BCSE00133

# Machine Learning Lab

DR GAURAV KUMAR

ASST. PROF, CEA, GLA UNIVERSITY


Welcome,
my awesome students!
LET'S WORK TOGETHER, DO OUR BEST TO LEARN AS MUCH AS WE CAN, AND RESPECT AND SUPPORT EACH OTHER THROUGH ALL THE CHALLENGES. LET'S DO THIS!

# To learn machine learning, we will use the Python programming language

we need to install it in our computer system with compatible IDEs (Integrated Development Environment).


# python™

GUIDO VAN ROSSUM, FEBRUARY 20, 1991

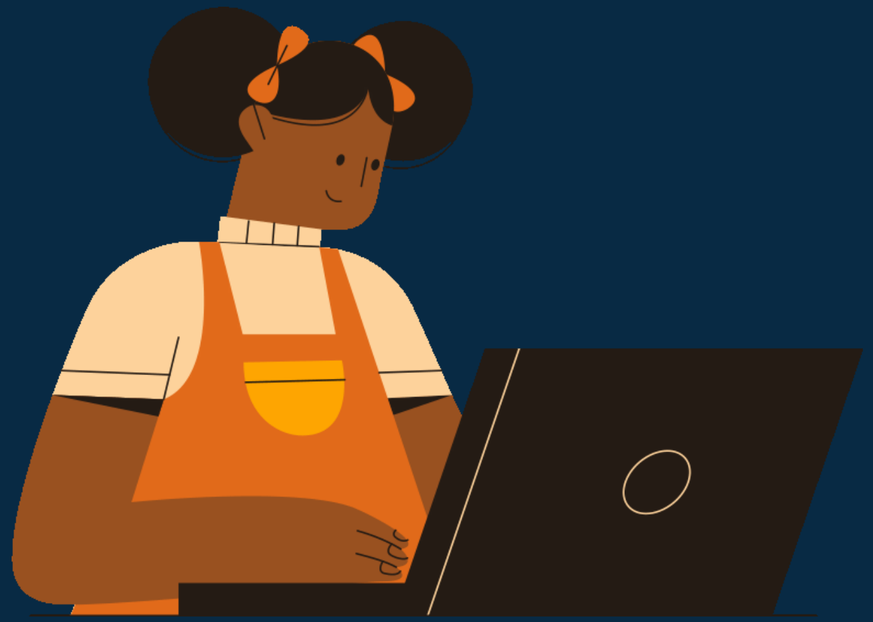
- Python is a general purpose, dynamic, high-level, and interpreted programming language.
- It supports Object Oriented programming approach to develop applications.
- It is simple and easy to learn and provides lots of high-level data structures.
- Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc.

# WHERE IS PYTHON USED

The various areas of Python use are given below.

- Data Science
- Date Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions

# IDE
## (Integrated Development Environment)



- Text/Code Editor
  - A code editor is a tool that is used to write and edit code. For Ex- Sublime Text, Atom, Vim etc.
  - They are usually lightweight and can be great for learning.
  - However, once your program gets larger, you need to test and debug your code, that's where IDEs come in.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# IDE
## (Integrated Development Environment)

- IDE
  - An IDE understand your code much better than a text editor.
  - It usually provides features such as build automation, code linting, testing and debugging.
  - This can significantly speed up your work.

Note: Code Linting highlights syntactical and stylistic problems


# Learn to install Python and an IDE with the help of Anaconda distribution

- WITH OVER 25 MILLION USERS WORLDWIDE, ANACONDA DISTRIBUTION IS THE WORLD' S MOST POPULAR AND EASIEST WAY TO LEARN AND PERFORM DATA SCIENCE AND MACHINE LEARNING.
- IT IS A FREE AND OPEN-SOURCE PLATFORM FOR PYTHON/R PROGRAMMING LANGUAGES CAN BE EASILY INSTALLED ON ANY OS SUCH AS WINDOWS, LINUX, AND MAC OS.
- IT PROVIDES MORE THAN 1500 LIBRARY PACKAGES WHICH ARE SUITABLE FOR DEVELOPING MACHINE LEARNING AND DEEP LEARNING MODELS.
- ANACONDA DISTRIBUTION PROVIDES INSTALLATION OF PYTHON WITH VARIOUS IDE'S SUCH AS JUPYTER NOTEBOOK, JUPYTERLAB, SPYDER, APACHE ZEPPELIN, PYTHON TOOLS FOR VISUAL STUDIO, ETC.

ANACONDA

GLA UNIVERSITY MATHURA
Accredited with A Grade by NAAC

# WHAT'S YOUR END GOAL?

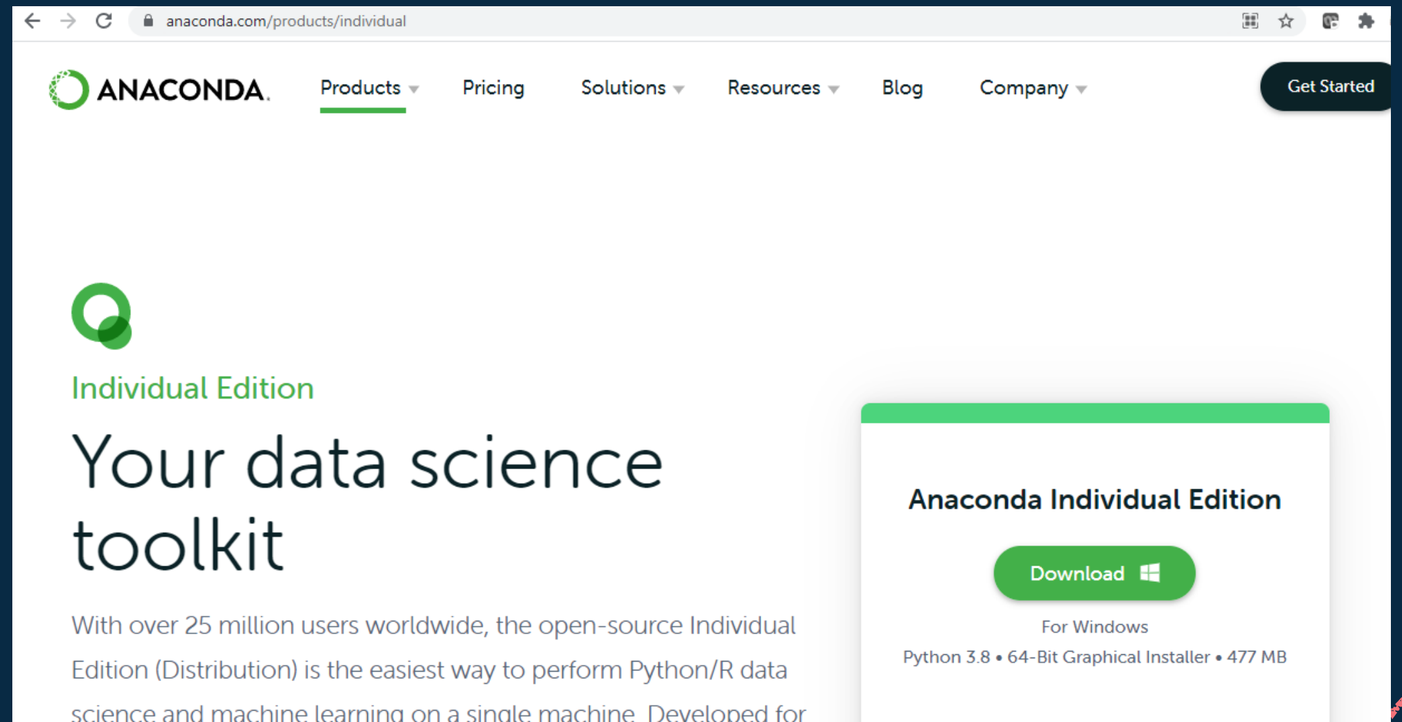
- WEB DEVELOPMENT
  - PYCHARM PROFESSIONAL, VS CODE
- DATA SCIENCE
  - SPYDER, JUPYTER NOTEBOOK, PYCHARM PROFESSIONAL
- SCRIPTING
  - SUBLIME, ATOM, PYCHARM COMMUNITY, ECLIPSE + PYDEV

