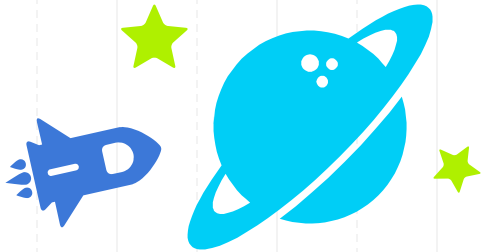


PANDAS

This package is required for data preprocessing.

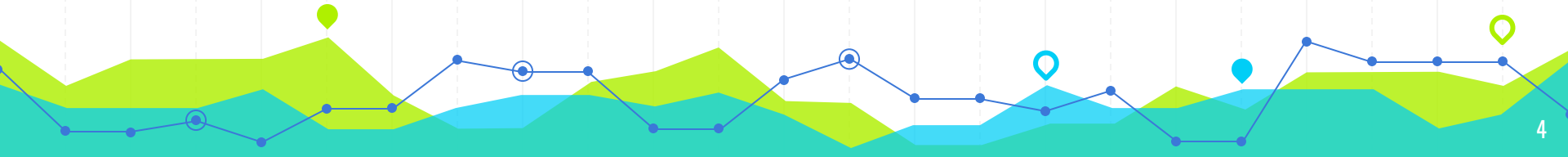
It’s faster and used for cleaning, restructuring or merging etc.

PANDAS OPERATIONS

- Reading Files
- Data Operations
- GroupBy
- Unstack
- Merge
- Index

“

import pandas as pd



Convert tuples to Series

```
>>> # converting tuple to Series
>>> h = ('AA', '2012-02-01', 100, 10.2)
>>> s = pd.Series(h)
>>> type(s)
<class 'pandas.core.series.Series'>
```

Convert dictionaries to Series

```
>>> # converting dict to Series
>>> d = {'name' : 'IBM', 'date' : '2010-09-08', 'shares' : 100, 'price' : 10.2}
>>> ds = pd.Series(d)
>>> type(ds)
<class 'pandas.core.series.Series'>
```

Tuple conversion with custom index

```
>>> f = ['FB', '2001-08-02', 90, 3.2]
>>> f = pd.Series(f, index = ['name', 'date', 'shares', 'price'])
>>> print(f)
name          FB
date      2001-08-02
shares        90
price         3.2
dtype: object
```

Provide index in list

```
>>> f['shares']
90
>>> f[0]
'FB'
>>> f[['shares', 'price']]
shares    90
price    3.2
dtype: object
```

Dataframes

```
>>> data = { 'name' : ['AA', 'IBM', 'GOOG'],
...       'date' : ['2001-12-01', '2012-02-10', '2010-04-09'],
...       'shares' : [100, 30, 90],
...       'price' : [12.3, 10.3, 32.2]
... }
>>> df = pd.DataFrame(data)
>>> type(df)
<class 'pandas.core.frame.DataFrame'>
```

Dataframes

```
>>> df
        date  name  price  shares
0  2001-12-01    AA   12.3     100
1  2012-02-10   IBM   10.3      30
2  2010-04-09  GOOG   32.2      90
```

Add additional Column

```
>>> df['owner'] = 'Unknown'
>>> df
        date  name  price  shares    owner
0  2001-12-01    AA   12.3     100  Unknown
1  2012-02-10   IBM   10.3      30  Unknown
2  2010-04-09  GOOG   32.2      90  Unknown
```

Change default index of row

```
>>> df.index = ['one', 'two', 'three']
>>> df
            date  name  price  shares    owner
one    2001-12-01    AA   12.3     100  Unknown
two    2012-02-10   IBM   10.3      30  Unknown
three  2010-04-09  GOOG   32.2      90  Unknown
```

Setting column as index

```
>>> df = df.set_index(['name'])
>>> df
         date  price  shares   owner
name
AA   2001-12-01   12.3     100  Unknown
IBM  2012-02-10   10.3      30  Unknown
GOOG 2010-04-09   32.2      90  Unknown
```

Access data using column index

```
>>> # access data using column-index
>>> df['shares']
name
AA      100
IBM      30
GOOG     90
Name: shares, dtype: int64
```

Access data using row index

```
>>> # access data by row-index
>>> df.iloc[0]
date      2001-12-01
price           12.3
shares           100
owner        Unknown
Name: AA, dtype: object
```

