The contents contained in this Jupyter notebook may be freely distributed for teaching-learning purpose. When more than 50% of this notebook is used for a presentation or distribution it may be cited as follows:

Python Tutorial: developed by Naveen Kumar (Professor, Department of Computer Science), University of Delhi, Ankit Rajpal (Assistant Professor, Department of Computer Science), Sheetal Rajpal (Assistant Professor, Dyal Singh College).

In [ ]:
```
#@title
from google.colab import drive
drive.mount('/content/drive/')
import os
os.chdir('/content/drive/My Drive/DAVWorkshop')
```

Pandas

- High-level data manipulation and analysis tool developed by Wes McKinney.
- Built on the Numpy package and its key data structure is called the DataFrame.
- DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

In [ ]:
```
dir(pd)
```

In [ ]:
```
import pandas as pd
```

Pandas Series

Series is a *one-dimensional labeled array* capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).

In [ ]:
```
# Creating Pandas Series using lists

animals = ['Tiger', 'Bear', 'Lion']
animalsSeries = pd.Series(animals)
print(animalsSeries)
```

In [ ]:
```
print(type(animalsSeries))
```

In [ ]:
```
percentage = [95, 99, 92, 99.5]
percSeries = pd.Series(percentage)
print(percSeries)
```

In [ ]:
```
percSeries + 0.5
```

In [ ]:
```
percentage = [95, 99, 92, 99.5, None]
percSeries = pd.Series(percentage)
print(percSeries)
```

In [ ]:
```
percentage = [95, 99, 92, 99.5, None]
percSeries = pd.Series(percentage, index = [101, 102, 103, 104, 105])
```

```
print(percSeries)
```

In [ ]:

```
percSeries +0.5
```

In [ ]:

```
student = ['Raman', 102, 99]
studentSeries = pd.Series(student)
studentSeries
```

In [ ]:

```
percSeries
```

Indexing

In [ ]:

```
# Retrieving percentage of second student
percSeries[1]
```

Using keyword argument index to explicitly specify indexes

In [ ]:

```
#Another way of creating a series
animals = ['Tiger', 'Bear', 'Lion']
animalSeries = pd.Series(animals, index = ['One', 'Two', 'Three'])
# Index of a Series can be of String type
print(animalSeries)
```

In [ ]:

```
animalSeries['Three']
```

In [ ]:

```
animals = ['Tiger', 'Bear', 'Lion']
nations = ['India', 'America', 'Sri Lanka']
nationalAnimals = pd.Series(animals, index = nations)
nationalAnimals
```

In [ ]:

```
nationalAnimals.index
```

In [ ]:

```
nationalAnimals.values
```

In [ ]:

```
newSeries =pd.Series(nationalAnimals.index, index=nationalAnimals.values)
```

In [ ]:

```
newSeries
```

In [ ]:

```
percentage = [95, 99, 92, 99.5, None]
rollNumbers = [101, 102, 103, 104, 105]
percSeries = pd.Series(percentage, index = rollNumbers)
print(percSeries)
```

In [ ]:

```
percentage = [95, 98, 99, 97, 92, 99.5, None]
rollNumbers = [101, 101, 102, 102, 103, 104, 105]
percSeries = pd.Series(percentage, index = rollNumbers)
print(percSeries)
```

In [ ]:

```
percSeries[101]
```

In [ ]:

```
# Retrieving percentage of student with roll number 102
percSeries[102]
```

In [ ]:

```
percSeries
```

In [ ]:

```
percentage = [95, 99, 92, 99.5, None]
rollNumbers = [101, 102, 103, 104, 105]
percSeries = pd.Series(percentage, index = rollNumbers)
print(percSeries)
```

Using **loc** and **iloc** to access elements of series:

```
loc: refers to actual label/index
iloc: refers to integer location
```

In [ ]:

```
#Querying a Series based on index label
print(percSeries.loc[102])
print(percSeries[102])
```

