Basic Python Syntax

List

```
lst=[1,2,3,4]
lst.append(5)
print(lst)
lst.extend([6,7,8])
print(lst)
lst.pop()

print(sum(lst))
lst.pop()
print(lst)
# lst.pop(3)
# print(lst)

lst2=[1,1,3,4,4,6,7,8,9,10]
print(lst2.count(1))
print(lst.index(1))
print(lst*2)
```

Sets

```
# No duplicates allowed
set_var=set(lst2)
print(set_var)
set_var.add(2)
set_var.remove(2)
print(set_var)
# set_var[1] or set_var[0] not allowed

set1=set(lst)
set2=set(lst2)
set3=set1.intersection(set2)
set4=set1.union(set2)
print(set3)
print(set4)
```

Dictionary

```
dictionary = {'CSE 101':'Structured Programming Language', 'CSE
107':'Object Oriented Programming', 'CSE 105':'Discrete Mathematics'}
print(dictionary)

for key in dictionary.keys():
    print(key)

for value in dictionary.values():
    print(value)
```

```
for item in dictionary.items():
    print(item)

for key,value in dictionary.items():
    print(key,value)
```

Functions in Python

```
```

Built-in Libaraies

Numpy

```
import numpy as np
a=np.array([1,2,3,4])
print(a)
print(type(a))
print(a.shape)
print(a.reshape(-1,1).shape)

b=np.array([5,6,7,8])
c=np.array([9,10,11,12])
array = np.array([a,b,c])
print(array)
print(array.shape)
print(array.reshape(4,3))
print(array[-2:,-2:])

# generate random numbers within a range
arr = np.arange(0,10,step=2)
print(arr)
np.linspace(0,10,50) # 50 evenly spaced numbers between 0 and 10

arr1 = np.arange(0,10)
# copy array
arr2 = arr1.copy()
arr2[0:5]=100
print(arr1)
print(arr2)
# conditional selection
arr1[arr1>5]
```

```
<IPython.core.display.Javascript object>
```

```
[0 1 2 3 4 5 6 7 8 9]
[100 100 100 100 100   5   6   7   8   9]
```

```
array([6, 7, 8, 9])
```

```
print(np.ones((2,3),dtype=int))
print(np.random.rand(2,3)) # uniform distribution
print(np.random.randn(2,3)) # normal distribution
print(np.random.randint(0,100,10)) # 10 random integers between 0 and
100
print(np.random.random_sample((1,5))) # 5 random numbers between 0 and
1
```

Pandas

```
import pandas as pd

df =
pd.DataFrame(np.arange(0,20).reshape(5,4),index=['A','B','C','D','E'],
columns=['W','X','Y','Z'])
df.head()

## Accessing the elements
## 1. loc 2. iloc
print(df.loc['A'])
print(type(df.loc['A']))
print(df.iloc[:2,:2])
print(df.iloc[1:3,1:3])

# convert dataframes to numpy array
print(df.iloc[:,:].values)
print(df.isnull().sum())
print(df['W'].value_counts())
print(df['W'].unique())

# read csv file
df = pd.read_csv('mercedesbenz.csv')
df.head()
df.info()
df.describe()

print(df['X0'].value_counts())
print(df[df['y']>100])
```

Working with CSV

```
# create csv from string
from io import StringIO, BytesIO
data = ('col1,col2,col3\n'
        'x,y,1\n'
        'a,b,2\n'
        'c,d,3')
df = pd.read_csv(StringIO(data))
print(df.head())
df2 = pd.read_csv(StringIO(data),usecols=['col1','col3'])
```

```
print(df2.head())
df.to_csv('Test.csv')

# Specifying columns data types
data = ('a,b,c,d\n'
        '1,2,3,4\n'
        '5,6,7,8\n'
        '9,10,11')
df = pd.read_csv(StringIO(data),dtype=object)
print(df.head())
print(df['a'][1])
df = pd.read_csv(StringIO(data),dtype={'b':int,'c':float,'a':'Int64'})
print(df.head())
print(df.dtypes)

# Index columns and training delimiters
data = ('index,a,b,c\n'
        '4,apple,bat,5.7\n'
        '8,orange,cow,10')
df = pd.read_csv(StringIO(data))
print(df.head())
df = pd.read_csv(StringIO(data),index_col=0)
print(df.head())
data = ('a,b,c\n'
        '4,apple,bat,\n'
        '8,orange,cow,')
df = pd.read_csv(StringIO(data),index_col=False)
print(df.head())

# Combining usecols and index_col
data = ('a,b,c\n'
        '4,apple,bat,\n'
        '8,orange,cow,')
df = pd.read_csv(StringIO(data),usecols=['b','c'],index_col=False)
print(df.head())

# Quoting and Escape characters. Very useful in NLP
data = 'a,b\n"hello, \\"Bob\\", nice to see you",5'
df = pd.read_csv(StringIO(data),escapechar='\\')
print(df.head())
```