Access data using row label

```
>>> # access data by row-index
>>> df.loc['AA']
date      2001-12-01
price           12.3
shares           100
owner        Unknown
Name: AA, dtype: object
```

Access all rows for a column

```
>>> # access all rows for a column
>>> df.ix[:, 'name']
0      AA
1     IBM
2    GOOG
Name: name, dtype: object
```

Access specific element of a column

```
>>> # access specific element from the DataFrame,
>>> df.ix[0, 'shares']
100
```

Del command to delete column

```
>>> del df['owner']
>>> df
        date  price  shares
name
AA    2001-12-01   12.3     100
IBM   2012-02-10   10.3      30
GOOG  2010-04-09   32.2      90
```

Drop command to delete column

```
>>> df.drop('shares', axis = 1)
          date  price
name
AA    2001-12-01   12.3
IBM   2012-02-10   10.3
GOOG  2010-04-09   32.2
```

Reading Files

```
>>> casts = pd.read_csv('cast.csv', index_col=None)
>>> casts.head()

>>> titles = pd.read_csv('titles.csv', index_col =None)
>>> titles.tail()
```

Set Max Limits

```
>>> pd.set_option('max_rows', 10, 'max_columns', 10)

>>> titles

>>> casts
```

Length function

```
>>> len(titles)

50000
```

Access Limited Rows

```
>>> titles.head(3)
                    title  year
0             The Rising Son  1990
1  The Thousand Plane Raid  1969
2         Crucea de piatra  1993
```

Column Selection

```
>>> t = titles['title']
>>> type(t)
<class 'pandas.core.series.Series'>
>>> t.head()
0           The Rising Son
1    The Thousand Plane Raid
2         Crucea de piatra
3                  Country
4              Gaiking II
Name: title, dtype: object
```

Row Selection

```
>>> titles.iloc[0]
title    The Rising Son
year             1990
Name: 0, dtype: object
>>>
```

Filter data

```
>>> # movies after 1985
>>> after85 = titles[titles['year'] > 1985]
>>> after85.head()
                title  year
0    The Rising Son  1990
2  Crucea de piatra  1993
3           Country  2000
4       Gaiking II  2011
5      Medusa (IV)  2015
```

Filter data

```
>>> # movies after 1985
>>> after85 = titles[titles['year'] > 1985]
>>> after85.head()
                title  year
0    The Rising Son  1990
2  Crucea de piatra  1993
3           Country  2000
4       Gaiking II  2011
5      Medusa (IV)  2015
```

Joining two conditions

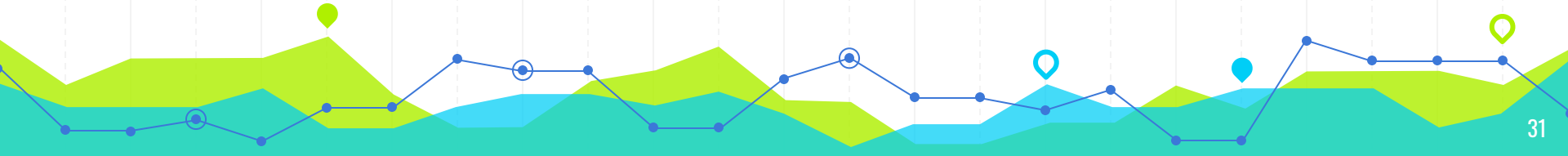
```
>>> # display movie in years 1990 - 1999
>>> t = titles
>>> movies90 = t[ (t['year']>=1990) & (t['year']<2000) ]
>>> movies90.head()
                    title  year
0             The Rising Son  1990
2          Crucea de piatra  1993
```

Joining two conditions

```
>>> # display movie in years 1990 - 1999
>>> t = titles
>>> movies90 = t[ (t['year']>=1990) & (t['year']<2000) ]
>>> movies90.head()
                    title  year
0             The Rising Son  1990
2          Crucea de piatra  1993
```

Sorting

```
>>> # find all movies named as 'Macbeth'
>>> t = titles
>>> macbeth = t[ t['title'] == 'Macbeth']
>>> macbeth.head()
        title  year
4226   Macbeth  1913
9322   Macbeth  2006
```



Sorting same with sort_index()

```
>>> # by default, sort by index i.e. row header
>>> macbeth = t[ t['title'] == 'Macbeth'].sort_index()
>>> macbeth.head()
       title  year
4226   Macbeth  1913
9322   Macbeth  2006
```