In [ ]:

```
percSeries
```

In [ ]:

```
#Querying a Series based on index location
print(percSeries.iloc[2])
print(percSeries.iloc[-2])
```

Typecasting

Type of the elements of the series can be casted to another type using astype

In [ ]:

```
numbers = [1, 2, 3.2, 4.9, 5.5]
numSeries = pd.Series(numbers)
print('numSeries Before:')
print(numSeries)
numSeries = numSeries.astype(int)
print('\n\nnumSeries after:')
print(numSeries)
```

Creating Pandas Series using dictionary

In [ ]:

```
sportsDict = {'Archery': 'Bhutan',
              'Golf': 'Scotland',
              'Sumo': 'Japan',
              'Hockey': 'India'}          #Creating Series from Dictionary

sports = pd.Series(sportsDict)
print(sports)
```

In [ ]:

```
print(sports['Sumo'])
```

In [ ]:

```
sports.values
```

In [ ]:

```
sports.index
```

Creating a new series object with index and value part swapped with each other

In [ ]:

```
pd.Series(sports.index, index = sports.values)
```

Operations on Series

In [ ]:

```
sectionNames = ['A','B','C','D','E','F','G']
```

In [ ]:

```
sectionStrength = [34,50,60,54,45,40, 50]
```

In [ ]:

```
sections = pd.Series(sectionStrength, index = sectionNames)
print(sections)
```

In [ ]:

```
sections= sections+5
sections
```

QUESTION. What is the strength of Section B?

In [ ]:

```
#####################  TO DO ##################
```

In [ ]:

```
sections.head(3)
```

In [ ]:

```
#help(sections.head)
```

In [ ]:

```
sections.head(3)
```

In [ ]:

```
sections.tail()
```

In [ ]:

```
#Adding values of a series  using for loop:
def getClassStrength(sections):
    '''
    Objective: to compute the strength of the class across the sections
    Input:
        sections: series comprising sections of a class
    Output:
        classStrength: total number of students across the sections
    '''
    '''
    approach:
        initialize classStrength = 0
        iterate through iteritems
    '''
    classStrength = 0
    for item in sections:
        classStrength += item
    return classStrength

getClassStrength(sections)
```

In [ ]:

```
#Using numpy functions is much faster than for loops
print(sections.sum())
print(sections.mean())
```

In [ ]:

```
dir(pd.Series)
```

In [ ]:

```
sections
```

In [ ]:

```
#adds 5 to each value in Class using broadcasting - It is much faster
sections = sections + 5
sections
```

Appending Series to another Series

In [ ]:

```
#Creating Series using Dictionary
nationalSports = pd.Series({ 'Archery': 'Bhutan',
                            'Golf': 'Scotland',
                            'Sumo': 'Japan',
                            'Taekwondo': 'South Korea'})

#Creating Series using Lists
cricketLovingCountries = pd.Series(['Australia',
                                   'Barbados',
                                   'Pakistan',
                                   'England'],
                                  index = ['Cricket',
                                          'Cricket',
                                          'Cricket',
                                          'Cricket'])
print(nationalSports, "\n\n", cricketLovingCountries)
```

In [ ]:

```
allCountries = nationalSports.append(cricketLovingCountries)
print(allCountries)
```

In [ ]:

```
allCountries['Cricket']
```

Pandas Dataframe

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects.