Reading JSON

```
data='{ "employee_name": "James", "email": "james@gmail.com",
"job_profile": [{"title1":"Team Lead","title2":"Sr. Developer"}]}'
df1 = pd.read_json(data)
df1.head()
df2 = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-
databases/wine/wine.data',header=None)
df2.head()
```

```
# convert json to csv
df1.to_csv('employee.csv')
df2.to_csv('wine.csv')

# convert json to different json formats
# df1.to_json()
# df1.to_json(orient="index")
df1.to_json(orient="records")
df2.to_json(orient="records",lines=True)
```

Reading HTML

```
url_mcc = 'https://en.wikipedia.org/wiki/Mobile_country_code'
dfs = pd.read_html(url_mcc,match='Country',header=0)
dfs[0]
```

Reading Excel

```
fw = pd.read_excel('Farewell List.xlsx',header=0)
fw
```

Pickling

```
fw.to_pickle('fw_pickle')
fw_pickle = pd.read_pickle('fw_pickle')
fw.head()
```

Matplotlib

```
# Matplotlib
import matplotlib.pyplot as plt

x = np.arange(0,10)
y = np.arange(11,21)

plt.scatter(x,y,c='r')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('2D Graph')

y1 = x*x
plt.plot(x,y,'bo--',linewidth=2,markersize=8)
plt.legend(['X','X^2'])

## Creating subplots
plt.subplot(2,2,1)
plt.plot(x,y,'ro-')
plt.subplot(2,2,2)
plt.plot(x,y,'g*--')
```

```
## Compute x and y coordinates for points on sine and cosine curves
x = np.arange(0,4*np.pi,0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
plt.subplot(2,1,1)
plt.plot(x,y_sin,'r-')
plt.title('Sine')
plt.subplot(2,1,2)
plt.plot(x,y_cos,'g-')
plt.title('Cosine')

### Bar plot
x = [2,8,10]
y = [11,16,9]

x2 = [3,9,11]
y2 = [6,15,7]
plt.bar(x,y)
plt.bar(x2,y2,color='g')
plt.title('Bar Graph')
plt.xlabel('X axis')
plt.ylabel('Y axis')

### Histograms
a = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])
plt.hist(a,bins=20)
plt.title('Histogram')
plt.xlabel('X axis')
plt.ylabel('Y axis')

### Box plot
data = [np.random.normal(0,std,100) for std in range(1,4)]
plt.boxplot(data,vert=True,patch_artist=True)
plt.title('Box Plot')

### Pie chart
labels = 'Python','C++','Ruby','Java'
sizes = [215,130,245,210]
colors = ['gold','yellowgreen','lightcoral','lightskyblue']
explode = (0.05,0.05,0.05,0.05)
plt.pie(sizes,explode=explode,labels=labels,colors=colors,autopct='%1.
1f%%',shadow=True)
plt.axis('equal')
```

Seaborn

```
import seaborn as sns

df = sns.load_dataset('tips')
df.head()
```

```
### Correlation with Heatmap
print(df.corr())
sns.heatmap(df.corr())
```

Joint plot: Univariate Analysis

```
sns.jointplot(x='tip',y='total_bill',data=df,kind='hex')
sns.jointplot(x='tip',y='total_bill',data=df,kind='reg')
```

Pair plot: Bivariate analysis

```
sns.pairplot(df)
sns.pairplot(df,hue='sex')
```

Dist plot: helps to check the distribution of the columns feature

```
sns.displot(df['tip'],kde=True)
sns.displot(df['tip'],kde=False,bins=10)
```

Categorical Plots

```
# count plot: show the counts of observations in each categorical bin
using bars.
sns.countplot(x='sex',hue='smoker',data=df)
# sns.countplot(x='day',data=df)

### Bar plot
sns.barplot(data=df,x='day',y='tip')

### Box plot
sns.boxplot(data=df,x='day',y='tip',hue='smoker')

### Violin plot: combination of boxplot and kde plot
sns.violinplot(data=df,x='day',y='tip',hue='smoker',split=True)
```

Practice

```
iris_df = sns.load_dataset('iris')
iris_df.head()
# print(iris_df['species'].value_counts())

print(iris_df.corr())
sns.heatmap(iris_df.corr(),annot=True)

sns.jointplot(x='petal_length',y='petal_width',data=iris_df,kind='reg'
)
sns.jointplot(x='petal_length',y='petal_width',data=iris_df,kind='hex'
)

sns.pairplot(iris_df,hue='species')

sns.displot(iris_df['petal_length'],kde=True)
```