# Installation Guidelines

1

https://www.anaconda.com/products/individual

2 CLICK ON DOWNLOADS

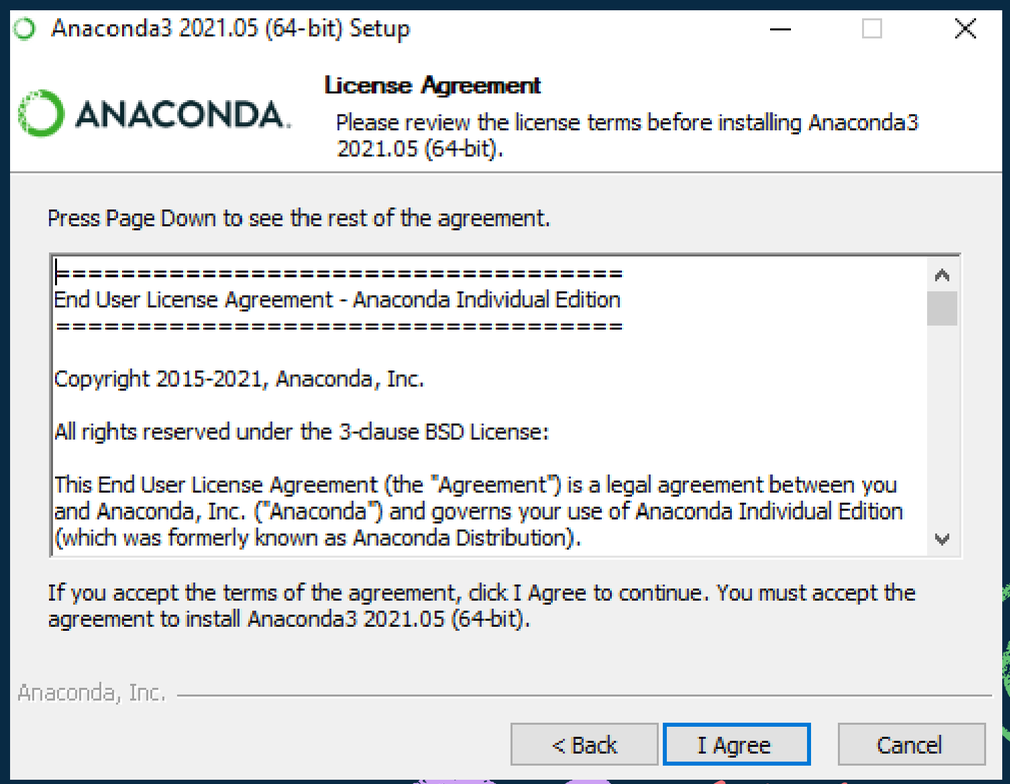


# ANACONDA

# Installation Guidelines

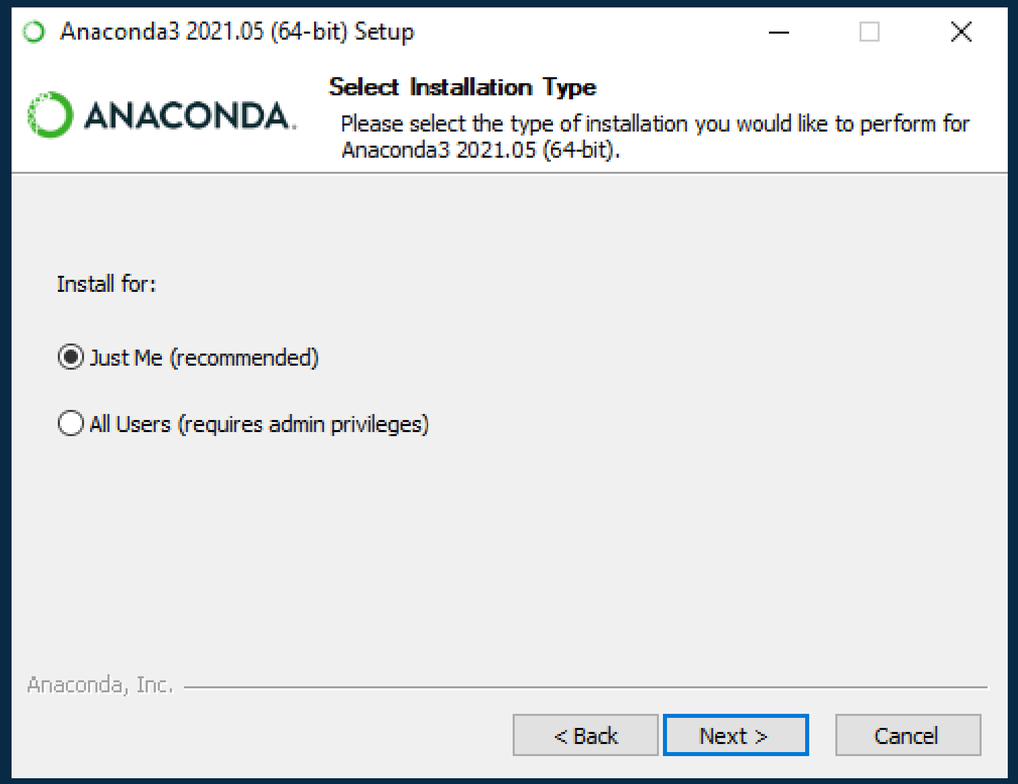
## 3 CLICK ON NEXT


## 4 ACCEPT THE AGREEMENT

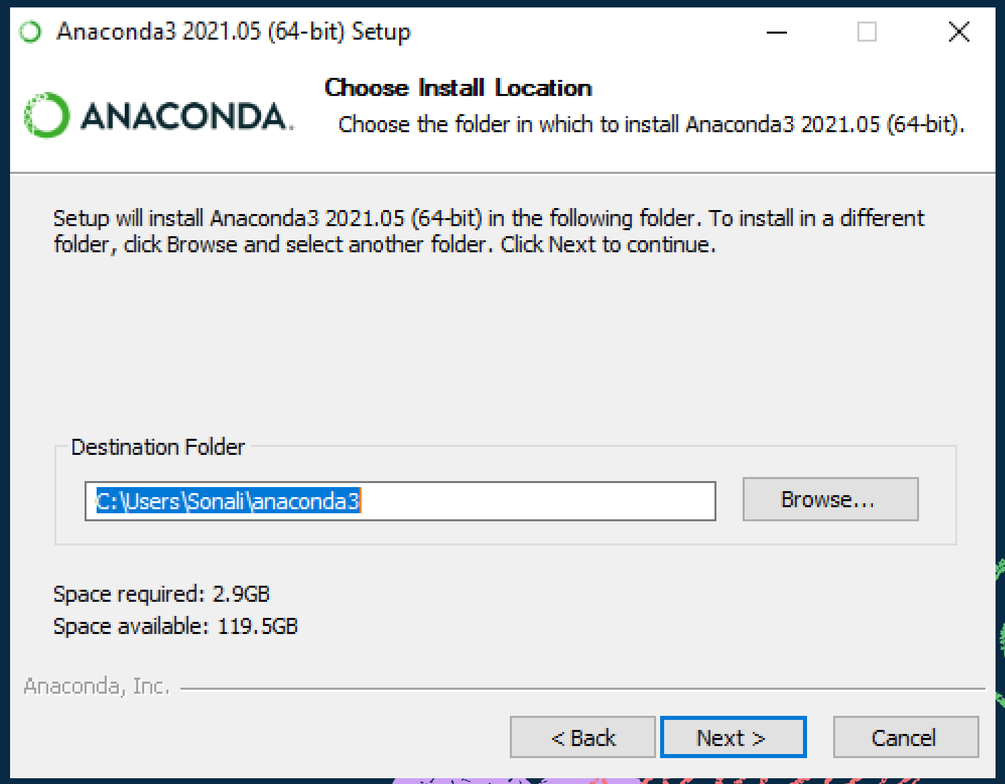


# Installation Guidelines

## 5 CLICK ON INSTALL FOR JUST ME

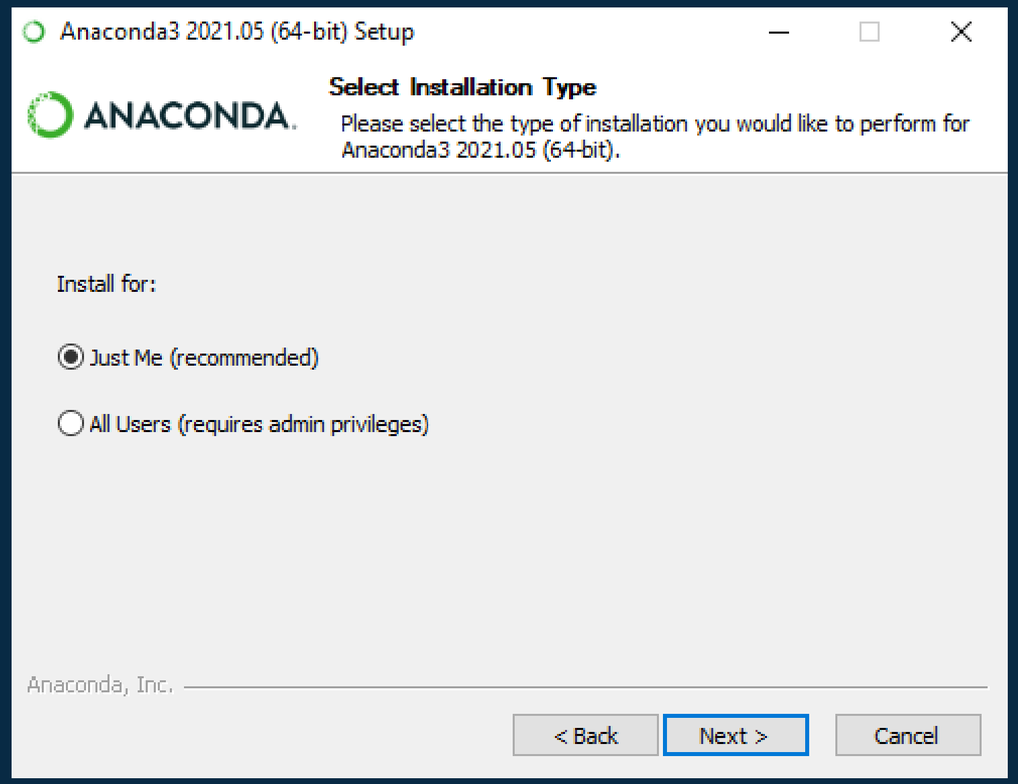



## 6 LEAVE THE DEFUALT INSTALLATION FOLDER AND CLIK ON NEXT

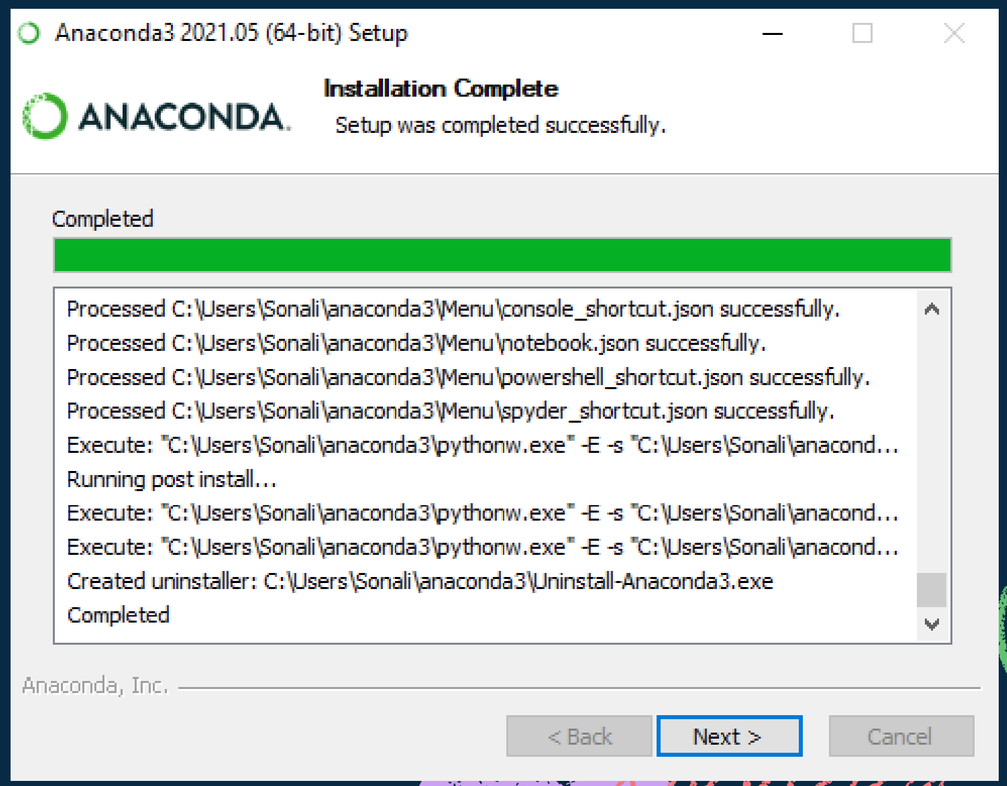


# Installation Guidelines

## 7 CLICK ON INSTALL FOR JUST ME AND CLICK ON NEXT



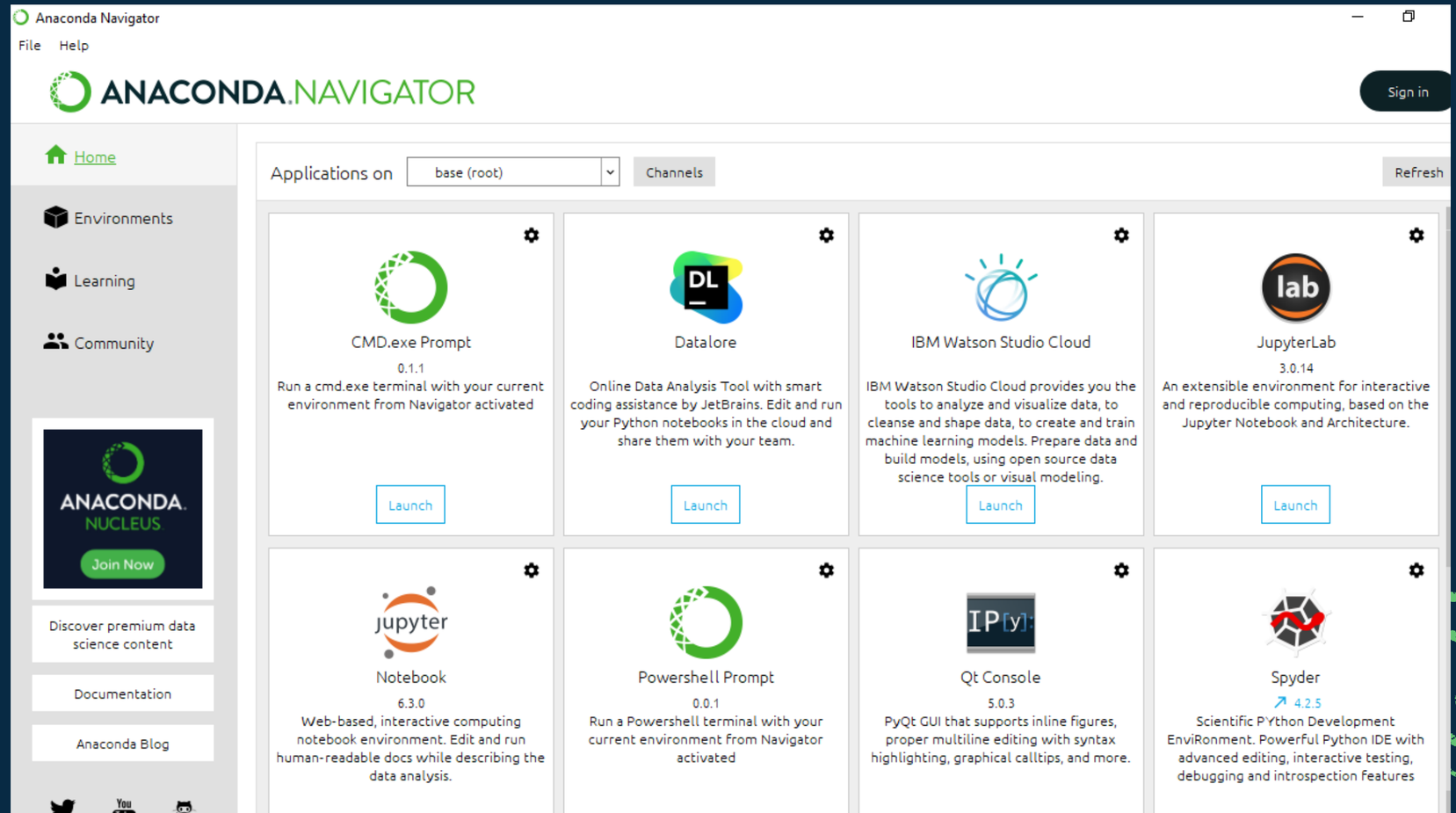

## 8 CLICK ON NEXT, NEXT AND FINISHED, INSTALLATION COMPLETED



# Sypder Installation Guidelines

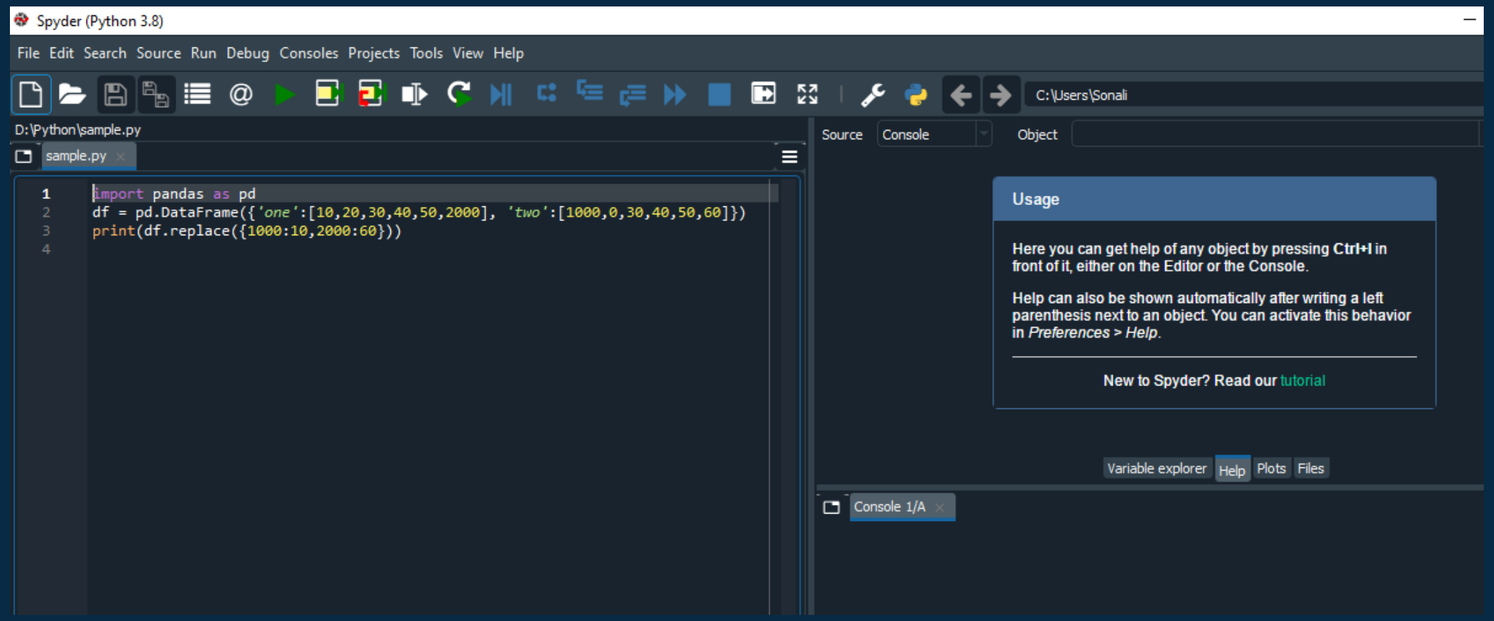
1. OPEN ANACONDA NAVIGATOR FROM WINDOW SEARCH BOX

2. AFTER OPENING THE NAVIGATOR, LAUNCH THE SPYDER IDE BY CLICKING ON THE LAUNCH BUTTON GIVEN BELOW THE SPYDER. IT WILL INSTALL THE SPYDER IDE IN YOUR SYSTEM.



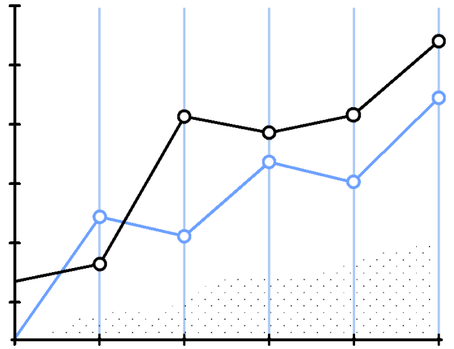
# Sypder Installation Guidelines

**3** OPEN SPYDER IDE, IT WILL LOOK LIKE THE BELOW IMAGE AND START WRITING YOUR PYTHON PROGRAMME

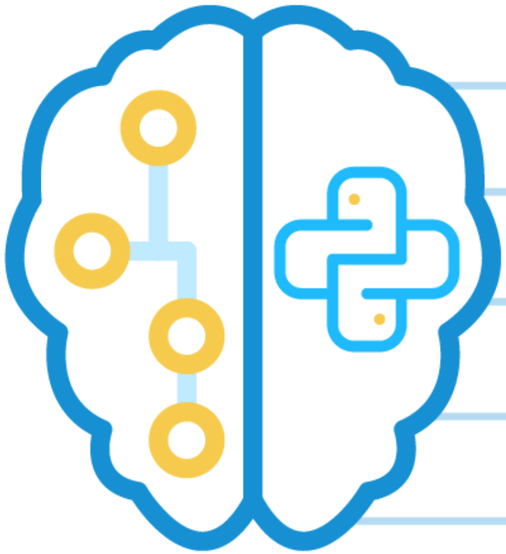



- WRITE YOUR FIRST PROGRAM, AND SAVE IT USING THE .PY EXTENSION.
- RUN THE PROGRAM USING THE TRIANGLE RUN BUTTON.
- YOU CAN CHECK THE PROGRAM'S OUTPUT ON CONSOLE PANE AT THE BOTTOM RIGHT SIDE.

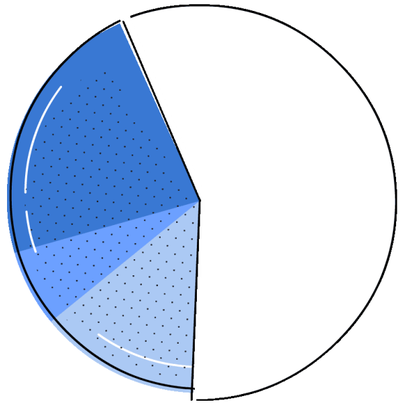
# Machine Learning Library

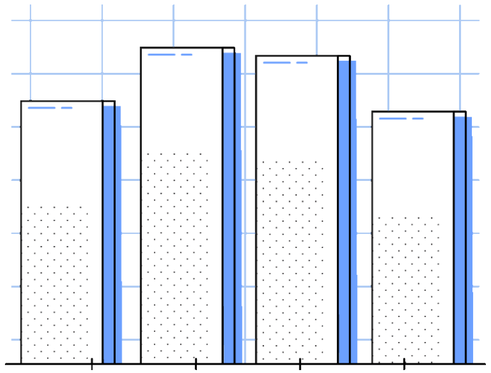


## Python Libraries for Machine Learning



- NumPy
- Matplotbit
- Pandas
- Seaborn
- Scikit-learn
- OpenCV
- Keras
- spaCy
- TensorFlow
- SciPy





~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

1


NUMPY STANDS FOR NUMERICAL PYTHON, IS USED FOR SCIENTIFIC COMPUTING AND HANDLE LARGE MULTI-DIMENSIONAL ARRAY AND MATRIX COMPUTATIONS

SUPPORT FOR MATHEMATICAL AND LOGICAL OPERATIONS SUCH AS DISCRETE FOURIER TRANSFORMATIONS, BASIC LINEAR ALGEBRA AND STATISTICAL OPERATIONS

CORE TASK: DATA MANIPULATION

2

# Pandas

PANDA MEANS PYTHON DATA ANALYSIS LIBRARY BUILT ON TOP OF THE NUMPY LIBRARY THAT OFFERS VARIOUS DATA STRUCTURES AND OPERATIONS FOR MANIPULATING NUMERICAL DATA AND TIME SERIES.

IT IS MAINLY POPULAR FOR IMPORTING AND ANALYZING DATA MUCH EASIER.

IT DEALS WITH MISSING DATA EFFECTIVELY

CORE TASK: DATA CLEANING AND LOADING

3

GLA UNIVERSITY MATHURA
Accredited with A Grade by NAAC

IT OFFERS POWERFUL VISUALIZATIONS AND PLOTTING LIBRARY FOR PYTHON


IT IS A FREE AND OPEN-SOURCE LIBRARY AND IS USUALLY CONSIDERED AS THE REPLACEMENT FOR MATLAB.

IT IS ALSO USED FOR CORRELATION ANALYSIS OF VARIABLES, AND OUTLIER DETECTOR WITH THE HELP OF A SCATTER PLOT.

CORE TASK: VISUALIZATIONS AND PLOTTING

4

SCIPY MEANS “SCIENTIFIC PYTHON,” EXTENSIVELY USED TO PERFORM HIGH-LEVEL COMPUTATIONS.

SciPy

IT IS USED FOR MULTI-DIMENSIONAL IMAGE OPERATIONS.

IT IS USED FOR ALGORITHMS OPTIMIZATIONS AND LINEAR ALGEBRA.

CORE TASK: HIGH LEVEL COMPUTATIONS

5

GLA UNIVERSITY MATHURA
Accredited with A Grade by NAAC


THIS PYTHON LIBRARY IS ASSOCIATED WITH NUMPY AND SCIPY

THE LIBRARY CONTAINS A NUMBER OF ALGORITHMS FOR IMPLEMENTING MACHINE LEARNING TASKS LIKE REGRESSION, REDUCING DIMENSIONALITY, AND MODEL SELECTION, ETC.

IT IS USED FOR CLUSTERING, FACTOR ANALYSIS, PRINCIPAL COMPONENT ANALYSIS, TO UNSUPERVISED NEURAL NETWORKS.

CORE TASK: WORKING WITH COMPLEX DATA

6

# TensorFlow

THE IS THE MOST POPULAR LIBRARY DEVELOPED BY GOOGLE IN COLLABORATION WITH BRAIN TEAM.

IT IS A PYTHON LIBRARY OPTIMIZED FOR SPEED, AND IT USES TECHNIQUES THAT ARE SUITABLE FOR QUICK ALGEBRA OPERATIONS.

WITH TENSORFLOW, IT IS POSSIBLE TO VISUALIZE EVERY PART OF THE GRAPH CAREFULLY THAT IT IS NOT POSSIBLE WITH OTHER PYTHON LIBRARIES.

IT IS USED WITH APPLICATIONS LIKE GOOGLE PHOTOS, GOOGLE VOICE SEARCH, ETC.

IT CAN BE USED ON CPU OR GPU FOR DISTRIBUTED COMPUTING.

7

# Keras

CORE TASK: DEEP LEARNING

KERA IS CONSIDERED ONE OF THE MOST AMAZING MACHINE LEARNING LIBRARIES FOR EXPRESSING NEURAL NETWORKS.

KERA IS MODULAR IN NATURE, AND IT IS INCREDIBLY EXPRESSIVE, FLEXIBLE, AND SUITABLE FOR INNOVATIVE RESEARCH.

IT IS HIGHLY POPULAR AMONG START-UPS THAT PLACE DEEP LEARNING AT THE CORE OF THEIR PRODUCTS. IT IS USED AT NETFLIX, INSTACART, YELP, UBER, ETC.


# NumPy(Numerical Python)

IT DEMONSTRATES HOW N-DIMENSIONAL () ARRAYS ARE REPRESENTED AND CAN BE MANIPULATED.

NUMPY IS A LIBRARY CONSISTING OF MULTIDIMENSIONAL ARRAY OBJECTS AND A COLLECTION OF ROUTINES FOR PROCESSING THOSE ARRAYS.

USING NUMPY, MATHEMATICAL AND LOGICAL OPERATIONS ON ARRAYS CAN BE PERFORMED.

# NumPy(Numerical Python)

## Numpy Array

It is similar to array that stores the same type of elements

Dimension is the number of indices or subscripts, that we require in order to specify an individual element of an array.

