SunilOS

Pandas Library
Lets play with Tabular Data

www.sunilos.com

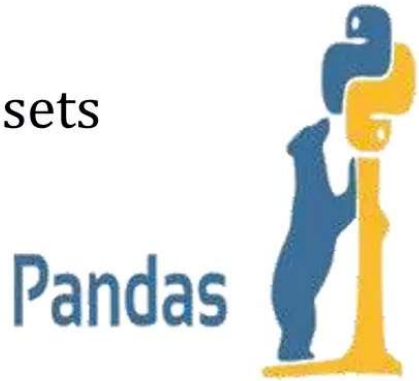
www.raystec.com

Think IT Think Us
RAYS

What is Pandas ?

- Pandas is open source.
- BSD- licensed Python library providing high – Performance.
- Easy to use for data structures and data analysis.
- Pandas use for different types of data.
  - Tabular data with heterogeneously-typed columns.
  - Ordered and unordered time series data.

  - Arbitrary matrix data with row & column labels
  - Unlabeled data
  - Any other form of observational or statistical data sets



Pandas is Data Frame

SunilOS

	A	B	C	D	E	F
1	Country	Salesperson	Order Date	OrderID	Units	Order Amount
2	USA	Fuller	1/01/2011	10392	13	1,440.00
3	UK	Gloucester	2/01/2011	10397	17	716.72
4	UK	Bromley	2/01/2011	10771	18	344.00
5	USA	Finchley	3/01/2011	10393	16	2,556.95
6	USA	Finchley	3/01/2011	10394	10	442.00
7	UK	Gillingham	3/01/2011	10395	9	2,122.92
8	USA	Finchley	6/01/2011	10396	7	1,903.80
9	USA	Callahan	8/01/2011	10399	17	1,765.60
10	USA	Fuller	8/01/2011	10404	7	1,591.25
11	USA	Fuller	9/01/2011	10398	11	2,505.60
12	USA	Coghill	9/01/2011	10403	18	855.01
13	USA	Finchley	10/01/2011	10401	7	3,868.60
14	USA	Callahan	10/01/2011	10402	11	2,713.50
15	UK	Rayleigh	13/01/2011	10406	15	1,830.78
16	USA	Callahan	14/01/2011	10408	10	1,622.40
17	USA	Farnham	14/01/2011	10409	19	319.20
18	USA	Farnham	15/01/2011	10410	16	802.00

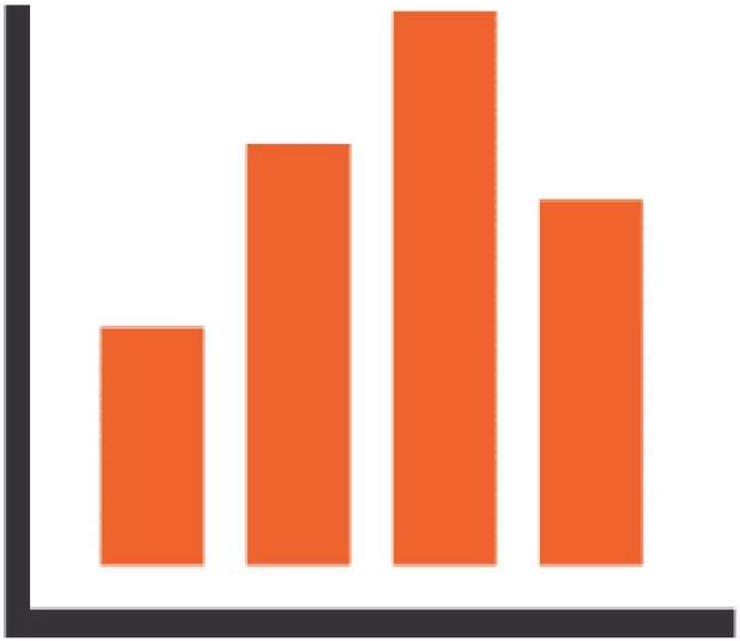
max std filter min count

Features of pandas

- It has a DataFrame Object with customized indexing.
- We can reshape data sets.
- We can perform aggregations and Transformations on the data sets
- It is used for data alignment and integration of the missing data.
- It provides the functionality of Time Series.
- Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.
- Handle multiple operations of the data sets such as subsetting, slicing, filtering, groupBy, re-ordering, and re-shaping.
- It integrates with the other libraries such as SciPy, and scikit-learn.
- Provides fast performance, and If you want to speed it, even more, you can use the **Cython**.

Usage of Pandas

- Modeling
- Visualization
- High Performance Computing
- Big Data
- Statistical Computing
- Numerical Computing
- Data Mining
- Text Processing
- Computing Language
- Educational Outreach
- Computational thinking



How to install pandas

- You can run following command to install pandas.
  - `pip install pandas`
  - OR
  - `conda install pandas`
- After installation you can check version of pandas by running the following command at python script mode.
  - `Import pandas`
  - `pandas.__version__`

Data Structures

- There are 2 data structures in pandas.
  - Series.(One D array)
  - DataFrame

Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

Series Data Structure

- Series is a one –dimensional array.
- It contain homogeneous data.
- Size is fixed.

