

Python

START

```
import quandl #API for finance/econ data
import numpy as np #math functions
import pandas as pd #dataframes
import matplotlib.pyplot as plt #plotting
import seaborn as sns
*****import pandas_datareader as web
```

DATA

DATA: getting or sending

Getting data from google

```
data=quandl.get('WIKI/AAPL', auth_token='ADvyLkjofoE4PsgxtUz')
*****data = web.DataReader("NKE","google")
```

Getting Annual / quarterly / monthly data from google

```
econ_quarterly = quandl.get(["FRED/GDP","FRED/UNRATE"],collapse =
"annual",auth_token="62LSmkDHdrcUQvyYrewV")
```

Getting multiple data from google

```
data=quandl.get(['WIKI/AAPL','WIKI/GOOGL','WIKI/IBM','WIKI/MSFT',
'WIKI/FB'],auth_token='ADvyLkjofoE4PsgxtUz')
```

Getting data from computer

```
data=pd.read_csv("People.csv") (in the AD student so not to find address)
```

Getting data from github with link

```
data=pd.read_csv('https://raw.githubusercontent.com/HrantDavtyan/Data_Scraping/master/Week%202/Datasets/Titanic.csv')
```

Sending data from python to excel

```
df.to_excel("econ.xlsx")
```

Again read from excel

```
econ_read = pd.read_excel("econ.xlsx")
```

```
econ_read.head()
```

```
df = pd.read_excel("econ.xlsx", sheet_name=1)
```

Sending data from python to csv

```
econ_quarterly.to_csv("econ_quarterly.csv")
```

DATA: standard functions

```
data.head(3)
```

```
data.tail(3)
```

```
data.info()
```

```
data.describe()
```

```
data.mean() / data_1.Value.mean()
```

```
data.median()
```

```
range=data.Open.max()-data.Open.min()
```

```
data.mode()
```

If we have several modes use:

```
data_1.Value.astype("int").mode()
```

```
data.std()
```

If we want to count the relative percentage change

```
pct_change=data_1.pct_change()
```

```
data_num.Survived.value_counts()/len(data_num)*100
```

If we want to add new_column=small_data['Open'].pct_change()

```
small_data['Open_change']=small_data.Open.pct_change()
```

If we want to see the maximum value of given column

```
data_1.Value.max()
```

If we want to filter the data

```
condition=data.satisfaction_level>0.5
```

```
Data[condition]
```

If we want to have the Value of date(index of 1991)

```
econ["1991"]
```

If we want to have the date of maximum value(column)

```
data_1.loc[data_1['Value'].idxmax()]
```

```
data_1.loc[data_1['Value']==data_1.Value.max()]
```

If we want to change the column into index

```
df = df.set_index("Date")
```

If we want to drop na-s

```
data=data.dropna()
```

If we want to drop a column (if axis=0, will try to drop a row)

#axis =0 for row, axis=1 for column

```
df_new = df.drop(["FRED/GDP - Value"],axis=1)
If we want to apply some function to all the columns
df.apply(np.std)
small_data.Open.apply(classifier_2)
If we want to filter values that are higher than median
#with making =, we ask for the values, w/o =, we will see True/False
unrate[unrate>unrate.median()]
```

DATA: conditions

```
If we have a range (160,170) we need and and
condition_1=data_2.median()>160
condition_2=data_2.median()<170
data_2.median()[condition_1 & condition_2].count()
```

```
If we have a range x>170, x<160 we need or or
condition_1=data_2.median()<160
condition_2=data_2.median()>170
data_2.median()[condition_1 | condition_2].count()
```

Class Case

```
cond1=data_full.Sex==1
cond2=data_full.Sex==0
```

```
data_males=data_full[cond1]
males_survived=data_males.Survived.value_counts()[1]
```

```
data_females=data_full[cond2]
females_survived=data_females.Survived.value_counts()[1]
```

```
males_survived/females_survived
```

```
females_survived/males_survived
```

```
If we have a range (160,170)
# (-0.8,-0.4), both conditions are satisfied
# &-and |-or
condition_1=small_data.Open_change>-0.4
condition_2=small_data.Open_change<-0.8
small_data.Open_change[condition_1 & condition_2]
```

```
small_data.Open_change[condition_1 | condition_2] #count()- will say the total count
```

DATA: columns

data which has many columns and a column titled **my_col** among them, which is the 3rd one:

- `my_column = data.my_col`
- `my_column = data["my_col"]`
- `my_column = data.iloc[:,2]`

If we want to choose first column and first 10 rows by index

```
econ.iloc[0:10,0]
```

If we want to choose all rows and 2nd column only and saving in a new variable

```
unrate = df.iloc[:,1]
```

If we want to choose several different columns we put a list for selecting several columns

```
data_1=data.iloc[:, [1,2,5]]
```

If we want to choose all rows and given names of column

```
data_opening=data.loc[:,["WIKI/AAPL - Open", "WIKI/GOOGL - Open", "WIKI/IBM - Open",  
"WIKI/MSFT - Open", "WIKI/FB - Open"]]
```

If we want to choose a column with the name

```
data['Open']
```

If we want to rename a column

```
#df=df.rename(columns={'old_name':'new_name'})
```

```
small_data=small_data.rename(columns={'Pct_change':'percentage_change'})
```

Or easiest, just copy paste and change

```
small_data.columns=['Open', 'High', 'Low', 'Close', 'Pct_change', 'Classes']
```

DATA: plotting

```
plt.plot(data)
```

```
data.plot()
```

If we want to plot data in green and with --es

```
plt.plot(data,"g--")
```

```
plt.plot(data.Open, "y")
```

```
plt.show()
```