Number of dimensions is called the Rank of an Array.

The shape of an array can be defined as the number of elements in each dimension.

## Syntax

import numpy as np

```
IMPORT NUMPY AS NP
A = NP.ARRAY([1,2,3])
PRINT(A)
```

[1,2,3]

```
IMPORT NUMPY AS NP
MY_LIST=[1,2,3,4,5,6]
A = NP.ARRAY(MY_LIST)
PRINT(A)
```

[1,2,3, 4, 5, 6]

# NumPy(Numerical Python)

## Numpy Array

```
IMPORT NUMPY AS NP
MY_LIST=[[1,2,3], [4,5,6]]
A = NP.ARRAY(MY_LIST)
PRINT(A)
```

```
PRINT(TYPE(A))   → <CLASS 'NUMPY.NDARRAY'>
PRINT(A.NDIM)    → 2
PRINT(A.SHAPE)   → 2,3
PRINT(A.DTYPE)   → INT32
PRINT(A.SIZE)    → 6
```

Output

```
[[1 2 3]
 [4 5 6]]
```

# Lab Assignment

PRACTISE NUMPY ARRAY- HTTPS://WWW.W3SCHOOLS.COM/PYTHON/NUMPY/

1. NUMPY CREATING ARRAYS
2. NUMPY ARRAY INDEXING
3. NUMPY ARRAY SLICING
4. NUMPY DATA TYPES
5. NUMPY COPY VS VIEW
6. NUMPY ARRAY SHAPE
7. NUMPY ARRAY RESHAPE
8. NUMPY ARRAY ITERATING
9. NUMPY ARRAY JOIN
10. NUMPY ARRAY SPLIT
11. NUMPY ARRAY SEARCH
12. NUMPY ARRAY SORT
13. NUMPY ARRAY FILTER

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Panda Library

PANDAS IS DEFINED AS AN OPEN-SOURCE LIBRARY THAT PROVIDES HIGH-PERFORMANCE DATA MANIPULATION IN PYTHON.

THE NAME OF PANDAS IS DERIVED FROM THE WORD PANEL DATA, WHICH MEANS AN ECONOMETRICS FROM MULTIDIMENSIONAL DATA.

IT IS USED FOR DATA ANALYSIS IN PYTHON AND DEVELOPED BY WES MCKINNEY IN 2008.

PANDAS IS BUILT ON TOP OF THE NUMPY PACKAGE, MEANS NUMPY IS REQUIRED FOR OPERATING THE PANDAS.

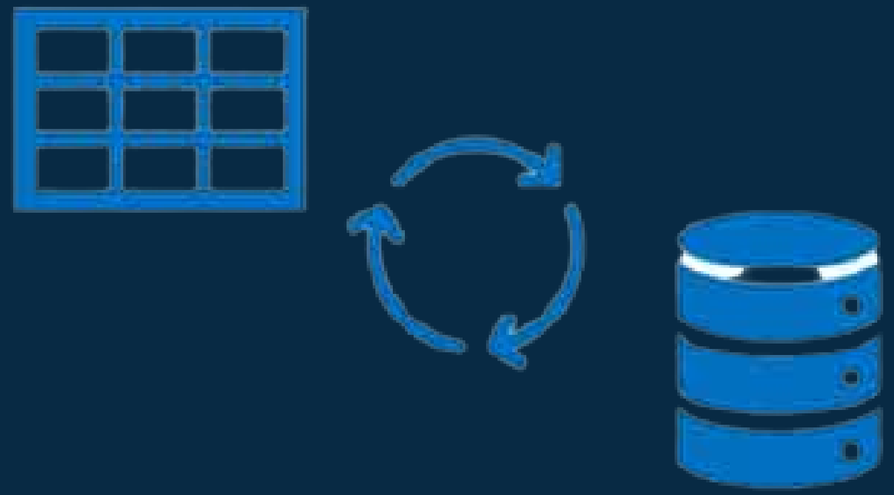
~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Panda Library

DATA ANALYSIS REQUIRES LOTS OF PROCESSING, SUCH AS RESTRUCTURING, CLEANING OR MERGING, ETC.

USING PANDAS, WE CAN ACCOMPLISH FIVE TYPICAL STEPS IN THE PROCESSING AND ANALYSIS OF DATA —

LOAD, PREPARE, MANIPULATE, MODEL, AND ANALYZE.



THERE ARE DIFFERENT TOOLS ARE AVAILABLE FOR FAST DATA PROCESSING, SUCH AS NUMPY, SCIPY, CYTHON, AND PANDA. BUT WE PREFER PANDAS BECAUSE WORKING WITH PANDAS IS FAST, SIMPLE AND MORE EXPRESSIVE THAN OTHER TOOLS.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Key Features of Pandas

- FAST AND EFFICIENT DATAFRAME OBJECT WITH DEFAULT AND CUSTOMIZED INDEXING.
- TOOLS FOR LOADING DATA INTO IN-MEMORY DATA OBJECTS FROM DIFFERENT FILE FORMATS.
- DATA ALIGNMENT AND INTEGRATED HANDLING OF MISSING DATA.
- RESHAPING AND PIVOTING OF DATE SETS.
- LABEL-BASED SLICING, INDEXING AND SUBSETTING OF LARGE DATA SETS.
- COLUMNS FROM A DATA STRUCTURE CAN BE DELETED OR INSERTED.
- GROUP BY DATA FOR AGGREGATION AND TRANSFORMATIONS.
- HIGH PERFORMANCE MERGING AND JOINING OF DATA.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Python Pandas Data Structure

THE PANDAS PROVIDES TWO DATA STRUCTURES FOR PROCESSING THE DATA, I.E.,

SERIES AND DATAFRAME

1) SERIES

- IT IS DEFINED AS A ONE-DIMENSIONAL ARRAY THAT IS CAPABLE OF STORING VARIOUS DATA TYPES.
- THE ROW LABELS OF SERIES ARE CALLED THE INDEX.
- WE CAN EASILY CONVERT THE LIST, TUPLE, AND DICTIONARY INTO SERIES USING "SERIES' METHOD. A SERIES CANNOT CONTAIN MULTIPLE COLUMNS.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Creating Series from Array

BEFORE CREATING A SERIES, FIRSTLY, WE HAVE TO IMPORT THE NUMPY MODULE AND THEN USE ARRAY() FUNCTION IN THE PROGRAM.

```
IMPORT PANDAS AS PD
IMPORT NUMPY AS NP
INFO = NP.ARRAY(['P','A','N','D','A','S'])
A = PD.SERIES(INFO)
PRINT(A)
```

or

```
IMPORT PANDAS AS PD
INFO = ['P','A','N','D','A','S']
A = PD.SERIES(INFO)
PRINT(A)
```

Output

```
0    P
1    A
2    N
3    D
4    A
5    S

DTYPE: OBJECT
```

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

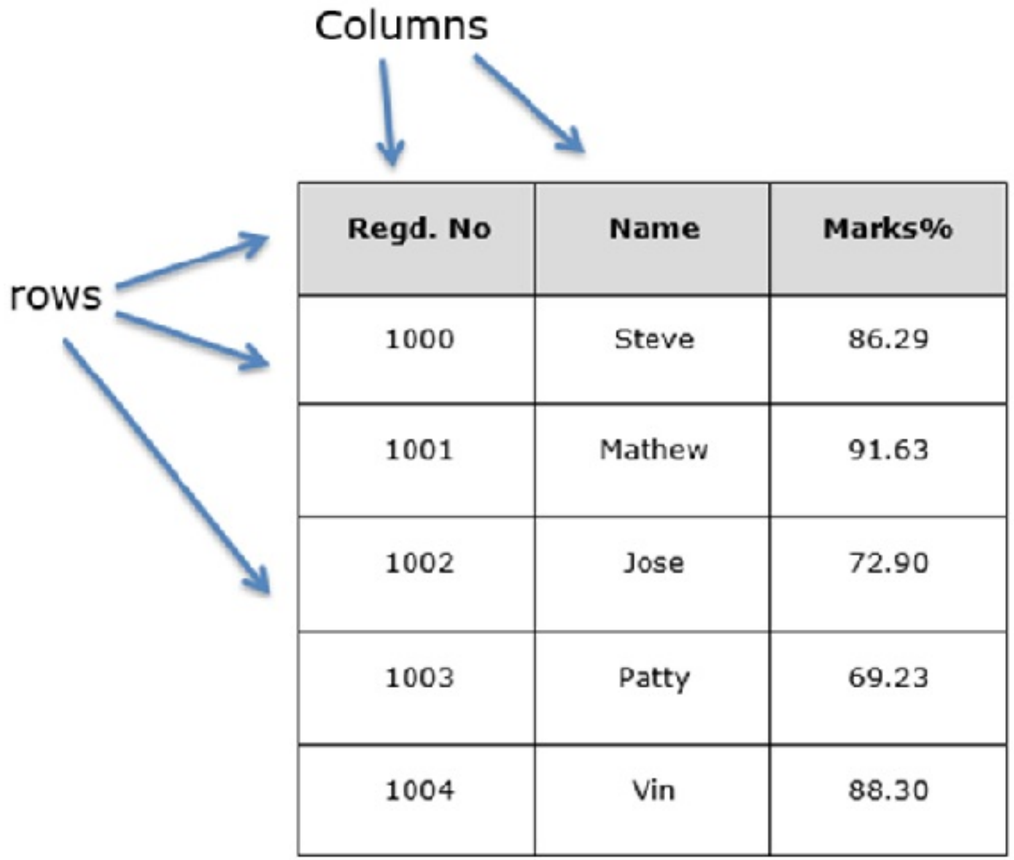
# Python Pandas DataFrame

- A DATA FRAME IS A TWO-DIMENSIONAL DATA STRUCTURE, I.E., DATA IS ALIGNED IN A TABULAR FASHION IN ROWS AND COLUMNS.

FEATURES OF DATAFRAME

- POTENTIALLY COLUMNS ARE OF DIFFERENT TYPES
- SIZE – MUTABLE
- LABELLED AXES (ROWS AND COLUMNS)
- CAN PERFORM ARITHMETIC OPERATIONS ON ROWS AND COLUMNS

A DATA FRAME WITH STUDENT' S DATA




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# pandas.DataFrame

A PANDAS DATAFRAME CAN BE CREATED USING THE FOLLOWING CONSTRUCTOR

PANDAS.DATAFRAME( DATA, INDEX, COLUMNS, DTYPE, COPY)

Optional

1 DATA- TAKES VARIOUS FORMS LIKE NDARRAY, SERIES, MAP, LISTS, DICT, CONSTANTS AND ALSO ANOTHER DATAFRAME.

2 INDEX- FOR THE ROW LABELS, THE INDEX TO BE USED FOR THE RESULTING FRAME IS OPTIONAL DEFAULT NP.ARANGE(N) IF NO INDEX IS PASSED.

3 COLUMNS- FOR COLUMN LABELS, THE OPTIONAL DEFAULT SYNTAX IS - NP.ARANGE(N). THIS IS ONLY TRUE IF NO INDEX IS PASSED.

4 DTYPE-THE DATA TYPE OF EACH COLUMN.

5 COPY-THIS COMMAND (OR WHATEVER IT IS) IS USED FOR COPYING OF DATA, IF THE DEFAULT IS FALSE.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Create DataFrame

A PANDAS DATAFRAME CAN BE CREATED USING VARIOUS INPUTS LIKE –

- LISTS
- DICT
- SERIES
- NUMPY NDARRAYS
- ANOTHER DATAFRAME

# Create a DataFrame using List

```
IMPORT PANDAS AS PD
DATA = [1,2,3,4,5]
DF = PD.DATAFRAME(DATA)
PRINT(DF)
```

Output

| | 0 |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Example 2

```
IMPORT PANDAS AS PD
DATA = [['ALEX',10],['BOB',12],['CLARKE',13]]
DF = PD.DATAFRAME(DATA,COLUMNS=['NAME','AGE'])
PRINT(DF)
```

Output

| | NAME | AGE |
|---|---|---|
| 0 | ALEX | 10 |
| 1 | BOB | 12 |
| 2 | CLARKE | 13 |

# Create a DataFrame from Dict of ndarrays / Lists


```
IMPORT PANDAS AS PD
DATA = {'NAME':['TOM', 'JACK', 'STEVE', 'RICKY'],'AGE':
[28,34,29,42]}
DF = PD.DATAFRAME(DATA)
PRINT(DF)
```

Output

| | AGE | NAME |
|---|---|---|
| 0 | 28 | TOM |
| 1 | 34 | JACK |
| 2 | 29 | STEVE |
| 3 | 42 | RICKY |

## Let us now create an indexed DataFrame using arrays.


```
IMPORT PANDAS AS PD
DATA = {'NAME':['TOM', 'JACK', 'STEVE', 'RICKY'],'AGE':[28,34,29,42]}
DF = PD.DATAFRAME(DATA, INDEX ['RANK1','RANK2','RANK3','RANK4'])
PRINT(DF)
```

Output

| | AGE | NAME |
|---|---|---|
| RANK1 | 28 | TOM |
| RANK2 | 34 | JACK |
| RANK3 | 29 | STEVE |
| RANK4 | 42 | RICKY |

# Pandas Reading Files

READING FROM CSV FILE

```
DF = PD.READ_CSV('A.CSV')
```

```
IMPORT PANDAS
DF = PANDAS.READ_CSV('HRDATA.CSV')
PRINT(DF)
```

Output

| | NAME | HIRE DATE | SALARY | LEAVES REMAINING |
|---|---|---|---|---|
| 0 | JOHN IDLE | 08/15/14 | 50000.0 | 10 |
| 1 | SMITH GILLIAM | 04/07/15 | 65000.0 | 8 |
| 2 | PARKER CHAPMAN | 02/21/14 | 45000.0 | 10 |
| 3 | JONES PALIN | 10/14/13 | 70000.0 | 3 |
| 4 | TERRY GILLIAM | 07/22/14 | 48000.0 | 7 |
| 5 | MICHAEL PALIN | 06/28/13 | 66000.0 | |

PRINT(DF.TO_STRING()) TO PRINT THE ENTIRE DATAFRAME.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Pandas Reading Files

READING FROM JSON (JSON STANDS FOR JAVASCRIPT OBJECT NOTATION)

```
DF =PD.READ_JSON('HRDATA.JSON')
```

```
IMPORT PANDAS AS PD
DATA = PD.READ_JSON('HRDATA.JSON')
PRINT(DATA)
```

Output

| | NAME | HIRE DATE | SALARY | LEAVES REMAINING |
|---|---|---|---|---|
| 0 | JOHN IDLE | 08/15/14 | 50000.0 | 10 |
| 1 | SMITH GILLIAM | 04/07/15 | 65000.0 | 8 |
| 2 | PARKER CHAPMAN | 02/21/14 | 45000.0 | 10 |
| 3 | JONES PALIN | 10/14/13 | 70000.0 | 3 |
| 4 | TERRY GILLIAM | 07/22/14 | 48000.0 | 7 |
| 5 | MICHAEL PALIN | 06/28/13 | 66000.0 | |

# Pandas - Analyzing DataFrames

## VIEWING THE DATA

- ONE OF THE MOST USED METHOD FOR GETTING A QUICK OVERVIEW OF THE DATAFRAME, IS THE HEAD() METHOD.
- THE HEAD() METHOD RETURNS THE HEADERS AND A SPECIFIED NUMBER OF ROWS, STARTING FROM THE TOP.

```
IMPORT PANDAS AS PD

DF = PD.READ_CSV('DATA.CSV')

PRINT(DF.HEAD(4))
```

Output

| | DURATION | PULSE | MAXPULSE | CALORIES |
|---|---|---|---|---|
| 0 | 60 | 110 | 130 | 409.1 |
| 1 | 60 | 117 | 145 | 479.0 |
| 2 | 60 | 103 | 135 | 340.0 |
| 3 | 45 | 109 | 175 | 282.4 |

IF THE NUMBER OF ROWS IS NOT SPECIFIED, THE HEAD() METHOD WILL RETURN THE TOP 5 ROWS.

# Pandas - Analyzing DataFrames


## VIEWING THE DATA

- THERE IS ALSO A TAIL() METHOD FOR VIEWING THE LAST ROWS OF THE DATAFRAME.
- THE TAIL() METHOD RETURNS THE HEADERS AND A SPECIFIED NUMBER OF ROWS, STARTING FROM THE BOTTOM.

```
PRINT(DF.TAIL())
```

## INFO ABOUT THE DATA

```
PRINT(DF.INFO())
```

```
<CLASS 'PANDAS.CORE.FRAME.DATAFRAME'>
 RANGEINDEX: 169 ENTRIES, 0 TO 168
 DATA COLUMNS (TOTAL 4 COLUMNS):
  #   COLUMN    NON-NULL COUNT  DTYPE
 ---  ------    --------------  -----
  0   DURATION  169 NON-NULL    INT64
  1   PULSE     169 NON-NULL    INT64
  2   MAXPULSE  169 NON-NULL    INT64
  3   CALORIES  164 NON-NULL    FLOAT64
 DTYPES: FLOAT64(1), INT64(3)
 MEMORY USAGE: 5.4 KB
 NONE
```

# Pandas - Analyzing DataFrames

## PANDAS - CLEANING DATA

DATA CLEANING MEANS FIXING BAD DATA IN YOUR DATA SET.

BAD DATA COULD BE:

- EMPTY CELLS
- DATA IN WRONG FORMAT
- WRONG DATA
- DUPLICATES

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Pandas - Cleaning Data

## Input

| | NAME | HIRE DATE | SALARY | LEAVES REMAINING |
|---|---|---|---|---|
| 0 | JOHN IDLE | 08/15/14 | 50000.0 | 10 |
| 1 | SMITH GILLIAM | 04/07/15 | | 8 |
| 2 | PARKER CHAPMAN | 02/21/14 | 45000.0 | 10 |
| 3 | JONES PALIN | 100713 | 70000.0 | 3 |
| 4 | TERRY GILLIAM | 07/22/14 | 5 | NAN |
| 5 | MICHAEL PALIN | 06/28/13 | 66000.0 | 5 |
| 6 | MICHAEL PALIN | 06/28/13 | 66000.0 | 5 |

NAN STANDS FOR NOT A NUMBER AND IS ONE OF THE COMMON WAYS TO REPRESENT THE MISSING VALUE IN THE DATA.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Pandas - Cleaning Empty Cells


Empty cells can potentially give you a wrong result when you analyze data.

