

Chapter 1: Python Pandas-I

PANDAS

- Pandas or Python Pandas is Python's library for data analysis.
- Pandas has derived its name from "Panel data system", (term used for structured data sets).
- It is useful for data analysis and manipulation.

Data analysis: refers to process of evaluating big data sets using statistical tools to discover useful information and conclusions to support business decision –making.

- Pandas provide powerful and easy-to-use data structures, as well as the means to quickly perform operations on these structures.

WHY Pandas ?

It is capable of many tasks including

- It can read or write in many different data formats(integer, float, double etc)
- It can calculate in all ways data is organised i.e across rows and columns.
- It can easily select subsets of data from bulky data sets and even combine multiple datasets together.
- It has functionality to find and fill missing data
-

DATA STRUCTURE:-

It refers to specialized way of storing and organizing data in a computer so that it can be accessed and we can apply a specific type of functionality on them as per requirements.

Pandas deals with 3 data structure

1. Series
2. Data Frame
3. Panel

We are having only **Series** and **data frame** in our syllabus

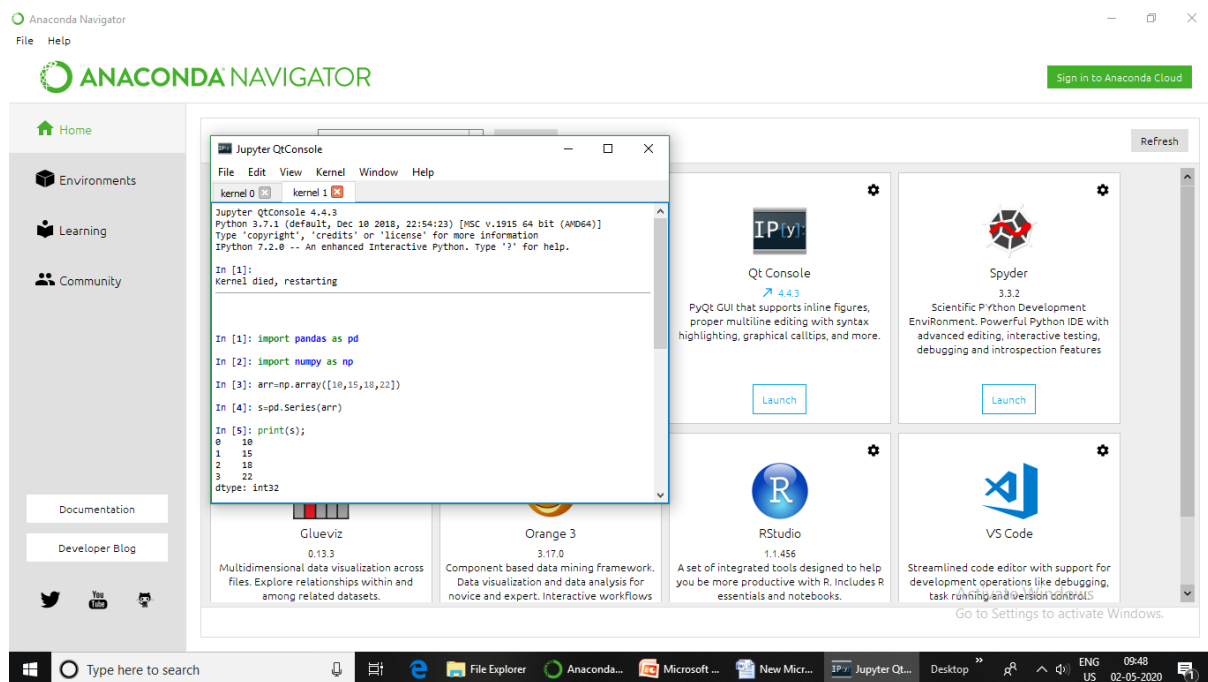
Series :- Series is a one-dimensional array like structure with homogeneous data(meaning –of the same kind), which can be used to handle and manipulate data. It is special because of its index attribute, which has incredible(Unbelievable) Functionality and is heavily mutable.

