Unit 1: Data Handling using Pandas

Introduction

Python Pandas Pandas is a software library written for the Python programming language for data manipulation and analysis. Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

There are 3 well-established python libraries namely NumPy, Pandas and Matplotlib specially for scientific and analytical use. These libraries allow us to manipulate, transform and visualise data easily and efficiently.

Using the Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data. These steps are- load, prepare, manipulate, model and analyse

Benefits of Pandas

The benefits of pandas over using the languages are

- Data representation: It can easily represent data in a form naturally suited for data analysis through its DataFrame and Series data structures in a concise manner. Doing the equivalent in C/C++ or Java would require many lines of custom code, as these languages were not built for data analysis but rather networking and kernel development.
- Clear code: The clear API of the Pandas allows you to focus on the core part of the code. So, it provides clear code.

Matplotlib

It is an amazing visualization library in Python that used for 2D plots of arrays. It is a multi-platform data visualization library which build NumPy arrays. Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers and various graphical user interface toolkits. To get matplotlib up and running in our environment, we need to import it. import matplotlib.pyplot as plt.

Data structures in Pandas:

Data structure is defined as the storage and management of the data for its efficient and easy access in the future where the data is collected, modified and the various types of operations are performed on the data respectively. Pandas provides two data structures for processing the data, which are explained below:

(1) Series: It is one dimensional object similar to an array, list or column in a table. It will assign a labelled index to each item in the series. By default, each item will receive an index label from 0 to N, where N is the length of the series minus one.

(2) DataFrame: It is a tabular data structure comprised of rows and columns. DataFrame is defined as a standard way to store data which has two different indexes i.e., row index and column index.

Know the Terms:

Pandas: The Pandas is a high-performance open source library for data analysis in Python.

Matplotlib: It is a visualization library in Python that used for 2D plots of arrays.

Series: It is a one-dimensional array containing a sequence of values. Each value has a data label associated with it also called its index. DATA HANDLING USING PANDAS-I 7

Selection: This function returns data corresponding to axis labels matching criteria.

Indexing: This function is used to get or set the index labels of the given series object.

Slicing: Slicing is a powerful approach to retrieve subsets of data from a Pandas object.

Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56, … Pandas Series is a one-dimensional labelled array capable of holding any data type (integer values, string values, double value and more). A Series represents a single column in memory. Series can only contain single list with index, whereas DataFrames can be made of more than one series or we can say that a DataFrames is a collection of series.

Key Points of Series

● Homogeneous data

● Size Immutable

● Values of Data Mutable

A pandas Series can be created using the following constructor − pandas. Series (data, index, dtype, copy)

CREATION OF SERIES

A series can be created using various inputs like:

● List

● Tuple

● Dictionary

● Array

● Scalar value or constant

Create an Empty Series

A basic series, which can be created is an Empty Series.

1. Write a program to create an empty series

#import the pandas library and aliasing as pd

Ans.

```
import pandas as pd

s =pd.Series()

print (s)
```

Output is as follows: Series ([], dtype: float64)

Create a Series from List

2. Write a program to create a series by given list [‘red’,’green’,’blue’].

Ans.

```
import pandas as pd
c=[‘red’,’green’,’blue’]
p=pd.Series(c)
print (p)
```

OR

```
import pandas as pd
c=[‘red’,’green’,’blue’]
p=pd.Series(data=c)
print (p)
```

OR

```
import pandas as pd
p=pd.Series([‘red’,’green’,’blue’])
print (p)
```

Output:

```
0   red
1   green
2   blue
```

We did not pass any index, so by default, it assigned the indexes ranging from 0 to len(data)-1, i.e., 0 to 2.

3. Modify the above program and change the index values to r, g, b

Ans.

```
import pandas as pd

c=[‘red’,’green’,’blue’]

p=pd.Series(c,index=[‘r’,’g’,’b’])
```

```
print (p)
```

OR

```
import pandas as pd
p=pd.Series([‘red’,’green’,’blue’],index=[r,g,b])
print (p)
```

OR

```
import pandas as pd
c=[‘red’,’green’,’blue’]
p=pd.Series(c,[r,g,b])
print (p)
```

OR

```
import pandas as pd
c=[‘red’,’green’,’blue’]
p=pd.Series(data=c,index=[r,g,b])
print (p)
```

Output:

```
r       red
g       green
b       blue
```

Create a Series from Tuple

4. Write a program to create a series from a given Tuple data data=(‘1’,’Aman’,86.3,’A’).

Ans:

```
import pandas as pd
data=(‘1’,’Aman’,86.3,’A’)
p=pd.Series(data)
print (p)
```

Output

```
0  1
1  Aman
2  86.3
3  A
```

Create a Series from Dictionary

A dictionary can be passed as input and if no index is specified, then keys of the dictionary are used to represent the index of the Series.