## Remove Rows

- One way to deal with empty cells is to remove rows that contain empty cells
- This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

```
import pandas as pd

df = pd.read_csv('data.csv')

new_df = df.dropna()

print(new_df.to_string())
```

By default, the dropna() method returns a new DataFrame, and will not change the original.

If you want to change the original DataFrame, use the inplace = True argument: `df.dropna(inplace = True)`

Data Set- https://www.w3schools.com/python/pandas/pandas_cleaning_empty_cells.asp

# Pandas - Cleaning Empty Cells

## Replace Empty Values

- Another way of dealing with empty cells is to insert a new value instead.
- This way you do not have to delete entire rows just because of some empty cells.
- The fillna() method allows us to replace empty cells with a value

```
import pandas as pd

df = pd.read_csv('data.csv')

df.fillna(130, inplace = True)
```

Replace only for a specified columns

```
import pandas as pd

df = pd.read_csv('data.csv')

df["Calories"].fillna(130, inplace = True)
```

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Pandas - Cleaning Empty Cells

## Replace using mean, median, or mode

- A common way to replace empty cells, is to calculate the mean, median or mode value of the column.

- Pandas uses the mean() median() and mode() methods to calculate the respective values for a specified column

```
import pandas as pd

df = pd.read_csv('data.csv')

x = df["Calories"].mean()

df["Calories"].fillna(x, inplace = True)
```

mean = the average value (the sum of all values divided by number of values).

# Pandas - Cleaning Empty Cells


CALCULATE THE MEDIAN, AND REPLACE ANY EMPTY VALUES WITH IT

MEDIAN = THE VALUE IN THE MIDDLE, AFTER YOU HAVE SORTED ALL VALUES ASCENDING.

```
IMPORT PANDAS AS PD

DF = PD.READ_CSV('DATA.CSV')

X = DF["CALORIES"].MEDIAN()

DF["CALORIES"].FILLNA(X, INPLACE = TRUE)
```

# Pandas - Cleaning Empty Cells


CALCULATE THE MODE, AND REPLACE ANY EMPTY VALUES WITH IT:

MODE = THE VALUE THAT APPEARS MOST FREQUENTLY.

```
IMPORT PANDAS AS PD

DF = PD.READ_CSV('DATA.CSV')

X = DF["CALORIES"].MODE()[0]

DF["CALORIES"].FILLNA(X, INPLACE = TRUE)
```

# Pandas - Cleaning Data of Wrong Format

CELLS WITH DATA OF WRONG FORMAT CAN MAKE IT DIFFICULT, OR EVEN IMPOSSIBLE, TO ANALYZE DATA.

| | DURATION | DATE | PULSE | MAXPULSE |
|---|---|---|---|---|
| 0 | 60 | '2020/12/01' | 110 | 130 |
| 1 | 60 | '2020/12/02' | 117 | 145 |
| 2 | 60 | '2020/12/03' | 103 | 135 |
| 3 | 45 | '2020/12/04' | 109 | 175 |
| 4 | 45 | 20201205 | 117 | 148 |
| 5 | 60 | '2020/12/06' | 102 | 127 |
| 6 | 60 | NAN | 110 | 136 |
| 7 | 450 | '2020/12/08' | 104 | 134 |

IN OUR DATA FRAME, WE HAVE TWO CELLS WITH THE WRONG FORMAT. CHECK OUT ROW 4 AND 6, THE 'DATE' COLUMN SHOULD BE A STRING THAT REPRESENTS A DATE

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Pandas - Cleaning Data of Wrong Format

| | DURATION | DATE | PULSE | MAXPULSE |
|---|---|---|---|---|
| 0 | 60 | '2020/12/01' | 110 | 130 |
| 1 | 60 | '2020/12/02' | 117 | 145 |
| 2 | 60 | '2020/12/03' | 103 | 135 |
| 3 | 45 | '2020/12/04' | 109 | 175 |
| 4 | 45 | 20201205 | 117 | 148 |
| 5 | 60 | '2020/12/06' | 102 | 127 |
| 6 | 60 | NAN | 110 | 136 |
| 7 | 450 | '2020/12/08' | 104 | 134 |

TO FIX IT, YOU HAVE TWO OPTIONS:

- REMOVE THE ROWS, OR
- CONVERT ALL CELLS IN THE COLUMNS INTO THE SAME FORMAT.

# Pandas - Cleaning Data of Wrong Format

CONVERT ALL CELLS IN THE COLUMNS INTO THE SAME FORMAT.

LET'S TRY TO CONVERT ALL CELLS IN THE 'DATE' COLUMN INTO DATES.

PANDAS HAS A TO_DATETIME() METHOD FOR THIS:

```
IMPORT PANDAS AS PD

DF = PD.READ_CSV('DATA.CSV')

DF['DATE'] = PD.TO_DATETIME(DF['DATE'])

PRINT(DF.TO_STRING())
```

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Pandas - Cleaning Data of Wrong Format

| | DURATION | DATE | PULSE | MAXPULSE |
|---|---|---|---|---|
| 0 | 60 | '2020/12/01' | 110 | 130 |
| 1 | 60 | '2020/12/02' | 117 | 145 |
| 2 | 60 | '2020/12/03' | 103 | 135 |
| 3 | 45 | '2020/12/04' | 109 | 175 |
| 4 | 45 | '2020/12/05' | 117 | 148 |
| 5 | 60 | '2020/12/06' | 102 | 127 |
| 6 | 60 | NAT | 110 | 136 |
| 7 | 450 | '2020/12/08' | 104 | 134 |

AS YOU CAN SEE FROM THE RESULT, THE DATE IN ROW 4 WAS FIXED, BUT THE EMPTY DATE IN ROW 6 GOT A NAT (NOT A TIME) VALUE, IN OTHER WORDS AN EMPTY VALUE.

ONE WAY TO DEAL WITH EMPTY VALUES IS SIMPLY REMOVING THE ENTIRE ROW.

# Pandas - Cleaning Data of Wrong Format


## REMOVING ROWS

THE RESULT FROM THE CONVERTING IN THE EXAMPLE ABOVE GAVE US A NAT VALUE, WHICH CAN BE HANDLED AS A NULL VALUE, AND WE CAN REMOVE THE ROW BY USING THE DROPNA() METHOD.

```
DF.DROPNA(SUBSET=['DATE'], INPLACE = TRUE)
```

```
IMPORT PANDAS AS PD

DF = PD.READ_CSV('DATA.CSV')

DF['DATE'] = PD.TO_DATETIME(DF['DATE'])

DF.DROPNA(SUBSET=['DATE'], INPLACE = TRUE)

PRINT(DF.TO_STRING())
```

# Pandas - Cleaning Data of Wrong Format

## PANDAS - FIXING WRONG DATA

"WRONG DATA" DOES NOT HAVE TO BE "EMPTY CELLS" OR "WRONG FORMAT", IT CAN JUST BE WRONG, LIKE IF SOMEONE REGISTERED "199" INSTEAD OF "1.99".

## Ways to do

REPLACING VALUES

REMOVING ROWS

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Pandas - Cleaning Data of Wrong Format


## PANDAS - FIXING WRONG DATA

| | DURATION | DATE | PULSE | MAXPULSE |
|---|---|---|---|---|
| 0 | 60 | '2020/12/01' | 110 | 130 |
| 1 | 60 | '2020/12/02' | 117 | 145 |
| 2 | 60 | '2020/12/03' | 103 | 135 |
| 3 | 45 | '2020/12/04' | 109 | 175 |
| 4 | 45 | '2020/12/05' | 117 | 148 |
| 5 | 60 | '2020/12/06' | 102 | 127 |
| 7 | 450 | '2020/12/08' | 104 | 134 |

## REPLACING VALUES

```
FOR X IN DF.INDEX:
  IF DF.LOC[X, "DURATION"] > 120:
    DF.LOC[X, "DURATION"] = 120
```

# Pandas - Cleaning Data of Wrong Format


## PANDAS - FIXING WRONG DATA

## REMOVING ROWS

| | DURATION | DATE | PULSE | MAXPULSE |
|---|---|---|---|---|
| 0 | 60 | '2020/12/01' | 110 | 130 |
| 1 | 60 | '2020/12/02' | 117 | 145 |
| 2 | 60 | '2020/12/03' | 103 | 135 |
| 3 | 45 | '2020/12/04' | 109 | 175 |
| 4 | 45 | '2020/12/05' | 117 | 148 |
| 5 | 60 | '2020/12/06' | 102 | 127 |
| 7 | 450 | '2020/12/08' | 104 | 134 |

```
FOR X IN DF.INDEX:
  IF DF.LOC[X, "DURATION"] > 120:
    DF.DROP(X, INPLACE = TRUE)
```

# Pandas - Removing Duplicates

| | DURATION | DATE | PULSE | MAXPULSE |
|---|---|---|---|---|
| 0 | 60 | '2020/12/01' | 110 | 130 |
| 1 | 60 | '2020/12/02' | 117 | 145 |
| 2 | 60 | '2020/12/03' | 103 | 135 |
| 3 | 45 | '2020/12/04' | 109 | 175 |
| 4 | 45 | '2020/12/04' | 109 | 175 |
| 5 | 60 | '2020/12/06' | 102 | 127 |
| 6 | 450 | '2020/12/08' | 104 | 134 |

TO DISCOVER DUPLICATES, WE CAN USE THE DUPLICATED() METHOD.

THE DUPLICATED() METHOD RETURNS A BOOLEAN VALUES FOR EACH ROW

# Pandas - Removing Duplicates

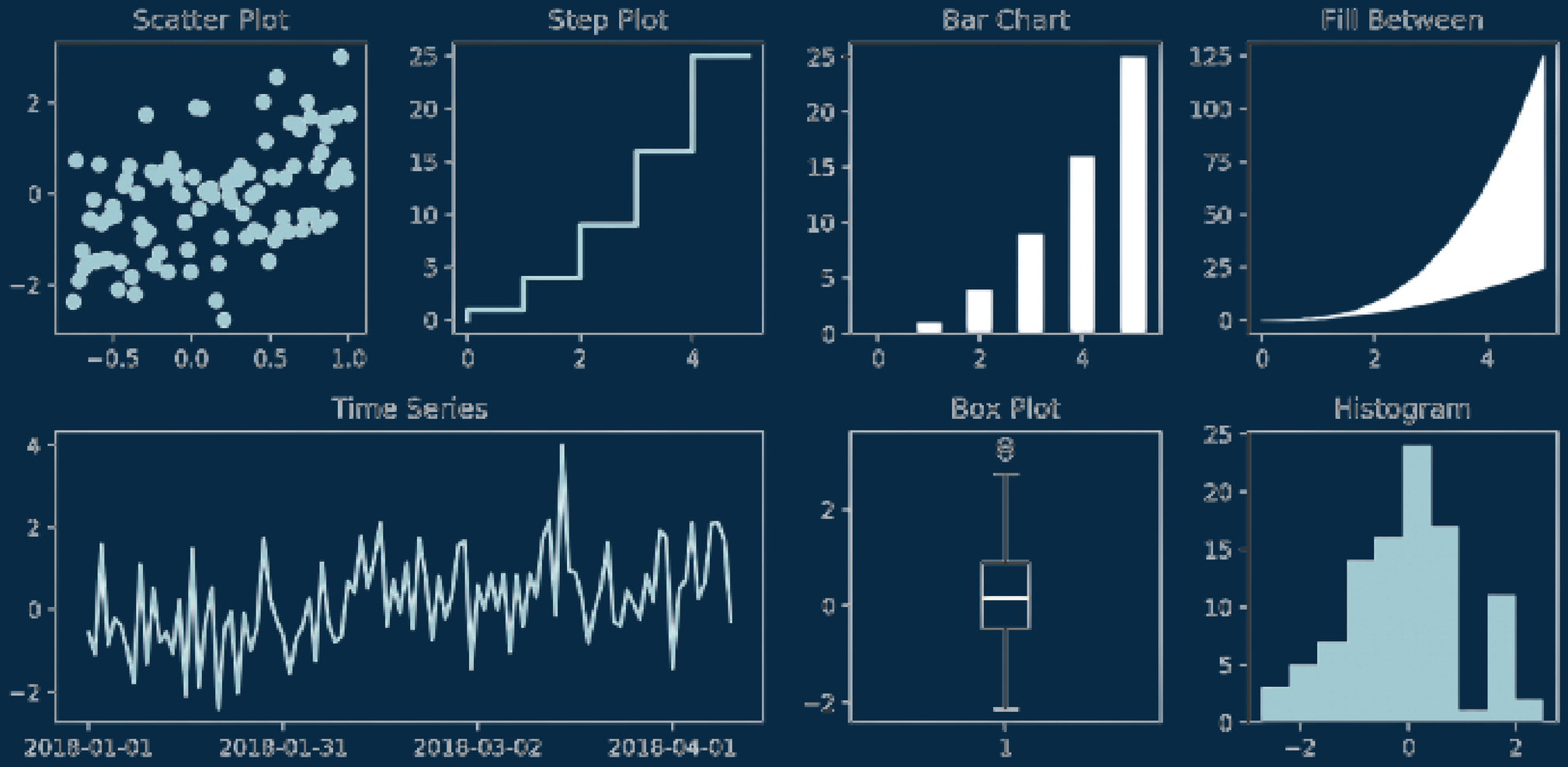
```
PRINT(DF.DUPLICATED())
```

RETURNS TRUE FOR EVERY ROW THAT IS A DUPLICATE, OTHWERWISE FALSE:

TO REMOVE DUPLICATES, USE THE DROP_DUPLICATES() METHOD.

```
DF.DROP_DUPLICATES(INPLACE = TRUE)
```

# Matplotlib Pyplot (Draw a Graph)




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

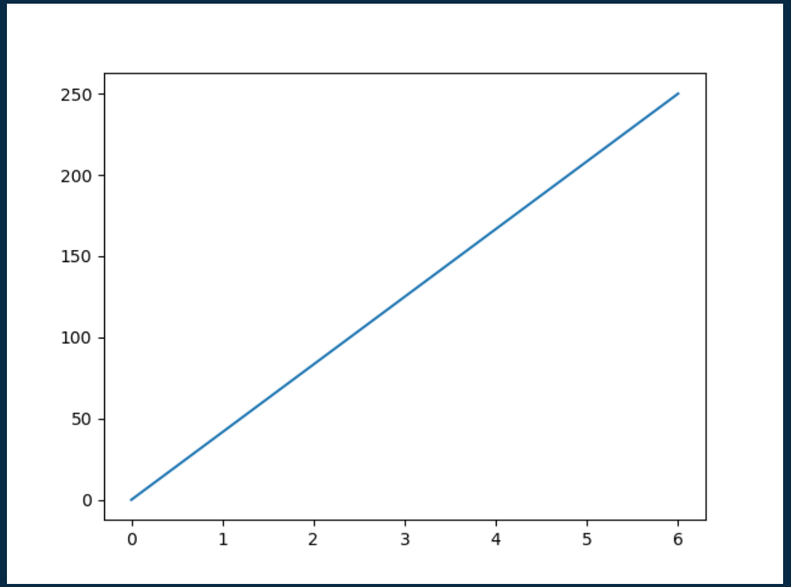
# Matplotlib Pyplot (Draw a Graph)

MOST OF THE MATPLOTLIB UTILITIES LIES UNDER THE PYPLOT SUBMODULE

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
```

EXAMPLE- DRAW A LINE IN A DIAGRAM FROM POSITION (0,0) TO POSITION (6,250)

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

XPOINTS = NP.ARRAY([0, 6])
YPOINTS = NP.ARRAY([0, 250])

PLT.PLOT(XPOINTS, YPOINTS)
PLT.SHOW()
```




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Pyplot (Draw a Graph)

## PLOTTING X AND Y POINTS

PLT.PLOT(XPOINTS, YPOINTS)

THE PLOT() FUNCTION IS USED TO DRAW POINTS (MARKERS) IN A DIAGRAM.

BY DEFAULT, THE PLOT() FUNCTION DRAWS A LINE FROM POINT TO POINT.

THE FUNCTION TAKES PARAMETERS FOR SPECIFYING POINTS IN THE DIAGRAM.

PARAMETER 1 IS AN ARRAY CONTAINING THE POINTS ON THE X-AXIS.

PARAMETER 2 IS AN ARRAY CONTAINING THE POINTS ON THE Y-AXIS.

IF WE NEED TO PLOT A LINE FROM (1, 3) TO (8, 10),
WE HAVE TO PASS TWO ARRAYS [1, 8] AND [3, 10] TO THE PLOT FUNCTION.