It has two parts:--

1. Data part(An array of actual data)

Index	Data
0	10
1	15
2	18
3	22

- Pandas data structures is enhanced versions of NumPy structured array.
- FOR WORKING IN PANDAS WE GENERALLY IMPORT BOTH **PANDAS** AND **NUMPY LIBRARIES**
- **NumPy** is used because in Pandas' some function return result in form of NumPy arrays(**Pandas library's data manipulation capabilities have been built over NumPy library**)



```
import pandas as pd
S=pd.Series([10,15,18,22])

print (S)
```

CREATION OF SERIES FROM

- Nddarray

```
import pandas as pd
import numpy as np
Arr=[31,28,31,30]
```

Output

print (obj3)

31

28

CREATION OF SERIES FROM

- Pyhthon Dictionary

```
import pandas as pd
obj=pd.Series({'Jan':31,'Feb':28,'Mar':31})
print(obj)
```

Output

print (obj)

Feb 28

Jan 31

CREATION OF SERIES FROM

- Scalar Value

```
import pandas as pd

obj=pd.Series(15, index=range(1,6,2))

ser=pd.Series('Yet to
```

Output

print (obj)

1 15

3 15

Output

print (ser)

Indore Yet to start

Delhi Yet to start

Mathematical function

- The Series() allows us to define a function that can calculate values for data

sequence.

- eg

```
import pandas as pd
```

```
import numpy as np
```

```
a=np.arange(9,13)
```

```
print (a)
```

```
[ 9 10 11 12]
```

```
S=pd.Series(index=a,data=a*2)
```

```
S
```

```
Out[6]:
```

```
9    18
```

```
10   20
```

11 22

12 24

dtype: int32

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Documentation

Developer Blog

Applications on base (root) Channels Refresh

Jupyter QtConsole

File Edit View Kernel Window Help

kernel 0 kernel 1

```
Jupyter QtConsole 4.4.3
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.2.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import pandas as pd
In [2]: import numpy as np
In [3]: a=np.arange(9,13)
In [4]: print (a)
[ 9 10 11 12]
In [5]: S=pd.Series(index=a,data=a*2)
In [6]: S
Out[6]:
9    18
10   20
11   22
12   24
dtype: int32
In [7]:
```

Qt Console 4.4.3

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch

Spyder 3.3.2

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch

RStudio 1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code

Streamlined code editor with support for development operations like debugging, task running and version control.

Go to Settings to activate Windows.

Glueviz 0.13.3

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3 3.17.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

Type here to search

IP CLASS xii

Microsoft...

Anacond...

IP Jupyter Q...

Typing M...

Desktop

ENG US

16:25

05-05-2020

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Documentation

Developer Blog

Applications on base (root) Channels Refresh

Jupyter QtConsole

File Edit View Kernel Window Help

```
Jupyter QtConsole 4.4.3
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.2.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import pandas as pd
In [2]: import numpy as np
In [3]: Lst=[9,10,11,12]
In [4]: obj1=pd.Series(data=(2*Lst))
In [5]: obj1
Out[5]:
0    9
1   10
2   11
3   12
4    9
5   10
6   11
7   12
dtype: int64
In [6]:
```

Qt Console 4.4.3

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch

Spyder 3.3.2

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch

RStudio 1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code

Streamlined code editor with support for development operations like debugging, task running and version control.

Go to Settings to activate Windows.

Glueviz 0.13.3

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3 3.17.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

Type here to search

IP CLASS xii

Microsoft...

Anacond...

IP Jupyter Q...

Typing M...

Desktop

ENG US

16:25

05-05-2020

SERIES ATTRIBUTES

- When we create Series all information related to it (such as size, its datatype etc) is available through attributes .
- We can use these attributes in the following format to get information about the Series object.
`<series object>.<attribute name>`

ATTRIBUTE	DESCRIPTION
Series.itemsize	Return the size of the dtype of the item of the underlying data <code>s.itemsize</code> 4
Series.nbytes	Return the number of bytes in the underlying data <code>print(s.nbytes)</code> 16 (nbytes is equal to the size*itemsize)
Series.ndim	Return the number of dimensions of the underlying data <code>s.ndim</code> Out[6]: 1

Series.itemsize	Return the size of the dtype of the item of the underlying data s.itemsize 4
Series.nbytes	Return the number of bytes in the underlying data print(s.nbytes) 16 (nbytes is equal to the size*itemsize)
Series.ndim	Return the number of dimensions of the underlying data s.ndim Out[6]: 1

After creating Series type object, we can access it in many ways. We can access its