10	20	30	40	50	60	70

Series Data Structure Cont.

- Series is One Dimensional array to hold different types of values.
  - `pandas.Series( data, index, dtype, copy)`
- **data:** It can be any list, dictionary, or scalar value.
- **index:** The value of the index should be unique and hashable. It must be of the same length as data. If we do not pass any index, default np.arrange(n) will be used.
- **dtype:** It refers to the data type of series.
- **copy:** It is used for copying the data.

Creating a Series

- We can create Series by using various inputs:
  - Array
  - Dictonary
  - Scalar value

Using Array

- `import pandas as pd`
- `import numpy as np`
- `info = np.array(['R','a','y','s','T','e','c', 'h'])`
- `a = pd.Series(info)`
- `print(a)`

Using Dictionary and scalar Value

- **Using dictionary**
  - `import pandas as pd`
  - `info = {1:'Rays',2:'Tech'}`
  - `a = pd.Series(info)`
  - `print(a)`
- **Using Scalar Value**
  - ```
    import pandas as pd
    a = pd.Series(4,index=[1,2,3,4,5])
    print(a)
    ```
    ```

Series Object attribute:

```
import pandas as pd
x = pd.Series([1,2,3],index = ['a','b','c'])
```

 # It returns a tuple of shape of the data.

```
print("Shape:",x.shape)
```

 # It returns the size of the data.

```
print("Size:",x.size)
```

 # Defines the index of the Series.

```
print("Index:",x.index)
```

 # It returns the number of dimensions in the data.

```
print("Dimension:",x.ndim)
```

 # It returns the values of series as list

```
print("values:",x.values)
```

Series Object attribute (cont.)

- **# It returns the data type of the data.**

```
print("Data Type:",x.dtype)
```

- **# It returns the number of bytes in the data**

```
print("Bytes:",x.nbytes)
```

- **# It returns True if Series object is empty, otherwise returns false.**

```
print("is empty:",x.empty)
```

- **#It returns True if there are any NaN values, otherwise returns false.**

```
print("is empty:",x.hasnans)
```

Pandas Operations

map()

```
import pandas as pd
import numpy as np
a = pd.Series(['Java', 'C', 'C++', np.nan])
print(a.map({'Java':'Core',"C":'Python'}))
```

std()

```
import numpy as np
print("Series:",np.std([1,2,3,4,5]))
```

count()

```
import pandas as pd
import numpy as np
i = pd.Series([2, 1, 1, np.nan, 3,4,5,5])
print(i.count())
```

Pandas Operations

filter()

```
import numpy as np
import pandas as pd
arr=np.array(([20,30,40],[50,70,80]))
df=pd.DataFrame(arr,index=["Vijay","Jay"],columns=
["Physics","Chemistry","Maths"])
print(df)
print(df.filter(items=["Physics"]))
```

Data Frame & Panel Data Structure

- DataFrame is a two–dimensional array.
- It contain heterogeneous data.
- Size is fixed.
- Data Type of Columns.
- Panel is three-dimensional array.
- It is hard to represent the Panel In graphical representation.
- But panel is consider as a Data Frame.

NAME	AGE	ADDRESS
Rahul	22	Indore
Ram	35	Ayodhya
Krishna	25	Mathura

COLUMN NAME	TYPE
NAME	String
AGE	Integer
ADDRESS	String

Creating DataFrame

- **# importing the pandas library**

```
import pandas as pd
#empty DataFrame
df = pd.DataFrame()
print (df)
# Using List
list=["Rays","Tech","SunilOs"]
print(pd.DataFrame(list))
# using Dict
info = {'ID' :[101, 102, 103],'Department'
:['B.Sc','B.Tech','M.Tech',]}
print(pd.DataFrame(info))
```

Creating DataFrame

- **#Using Dict of series**

```
info = {'one' : pd.Series([1, 2, 3, 4, 5, 6],
index=['a', 'b', 'c', 'd', 'e', 'f']),
  'two' : pd.Series([1, 2, 3, 4, 5, 6, 7, 8],
index=['a', 'b', 'c', 'd', 'e', 'f', 'g',
'h'])}

d1 = pd.DataFrame(info)
print(d1)
```

	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	4.0	4
e	5.0	5
f	6.0	6
g	NaN	7
h	NaN	8

DataFrame Implementation

- `import pandas as pd`
- `a list of strings`
- `x = ['Rays','Tech',"SunilOs"]`
- `Calling DataFrame constructor on list`

- `df = pd.DataFrame(x)`
- `print(df)`

Column Selection from a DataFrame:

```
from pandas import DataFrame,Series
x={'Id':Series([1,2,3,4,5]),
"Name":Series(["A","B","C","D","E"])}
df=DataFrame(x)
print(df)
print(df['Id'])
print(df["Name"])
```

Column Addition and Deletion to DataFrame

```
from pandas import DataFrame,Series
x={'Id':Series([1,2,3,4,5]),"Name":Series(["A"
,"B","C","D","E"])}
df=DataFrame(x)
print(df)
```

```
# ADD a Column from DataFrame
df['Subject']=Series(["C","C++","Java","python
","Java","Julia"])
```

```
del df["Id"] # Delete a Column from DataFrame
Using del function
print(df)
```

```
df.pop("Name") # Delete a Column from
DataFrame Using pop function
```

Row Addition from DataFrame

```
from pandas import DataFrame,Series
x={'Id':Series([1,2,3,4,5],index=['a','b','c','d','e']),
"Name":Series(["A","B","C"],index=['a','b','c'])
}
df=DataFrame(x)
d=DataFrame({"Subject":Series([1,2,3],index=['a','b','c'])})
print(df.append(d))
#OR
d=DataFrame([[1,"Ajay","Python"]],
columns=["Id","Name","Subject"])
print(df.append(d))
```

Row Deletion from DataFrame:

SunilOS

- `from pandas import DataFrame,Series`
-
```
x={'Id':Series([1,2,3,4,5],index=['a','b
','c','d','e']),
"Name":Series(["A","B","C"],index=['a','
b','c'])}
```
- `df=DataFrame(x)`
- `print(df.drop('a'))`

Row Selection from DataFrame

SunilOS

- `from pandas import DataFrame,Series`
- `x={'Id':Series([1,2,3,4,5],index=['a','b','c','d','e']),`
  `"Name":Series(["A","B","C"],index=['a','b','c'])}`
- `df=DataFrame(x)`
  `#by row label`
- `print(df.loc['a'])`
  `#by using row integer row location`
- `print(df.iloc[4])`
  `#by using slice operator :`
  `print(df[0:3])`

DataFrame Object Methods:

- **append():** Add the rows of other dataframe to the end of the given dataframe.
- **apply():** Allows the user to pass a function and apply it to every single value of the Pandas series.
- **count():** Count the number of non-NA cells for each column or row.
- **describe():** Calculate some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame.

DataFrame Object Methods: (cont.)

- **drop_duplicate():** Remove duplicate values from the DataFrame.
- **head():**Returns the first n rows for the object based on position.
- **mean():** Return the mean of the values for the requested axis.
- **merge():**Merge the two datasets together into one.
- **pivot_table():**Aggregate data with calculations such as Sum, Count, Average, Max, and Min.
- **query():**Filter the dataframe.
- **Sort_index():** Sort the dataframe.
- **sum():** Return the sum of the values for the requested axis by the user.
- **to_excel():** Export the dataframe to the excel file.
- **transpose():** Transpose the index and columns of the dataframe.

Importing Data:

- pd.read_csv(filename) : It read the data from CSV file.
- pd.read_table(filename) : It is used to read the data from delimited text file.
- pd.read_excel(filename) : It read the data from an Excel file.
- pd.read_sql(query,connection _object) : It read the data from a SQL table/database.
- pd.read_json(json _string) : It read the data from a JSON formatted string, URL or file.

Time-Series in Panda

- Time-series data is a data with equal no of intervals such as daily, monthly, yearly.
- Data is collected on a fraction of time

```
import pandas as pd
from datetime import datetime
import numpy as np


range_date = pd.date_range(start ='1/1/2019', end ='1/08/2019', freq ='Min')
print(range_date)
print(len(range_date))
```

Panda Date Time

- `import pandas as pd`
- `dmy = pd.date_range('2017-06-04', periods=5, freq='S')`
- **We can Format the String to date**
  - `import pandas as pd`
  - `import datetime`
  - `print(pd.to_datetime('18000706', format='%Y%m%d', errors='ignore') )`
  - `print(datetime.datetime(1800, 7, 6, 0, 0) )`
  - `print(pd.to_datetime('18000706', format='%Y%m%d', errors='coerce') )`
-

Pandas time offset

- It performs various operation on time i.e. adding and subtracting

```
import pandas as pd
# Create the Timestamp
p = pd.Timestamp('2018-12-12 06:25:18')
# Create the DateOffset
do = pd.tseries.offsets.DateOffset(n = 2)
# Add the dateoffset to given timestamp
new_timestamp = p + do
# Print updated timestamp
print(new_timestamp)
```

Pandas Time Period

- It represents the time period i.e. day, month, quarter, year.
- It converts frequency to time period.
  - `import pandas as pd`
  - `daily=pd.period_range("2000","2010",freq='D')`
  - `print(daily)`