If we want to define function that will plot the given name of column from the table

```
def my_plot(x):
```

```
    return plt.plot(df[x])
```

```
->>my_plot("FRED/UNRATE - Value")
```

If we want to plot data for distribution with seaborn

```
sns.distplot(data)
```

If we want to plot data with histogram

```
data_num.Fare.hist()
```

```
data_2.plot(kind='hist')
```

If we want to define function that will get the needed data and plot

```
def stockticker(x):
```

```
    return quandl.get(x,auth_token='ADvyLkjofdoE4PsgxtUz').plot()
```

```
->>stockticker("WIKI/FB")
```

If we want to plot the histogram of prices with only 15 bins, some specified color and add title for the graph as well as x and y axis

```
data_2.plot(kind='hist', bins=15, color='orchid')
```

```
plt.xlabel('Price')
```

```
plt.ylabel('Frequency')
```

```
plt.title('Property price ')
```

or

```
data.plot(x='average_monthly_hours', y='satisfaction_level', kind='scatter')
```

If we want to plot first two columns separately

```
for i in range(0,2):
```

```
    sns.distplot(data.iloc[:,i])
```

```
plt.show()
```

DATA:working on tables

*If we want to separate the data to objects and numerical
select and categorize numeric and text columns*

```
data_obj=data.select_dtypes(include=['object'])
```

```
data_num=data.select_dtypes(exclude=['object'])
```

*If we want to see the unique variables for column or list
#unique function exists only for columns*

```
data_obj.iloc[:,0].unique()
```

If we want to see the count of the given variables

```
data.churn.value_counts()
```

If we want to see how many columns and rows smth like info()

```
data.shape
```

If we want to create the range which automatically will calculate unique for all columns

#for loop is certainly better(kberri sagh columnneri meji uniqueness(sales, marketing))

```
for i in range(0,2):  
    print(data_obj.iloc[:,i].unique())
```

Another case if we want to see the columns which unique variables are less than 11

#data=data.columns: just add x and the column names will appear also

```
for x in data: #data=data.columns:  
    unique_values=data[x].unique()  
    num_unique=len(unique_values)  
    if num_unique<11:  
        print(x,unique_values)
```

Categorizing:

```
data.salary=data.salary.astype('category').cat.reorder_categories(['low', 'medium', 'high']).cat.codes  
data.salary.unique()----(['low', 'medium', 'high'])
```

If we want to create dummies than join it to our table. We should drop current column

```
departments=pd.get_dummies(data.department,prefix="department")  
data=data.join(departments)  
data=data.drop('department',axis=1) #dropping is required
```

If we want to look different column with index we choose pivot tables with different functions

```
data_pivot=pd.pivot_table(data=data,index=["salary"],values='satisfaction_level',aggfunc=np.average)  
data_pivot=pd.pivot_table(data=small_data,index=["Classes"],values='Open',aggfunc=np.std)  
Data_pivot
```

Using where instead of excel if, we created a new column

```
small_data['Classes']=np.where(small_data.Open>small_data.Open.median(),"Higher","Lower")  
data_obj.Sex=np.where(data_obj.Sex=="male","0","1")
```

Embarked getting smth similar to dummies

```
data_obj.Embarked.unique()
```

Joining two tables

```
data_full=data_num.join(data_obj)
```

General functions

One can merge strings in python by just putting + in between.

```
full_name = first_name + " " + last_name
my_list = [first_name, last_name, my_var]
print(my_list)
['Hrant ', 'Davtyan', 5.05]
```

print() - prints the content,

type() - checks the type of the given argument (list or string etc.)

range() - creates range of integers (e.g. range(5) will provide integers from 0 to 5)

If we want to select #selects very first element of the list

```
names[0] #selects very first element of the list
names[0][0] - Selects the first letter of the first element
```

Shows how many items in the list

len(names)-

Join the given to the list

```
names.append("Joroh")
```

Sorts the names in alphabetical order

```
names.sort()
```

Gets all the letters up / low

```
full_name = full_name.upper()
```

```
full_name.lower()
```

Gets the first letter up

```
full_name = full_name.capitalize()
```

Finds the count of the given in a specific place

```
full_name.find("T")
```

Give range and define the steps it should count next

```
range(1,10,2) # range(start,end,step)
```

Gives the square root

```
np.sqrt(25)
```

Shows how many digits:

```
len(str(55))
```

Define functions

```
1. def sq(num):
```

```
    return num**2
```

```
2. def superify(names):
```

```
    return ("Super"+" " + names)
```

```
3. def gender_detector(name):
```

```
    if name[0]=="J":
```

```
        return "Male"
```

```
    elif name[0]=="K":
```

```
        return "Female"
```

```
    else:
```

```
        return "Other"
```

```
4. def converter(x):
```

```
    if x[-3:]=="EUR":
```

```
        value=int(x[:3])*565
```

```
        print(value, " AMD")
```

```
    elif x[-3:]=="USD":
```

```
        value=int(x[:3])*478
```

```
        print(value, " AMD")
```

```
    else:
```

```
        print("Unknown")
```

```
5. def classifier(x):
```

```
    if x>0:
```

```
        output="Higher"
```

```
    else:
```

```
        output="Lower"
```



```
return output
```

```
6.def classifier_2(x):
```

```
    if x>small_data.Open.median():
```

```
        output="H"
```

```
    elif x==small_data.Open.median():
```

```
        output='E'
```

```
    else:
```

```
        output="L"
```

```
    return output
```

```
7.def stockticker(x):
```

```
    return quandl.get(x,auth_token='ADvyLkjofdoE4PsgxtUz').plot()
```

If we want to choose every 3rd item with for loop

```
for i in range(len(name_list)):
```

```
    if i%3==2:
```

```
        print(name_list[i])
```