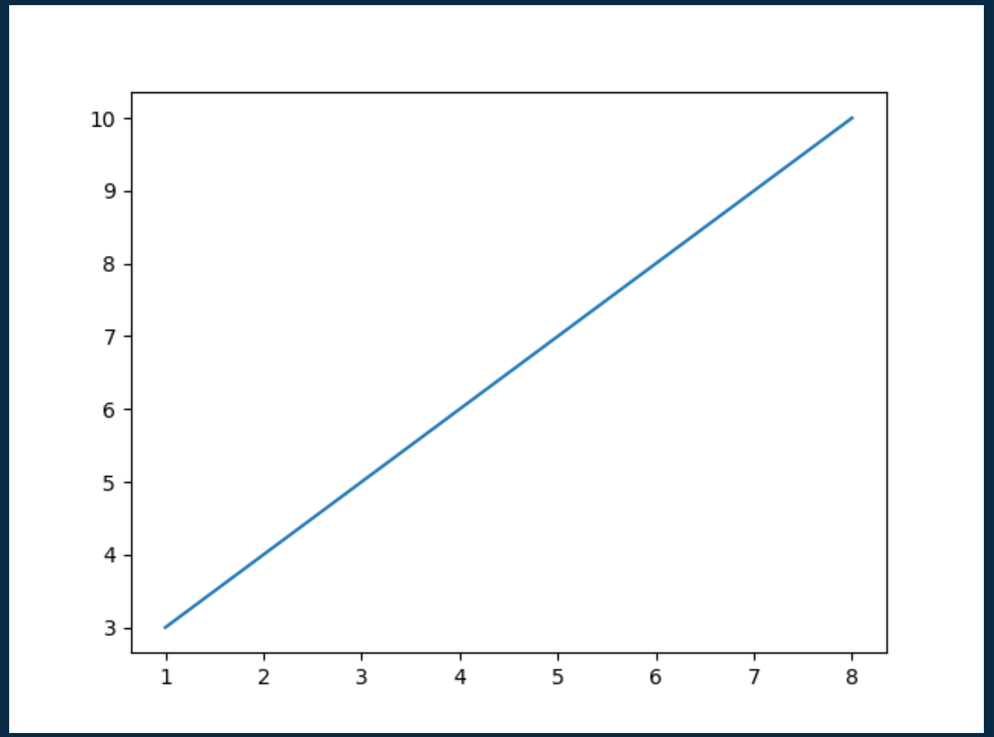
~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Pyplot (Draw a Graph)

## PLOTTING X AND Y POINTS

DRAW A LINE IN A DIAGRAM FROM POSITION (1, 3) TO POSITION (8, 10):

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

XPOINTS = NP.ARRAY([1, 8])
YPOINTS = NP.ARRAY([3, 10])

PLT.PLOT(XPOINTS, YPOINTS)
PLT.SHOW()
```



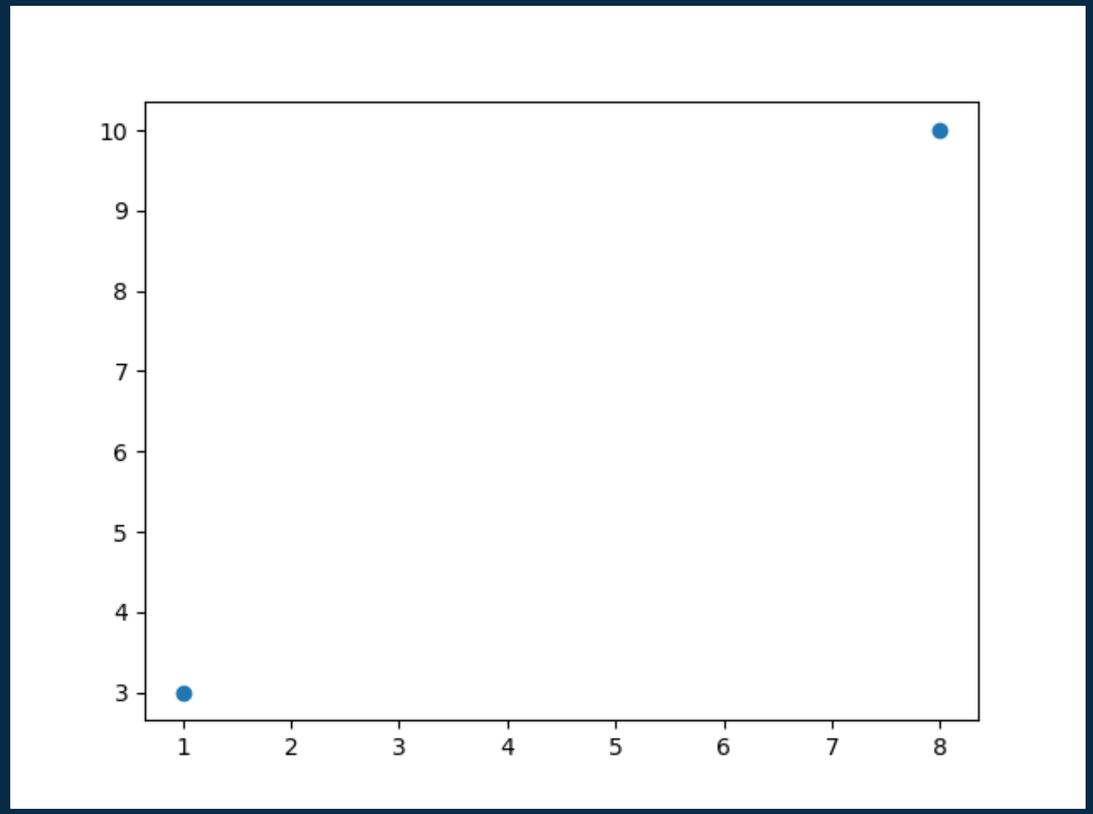

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Pyplot (Draw a Graph)

## PLOTTING WITHOUT LINE

DRAW TWO POINTS IN THE DIAGRAM, ONE AT POSITION (1, 3) AND ONE IN POSITION (8, 10):

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

XPOINTS = NP.ARRAY([1, 8])
YPOINTS = NP.ARRAY([3, 10])

PLT.PLOT(XPOINTS, YPOINTS, 'O')
PLT.SHOW()
```



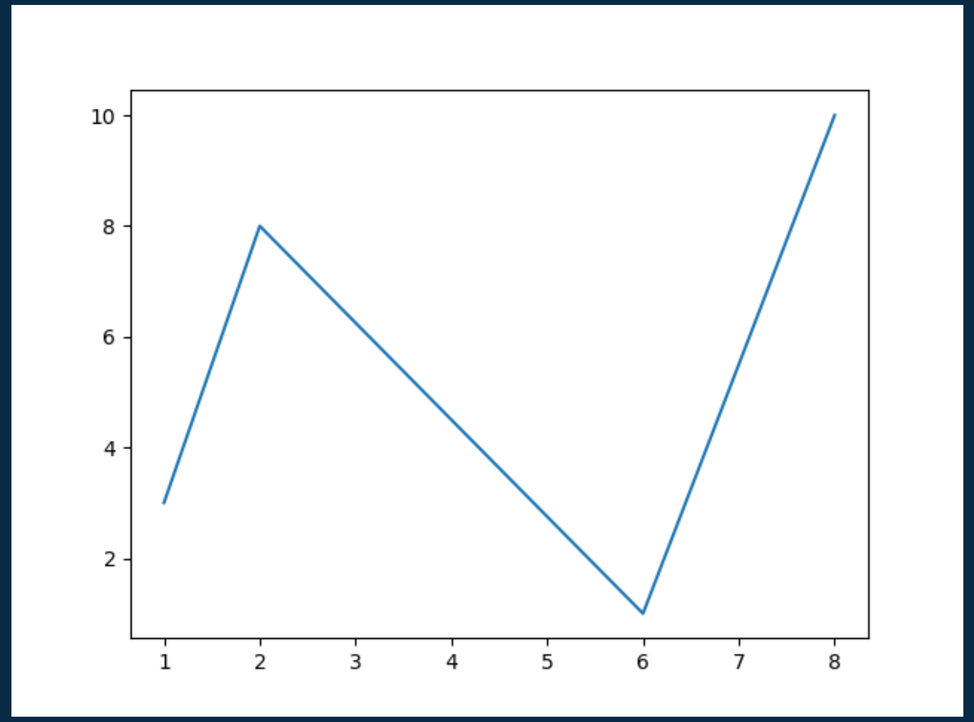

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Pyplot (Draw a Graph)

## MULTIPLE POINTS

DRAW A LINE IN A DIAGRAM FROM POSITION (1, 3) TO (2, 8) THEN TO (6, 1) AND FINALLY TO POSITION (8, 10):

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

XPOINTS = NP.ARRAY([1, 2, 6, 8])
YPOINTS = NP.ARRAY([3, 8, 1, 10])

PLT.PLOT(XPOINTS, YPOINTS)
PLT.SHOW()
```




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

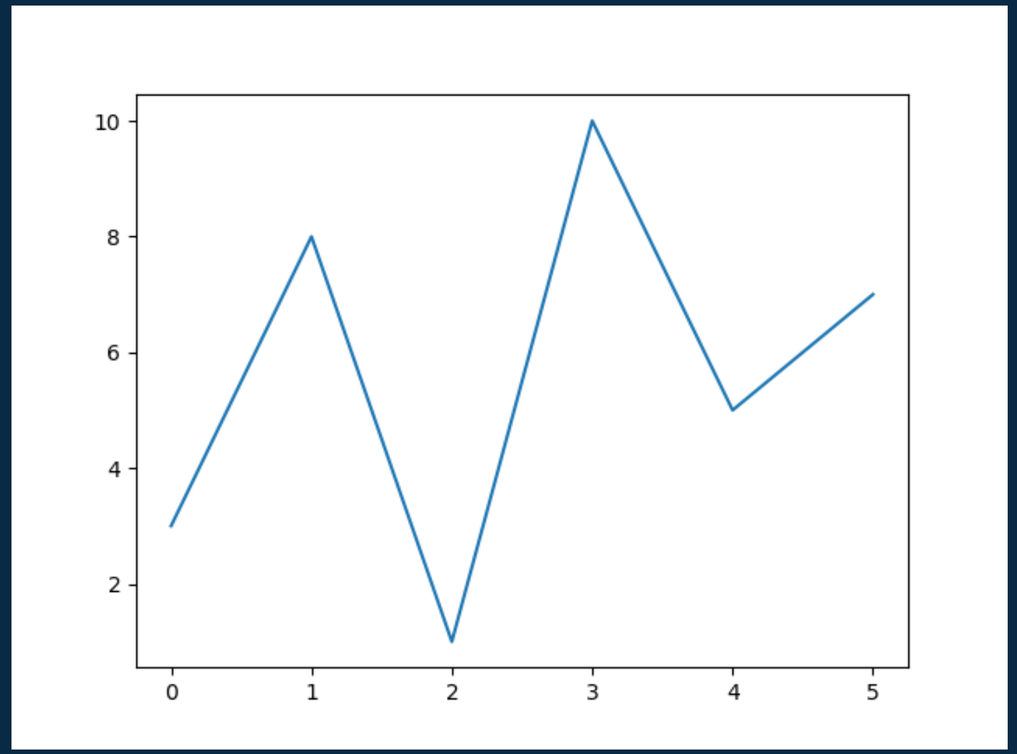
# Matplotlib Pyplot (Draw a Graph)

## Default X-Points

If we do not specify the points in the x-axis, they will get the default values 0, 1, 2, 3, (etc. depending on the length of the y-points.

Plotting without x-points

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)
plt.show()
```




The x-points in the example above is [0, 1, 2, 3, 4, 5].

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

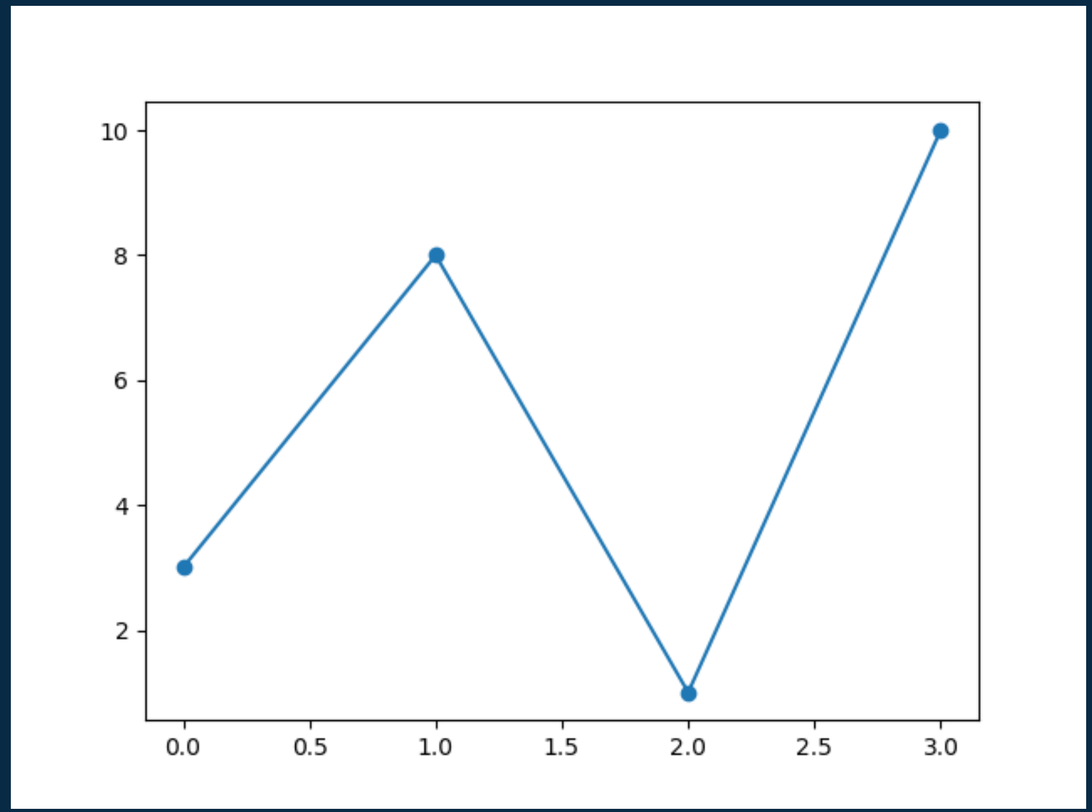
# Matplotlib Pyplot (Draw a Graph)

## MATPLOTLIB MARKERS

YOU CAN USE THE KEYWORD ARGUMENT MARKER TO EMPHASIZE EACH POINT WITH A SPECIFIED MARKER

MARK EACH POINT WITH A CIRCLE

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

YPOINTS = NP.ARRAY([3, 8, 1, 10])

PLT.PLOT(YPOINTS, MARKER = 'O')
PLT.SHOW()
```




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

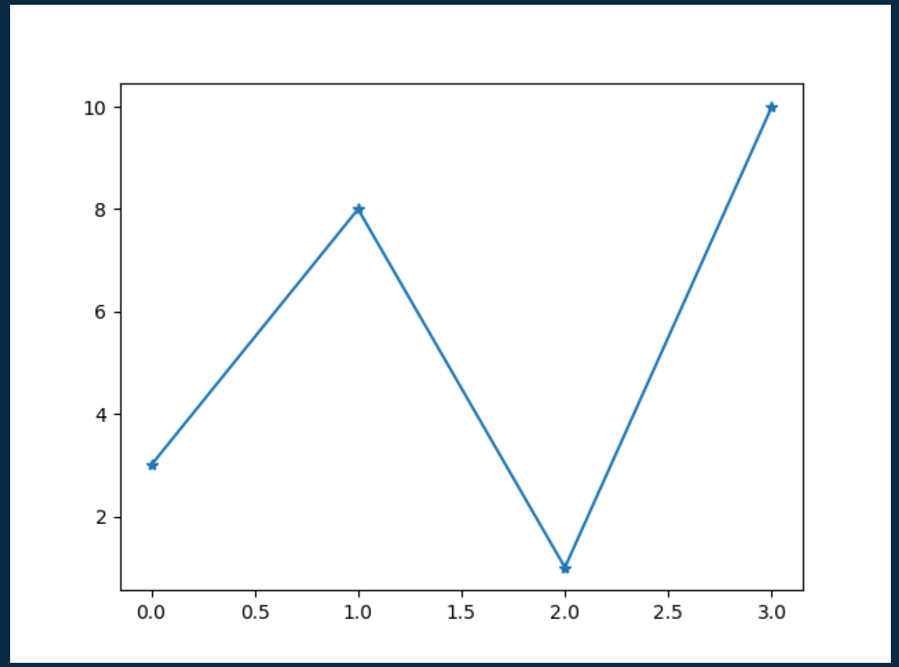
# Matplotlib Pyplot (Draw a Graph)

## MATPLOTLIB MARKERS

YOU CAN USE THE KEYWORD ARGUMENT MARKER TO EMPHASIZE EACH POINT WITH A SPECIFIED MARKER

MARK EACH POINT WITH A STAR

```
PLT.PLOT(YPOINTS, MARKER = '*')
```



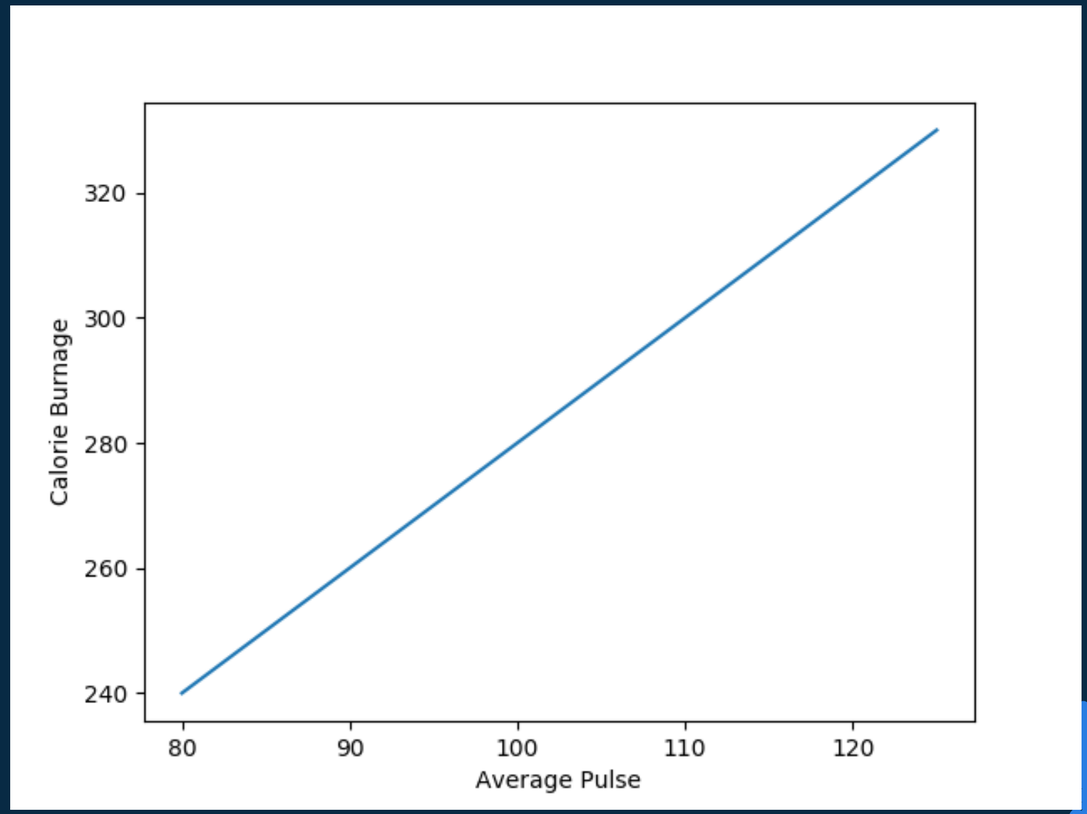

MARKER DESCRIPTION
'O' CIRCLE
'*' STAR
'.' POINT
',' PIXEL
'X' X
'X' X (FILLED)
'+' PLUS
'P' PLUS (FILLED)
'S' SQUARE

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Labels and Title

WITH PYPLOT, YOU CAN USE THE XLABEL() AND YLABEL() FUNCTIONS TO SET A LABEL FOR THE X- AND Y-AXIS