- indexes separately
- Its data separately
- Access individual elements and slices

1. Accessing individual elements

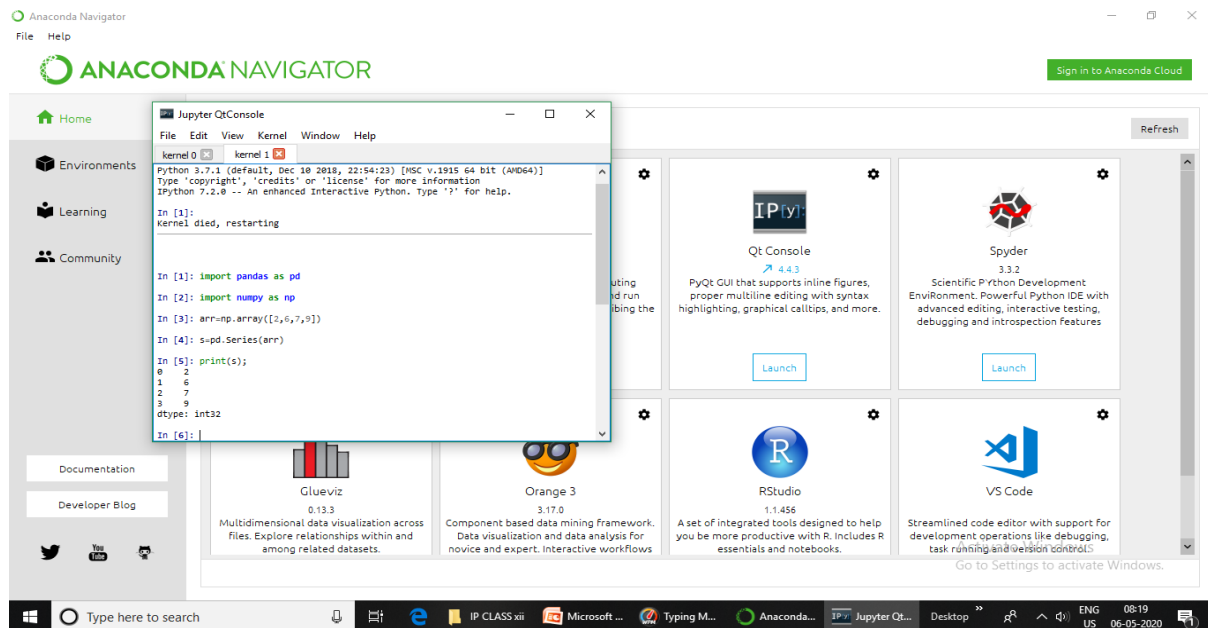
- To access individual elements of a series object, we can give index in square brackets along with its name

eg Series object name [valid index]

2. Extracting Slices from Series Object

- We can extract slices too from a Series object .
- Slicing is a powerful way to retrieve subsets of data from a pandas object.
- Slicing takes place position wise and not the index wise in a series object.

Eg obj1	position
	0
	1



S[1:]

S[1:3]

After creating Series type object, we can perform various types of operations on pandas SERIES OBJECTS.

- **Modifying Elements of Series Object**
- The **head()** and **tail()** functions
- **Vector Operations on Series Objects**
- **Arithmetic on Series objects**
- **Filtering Entries**

