Python Pandas Cheat Sheet

Pandas is one of the most popular packages in Python. It is widely used for data manipulation, data cleaning and wrangling. Panda's package comes up with multiple feature-rich functions and options which could be overwhelming. This pandas cheat sheet might be a handy tool in such instances where one could quickly brush up the basics of Pandas.

Data Structures:

- Series and Data frame are two prominent data structures of Pandas library.
- Series is a one-dimensional labelled array capable of holding any data type
- Data Frame is a 2-dimensional labelled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table.

Importing Pandas:

```
import pandas as pd
```

Creating Series:

```
a = pd.Series([1,2,3,4,5])
print(a)
```

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

Creating Data Frame

```
b = { 'Student' : ['A','B','C','D','E','F'],'ID' : [1,2,3,4,5,6]}
print(type(b))

df = pd.DataFrame(b)
print(df)
```

```
<class 'dict'>
  Student  ID
0       A   1
1       B   2
2       C   3
3       D   4
4       E   5
5       F   6
```

To retrieve basic information from a Data Frame:

```
df.head()
```

```
  Student  ID
0       A   1
1       B   2
2       C   3
3       D   4
4       E   5
```

```
df.tail()
```

```
  Student  ID
1       B   2
2       C   3
3       D   4
4       E   5
5       F   6
```

```
df.shape
```

```
(6, 2)
```

```
df.Student
```

```
0    A
1    B
2    C
3    D
4    E
5    F
Name: Student, dtype: object
```

```
df['Student']
```

```
0    A
1    B
2    C
3    D
4    E
5    F
Name: Student, dtype: object
```

```
df.columns
```

```
Index(['Student', 'ID'], dtype='object')
```

```
df['ID'].describe()
```

```
count    6.000000
mean     3.500000
std      1.870829
min      1.000000
25%      2.250000
50%      3.500000
75%      4.750000
max      6.000000
Name: ID, dtype: float64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Student  6 non-null      object
 1   ID       6 non-null      int64 
dtypes: int64(1), object(1)
memory usage: 224.0+ bytes
```

```
df.values
```

```
array([['A', 1],
       ['B', 2],
       ['C', 3],
       ['D', 4],
       ['E', 5],
       ['F', 6]], dtype=object)
```

```
df
```

```
  Student  ID
0       A   1
1       B   2
2       C   3
3       D   4
4       E   5
5       F   6
```

To use a Column as Index:

```
df.set_index('ID',inplace = True)
df
```

```
   Student
ID
1        A
2        B
3        C
4        D
5        E
6        F
```

To Sort the Data Frame by Index/Column:

```
df.sort_index(axis = 0,ascending = False)
```

```
   Student
ID
6        F
5        E
4        D
3        C
2        B
1        A
```

```
df.sort_values(by = 'ID',ascending = False)
```

```
   Student
ID
6        F
5        E
4        D
3        C
2        B
1        A
```

To select rows/columns of a data frame based on index value :

```
df.iloc[[0,1,2]]
```

```
   Student
ID
1        A
2        B
3        C
```

To select rows/columns of a data frame based on Label/Name

```
df.loc[1:3]
```

```
   Student
ID
1        A
```

```
2        B
3        C
```

```
df.loc[1:3,'Student']
```

```
ID
1    A
2    B
3    C
Name: Student, dtype: object
```

Concatenate Data Frames: (by column)

```
d1 = pd.DataFrame([['a',1],['b',2]],columns= ['name','number'])
d2 = pd.DataFrame([['c',3,'lion'],['d',4,'tiger']],columns=
['letter','number','animal'])
pd.concat([d1,d2],axis = 1)
```

```
  name  number letter  number animal
0    a       1      c       3   lion
1    b       2      d       4  tiger
```

```
d1 = pd.DataFrame([['a',1],['b',2]],columns= ['name','number'])
d2 = pd.DataFrame([['c',3,'lion'],['d',4,'tiger']],columns=
['letter','number','animal'])
pd.concat([d1,d2],axis = 0,sort = True)
```

```
  animal letter name  number
0    NaN    NaN    a       1
1    NaN    NaN    b       2
0   lion      c  NaN       3
1  tiger      d  NaN       4
```

Merging Data Frames: This works similar to SQL joins (left join, right,outer, inner)

Merging based on a Column

```
d1 = pd.DataFrame({  "city":
["lucknow","kanpur","agra","delhi"],"temperature" : [32,45,30,40]})
d2 = pd.DataFrame({"city":["delhi","lucknow","kanpur"],"humidity" :
[68,65,75]})
print(d1)
print(d2)
df = pd.merge(d1,d2,on='city')
df
```

```
      city  temperature
0  lucknow           32
1   kanpur           45
2     agra           30
3    delhi           40
      city  humidity
0    delhi        68
1  lucknow        65
2   kanpur        75

      city  temperature  humidity
0  lucknow           32        65
1   kanpur           45        75
2    delhi           40        68
```

Outer join

```
pd.merge(d1,d2,on=['city'],how='outer')
```

```
      city  temperature  humidity
0  lucknow           32      65.0
1   kanpur           45      75.0
2     agra           30       NaN
3    delhi           40      68.0
```

left join

```
pd.merge(d1,d2,on=['city'],how='left')
```

```
      city  temperature  humidity
0  lucknow           32      65.0
1   kanpur           45      75.0
2     agra           30       NaN
3    delhi           40      68.0
```

Reading External files :

```
df2 = pd.read_excel('df_xl.xlsx')

df1 = pd.read_csv("weather_data.csv")
```

Applying function to column/s in a data frame:

By defining a function

```
def hot_temp(x) :
   return x>30
```

Handling missing values:

```
df.fillna(value=50)
```

```
      city  temperature  humidity
0  lucknow           32        65
1   kanpur           45        75
2    delhi           40        68
```