```
IMPORT NUMPY AS NP
IMPORT MATPLOTLIB.PYPLOT AS PLT

X = NP.ARRAY([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
Y = NP.ARRAY([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

PLT.PLOT(X, Y)

PLT.XLABEL("AVERAGE PULSE")
PLT.YLABEL("CALORIE BURNAGE")

PLT.SHOW()
```




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Subplots

## Display Multiple Plots

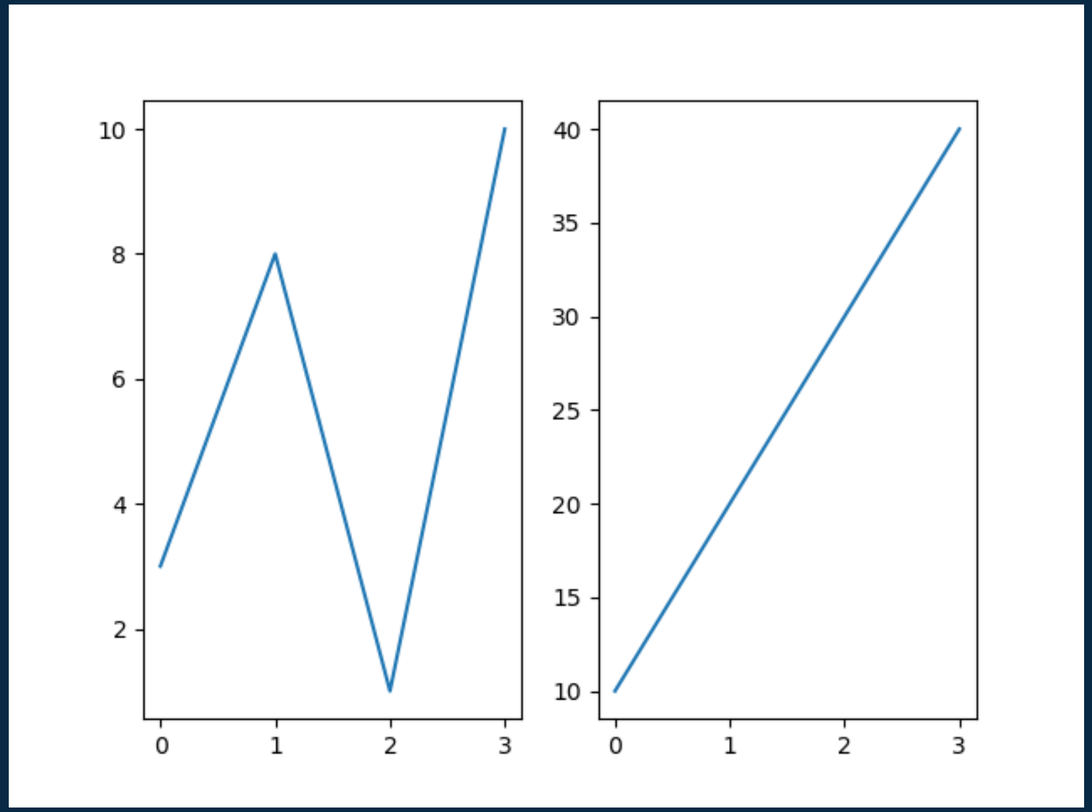
With the subplots() function you can draw multiple plots in one figure

The subplots() function takes three arguments that describes the layout of the figure.

The layout is organized in rows and columns, which are represented by the first and second argument.

The third argument represents the index of the current plot.

```
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.
```

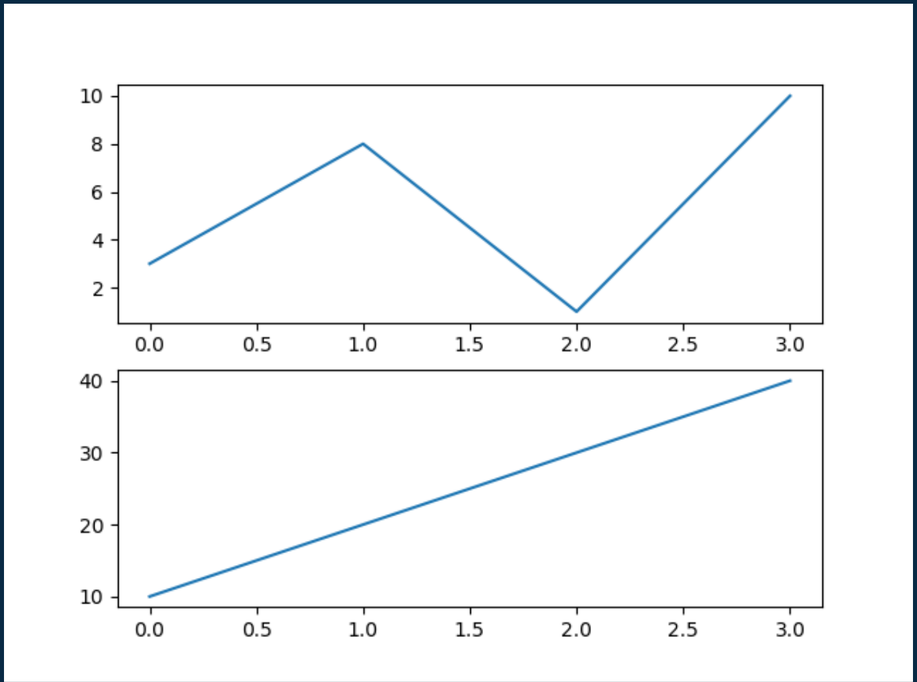



~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Subplots

```
PLT.SUBPLOT(1, 2, 2)
#THE FIGURE HAS 1 ROW, 2 COLUMNS, AND THIS PLOT IS THE SECOND PLOT.
```

SO, IF WE WANT A FIGURE WITH 2 ROWS AN 1 COLUMN (MEANING THAT THE TWO PLOTS WILL BE DISPLAYED ON TOP OF EACH OTHER INSTEAD OF SIDE-BY-SIDE), WE CAN WRITE THE SYNTAX LIKE THIS:

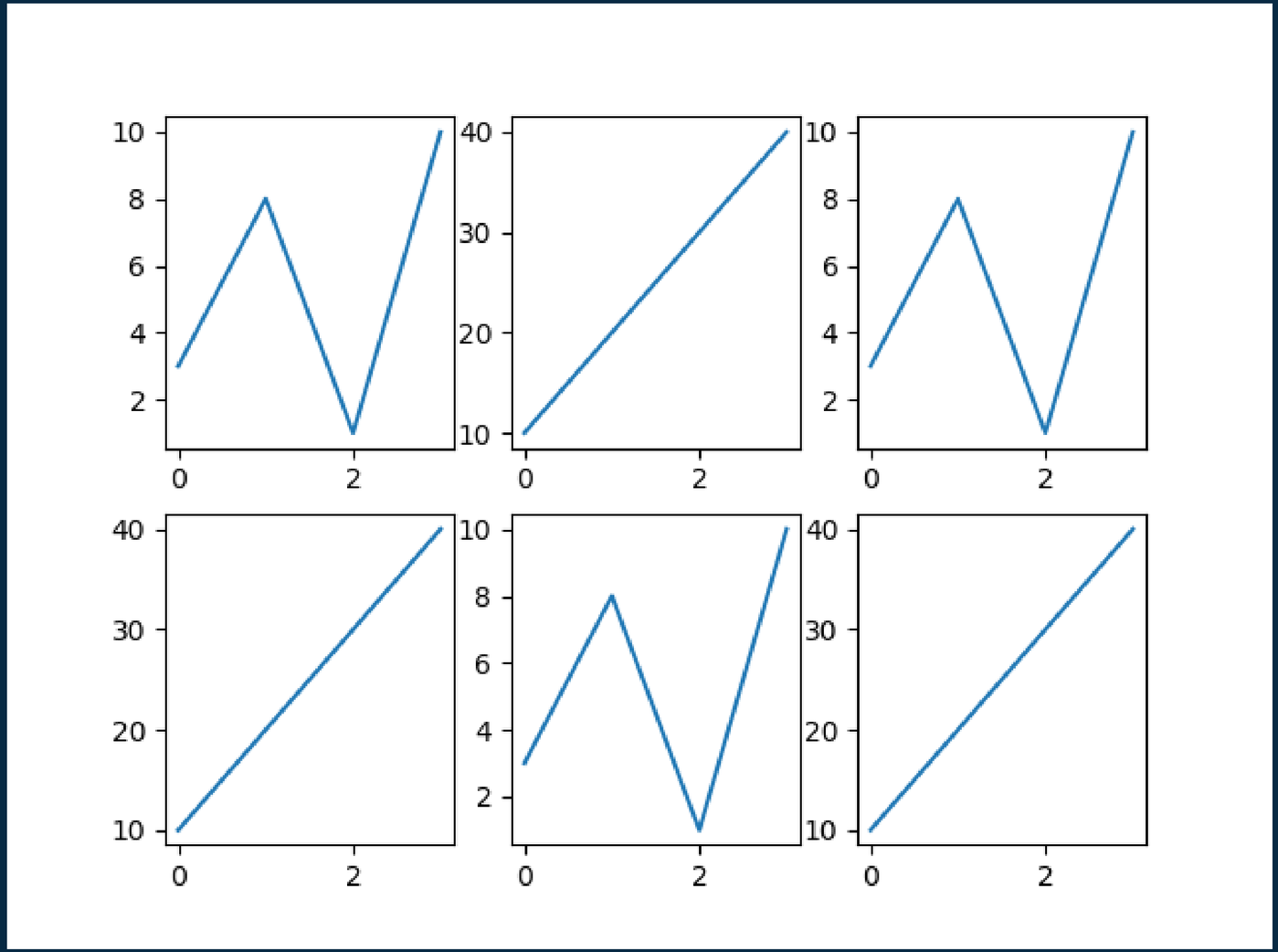



```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

#PLOT 1:
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([3, 8, 1, 10])

PLT.SUBPLOT(2, 1, 1)
PLT.PLOT(X,Y)


#PLOT 2:
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([10, 20, 30, 40])
PLT.SUBPLOT(2, 1, 2)
PLT.PLOT(X,Y)
PLT.SHOW()
```

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Subplots

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([3, 8, 1, 10])
PLT.SUBPLOT(2, 3, 1)
PLT.PLOT(X,Y)
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([10, 20, 30, 40])
PLT.SUBPLOT(2, 3, 2)
PLT.PLOT(X,Y)
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([3, 8, 1, 10])
PLT.SUBPLOT(2, 3, 3)
PLT.PLOT(X,Y)
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([10, 20, 30, 40])
PLT.SUBPLOT(2, 3, 4)
PLT.PLOT(X,Y)
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([3, 8, 1, 10])
PLT.SUBPLOT(2, 3, 5)
PLT.PLOT(X,Y)
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([10, 20, 30, 40])
PLT.SUBPLOT(2, 3, 6)
PLT.PLOT(X,Y)
PLT.SHOW()
```

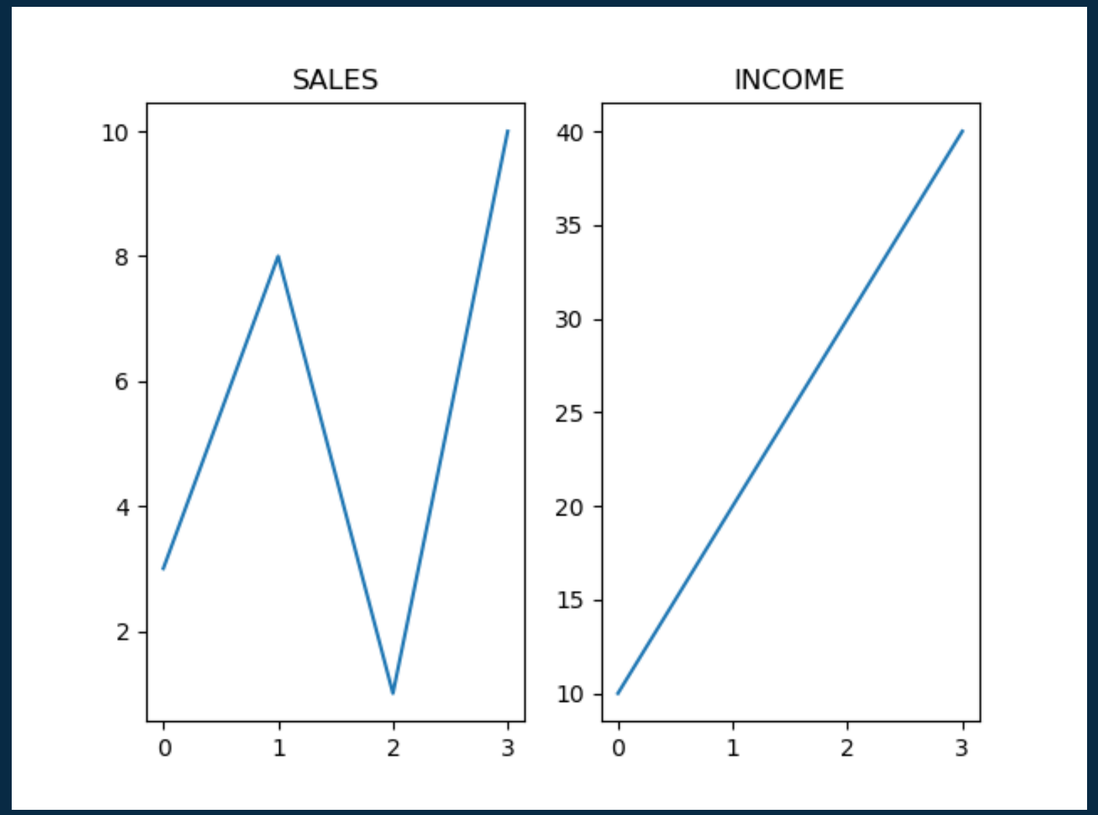


~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Title

YOU CAN ADD A TITLE TO EACH PLOT WITH THE TITLE() FUNCTION `PLT.TITLE("SALES")`

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP
#PLOT 1:
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([3, 8, 1, 10])
PLT.SUBPLOT(1, 2, 1)
PLT.PLOT(X,Y)
PLT.TITLE("SALES")
#PLOT 2:
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([10, 20, 30, 40])
PLT.SUBPLOT(1, 2, 2)
PLT.PLOT(X,Y)
PLT.TITLE("INCOME")
PLT.SHOW()
```

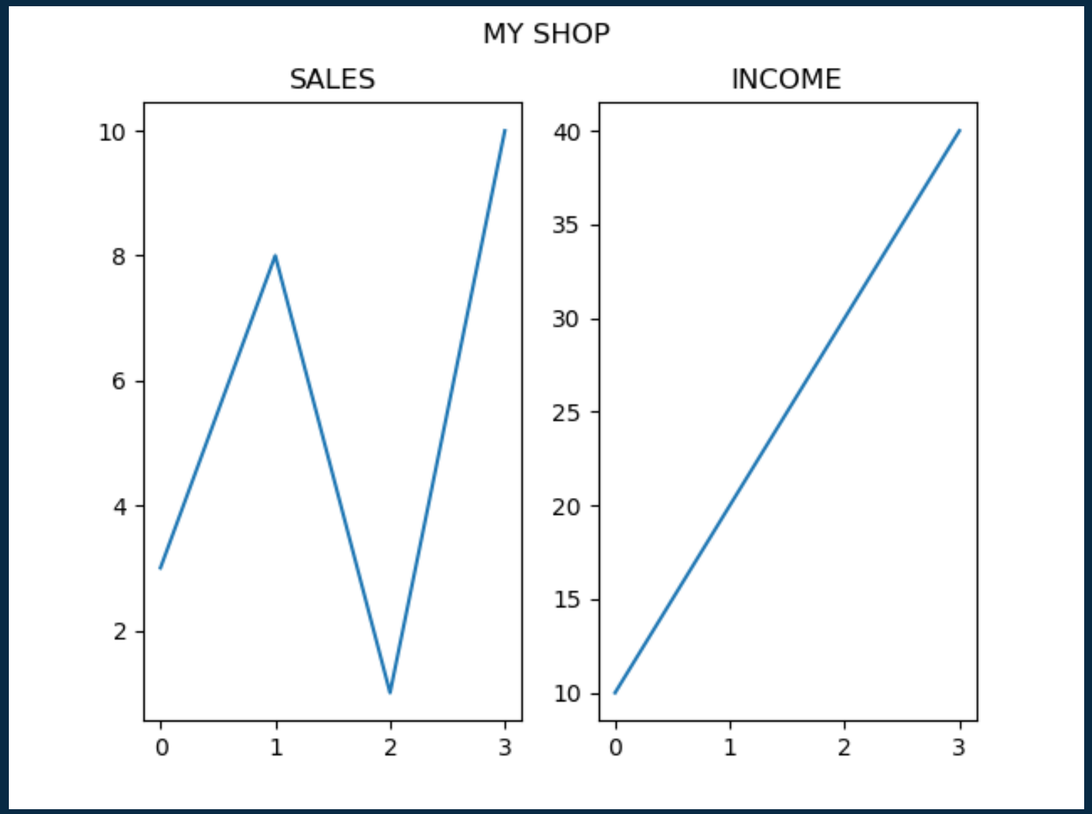



~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Super Title

YOU CAN ADD A TITLE TO THE ENTIRE FIGURE WITH THE SUPTITLE() FUNCTION

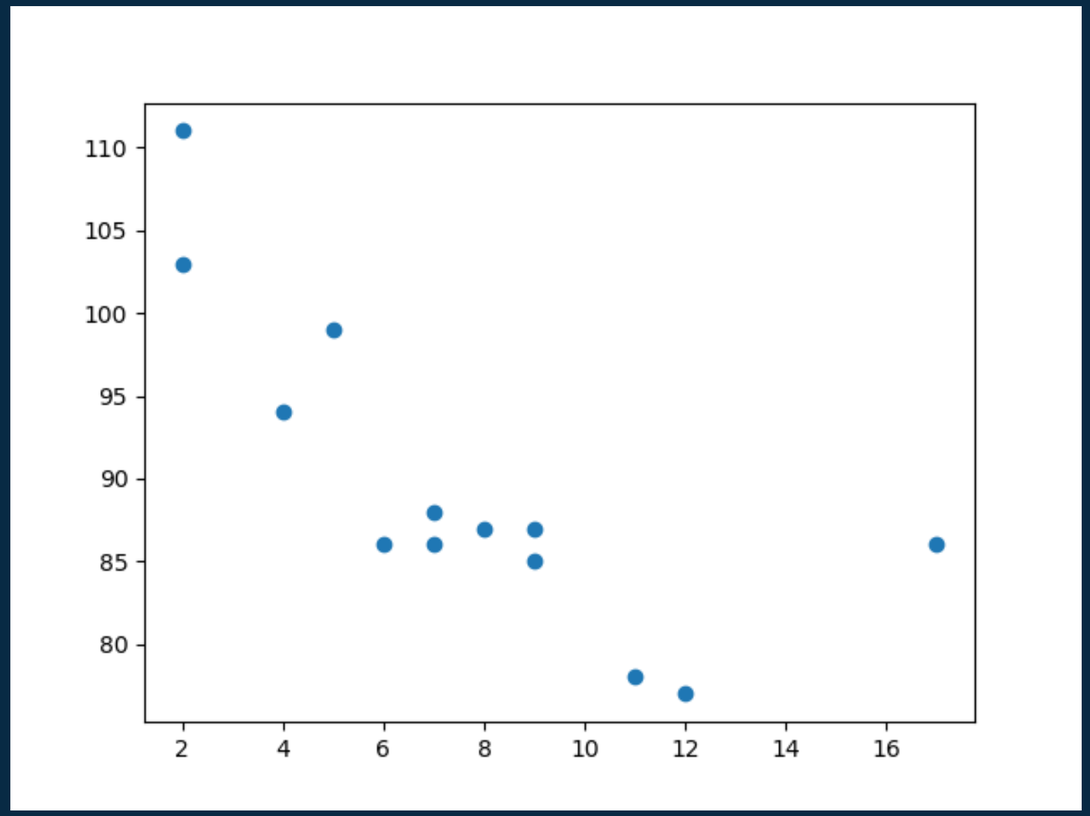
```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP
#PLOT 1:
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([3, 8, 1, 10])
PLT.SUBPLOT(1, 2, 1)
PLT.PLOT(X,Y)
PLT.TITLE("SALES")
#PLOT 2:
X = NP.ARRAY([0, 1, 2, 3])
Y = NP.ARRAY([10, 20, 30, 40])
PLT.SUBPLOT(1, 2, 2)
PLT.PLOT(X,Y)
PLT.TITLE("INCOME")
PLT.SUPTITLE("MY SHOP")
PLT.SHOW()
```




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Scatter

- WITH PYPLOT, YOU CAN USE THE SCATTER() FUNCTION TO DRAW A SCATTER PLOT.
- THE SCATTER() FUNCTION PLOTS ONE DOT FOR EACH OBSERVATION. IT NEEDS TWO ARRAYS OF THE SAME LENGTH, ONE FOR THE VALUES OF THE X-AXIS, AND ONE FOR VALUES ON THE Y-AXIS