1. Modifying Elements of Series Object

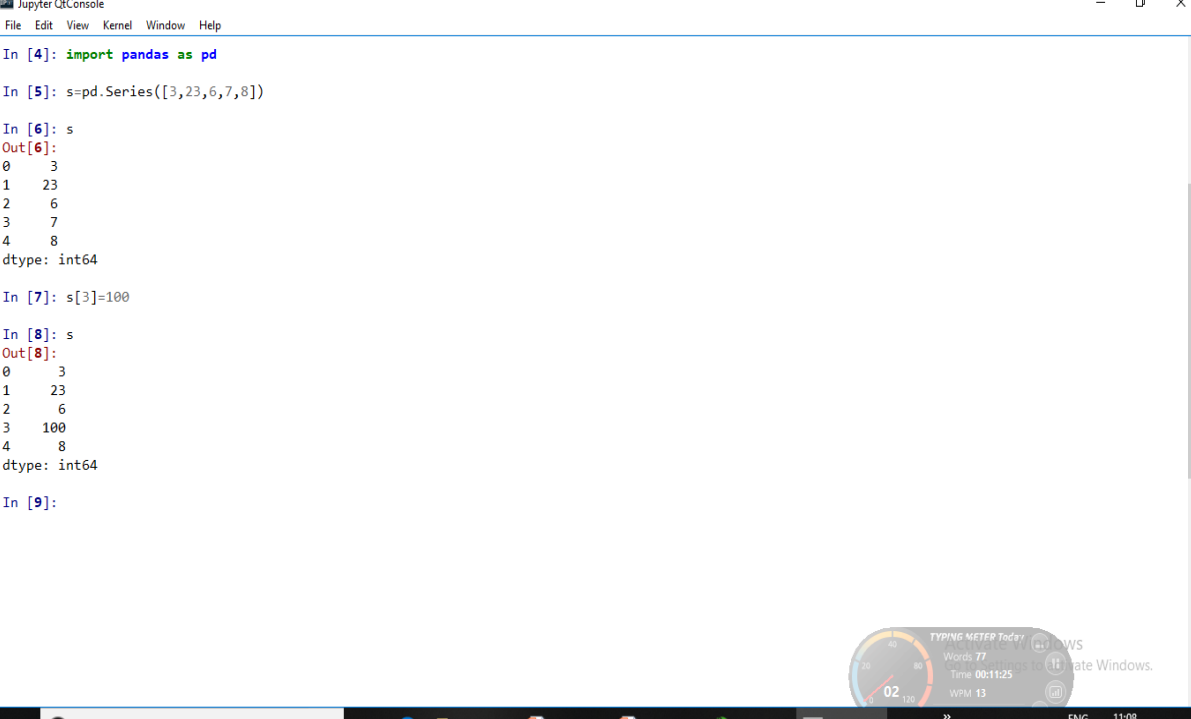
The data values of a Series object can be easily modified through item assignment

eg (a) `Series object[index]= newvalue`

above assignment will change the data value of the given index in Series object.

(b) `Series object[star:stop]=newvalue`

above assignment will replace all the values falling in given slice



The screenshot shows a Jupyter QtConsole window with the following code and output:

```
In [4]: import pandas as pd
In [5]: s=pd.Series([3,23,6,7,8])
In [6]: s
Out[6]:
0    3
1   23
2    6
3    7
4    8
dtype: int64
In [7]: s[3]=100
In [8]: s
Out[8]:
0    3
1   23
2    6
3  100
4    8
dtype: int64
In [9]:
```

The console window has a menu bar (File, Edit, View, Kernel, Window, Help) and a toolbar. The Windows taskbar at the bottom shows the search bar, task view button, and several open applications including IP CLASS xii, may9, may8, Anacond..., Jupyter Q..., and Desktop. A typing meter overlay is visible in the bottom right corner of the console window.

head():- It is used to access the first **n rows** of a Series.

pandas object.head()

tail():- returns last n rows from a pandas object.

pandas object.head()

```
import pandas as pd
s=pd.Series([2,3,21,12,31,7,8])
s
Out[3]:
0    2
1    3
2   21
3   12
```

```
s
Out[7]:
0    2
1    3
2   21
3   12
4   31
5    7
6    8
```

```
import pandas as pd
s=pd.Series([2,3,21,12,31,7,8])
s
Out[3]:
0    2
1    3
2   21
3   12
4   31
5    7
```

```
import pandas as pd

s=pd.Series([2,3,21,12,31,
7,8])

s

Out[3]:

0    2
1    3
2   21
3   12
```

```
s+2
Out[10]:
0    4
1    5
2   23
3   14
4   33
5    9
6   10
dtype: int64
```

```
s*3
Out[11]:
0    6
1    9
2   63
3   36
4   93
5   21
6   24
```

```
import pandas as pd

s=pd.Series([2,3,21,12,31,7,8])

s

Out[3]:

0    2
1    3
2   21
3   12
4   31
-    -
```

```
s>15
Out[12]:
0    False
1    False
2     True
3    False
4     True
5    False
6    False
```

```
s[s>15]
Out[17]:
2   441
3   144
4   961
5    49
6    64
dtype:
```

```
import pandas as pd

s=pd.Series([2,3,4,1])

s2=pd.Series([6,7,8,9])

s+s2

Out[25]:

0    8
1   10
```