Python For Data Science *Cheat Sheet*

Pandas

Learn Python for Data Science **Interactively** at www.DataCamp.com

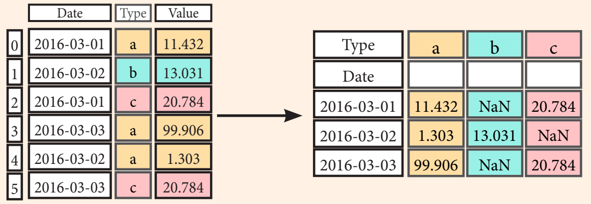
df.describe()

Reshaping Data

Pivot

```
>>> df3= df2.pivot(index='Date',
                   columns='Type',
                   values='Value')
```

Spread rows into columns



	Date	Type	Value
0	2016-03-01	a	11.432
1	2016-03-02	b	13.031
2	2016-03-01	c	20.784
3	2016-03-03	a	99.906
4	2016-03-02	a	1.303
5	2016-03-03	c	20.784

Type	a	b	c
Date			
2016-03-01	11.432	NaN	20.784
2016-03-02	1.303	13.031	NaN
2016-03-03	99.906	NaN	20.784

Pivot Table

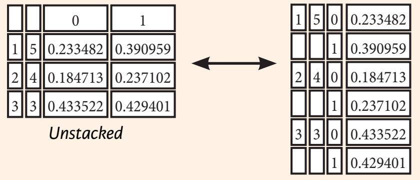
```
>>> df4 = pd.pivot_table(df2,
                         values='Value',
                         index='Date',
                         columns='Type'])
```

Spread rows into columns

Stack / Unstack

```
>>> stacked = df5.stack()
>>> stacked.unstack()
```

`>>> stacked = df5.stack()`	Pivot a level of column labels
`>>> stacked.unstack()`	Pivot a level of index labels



		0	1
1	5	0.233482	0.390959
2	4	0.184713	0.237102
3	3	0.433522	0.429401

Unstacked

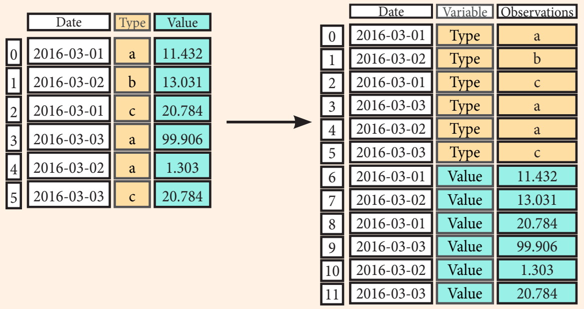
1	5	0	0.233482
		1	0.390959
2	4	0	0.184713
		1	0.237102
3	3	0	0.433522
		1	0.429401

Stacked

Melt

```
>>> pd.melt(df2,
            id_vars=["Date"],
            value_vars=["Type", "Value"],
            value_name="Observations")
```

Gather columns into rows



	Date	Type	Value
0	2016-03-01	a	11.432
1	2016-03-02	b	13.031
2	2016-03-01	c	20.784
3	2016-03-03	a	99.906
4	2016-03-02	a	1.303
5	2016-03-03	c	20.784

	Date	Variable	Observations
0	2016-03-01	Type	a
1	2016-03-02	Type	b
2	2016-03-01	Type	c
3	2016-03-03	Type	a
4	2016-03-02	Type	a
5	2016-03-03	Type	c
6	2016-03-01	Value	11.432
7	2016-03-02	Value	13.031
8	2016-03-01	Value	20.784
9	2016-03-03	Value	99.906
10	2016-03-02	Value	1.303
11	2016-03-03	Value	20.784

Iteration

`>>> df.iteritems()`	(Column-index, Series) pairs
`>>> df.iterrows()`	(Row-index, Series) pairs

Advanced Indexing

Also see NumPy Arrays

Selecting

df3.iloc[:, 1:5] | np.array(df3).

`>>> df3.loc[:,(df3>1).any()]`	Select cols with any vals >1
`>>> df3.loc[:,(df3>1).all()]`	Select cols with vals > 1
`>>> df3.loc[:,df3.isnull().any()]`	Select cols with NaN
`>>> df3.loc[:,df3.notnull().all()]`	Select cols without NaN

Indexing With isin

`>>> df[(df.Country.isin(df2.Type))]`	Find same elements
`>>> df3.filter(items="a","b"])`	Filter on values
`>>> df.select(lambda x: not x%5)`	Select specific elements

Where

`>>> s.where(s > 0)`	Subset the data

Query

`>>> df6.query('second > first')`	Query DataFrame

Setting/Resetting Index

```
>>> df.set_index('Country')
>>> df4 = df.reset_index()
>>> df = df.rename(index=str,
                   columns={"Country":"cntry",
                            "Capital":"cptl",
                            "Population":"ppltn"})
```