```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

X = NP.ARRAY([5,7,8,7,2,17,2,9,4,11,12,9,6])
Y = NP.ARRAY([99,86,87,88,111,86,103,87,94,78,77,85,86])

PLT.SCATTER(X, Y)

PLT.SHOW()
```

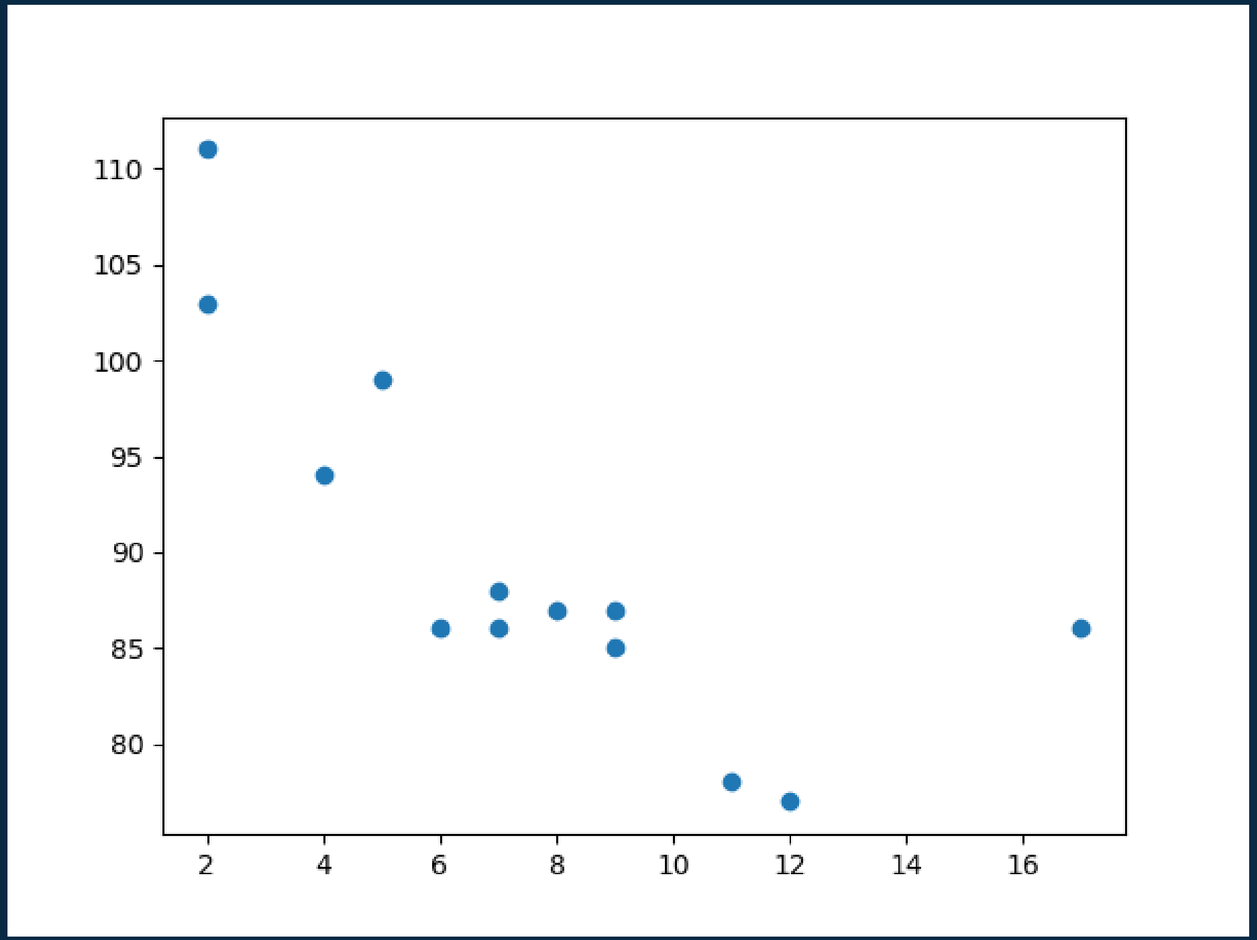
~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Scatter

- THE OBSERVATION IN THE EXAMPLE BELOW IS THE RESULT OF 13 CARS PASSING BY.
- THE X-AXIS SHOWS HOW OLD THE CAR IS.
- THE Y-AXIS SHOWS THE SPEED OF THE CAR WHEN IT PASSES.

ARE THERE ANY RELATIONSHIPS BETWEEN THE OBSERVATIONS?

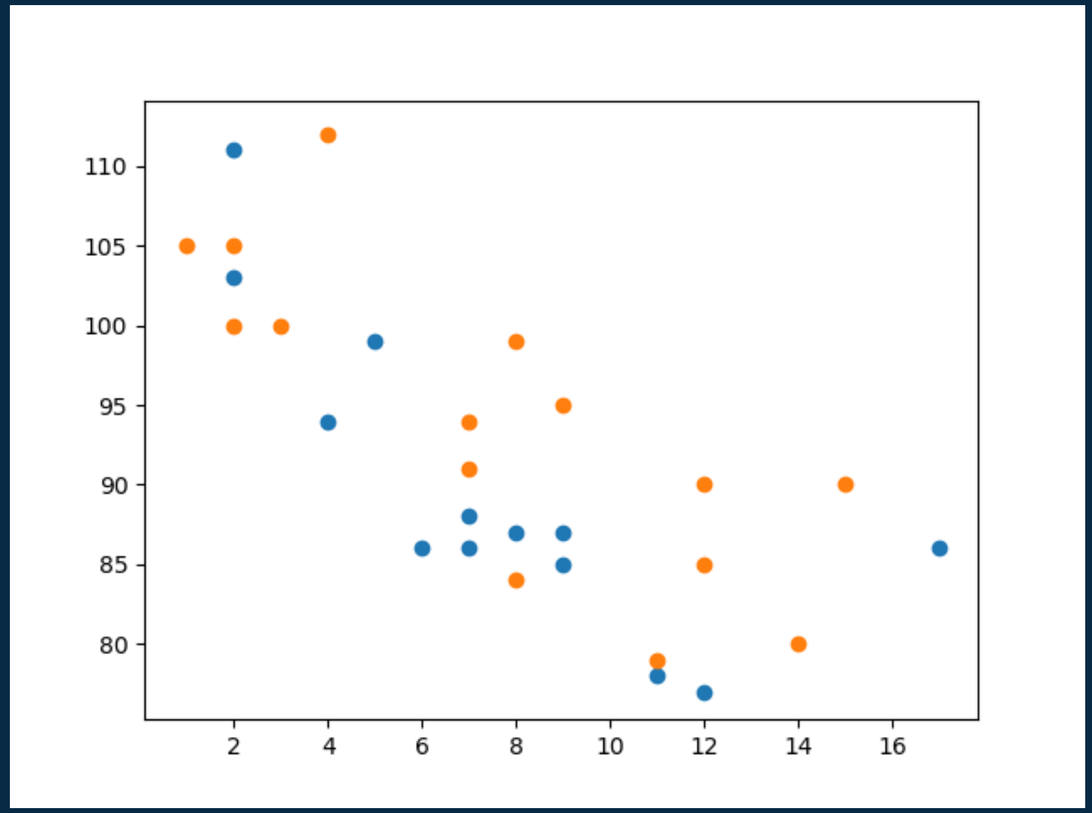
- IT SEEMS THAT THE NEWER THE CAR, THE FASTER IT DRIVES,



# Matplotlib Scatter (Compare Plots)

IN THE EXAMPLE ABOVE, THERE SEEMS TO BE A RELATIONSHIP BETWEEN SPEED AND AGE, BUT WHAT IF WE PLOT THE OBSERVATIONS FROM ANOTHER DAY AS WELL? WILL THE SCATTER PLOT TELL US SOMETHING ELSE?

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP
#DAY ONE, THE AGE AND SPEED OF 13 CARS:
X = NP.ARRAY([5,7,8,7,2,17,2,9,4,11,12,9,6])
Y = NP.ARRAY([99,86,87,88,111,86,103,87,94,78,77,85,86])
PLT.SCATTER(X, Y)
#DAY TWO, THE AGE AND SPEED OF 15 CARS:
X = NP.ARRAY([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
Y =
NP.ARRAY([100,105,84,105,90,99,90,95,94,100,79,112,91,80
,85])
PLT.SCATTER(X, Y)
PLT.SHOW()
```



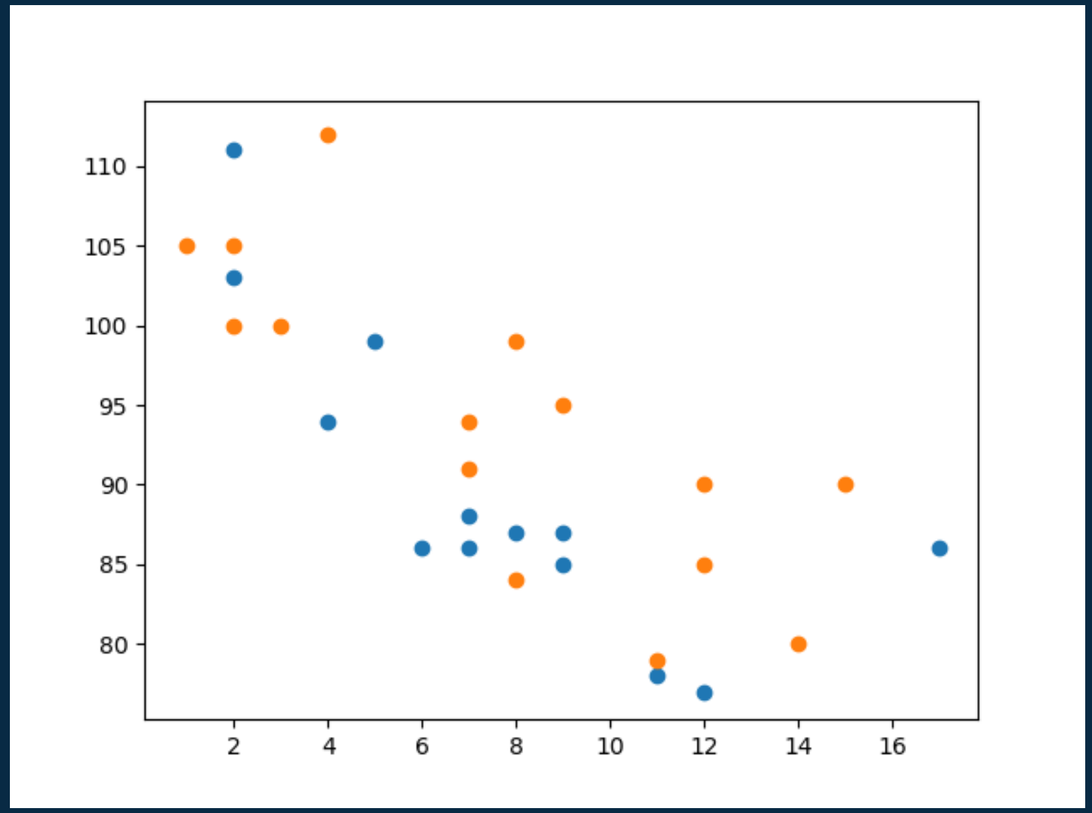

BY COMPARING THE TWO PLOTS, I THINK IT IS SAFE TO SAY THAT THEY BOTH GIVES US THE SAME CONCLUSION: THE NEWER THE CAR, THE FASTER IT DRIVES.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Scatter (Compare Plots)

IN THE EXAMPLE ABOVE, THERE SEEMS TO BE A RELATIONSHIP BETWEEN SPEED AND AGE, BUT WHAT IF WE PLOT THE OBSERVATIONS FROM ANOTHER DAY AS WELL? WILL THE SCATTER PLOT TELL US SOMETHING ELSE?

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP
#DAY ONE, THE AGE AND SPEED OF 13 CARS:
X = NP.ARRAY([5,7,8,7,2,17,2,9,4,11,12,9,6])
Y = NP.ARRAY([99,86,87,88,111,86,103,87,94,78,77,85,86])
PLT.SCATTER(X, Y)

#DAY TWO, THE AGE AND SPEED OF 15 CARS:
X = NP.ARRAY([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
Y =
NP.ARRAY([100,105,84,105,90,99,90,95,94,100,79,112,91,80
,85])
PLT.SCATTER(X, Y)
PLT.SHOW()
```



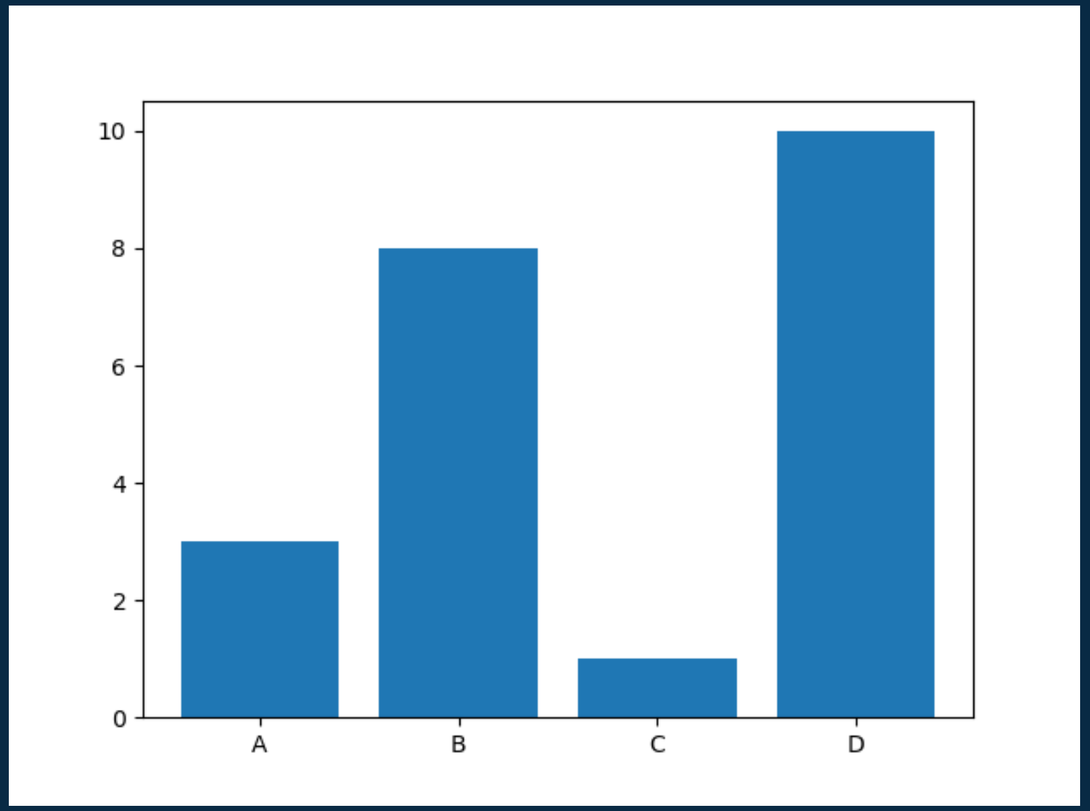

BY COMPARING THE TWO PLOTS, I THINK IT IS SAFE TO SAY THAT THEY BOTH GIVES US THE SAME CONCLUSION: THE NEWER THE CAR, THE FASTER IT DRIVES.

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Bars

WITH PYPLOT, WE CAN USE THE BAR() FUNCTION TO DRAW BAR GRAPHS

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

X = NP.ARRAY(["A", "B", "C", "D"])
Y = NP.ARRAY([3, 8, 1, 10])

PLT.BAR(X,Y)
PLT.SHOW()
```




