**Python For Network Analysis and Machine Learning:**

**Pandas Basics:**

**Basic libraries:**

import pandas as pd *Data manipulation, cleaning and munging.*

import numpy as np *Mathematical functions on array type objects*

import matplotlib as plt *Data visualization and plotting*

**Create Data:**

* x = 1,2,3,4,5
* y=['one','two','three','four','five']
* *pd.Series(<data>, <index>)*
  + Series1 = pd.Series(x)
  + Series2 = pd.Series(x,index=y)
  + Series3 = pd.Series(np.random.random(100))
  + Series4 = pd.Series({'2001: A Space Odyssey': 'Hal', 'Star Wars': 'Darth Vader', 'Avengers':'Loki'})
* *pd.DataFrame(<data>,<index>,[<column names>]...)*
  + Data1 = pd.DataFrame(Series2)
  + Data2 = pd.DataFrame(Series2,columns=['Value1'])
  + Data3 = pd.DataFrame(Series4, columns=['Villain'])

**Import Data:**

* pd.read\_csv(<file\_name>,sep=,names=['<column names>']) → read CSV into data frame.
  + Note: pandas libary contains several functions for easly importing different data types including pd.read\_excel, pd.read\_json, pd. pd.read\_sql,pd.read\_html, and many more.
* <data> = read\_from\_regex(r=<pattern>,input='<file>',columns=['<columnnames>'])

**Referencing Series:**

* Series1[0] → returns: 1
* Series1[0:3] → returns: 1,2,3
* Series4['Star Wars'] → returns: 'Darth Vader'
* Series4[['Star Wars', '2001: A Space Odyssey']] → returns: 'Darth Vader' 'Hal’
  + *Note the double brackets when indexing on a list.*
* Series3.head() → *Returns the first x records in a series, default 5.*
* Series3.tail(10) → *Returns the last 10 records in a series, default 5.*

**Referencing Data Frames:**

* *<DataFrame>.loc[<index>]*
  + Data3.loc['Avengers'] → Returns row at index Avengers
  + Data3.loc[['Avengers','StarWars']] → Returns Avengers and StarWars Rows
* *<DataFrame>.sample[n]*
  + Data3.sample(2) → samples two rows from Data3

**Manipulating Series:**

* *<NewSeries> = <Series><value>*
  + *Series1\_2 = Series1+2*
  + *Series1\_combined = Series1.add(Series3)*
  + *function1 = lambda x: x+2*
  + *Series1\_3 = Series1.apply(function1)*
* *<NewSeries> = <Series>.str.contains(<regex\_pattern>)*
  + *Series4\_1 = Series4.str.contains('Hal') → Returns a boolean vector if string contains 'Hal'*
  + *Series4\_2 = Series4[Series4.str.contains('Hal')] → Returns new series only where the conditions are met.*
  + *Series4\_2 = Series4[Series4.str.contains('Hal')].reset\_index(drop=True) →Creates new index for filtered data.*
* *<Series>*.sort()  *→ sorts by value*
* *<Series>*.sort\_index() *→ sorts by index*
* *<Series>*.sort(ascending=False) *→ sorts in descending order*
* *<Series>*.append(<values>) *→ adds values to series, can be single value, list, series etc.*
* *<Series>[<index#>] = <value> → sets Series index to value.*
* *<Series>.*dropna()  *→ drops null values from series*
* *<Series>.*fillna(value=<x>) *→ fills null values with x*

**Manipulating DataFrames**

* *<DataFame>['<newcolumn>'] =* Value
  + Data2['new'] = 0 *→ sets every value in column to 0*
  + Data2['new'] = Data2['Value1']\*3 *→ multiplies each value in 'Value1' by 3.*
  + Data3['Random'] = np.random.random(len(Data3)) → *creates new column with random numbers.*
  + Data3['StarWars'] = Data3['Villain'].str.contains('Darth') *→ returns Data3['StarWars'] as boolean if Villian column contains the phrase 'Darth'*
* *<DataFrame>.*sort(column=*'<Column Name>') → defaults to Data Frame index*
* *<DataFrame>.*sort\_index()
* *<DataFrame>.*sort(ascending=False)
* *<DataFrame>.*T *→ transposes data*
* *<DataFrame>.*sum(axis = 0) *→ provides sums of columns*
* *<DataFrame>.*sum(axis = 1) *→ provides sums of rows*
* *<DataFrame>*.dropna()  *→ drops some or all of NA from data frame.*