In [44]:

```
import pandas as pd
data = {'Name': 'Ashish', 'Item Purchased': 'Bread',   'Cost': 22.50}
series = pd.Series(data)
print(series)
```

```
Name                Ashish
Item Purchased       Bread
Cost                  22.5
dtype: object
```

In [45]:

```
import pandas as pd
data = {'Name': 'Ashish', 'Item Purchased': 'Bread',   'Cost': 22.50}
df = pd.DataFrame(data, index=[0])
print(df)
```

```
     Name Item Purchased  Cost
0  Ashish          Bread  22.5
```

In [48]:

```
#Creating a Dataframe from multiple Series: Series indices are promoted to column 

import pandas as pd
purchase_1 = pd.Series({'Name': 'Ashish',
                        'Item Purchased': 'Bread',
                        'Cost': 22.50})
purchase_2 = pd.Series({'Name': 'Nikita',
                        'Items Purchased': 'Vegetables',
                        'Cost': 90.00})
purchase_3 = pd.Series({'Name': 'Vinod',
                        'Item Purchased': 'Milk',
                        'Cost': 75.00})
print(purchase_1,"\n\n", purchase_2,"\n\n", purchase_3)
```

```
Name                Ashish
Item Purchased       Bread
Cost                  22.5
dtype: object

 Name                     Nikita
Items Purchased     Vegetables
Cost                      90.0
dtype: object

 Name               Vinod
Item Purchased      Milk
Cost                75.0
dtype: object
```

In [49]:
```
purchase = pd.DataFrame([purchase_1, purchase_2, purchase_3])             #, in
purchase
```

Out[49]:

	Name	Item Purchased	Cost	Items Purchased
0	Ashish	Bread	22.5	NaN
1	Nikita	NaN	90.0	Vegetables
2	Vinod	Milk	75.0	NaN

In [50]:
```
type(purchase)
```

Out[50]:
```
pandas.core.frame.DataFrame
```

Indexing: To retrieve a particular column of dataframe

In [52]:
```
purchase
```

Out[52]:

	Name	Item Purchased	Cost	Items Purchased
0	Ashish	Bread	22.5	NaN
1	Nikita	NaN	90.0	Vegetables
2	Vinod	Milk	75.0	NaN

In [55]:
```
purchase.Name
```

Out[55]:
```
0    Ashish
1    Nikita
2     Vinod
Name: Name, dtype: object
```

In [61]:
```
#Creating a Dataframe from multiple Series: Series indices are promoted to column

import pandas as pd
purchase_1 = pd.Series({'Name': 'Ashish',
                        'Item Purchased': 'Bread',
                        'Cost': 22.50})
purchase_2 = pd.Series({'Name': 'Nikita',
                        'Item Purchased': 'Vegetables',
                        'Cost': 90.00})
purchase_3 = pd.Series({'Name': 'Vinod',
                        'Item Purchased': 'Milk',
                        'Cost': 75.00})
```

```
print(purchase_1,"\n\n", purchase_2,"\n\n", purchase_3)
purchase = pd.DataFrame([purchase_1, purchase_2, purchase_3])          #, in
purchase
```

```
Name              Ashish
Item Purchased     Bread
Cost                22.5
dtype: object

 Name                Nikita
Item Purchased    Vegetables
Cost                    90.0
dtype: object

 Name              Vinod
Item Purchased     Milk
Cost               75.0
dtype: object
```

Out[61]:

	Name	Item Purchased	Cost
0	Ashish	Bread	22.5
1	Nikita	Vegetables	90.0
2	Vinod	Milk	75.0

In [62]:

```
purchase
```

Out[62]:

	Name	Item Purchased	Cost
0	Ashish	Bread	22.5
1	Nikita	Vegetables	90.0
2	Vinod	Milk	75.0

In [63]:

```
purchase.dtypes
```

Out[63]:

```
Name               object
Item Purchased     object
Cost              float64
dtype: object
```

In [ ]:

```
#Selecting a Column from a Dataframe
customers = purchase['Name']
print(customers)
print(type(customers))
```

In [ ]:

```
#Retrieving items purchased by first two customers
purchase.loc[0:1, 'Item Purchased']
```

Using nested list to create a dataframe

In [65]:

```
grocery = [[22,'Bread'], [90, 'Vegetables'], [75, 'Milk']]
numbers = [1,2,3]

header = ['Cost', 'Item Purchased']
foodCosts = pd.DataFrame(data=grocery, index=numbers, columns=header)
foodCosts
```