Sorting with sort_values()

```
>>> # sort by year
>>> macbeth = t[ t['title'] == 'Macbeth'].sort_values('year')
>>> macbeth.head()
        title  year
4226   Macbeth  1913
17166  Macbeth  1997
```

Null Values

```
>>> casts.iloc[3:4]
                title  year    name   type            character   n
3  Secret in Their Eyes  2015  $hutter  actor       2002 Dodger Fan
NaN
```

isNull

```
>>> c = casts
>>> c['n'].isnull().head()
0    False
1    False
2    False
3     True
4     True
```

NotNull

```
>>> c['n'].notnull().head()
0    True
1    True
```

Passing condition to get null value

```
>>> c[c['n'].isnull()].head(3)
                        title  year     name   type             character   n
3    Secret in Their Eyes  2015  $hutter  actor        2002 Dodger Fan NaN
```

Fill default value using fillna

```
>>> c_fill = c[c['n'].isnull()].fillna('NA')
>>> c_fill.head(2)
```

String Operations

```
>>> t = titles
>>> t[t['title'] == 'Maa']
      title  year
38880   Maa  1968
```

String Operations

```
>>> t[t['title'].str.startswith("Maa ")].head(3)
                title  year
19     Maa Durga Shakti  1999
3046      Maa Aur Mamta  1970
```

Count Values

```
>>> t['year'].value_counts().head()
2016    2363
2017    2138
```

Plots

```
>>> import matplotlib.pyplot as plt
>>> t = titles
>>> p = t['year'].value_counts()
>>> p.plot()
<matplotlib.axes._subplots.AxesSubplot object at 0xaf18df6c>
>>> plt.show()
```

Sort and Plot

```
>>> p.sort_index().plot()
<matplotlib.axes._subplots.AxesSubplot object at 0xa9cd134c>
>>> plt.show()
```

GroupBy same as values_count()

```
>>> cg = c.groupby(['year']).size()
>>> cg.plot()
<matplotlib.axes._subplots.AxesSubplot object at 0xa9f14b4c>
>>> plt.show()
```

Group By with multiple values

```
>>> cf.groupby(['year', 'title']).size().head()
year  title
2003  The In-Laws                    1
```

Group By with Max Rating

```
>>> c.groupby(['year']).n.max().head()
year
1912     6.0
1913    14.0
```

Group By with Min Rating

```
>>> c.groupby(['year']).n.min().head()
year
1912    6.0
1913    1.0
```

Group By with Mean Rating

```
>>> c.groupby(['year']).n.mean().head()
year
1912    6.000000
1913    4.142857
```

Group By with custom field

```
>>> # decade conversion : 1985//10 = 198, 198*10 = 1980
>>> decade = c['year']//10*10
>>> c_dec = c.groupby(decade).n.size()
>>>
>>> c_dec.head()
year
1910     669
1920    1121
```

Create new dataframe based on type

```
>>> c = casts
>>> c_decade = c.groupby( ['type', c['year']//10*10] ).size()
>>> c_decade
>>> c_decade.unstack()
```

Plot new data frame

```
>>> c_decade.unstack().plot() #unstack(0) to plot side by side
<matplotlib.axes._subplots.AxesSubplot object at 0xb1cec56c>
>>> plt.show()
>>> c_decade.unstack().plot(kind='bar')
<matplotlib.axes._subplots.AxesSubplot object at 0xa8bf778c>
>>> plt.show()
```

Merge Data Frames

```
c_amelia = casts[casts['title'] == 'Amelia']
c_amelia
release = pd.read_csv('release_dates.csv',index_col=None)
release = release[release['title'] == 'Amelia']
c_amelia.merge(release).head()
```

Merge data with itself

```
c.merge(casts, on=['title', 'year']).head()
```

Set and Reset Index

```
c.set_index('title')
c.set_index(['title','Year'])
c.reset_index('year')
```

Data Preprocessing

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Data Preprocessing

```
# Importing the dataset
dataset = pd.read_csv('Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 3].values
```

Data Preprocessing

```
# Taking care of missing data
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
imputer = imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
```

Data Preprocessing

```
# Encoding categorical data
# Encoding the Independent Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 0] = labelencoder_X.fit_transform(X[:, 0])
onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()
```

Data Preprocessing

```
# Encoding the Dependent Variable
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)
```

Data Preprocessing

```
# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)
```

Data Preprocessing

```
# Feature Scaling
"""from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""
```