Gentle Pandas

Python Mauritius UserGroup (pymug)

More info: mscc.mu/python-mauritius-usergroup-pymug/

Where Are We? 🏠

Why	Where
codes	github.com/pymug
share events	twitter.com/pymugdotcom
ping professionals	linkedin.com/company/pymug
all info	pymug.com
discuss	facebook.com/groups/318161658897893
tell friends by like	facebook.com/pymug

Support Us

Please support us by subscribing to our mailing list:
https://mail.python.org/mailman3/lists/pymug.python.org/

Abdur-Rahmaan Janhangeer
(Just a Python programmer)

twitter: @osdotsystem
github: github.com/abdur-rahmaanj

www.pythonmembers.club

Gentle Pandas

⚠️ Best to use ***Jupyter*** from ***Anaconda*** to try out the examples

Jupyter Shortcuts

`alt + enter` : run with new cell

`ctrl + enter` : run

Why Pandas

easy and powerful manipulation of indexed data

Pandas import

we import pandas as

```
import pandas as pd
```

Pandas Series

```
import pandas as pd
import numpy as np

data = np.array(['apple', 'banana', 'pear'])
series = pd.Series(data)
print(series)
```

```
0     apple
1    banana
2      pear
dtype: object
```

Series gives us indexes but we can specify our own

```
series = pd.Series(data, index=[80, 90, 100])
```

series can also be strings

Series from dictionary

```
data = {'a' : 0., 'b' : 1., 'c' : 2.}
series = pd.Series(data)
series
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

to access:

```
data['b']
```

```
1.0
```

Pandas Dataframe

Pandas dataframes are the real data type we'll be using with Pandas. Many Pandas series make a Pandas dataframe. A series is a column.

Reading data from file

```
df = pd.read_csv('<filename>.csv')
```

Creating data frame from scratch

```
# Data adapted from open data mauritius

data = {
    'Industry_Group': ['Black River', 'Flacq', 'Grand Port',
                       'Moka', 'Port Louis'],
    'Year':[2014, 2014, 2014, 2013, 2011],
    'Number_Of_SMEs': [77, 288, 91, 102, 224]
}

smes = pd.DataFrame(data)
smes
```

	Industry_Group	Year	Number_Of_SMEs
0	Black River	2014	77
1	Flacq	2014	288
2	Grand Port	2014	91
3	Moka	2013	102
4	Port Louis	2011	224

As with series, we can add `index=` to specify string indexes instead of 0, 1 ...

Viewing Data

```
# head gives data from the top

smes.head(3)
```

	Industry_Group	Year	Number_Of_SMEs
0	Black River	2014	77
1	Flacq	2014	288
2	Grand Port	2014	91

```
# tail gives from the end
smes.tail(3)
```

	Industry_Group	Year	Number_Of_SMEs
2	Grand Port	2014	91
3	Moka	2013	102
4	Port Louis	2011	224

Dataframe information

```
smes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
Industry_Group    5 non-null object
Year              5 non-null int64
Number_Of_SMEs    5 non-null int64
dtypes: int64(2), object(1)
memory usage: 140.0+ bytes
```

```
smes.describe()
```

	Year	Number_Of_SMEs
count	5.00000	5.00000
mean	2013.20000	156.40000
std	1.30384	94.11323
min	2011.00000	77.00000
25%	2013.00000	91.00000
50%	2014.00000	102.00000
75%	2014.00000	224.00000
max	2014.00000	288.00000

Indexes and selection

```
smes['Year']
```

```
0    2014
1    2014
2    2014
3    2013
4    2011
Name: Year, dtype: int64
```

```
smes['Year'] > 2013
```

```
0     True
1     True
2     True
3    False
4    False
Name: Year, dtype: bool
```

```
smes[smes['Year'] > 2013]
```

	Industry_Group	Year	Number_Of_SMEs
0	Black River	2014	77
1	Flacq	2014	288
2	Grand Port	2014	91

the above can also be stored in a variable

```
greater_than_2013 =  smes['Year'] > 2013
smes[greater_than_2013]
```

	Industry_Group	Year	Number_Of_SMEs
0	Black River	2014	77
1	Flacq	2014	288
2	Grand Port	2014	91

```
between = (smes['Year'] > 2011) & (smes['Year'] < 2014)
smes[between]
```

	Industry_Group	Year	Number_Of_SMEs
3	Moka	2013	102

```
smes['Year'].iloc[0]
```

```
2014
```

iloc

```
smes.iloc[0:3]
```

	Industry_Group	Year	Number_Of_SMEs
0	Black River	2014	77
1	Flacq	2014	288
2	Grand Port	2014	91

```
smes['Year'].iloc[0:3]
```

```
0    2014
1    2014
2    2014
Name: Year, dtype: int64
```

Column wise manipulations

```
data = {
    'width': [10, 20, 29],
    'height': [33, 32, 54]
}
info = pd.DataFrame(data)
info
```

	width	height
0	10	33
1	20	32
2	29	54

```
info['area'] = info['width'] * info['height']
info
```

	width	height	area
0	10	33	330
1	20	32	640
2	29	54	1566

Data Cleaning

```
# missing data
data = {
    'width': [10, 20, np.nan],
    'height': [33, 32, 54]
}
info = pd.DataFrame(data)
info
```

	width	height
0	10.0	33
1	20.0	32
2	NaN	54

```
info.dropna()
```

	width	height
0	10.0	33
1	20.0	32

```
info.replace(np.nan, 7)
```

	width	height
0	10.0	33
1	20.0	32
2	7.0	54

Some Stats

```
info.mean()
```

```
width     15.000000
height    39.666667
dtype: float64
```

```
info.median()
```

```
width     15.0
height    33.0
dtype: float64
```

```
info['width'].min()
```

```
10.0
```

```
info['height'].std()
```

```
12.42309676905615
```

```
info.groupby('height').sum()
```

	width
height	
32	20.0
33	10.0
54	0.0

```
info.groupby('height')[['width']].mean()
```

	width
height	
32	20.0
33	10.0
54	NaN

More to look at

- time series
- query eval
- pivot tables
- join, melt and duplicate
- sorting
- working with text