Set the index
Reset the index
Rename DataFrame

Reindexing

```
>>> s2 = s.reindex(['a','c','d','e','b'])
```

Forward Filling

```
>>> df.reindex(range(4),
               method='ffill')
   Country    Capital    Population
 0 Belgium    Brussels   11190846
 1   India  New Delhi  1303171035
 2  Brazil   Brasilia   207847528
 3  Brazil   Brasilia   207847528
```

Backward Filling

```
>>> s3 = s.reindex(range(5),
                   method='bfill')
 0    3
 1    3
 2    3
 3    3
 4    3
```

MultiIndexing

```
>>> arrays = [np.array([1,2,3]),
              np.array([5,4,3])]
>>> df5 = pd.DataFrame(np.random.rand(3, 2), index=arrays)
>>> tuples = list(zip(*arrays))
>>> index = pd.MultiIndex.from_tuples(tuples,
                                      names=['first', 'second'])
>>> df6 = pd.DataFrame(np.random.rand(3, 2), index=index)
>>> df2.set_index(["Date", "Type"])
```

Duplicate Data

`>>> s3.unique()`	Return unique values
`>>> df2.duplicated('Type')`	Check duplicates
`>>> df2.drop_duplicates('Type', keep='last')`	Drop duplicates
`>>> df.index.duplicated()`	Check index duplicates

Grouping Data

Aggregation

```
>>> df2.groupby(by=['Date','Type']).mean()
>>> df4.groupby(level=0).sum()
>>> df4.groupby(level=0).agg({'a':lambda x:sum(x)/len(x),
                              'b': np.sum})
```

Transformation

```
>>> customSum = lambda x: (x+x%2)
>>> df4.groupby(level=0).transform(customSum)
```

Missing Data

df.drop("column", 1).

`>>> df.dropna()`	Drop NaN values
`>>> df3.fillna(df3.mean())`	Fill NaN values with a predetermined value
`>>> df2.replace("a", "f")`	Replace values with others

Combining Data

data1

X1	X2
a	11.432
b	1.303
c	99.906

data2

X1	X3
a	20.784
b	NaN
d	20.784

Merge

```
>>> pd.merge(data1,
             data2,
             how='left',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN

```
>>> pd.merge(data1,
             data2,
             how='right',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
d	NaN	20.784

```
>>> pd.merge(data1,
             data2,
             how='inner',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN

```
>>> pd.merge(data1,
             data2,
             how='outer',
             on='X1')
```

X1	X2	X3
a	11.432	20.784
b	1.303	NaN
c	99.906	NaN
d	NaN	20.784

Join

```
>>> data1.join(data2, how='right')
```

Concatenate

Vertical

```
>>> s.append(s2)
```

Horizontal/Vertical

```
>>> pd.concat([s,s2],axis=1, keys=['One','Two'])
>>> pd.concat([data1, data2], axis=1, join='inner')
```

Dates

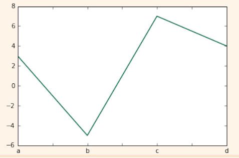
```
>>> df2['Date']= pd.to_datetime(df2['Date'])
>>> df2['Date']= pd.date_range('2000-1-1',
                               periods=6,
                               freq='M')
>>> dates = [datetime(2012,5,1), datetime(2012,5,2)]
>>> index = pd.DatetimeIndex(dates)
>>> index = pd.date_range(datetime(2012,2,1), end, freq='BM')
```

Visualization

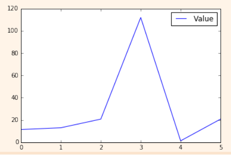
Also see Matplotlib

```
>>> import matplotlib.pyplot as plt
```

```
>>> s.plot()
>>> plt.show()
```



```
>>> df2.plot()
>>> plt.show()
```



CSV

```
pandas_csv= pd.read_csv("data.csv").
                        sep=";"
print(pandas_csv.iloc[:,:5])
numpy_csv= np.array(pandas_csv)
```

Info Dataset

```
pd.info()
pd.describe()
pd.head()
```

Columnlist:

```
pd.columns.tolist()
```

Count:

```
pd.column.count()
pd.count().
pd[column].value_counts
                (dropna=False)
```

loc V.S Where:

loc → remove values
where → replace with NaN.

FILTER: &

```
df2= df.loc[(df.col1<x)|(df.col2<y)]
```

Dataframe (craation)

```
df= pd.DataFrame(columns= col_names)
df= df.append(pd.Series( liste,
                          index= col_names),
                          ignore_index=True)
        └> row
        ┌> column
df['Name_column'] = liste.
```