5. A dictionary data ={‘a’:0.,’b’:1.,’c’:2} is given.
   Write a program to create series from dictionary data.

Ans.

```
import pandas as pd

data ={‘a’:0.,’b’:1.,’c’:2.}

s =pd.Series(data)

print(s)
```

Its output is as follows –

```
a 0.0

b 1.0

c 2.0

dtype: float64
```

If index is passed, the values will be displayed in the same sequence as index values are passed.

```
import pandas as pd

data ={‘a’:0.,’b’:1.,’c’:2.}

s =pd.Series(data,index=[‘b’,’c’,’a’])

print(s)
```

Its output is as follows –

```
b 1.0

c 2.0

a 0.0
```

You can also show the specified values by giving their keys as index values.

6. Modify the above program and display values of ‘a’ and ‘c’ only.

Ans.

```
import pandas as pd
data ={‘a’:0.,’b’:1.,’c’:2.}
s =pd.Series(data,index=[‘a’,’c’])
print(s)
```

```
Its output is as follows –
a 0.0
c 2.0
```

If index value is passed other than keys value then NaN (Not a Number) as value will be displayed.

7. Give the output:

Ans.

```
import pandas as pd

data ={‘a’:0,’b’:1.,’c’:2.}

s =pd.Series(data,index=[‘b’,’c’,’d’,’a’])

print(s)
```

Its output is as follows –

```
b 1.0

c 2.0

d NaN

a 0.0

dtype: float64
```

Observe − Index order is persisted and the missing element is filled with NaN (Not a Number).

Create a Series from Scalar/Constant value

If data is a scalar value, an index must be provided. The value will be repeated to match the length of index.

8. Write a program to create a series to print scalar value “5” four times.

Ans.

```
import pandas as pd
s =pd.Series(5, index=[0,1,2,3])
print(s)
```

Its output is as follows –

```
0      5
1      5
2      5
3      5
```

As you can see the “5” is printed 4 times because the length of index is 4.

Create a Series from ndarray

9. Write a program to create a series from ndarray with elements ‘a’,’b’,’c’,’d’

Ans.

```
import pandas as pd

import numpy as np

data =np.array([‘a’,’b’,’c’,’d’])

s =pd.Series(data)

print(s)
```

Its output is as follows –

```
0 a

1 b

2 c

3 d

dtype: object
```

10. Give the output:

```
import pandas as pd

import numpy as np

data =np.arange(10,15)

s =pd.Series(data)

print(s)
```

Ans.

```
0 10

1 11

2 12

3 13

4 14
```

REINDEXING

Reindexing means to conform the data to match a given set of labels along a particular axis. Reorder the existing data to match a new set of labels. Reindexing does not mean providing new index values; in fact, it is used to reorder the existing data or print specified data.

11. Give the output:

```
import pandas as pd
name=[‘Raj’,’Ankur’,’Harsh’]
p=pd.Series(name,index=[2,5,6])
print(p)
# Reindex the series and create a new series variable
p1=p.reindex([6,2,5])
print (p1)
```

Ans.

```
    2   Raj
    5   Ankur
    6   Harsh
    6   Harsh
    2   Raj
    5   Ankur
```

12. Give the output:

```
import pandas as pd
name=[‘Raj’,’Ankur’,’Harsh’]
p=pd.Series(name,index=[2,5,6])
print(p) p1=p.reindex([2,5])
print (p1)
```

Ans.

```
    2 Raj
    5 Ankur
    6 Harsh
    dtype: object

    2 Raj
    5 Ankur
    dtype: object
```

if other than existing index value is provided to reindex then NaN will be displayed.

13. Give the output:

```
import pandas as pd
name=[‘Raj’,’Ankur’,’Harsh’]
p=pd.Series(name,index=[2,5,6])
print(p) p1=p.reindex([2,4,5])
print (p1)
```

Ans.

```
    2 Raj
    5 Ankur
    6 Harsh
    dtype: object
    2 Raj
    4 NaN
    5 Ankur

    dtype: object
```

ALTER INDEX VALUES

The Series index function does not only allow you to display the index items, but you can also alter it as well. This example changes the actual index items and places the integer values as the index.

14. Give the output:

```
import pandas as pd
S = pd.Series([10,20,30,40,50], index = ['a', 'e', 'i', 'o', 'u'])
print(S)
# Assigning New Index Values
S.index = [1, 2, 3, 4, 5]
print("Series after new index values")
print(S)
```

Ans.

```
    a 10
    e 20
    i 30
    o 40
    u 50
    dtype: int64

    Series after new index values
    1 10
    2 20
    3 30
    4 40
    5 50
    dtype: int64
```

SIZE ATTRIBUTE

All pandas data structures are value-mutable (the values they contain can be altered). All pandas data structures are size mutable except Series. The length of a Series cannot be changed, i.e. number of columns and rows can't be altered once defined. Series is size immutable.

15. What is Series Size?

Ans. Size attribute returns the number of elements in the underlying data for the given series objects.

16. Give the output:

```
import pandas as pd
L=[10,20,30]
S=pd.Series(L)
print(S.size)
```

Ans. 3