneous array, size-immutable.
Data Frames	2	2D size-mutable tabular structure(Rows&Cols)
Panel	3	Multi-Dimensional(3D), size-mutable array.

NOTE:
DataFrame is widely used and one of the most important data structures. Panel is used much less.

Pandas Series
A Pandas Series is like a column in a table. It is a one-dimensional array holding data of any type. We can easily convert the list, tuple, and dictionary into series using "series' method.

1. Empty Series
2. Series using Arrays
3. Series using Lists
4. Series using Dictionary

1. Empty Series
We can easily create an empty series in Pandas which means it will not have any value.

Syntax:
<series object> = pandas.Series()

Example:
```
import pandas as pd
PySer = pd.Series()
print (PySer)  #Series([], dtype: float64)
```

Creating a Series using inputs:
1 Array
2 Scalar value
3. List
4 Dict

Creating Series from Array:
We have to import the numpy module and then use array() function in the program.

Example:
```
import pandas as pd
import numpy as np
PyData = np.array(['P','a','n','d','a','s'])
PySer = pd.Series(PyData)
print(PySer)
```

Scalar value
If we take the scalar values, then the index must be provided. The scalar value will be repeated for matching the length of the index.

Example:

```
import pandas as pd
PySer= pd.Series(4, index=[0, 1, 2, 3])
print(PySer)
print(type(PySer))
```

Create a Series from List

Example

Create a simple Pandas Series from a list:

```
import pandas as pd
PyArr=[1, 7, 2]
PyVar = pd.Series(PyArr)
print(PyVar)
```

Create a Series from dict

If the dictionary object is being passed as an input and the index is not specified, then the dictionary keys are taken in a sorted order to construct the index.

Example:

```
import pandas as pd
PyDict = {'PY' : 100, 'DS' : 91, 'ML' : 62}
PySer = pd.Series(PyDict)
print (PySer)
```

Example

```
import pandas as pd
Calories = {"day1": 420, "day2": 380, "day3": 390}
PyVar = pd.Series(Calories)
print(PyVar)
```

Example:

```
import pandas as pd
Calories = {"day1": 420, "day2": 380, "day3": 390}
PyVar = pd.Series(Calories, index = ["day1", "day2"])
print(PyVar)
```

Accessing data from series with Position:

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc. This label can be used to access a specified value.

Example

```
import pandas as pd
PyArr=[1, 7, 2]
PyVar = pd.Series(PyArr)
print(PyVar)
print(PyVar[0])
print(PyVar[1])
```

With the index argument, you can name your own labels.

Example:

```
import pandas as pd
PyArr=[1, 7, 2]
PyVar = pd.Series(PyArr, index = ["x", "y", "z"])
print(PyVar)
```

Example:

```
import pandas as pd
PyArr=[1, 7, 2]
PyVar=pd.Series(PyArr, index = ["x", "y", "z"])
print(PyVar["y"])
```

Series object attributes

The Series attribute is defined as any information related to the Series object such as size, datatype. etc.

Attributes	Description
Series.index	Defines the index of the Series.
Series.shape	It returns a tuple of shape of the data.
Series.dtype	It returns the data type of the data.
Series.size	It returns the size of the data.
Series.empty	It returns True if Series object is empty, otherwise returns false.
Series.hasnans	It returns True if there are any NaN values, otherwise returns false.
Series.nbytes	It returns the number of bytes in the data.
Series.ndim	It returns the number of dimensions in the data.

Example:

```
import pandas as pd
PySer=pd.Series(data=[11.2,18.6,22.5], index=['a','b','c'])
print(PySer.index)
print(PySer.values)
print(PySer.shape)
print(PySer.dtype)
print(PySer.size)
print(PySer.empty)
print(PySer.hasnans)
print(PySer.nbytes)
print(PySer.ndim)
```