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

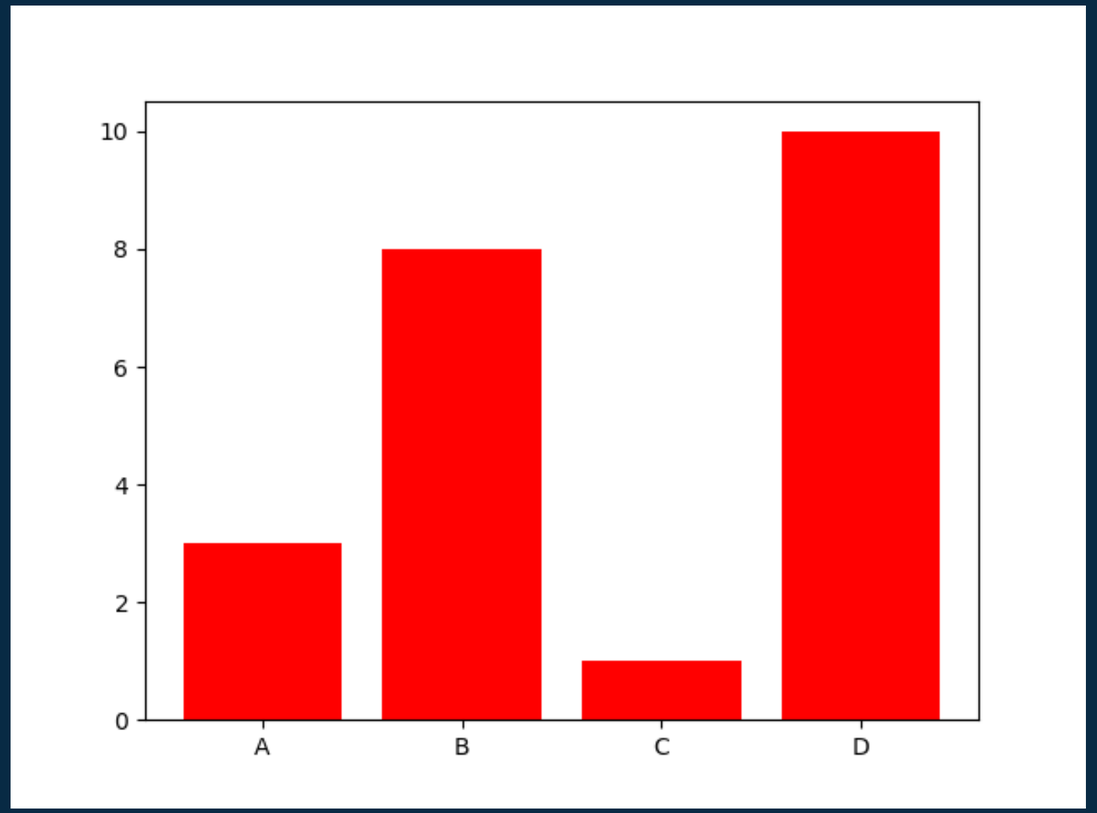
# Matplotlib Bar Color

WITH PYPLOT, WE CAN USE THE BAR() FUNCTION TO DRAW BAR GRAPHS

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

X = NP.ARRAY(["A", "B", "C", "D"])
Y = NP.ARRAY([3, 8, 1, 10])

PLT.BAR(X, Y, COLOR = "RED")
PLT.SHOW()
```

```
PLT.BAR(X, Y, WIDTH = 0.1)
```

THE DEFAULT WIDTH VALUE IS 0.8



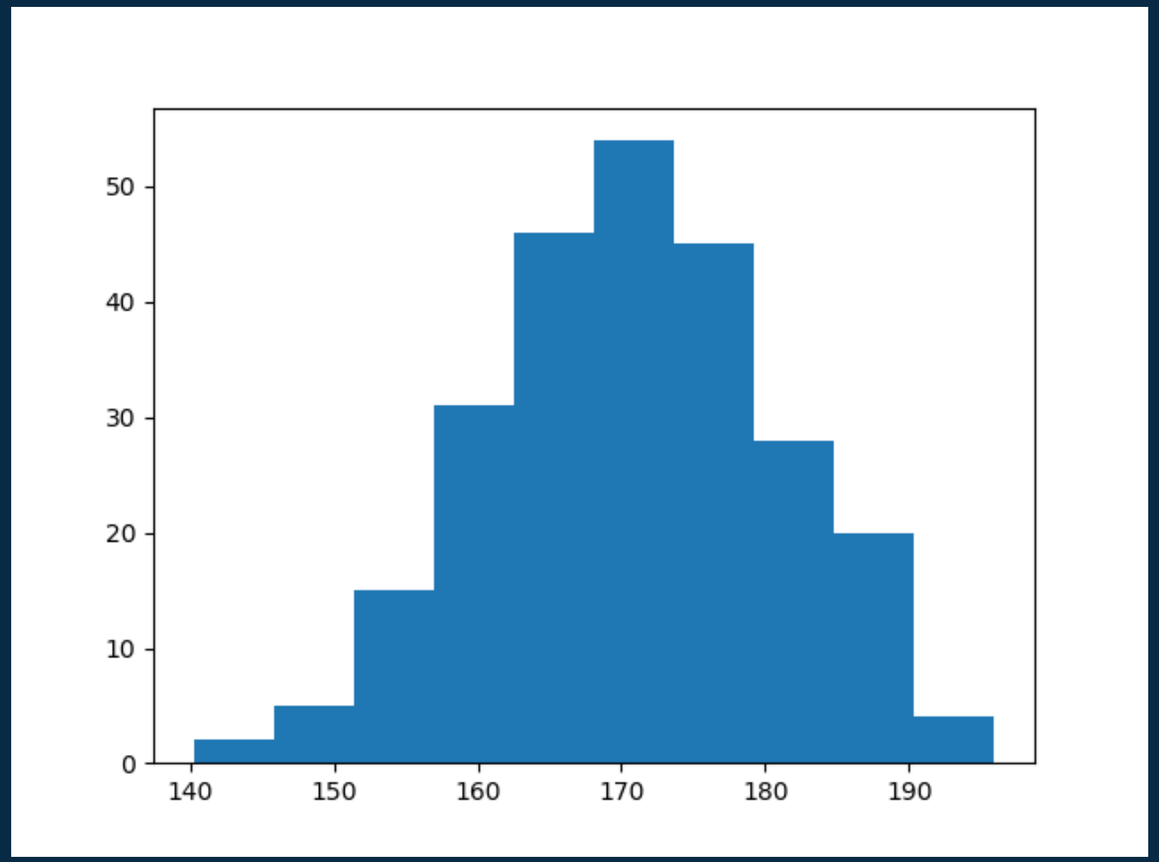

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Matplotlib Histograms

A HISTOGRAM IS A GRAPH SHOWING FREQUENCY DISTRIBUTIONS.

IT IS A GRAPH SHOWING THE NUMBER OF OBSERVATIONS WITHIN EACH GIVEN INTERVAL

EXAMPLE: SAY YOU ASK FOR THE HEIGHT OF 250 PEOPLE, YOU MIGHT END UP WITH A HISTOGRAM LIKE THIS:




YOU CAN READ FROM THE HISTOGRAM THAT THERE ARE APPROXIMATELY:

2 PEOPLE FROM 140 TO 145CM
5 PEOPLE FROM 145 TO 150CM
15 PEOPLE FROM 151 TO 156CM
31 PEOPLE FROM 157 TO 162CM
46 PEOPLE FROM 163 TO 168CM
53 PEOPLE FROM 168 TO 173CM
45 PEOPLE FROM 173 TO 178CM
28 PEOPLE FROM 179 TO 184CM
21 PEOPLE FROM 185 TO 190CM
4 PEOPLE FROM 190 TO 195CM

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Create Histogram

IN MATPLOTLIB, WE USE THE HIST() FUNCTION TO CREATE HISTOGRAMS

THE HIST() FUNCTION WILL USE AN ARRAY OF NUMBERS TO CREATE A HISTOGRAM, THE ARRAY IS SENT INTO THE FUNCTION AS AN ARGUMENT.

FOR SIMPLICITY WE USE NUMPY TO RANDOMLY GENERATE AN ARRAY WITH 250 VALUES, WHERE THE VALUES WILL CONCENTRATE AROUND 170, AND THE STANDARD DEVIATION IS 10.

```
IMPORT NUMPY AS NP

X = NP.RANDOM.NORMAL(170, 10, 250)

PRINT(X)
```

THIS WILL GENERATE A RANDOM RESULT, AND COULD LOOK LIKE THIS:

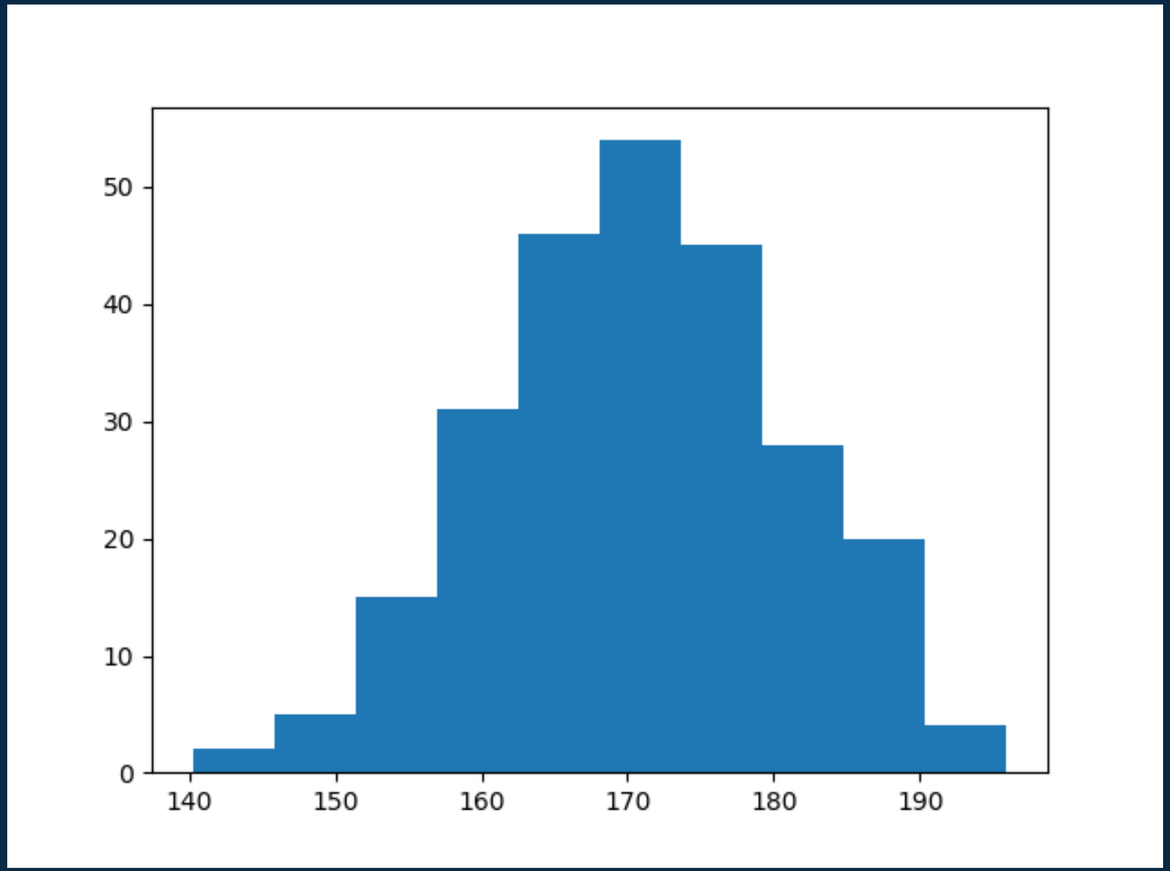
```
[167.62255766 175.32495609 152.84661337 165.50264047 163.17457988
 162.29867872 172.83638413 168.67303667 164.57361342 180.81120541
 170.57782187 167.53075749 176.15356275 176.95378312 158.4125473
 187.8842668  159.03730075 166.69284332 160.73882029 152.22378865
 164.01255164 163.95288674 176.58146832 173.19849526 169.40206527
 166.88861903 149.90348576 148.39039643 177.90349066 166.72462233
 177.44776004 170.93335636 173.26312881 174.76534435 162.28791953
 166.77301551 160.53785202 170.67972019 159.11594186 165.36992993
 178.38979253 171.52158489 173.32636678 159.63894401 151.95735707
 175.71274153 165.00458544 164.80607211 177.50988211 149.28106703
 179.43586267 181.98365273 170.98196794 179.1093176  176.91855744
 168.32092784 162.33939782 165.18364866 160.52300507 174.14316386
 163.01947601 172.01767945 173.33491959 169.75842718 198.04834503
 192.82490521 164.54557943 206.36247244 165.47748898 195.26377975
 164.37569092 156.15175531 162.15564208 179.34100362 167.22138242
 147.23667125 162.86940215 167.84986671 172.99302505 166.77279814
 196.6137667  159.79012341 166.5840824  170.68645637 165.62204521....
```

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

# Create Histogram

THE HIST() FUNCTION WILL READ THE ARRAY AND PRODUCE A HISTOGRAM

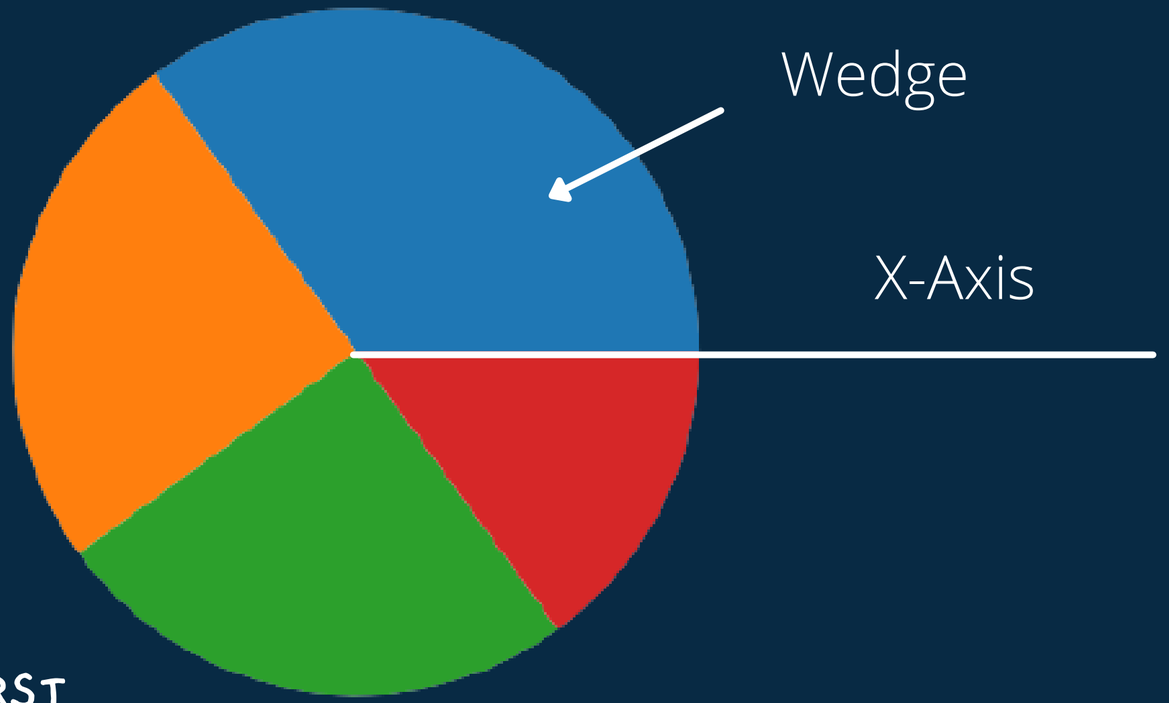
```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

X = NP.RANDOM.NORMAL(170, 10, 250)

PLT.HIST(X)
PLT.SHOW()
```



# Matplotlib Pie Charts

WITH PYPLOT, YOU CAN USE THE PIE() FUNCTION TO DRAW PIE CHARTS

```
IMPORT MATPLOTLIB.PYPLOT AS PLT
IMPORT NUMPY AS NP

Y = NP.ARRAY([35, 25, 25, 15])

PLT.PIE(Y)
PLT.SHOW()
```




BY DEFAULT THE PLOTTING OF THE FIRST WEDGE STARTS FROM THE X-AXIS AND MOVE COUNTERCLOCKWISE

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University

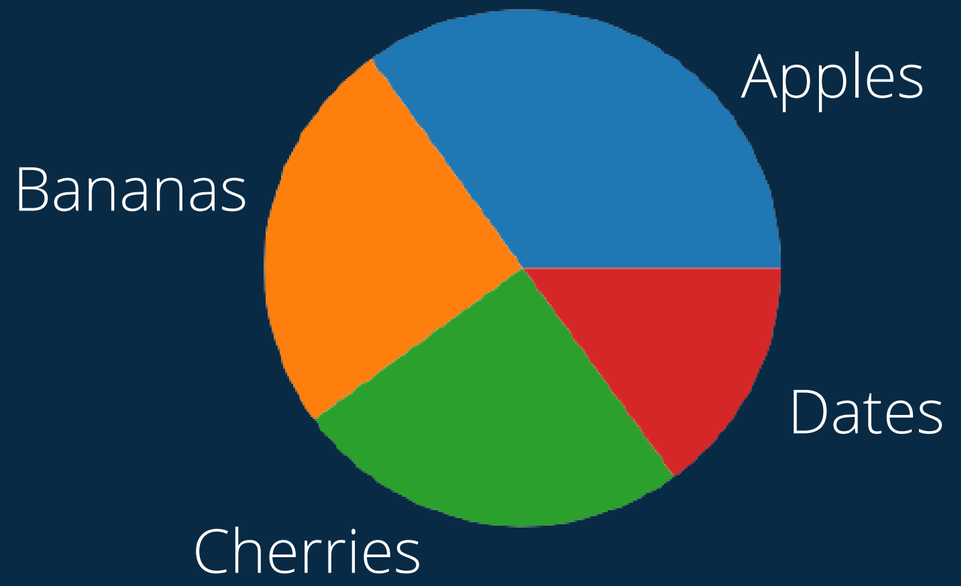
# Matplotlib Pie Charts

## Labels

Add labels to the pie chart with the label parameter.

The label parameter must be an array with one label for each wedge

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels)
plt.show()
```




~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University


# THANKYOU

gaurav.kumar@gla.ac.in

8586968801

~Dr Gaurav Kumar, Asst. Prof, CEA, GLA University