**Filtering Series:**

* *<Series>[<conditions>]*
  + Series1\_1 = Series1[Series1 > 2]  *→ Returns new series where all values are greater than 1*

**Filtering Dataframes**

* *<DataFrame> = <DataFrame>[['Column1','Column2']]*
  + Movies= Data3[[‘Villain’,'Random']] → *Returns new dataframe only with villian and random columns*
* *<DataFrame> = <DataFrame>[<Dataframe>['<Column Name>] <condition>]*
  + Filtered\_Movies = Movies[Movies['Random'] > .5] *→ Returns new dataframe where “Random’ is greater than .5*

**Merging Data Sets:**

* pd.concat(*[<series1>,<series2>]*, ignore\_index = True) *→ union/append/add series or datasets together* 
  + pd.merge*(<data1>, <data2>*) *→ basic merge, defaults to inner join.*
  + pd.merge(*<data1>, <data2>*, how='<merge type>') *→ change merge to 'outer', 'left' or 'right'.*
  + pd.merge(*<data1>, <data2>*, how='<merge type>', on=<columns>, ignore\_index = True) *→ if unspecified data is merged on indexes.*

**Data Exploration:**

* *Series*.abs() → *Returns absolute value of series elements*
* *Series*.count() → *counts total elements in series*
* *Series*.max() → *Returns series max*
* *Series*.min() → *Returns series min*
* *Series*.mean() → *Returns series mean*
* *Series*.median() → *Returns series median*
* *Series*.mode() → *Returns series mode*
* *Series*.quantile([q)] → *Returns the 'q' quantile of a series*
* *Series*.sum() → *Returns the sum of a series*
* *Series*.std() → *Returns the standard deviation of series*
* *Series*.var() → *Returns the variance of series*
* *Series*.describe() → *Returns series sum count, mean, standard deviation, min, quartiles, max for quantative data. Returns series count, unique elements, most common element, and the frequency of the most common element.*
* *Series*.unique() → *returns unique values*
* *Series*.value\_counts() → *Returns each unique value and counts the number of occurrences.*
* <*Series1>.*corr*(<Series2>) → returns correlation coefficient*

**Matplotlib and Charts:**

**Set Formating:**

%matplotlib inline

pd.options.display.mpl\_style = 'default'

**Plot Data:**

* *<Series>.*hist() *→ histogram of data.*
* *<Series>.*plot() → *line chart*
* *<Series>.*plot(kind=<'chart\_type''>) → kind='bar','barh' (horizontal bar), 'box', 'area', 'scatter','hexbin', 'pie'.
* *<Series>.*plot(kind=bar, stacked=True) → stacked bar chart
* plt.title('title') → give plot a title
* plt.xlabel('label') → label x axis
* plt.ylabel('ylabel) → label y axis
* *<DataFrame>.*plot*(*x=*'<column>',* y=*'<column>')*
  + *<DataFrame>.*plot(kind='scatter', color='blue', label='*<group\_name>*')
* *<Plot\_ObjectName>.*get\_figure().savefig('*<filename.png>*')

**Ip Address Library:**

* import ipaddress
  + ipaddress.ip\_address('192.168.0.1') → *converts string object to the ipaddress object (ipv4 or ipv6)*
  + ipaddress.ip\_network('192.168.0.1/28') → *converts string object to ip network object.*
  + ipaddress.ip\_address('192.168.0.1').is\_private → returns boolean for private or public addresses.

**User Agent Library:**

* from user\_agents import parse
  + parse('*<UserAgentString>*') → breaks out useragent string
  + user\_agent.browser → returns user agent browser
  + user\_agent.browser.family → returns browser family ie. Mozilla
  + user\_agent.browser.version → returns version number as integers.
  + user\_agent.browser.version\_string → returns string
  + user\_agent.os → returns operating system
  + user\_agent.os.family → returns OS family i.e. iOS
  + user\_agent.os.version → returns version number as list of ints.
  + user\_agent.os.version\_string → returns version number as string.
  + user\_agent.is\_pc() → True if user agent is PC
  + user\_agent.is\_mobile() → True if user agent is mobile
  + user\_agent.is\_tablet() → True if user agent is tablet
  + user\_agent.is\_touch\_capable → True if user agent is touch capable.
  + user\_agent.is\_bot → true if user agent is bot.