Out[65]:

	Cost	Item Purchased
1	22	Bread
2	90	Vegetables
3	75	Milk

In [68]:
```
foodCosts.loc[2, 'Item Purchased']
```

Out[68]:
```
'Vegetables'
```

In [69]:
```
foodCosts.iloc[2, 0]
```

Out[69]:
```
75
```

In [66]:
```
foodCosts['Item Purchased']
```

Out[66]:
```
1         Bread
2    Vegetables
3          Milk
Name: Item Purchased, dtype: object
```

In [67]:
```
foodCosts['Item Purchased'][2]
```

Out[67]:
```
'Vegetables'
```

Using dictionary to create a dataframe

In [ ]:
```
import pandas as pd
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],
        'year': [2000, 2001, 2002, 2001, 2002, 2003],
        'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
frame = pd.DataFrame(data)
frame
```

In [ ]:
```
frame.index
```

In [ ]:
```
frame.values
```

In [ ]:
```
frame.columns
```

In [ ]:
```
frame['year']
```

In [ ]:
```
frame.year
```

Reading csv file into a Dataframe

In [89]:
```
#Reading a CSV file into a Dataframe
shopping = pd.read_csv("Grocery.csv")
shopping.head(5)
```

Out[89]:

	Product	Category	Price	Quantity
0	Bread	Food	20	2
1	Milk	Food	60	5
2	Biscuit	Food	20	2
3	Bourn-Vita	Food	70	1
4	Maggi	Food	20	5

In [71]:

```
print(type(shopping))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [ ]:

```
shopping.head(5)
```

In [72]:

```
len(shopping)
```

Out[72]:

```
20
```

In [73]:

```
shopping.shape
```

Out[73]:

```
(20, 4)
```

In [74]:

```
shopping.shape[0]
```

Out[74]:

```
20
```

In [75]:

```
shopping.columns
```

Out[75]:

```
Index(['Product', 'Category', 'Price', 'Quantity'], dtype='object')
```

In [76]:

```
shopping.shape[1], shopping.columns.size
```

Out[76]:

```
(4, 4)
```

In [78]:

```
shopping.index.size
```

Out[78]:

```
20
```

Finding Number of unique values in a column

In [79]:

```
shopping
```

Out[79]:

	Product	Category	Price	Quantity
0	Bread	Food	20	2
1	Milk	Food	60	5
2	Biscuit	Food	20	2
3	Bourn-Vita	Food	70	1
4	Maggi	Food	20	5
5	Tea	Food	120	1
6	Soap	Hygiene	40	4
7	Brush	Hygiene	30	2
8	Detergent	Household	80	1
9	Hair-Oil	Hygiene	100	1
10	Perfume	Hygiene	150	1
11	Tiffin Box	Household	75	2
12	Pen	Stationary	5	10
13	Pencil	Stationary	2	10
14	T-Shirt	Clothes	250	3
15	Bottle	Household	80	2
16	Bucket	Household	200	1
17	Chips	Food	10	15
18	Juice	Food	100	4
19	Tissues	Hygiene	30	5

In [80]:

```
shopping['Category'].unique()
```

Out[80]:

```
array(['Food', 'Hygiene', 'Household', 'Stationary', 'Clothes'],
      dtype=object)
```

In [81]:

```
shopping['Category'].value_counts()
```

Out[81]:

```
Food          8
Hygiene       5
Household     4
Stationary    2
Clothes       1
Name: Category, dtype: int64
```

In [82]:

```
shopping.head(4)
```

Out[82]:

	Product	Category	Price	Quantity
0	Bread	Food	20	2
1	Milk	Food	60	5
2	Biscuit	Food	20	2
3	Bourn-Vita	Food	70	1

In [85]:
```
shopping['Total Price'] = shopping['Price'] * shopping['Quantity']
```

In [86]:
```
shopping.head(5)
```

Out[86]:

	Product	Category	Price	Quantity	Total Price
0	Bread	Food	20	2	40
1	Milk	Food	60	5	300
2	Biscuit	Food	20	2	40
3	Bourn-Vita	Food	70	1	70
4	Maggi	Food	20	5	100

In [87]:
```
shopping['temp'] = 9
```

In [88]:
```
shopping.head(5)
```

Out[88]:

	Product	Category	Price	Quantity	Total Price	temp
0	Bread	Food	20	2	40	9
1	Milk	Food	60	5	300	9
2	Biscuit	Food	20	2	40	9
3	Bourn-Vita	Food	70	1	70	9
4	Maggi	Food	20	5	100	9

In [84]:
```
shopping[['Product', 'Category']]
```

Out[84]:

	Product	Category
0	Bread	Food
1	Milk	Food
2	Biscuit	Food
3	Bourn-Vita	Food
4	Maggi	Food
5	Tea	Food
6	Soap	Hygiene
7	Brush	Hygiene
8	Detergent	Household
9	Hair-Oil	Hygiene
10	Perfume	Hygiene
11	Tiffin Box	Household
12	Pen	Stationary
13	Pencil	Stationary
14	T-Shirt	Clothes
15	Bottle	Household
16	Bucket	Household
17	Chips	Food
18	Juice	Food
19	Tissues	Hygiene

Accessing multiple columns of a dataframe

In [ ]:

```
shopping_new = shopping[['Product','Category']]
shopping_new
```

Adding new column to a dataframe

In [90]:

```
# Multiplying two Columns to get Total Price for an item
Total = shopping['Price']*shopping['Quantity']
Total
```

Out[90]:
```
0      40
1     300
2      40
3      70
4     100
5     120
6     160
7      60
8      80
9     100
10    150
11    150
12     50
13     20
14    750
15    160
16    200
17    150
18    400
19    150
dtype: int64
```

In [91]:
```
# Adding a new Column to the Dataframe
shopping['Total Price'] = Total
shopping
```

Out[91]:

	Product	Category	Price	Quantity	Total Price
0	Bread	Food	20	2	40
1	Milk	Food	60	5	300
2	Biscuit	Food	20	2	40
3	Bourn-Vita	Food	70	1	70
4	Maggi	Food	20	5	100
5	Tea	Food	120	1	120
6	Soap	Hygiene	40	4	160
7	Brush	Hygiene	30	2	60
8	Detergent	Household	80	1	80
9	Hair-Oil	Hygiene	100	1	100
10	Perfume	Hygiene	150	1	150
11	Tiffin Box	Household	75	2	150
12	Pen	Stationary	5	10	50
13	Pencil	Stationary	2	10	20
14	T-Shirt	Clothes	250	3	750
15	Bottle	Household	80	2	160
16	Bucket	Household	200	1	200
17	Chips	Food	10	15	150
18	Juice	Food	100	4	400
19	Tissues	Hygiene	30	5	150

In [92]:

```
#Sum Total of Price
shopping['Total Price'].sum()
```

Out[92]:

```
3250
```

In [93]:

```
#Adding Values in Total Price Columm to get total expenditure
totalExpenditure = shopping[['Total Price', 'Quantity', 'Price']].sum()
totalExpenditure
#print(type(totalExpenditure))
```

Out[93]:

```
Total Price    3250
Quantity         77
Price          1462
dtype: int64
```

In [96]:

```
shopping.describe()
```

Out[96]:

	Price	Quantity	Total Price
count	20.00000	20.000000	20.00000
mean	73.10000	3.850000	162.50000
std	66.56291	3.773523	165.58906
min	2.00000	1.000000	20.00000
25%	20.00000	1.000000	67.50000
50%	65.00000	2.000000	135.00000
75%	100.00000	5.000000	160.00000
max	250.00000	15.000000	750.00000

In [94]:

```
shopping[['Price','Quantity','Total Price']].agg(['sum','mean','max','min','count'
```

Out[94]:

	Price	Quantity	Total Price
sum	1462.000000	77.000000	3250.000000
mean	73.100000	3.850000	162.500000
max	250.000000	15.000000	750.000000
min	2.000000	1.000000	20.000000
count	20.000000	20.000000	20.000000
median	65.000000	2.000000	135.000000
var	4430.621053	14.239474	27419.736842

Sorting the contents of the dataframe w.r.t. a particular column

In [97]:

```
#help(shopping.sort_values)
#Sorting Data in decreasing order of item price
shoppingSorted = shopping.sort_values('Price',ascending=False)
shoppingSorted
```

Out[97]:

	Product	Category	Price	Quantity	Total Price
14	T-Shirt	Clothes	250	3	750
16	Bucket	Household	200	1	200
10	Perfume	Hygiene	150	1	150
5	Tea	Food	120	1	120
18	Juice	Food	100	4	400
9	Hair-Oil	Hygiene	100	1	100
15	Bottle	Household	80	2	160
8	Detergent	Household	80	1	80
11	Tiffin Box	Household	75	2	150
3	Bourn-Vita	Food	70	1	70
1	Milk	Food	60	5	300
6	Soap	Hygiene	40	4	160
19	Tissues	Hygiene	30	5	150
7	Brush	Hygiene	30	2	60
4	Maggi	Food	20	5	100
2	Biscuit	Food	20	2	40
0	Bread	Food	20	2	40
17	Chips	Food	10	15	150
12	Pen	Stationary	5	10	50
13	Pencil	Stationary	2	10	20

Applying an arithmetic operation on a column

In [ ]:

```
#Broadcasting in Pandas Dataframe
#Increase the quantity of each product by 1
shopping['Quantity'] = shopping['Quantity'] + 1
shopping
```

Summary Statistics

In [ ]:

```
shopping.describe()
```

Writing a dataframe to a csv file

In [ ]:

```
shopping.to_csv('GroceryV2.csv', header= True)
```

Grouping and Aggregation

In [ ]:

```
shopping.columns
```

In [ ]:

```
#Groupby in Pandas
#All products grouped according to the category
```

```
Categories = shopping.groupby('Category')
#Total expenditure per category
Categories['Total Price'].sum()
```

In [ ]:

```
#Total counts per category
Categories['Total Price'].count()
```

Deleting columns from a dataframe

In [ ]:

```
shopping.drop(['Total Price'],axis = 1)
```

Merging two dataframes

In [ ]:

```
#Combining data frames

shoppingNew = pd.read_csv('Grocery2.csv')
shoppingNew
```

In [ ]:

```
shopping.head()
```

In [ ]:

```
shoppingFinal = pd.concat([shopping,shoppingNew])
```

In [ ]:

```
shoppingFinal
```

In [ ]:

```
#Resetting the row index

shoppingFinal = shoppingFinal.reset_index(drop = True)
shoppingFinal
```

In [ ]:

```
#Altering the labels/Column headers
shoppingFinal = shoppingFinal.rename(columns = {'Product': 'Item'})
shoppingFinal.head(3)
```

In [ ]:

```
#Reindexing the columns
shoppingFinal.reindex(columns = ['Item','Category','Total Price','Price','Quantity
```

Data Plotting in Pandas

In [ ]:

```
globalTemp = pd.read_csv('GlobalTemperatures.csv')
globalTemp.head()
```

In [ ]:

```
#Histogram

hist = globalTemp['LandAverageTemperature'].plot(kind = 'hist',figsize = (12, 8))
hist.set_xlabel("Average Land Temperature")
hist.set_ylabel("Frequency")
```

In [ ]:

```
populationData = pd.read_csv('Demographicdata.csv',index_col = 0)
```

In [ ]:

```
populationData.head()
```

In [ ]:

```
#Bar Graph
populationData['Population(in millions)'].plot(kind = 'bar', figsize = (12, 6), ti
```

In [ ]: