Machine Learning With Python

Section 1 : Introduction to Machine Learning
Section 2 : Implementing ML Algorithms in Python
Section 3 : Simple Linear Regression
Section 4 : Multiple Linear Regression
Section 5 : Classification Algorithms: K-Nearest Neighbors
Section 6 : Classification Algorithms: Decision Tree
Section 7 : Classification Algorithms: Logistic regression
Section 8 : Clustering
Section 9 : Recommender System
Section 10: Conclusion

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning? 01

02 Application of machine learning

Machine learning Methods 03

04 What is Supervised learning?

What is Unsupervised learning? 05

06 Supervised learning vs Unsupervised learning

Introduction to Machine Learning

What is Machine Learning?

- Machine learning is a form of artificial intelligence, where the machine are taught to do tasks without being explicitly programmed.

Characteristics of Machine Learning:

1. Learns relationship between data.
2. It can predict events for other data.
3. Improve performance with respect to experience.

Meta Brains

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning? 01

02 **Application of machine learning**

Machine learning Methods 03

04 What is Supervised learning?

What is Unsupervised learning? 05

06 Supervised learning vs Unsupervised learning

Introduction to Machine Learning

Applications

- Medical Diagnosis
- Self-Driving Cars
- Spam email Filtering
- Stock market Trading
- Automatic Language Translation
- Product Recommendations
- Traffic Prediction
- Image Recognition
- Speech Recognition

And many more

Meta Brains

Introduction to Machine Learning

Real World Applications

- Detect cancer and brain tumours
- Tesla self-driving car
- Gmail spam emails filtering
- Google translator
- Netflix, YouTube, Amazon, and other such companies

  Google maps, face detection to unlock a device, Siri, Cortana, Alexa all these technologies use machine learning in the background

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning? 01

02 Application of machine learning

Machine learning Methods **03**

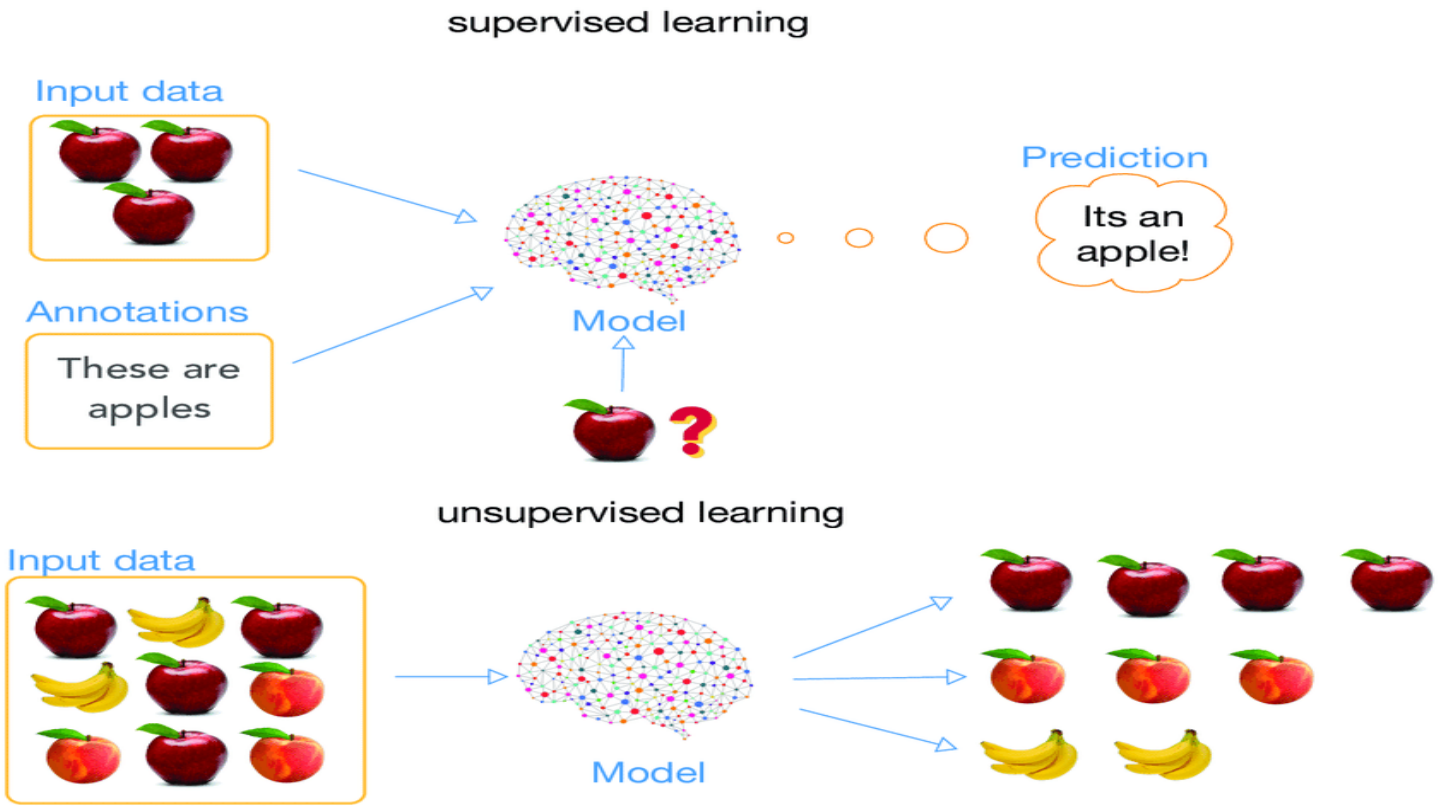
04 What is Supervised learning?

What is Unsupervised learning? 05

06 Supervised learning vs Unsupervised learning

Machine Learning

Machine learning Methods



Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning? 01

02 Application of machine learning

Machine learning Methods 03

04 What is Supervised learning?

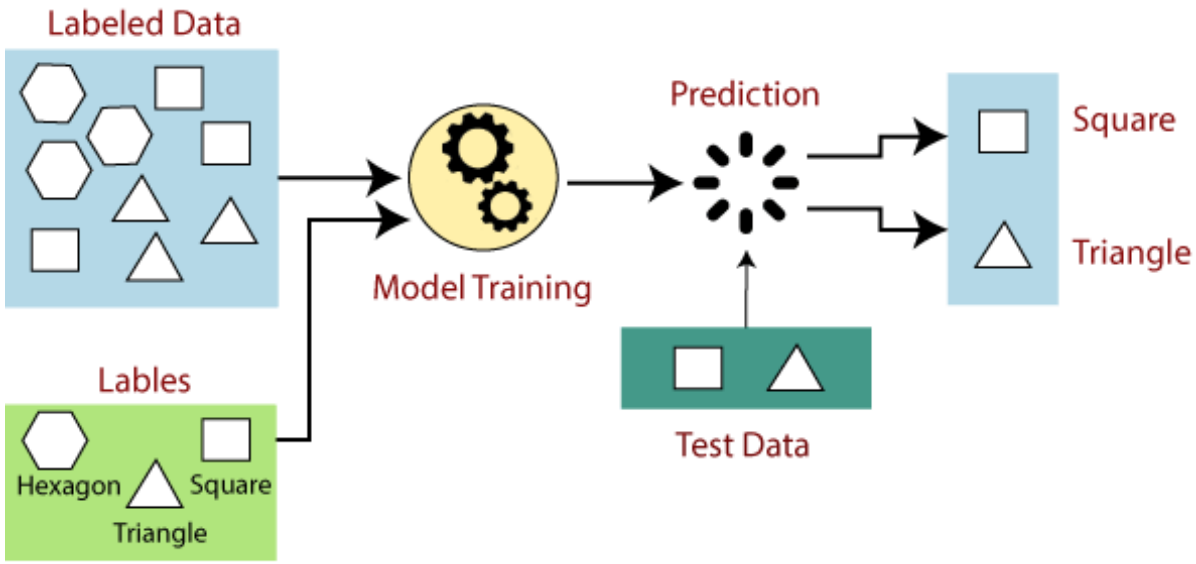
What is Unsupervised learning? 05

06 Supervised learning vs Unsupervised learning

Machine Learning

What is Supervised learning?

The term "supervise" refers to the act of watching and directing the completion of a work, project, or operation. It uses labelled training data and a collection of training sample to estimate a function.



Meta Brains

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning? 01

02 Application of machine learning

Machine learning Methods 03

04 What is Supervised learning?

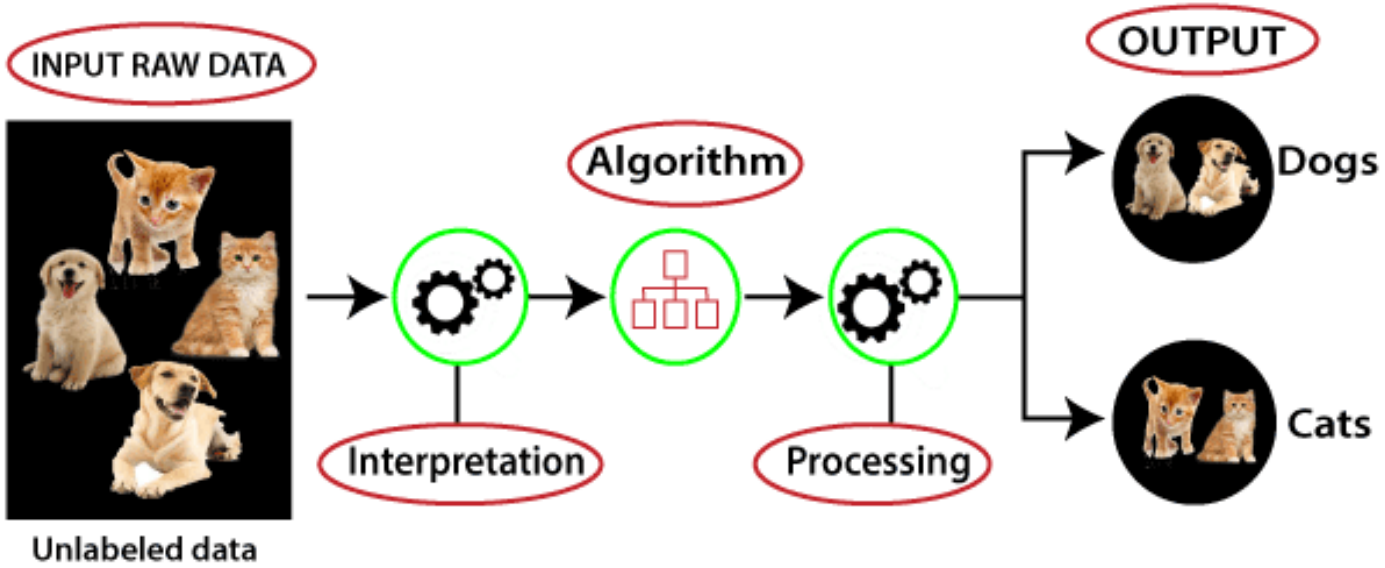
What is Unsupervised learning? 05

06 Supervised learning vs Unsupervised learning

Machine Learning

What is Unsupervised learning?

Unsupervised learning is a kind of machine learning in which the training data is supplied to the system without any pre-assigned labels or values.



Meta Brains

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

What is Machine Learning? 01

02 Application of machine learning

Machine learning Methods 03

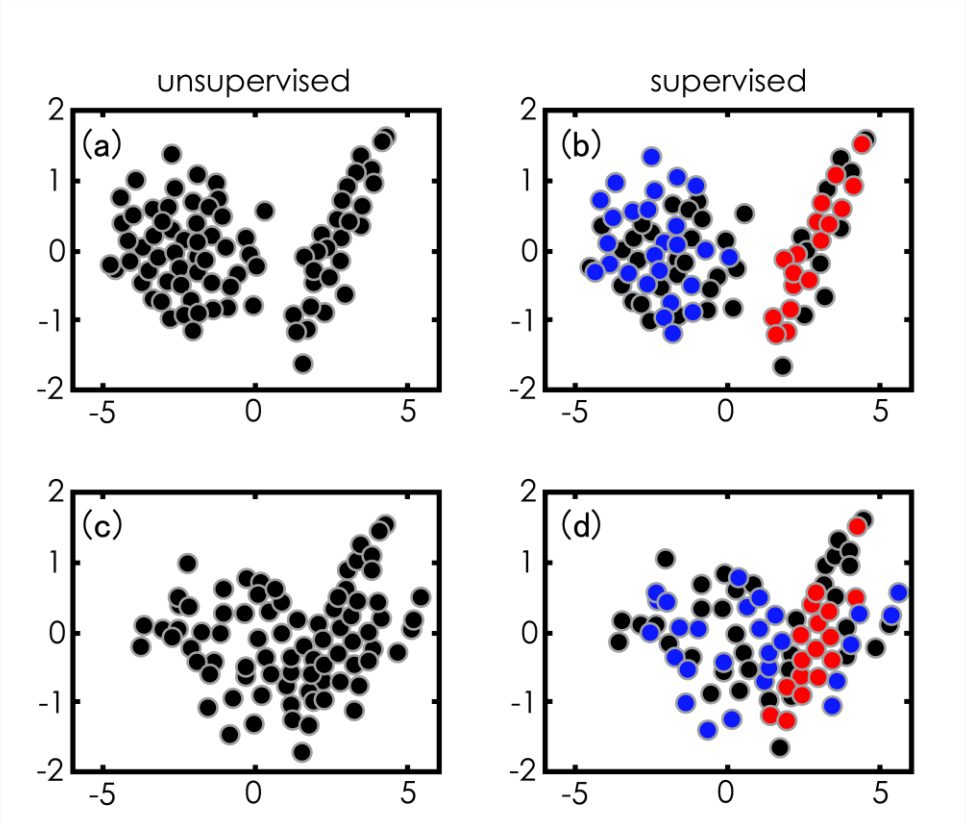
04 What is Supervised learning?

What is Unsupervised learning? 05

06 Supervised learning vs Unsupervised learning

Machine Learning

Supervised learning vs Unsupervised learning



Meta Brains

Quiz Time

Which of the following is TRUE about unsupervised machine learning?

1. *unsupervised learning comprises algorithms with no pre-existing outcomes.*
2. *learning algorithms with no control over quality of their predictions.*
3. *a semi-autonomous machine learning where researchers control some parts of the modelling process.*
4. *a fully autonomous machine learning with no human interference*

Meta Brains

Quiz Time

Which of the following is TRUE about unsupervised machine learning?

1. *unsupervised learning comprises algorithms with no pre-existing outcomes.*
2. *learning algorithms with no control over quality of their predictions.*
3. *a semi-autonomous machine learning where researchers control some parts of the modelling process.*
4. *a fully autonomous machine learning with no human interference*

Meta Brains

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction 01

02 Python libraries for Machine Learning

Setting up Python 03

04 What is Jupyter?

Anaconda Installation for Windows OS 05

06 Anaconda Installation for Mac OS

Anaconda Installation for Ubuntu OS 07

08 Implementing Python in Jupyter

Managing Directories in Jupyter Notebook 09

Python Language for Machine Learning

Python

Python is a widely used and efficient programming language that has lately become the language of choice for data scientists.

Libraries of Python

Following are the few libraries of Python.

Meta Brains

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 **Python libraries for Machine Learning**

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Python Language for Machine Learning

Libraries of Python

Numpy helps to deal with large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Scipy used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, special functions, signal and image processing, and other tasks common in science and engineering.

Matplotlib is a plotting library, and it is easy to use with Numpy because it's numeric is integrated with Numpy.

Meta Brains

Python Language for Machine Learning

Libraries of Python

Pandas is designed for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Scikit-learn (also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms. And it is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Meta Brains

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Python libraries for Machine Learning

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Introduction to Python

What is Python?

- Python is one of the most popular programming languages, created by **Guido van Rossum** and released in 1991.
- Python is an **interpreted** programming language as opposed to a compiled programming language.

Why python?

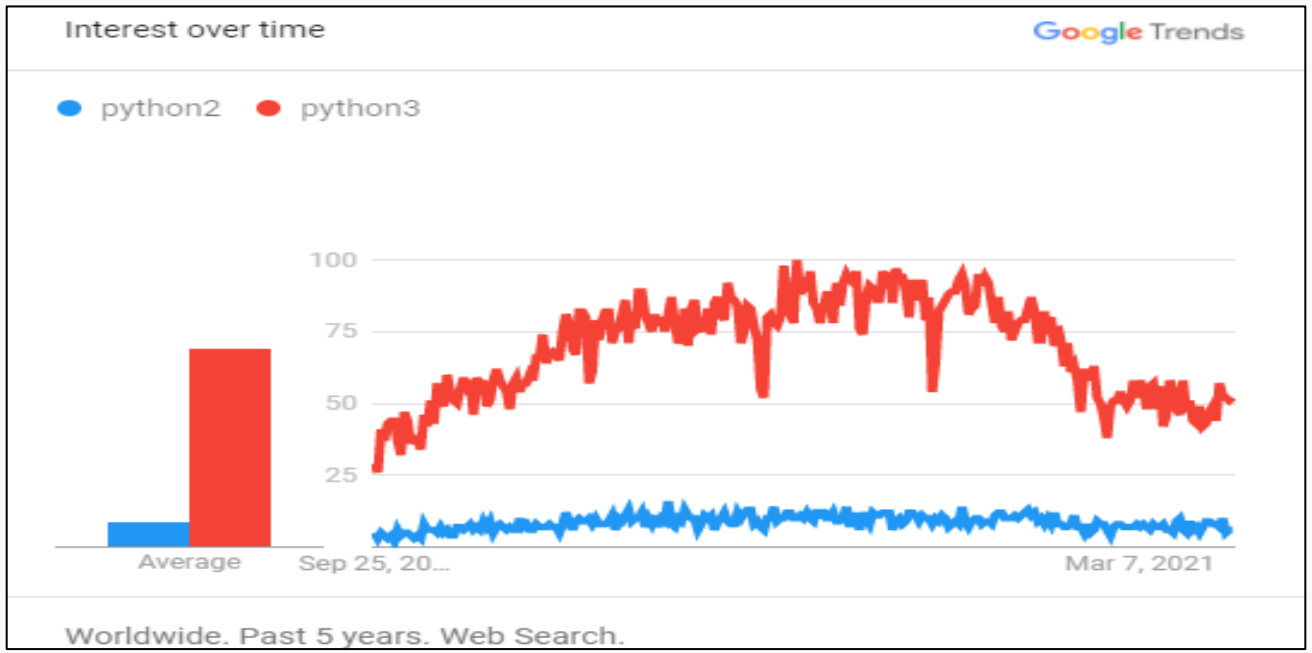
- Python is a beginner-friendly language due to its simplified syntax which is close to natural language.
- It has been designed to write clear, logical code for small and large-scale projects.
- It is used extensively in the industry for software development, data science, artificial intelligence, and machine learning.

Meta Brains

Setting Up Python

Python3 vs Python2

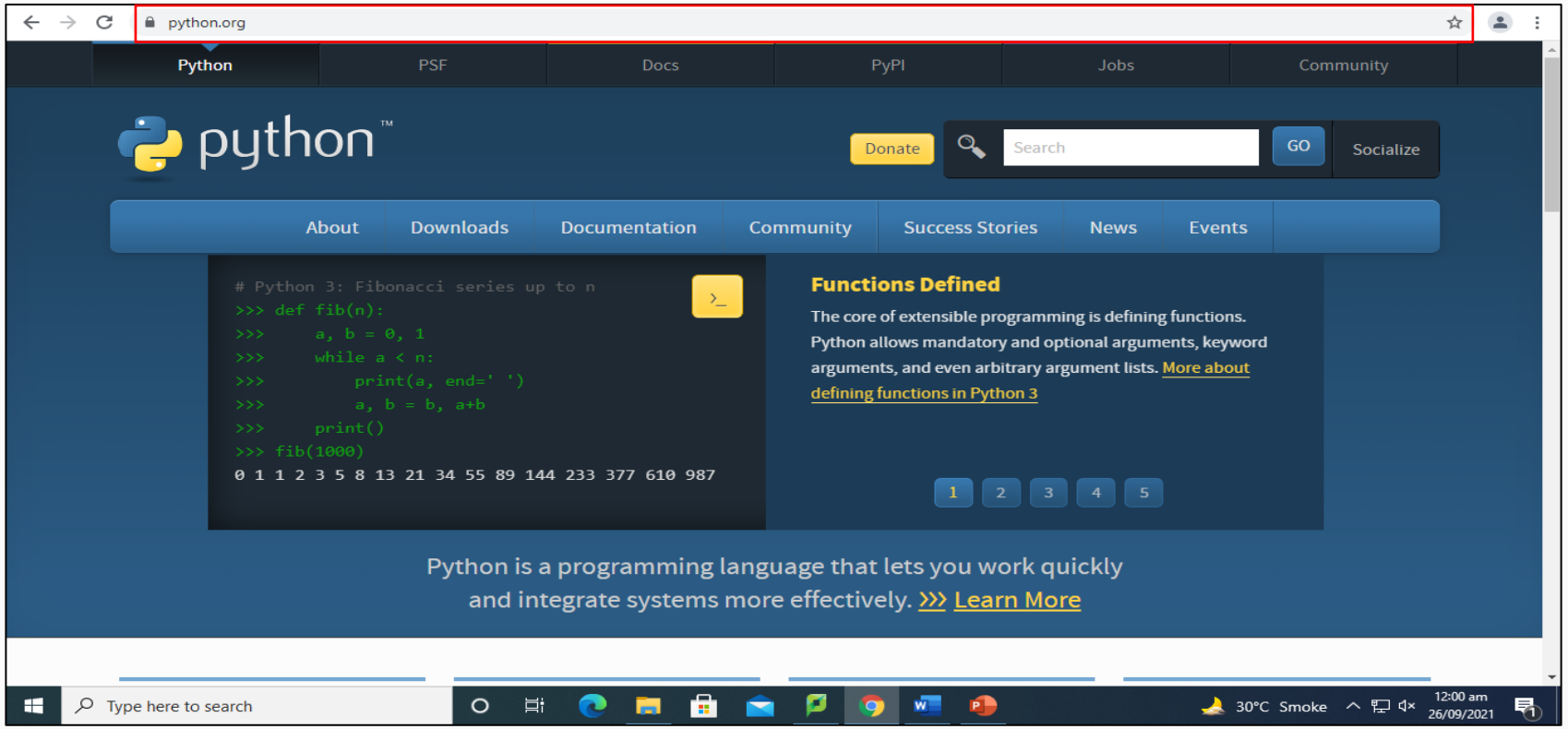
- Python2 is an outdated version of python. Python3 is the current version and is in-demand.
- In this course, we will be learning Python3 .



Meta Brains

Setting Up Python – Windows (1/8)

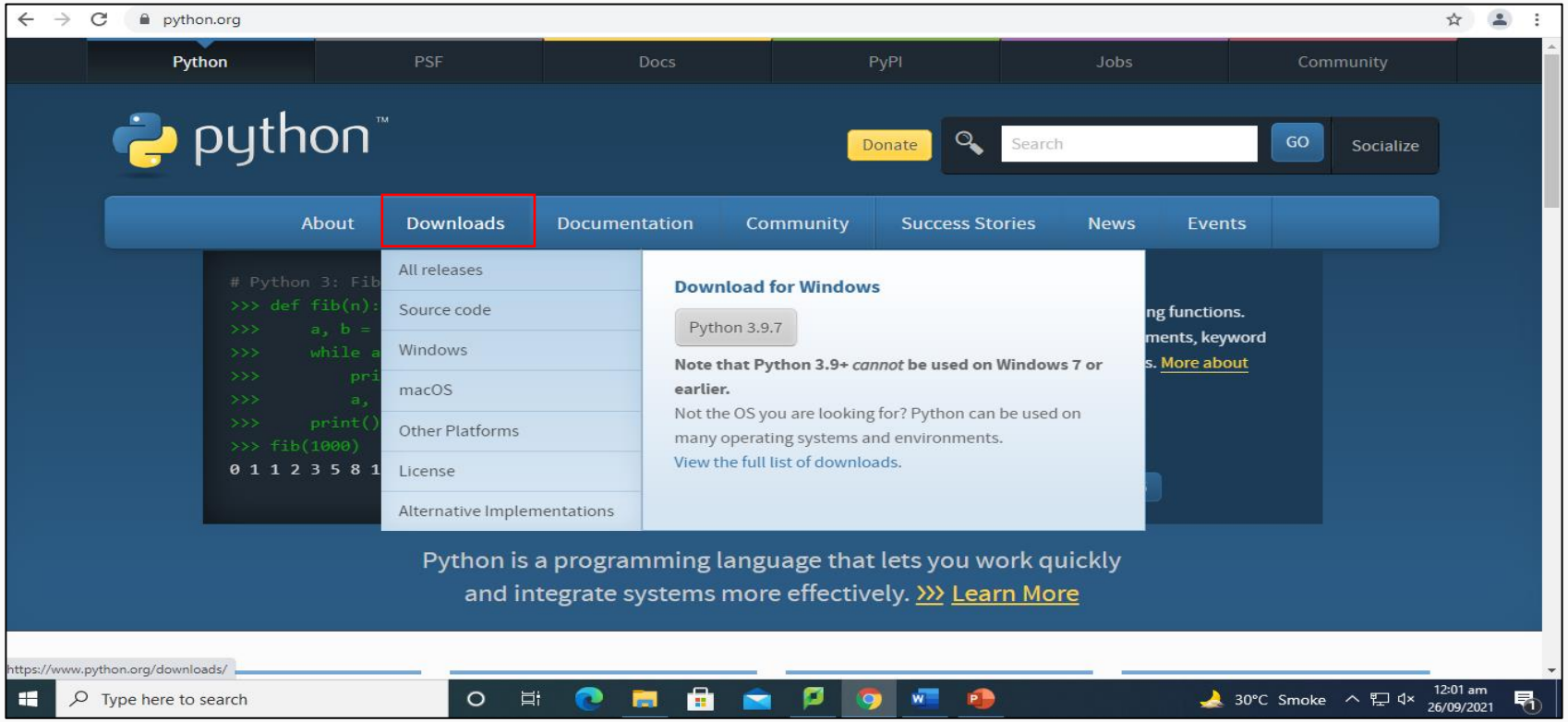
- Visit: https://www.python.org/



Meta Brains

Setting Up Python – Windows (2/8)

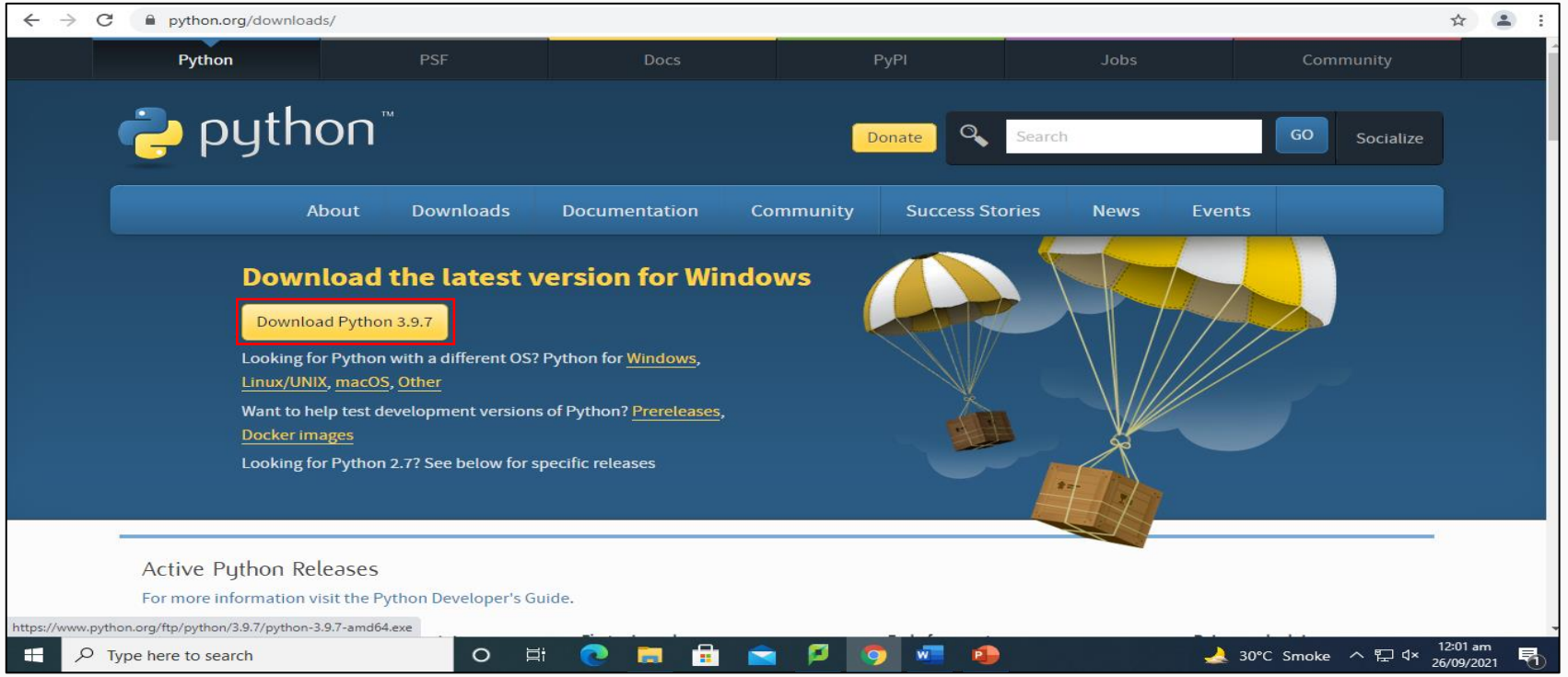
- Click on Downloads.



Meta Brains

Setting Up Python – Windows (3/8)

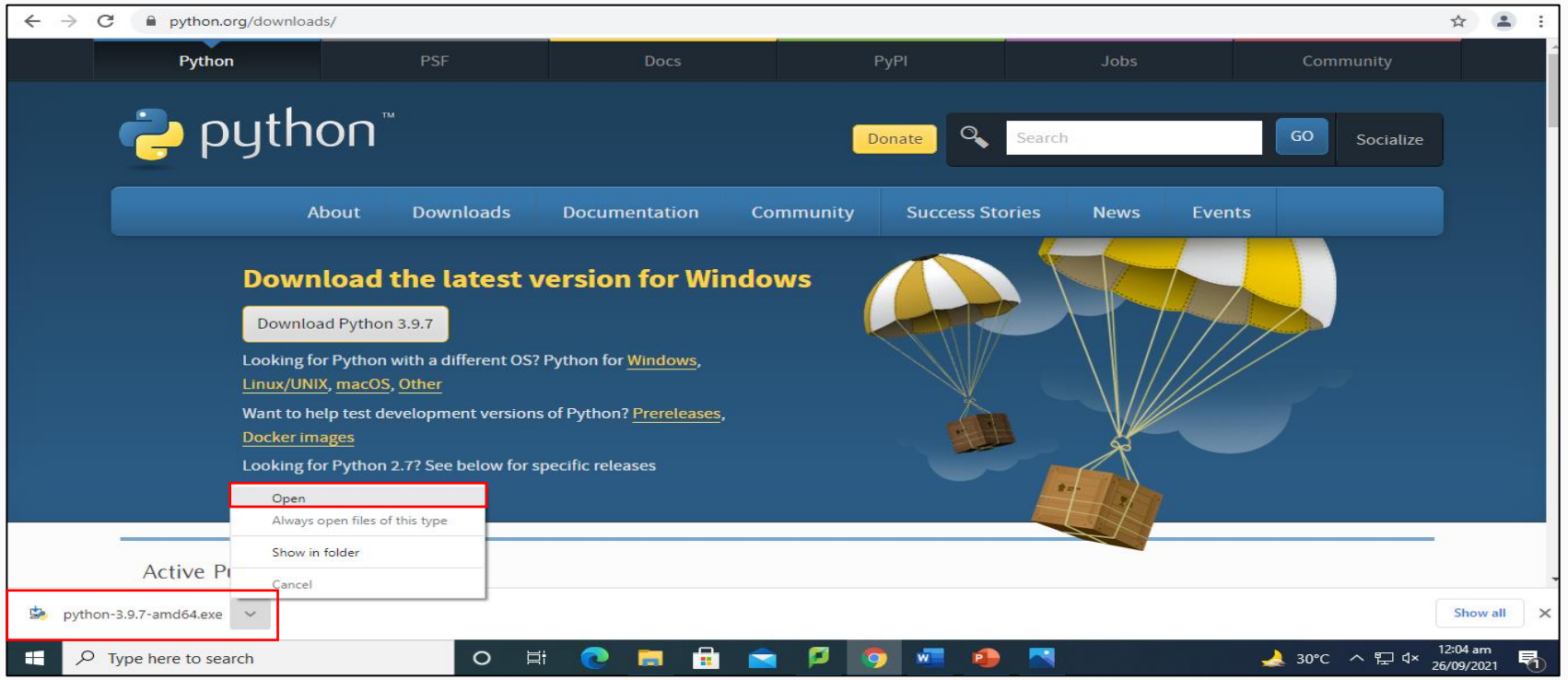
- This will show you the latest version of Python3 for your operating system. Download it.



Meta Brains

Setting Up Python – Windows (4/8)

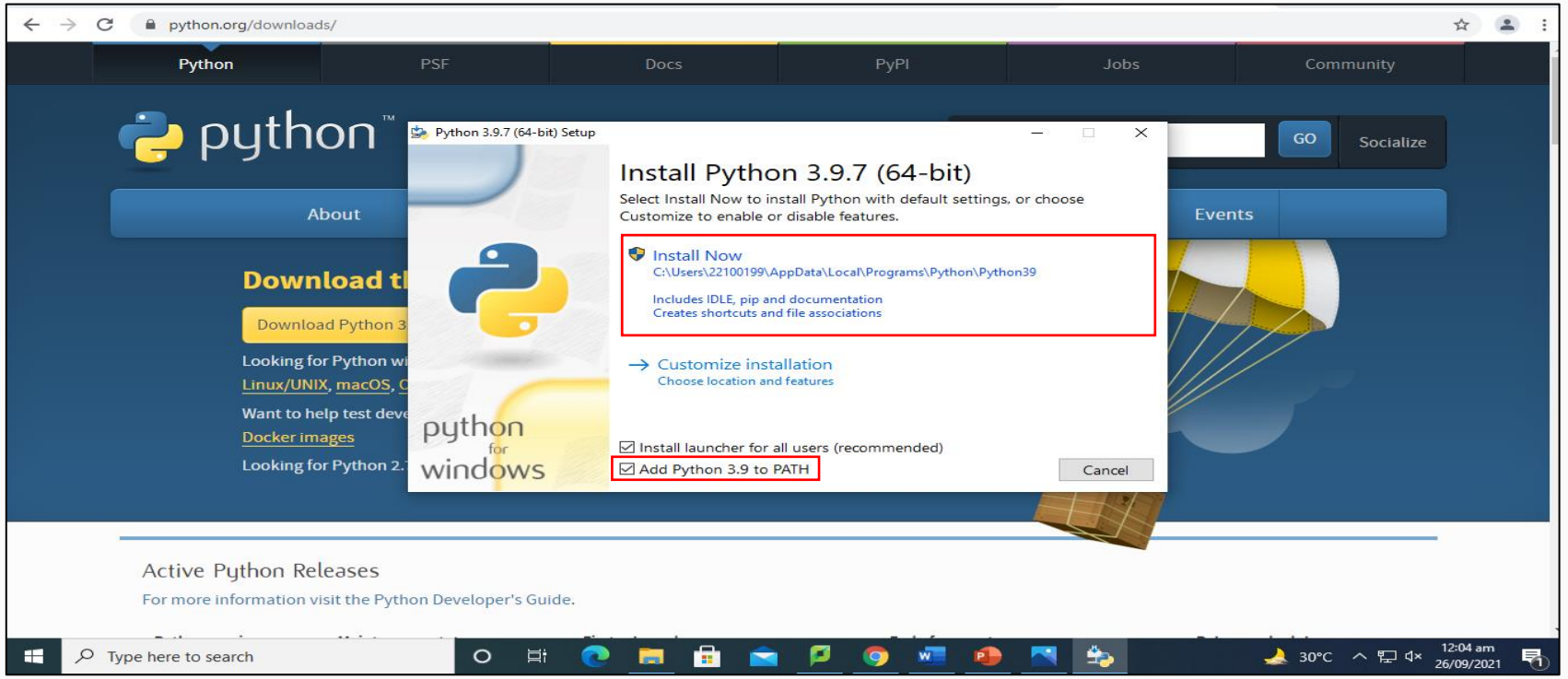
- Once downloaded, click Open.



Meta Brains

Setting Up Python – Windows (5/8)

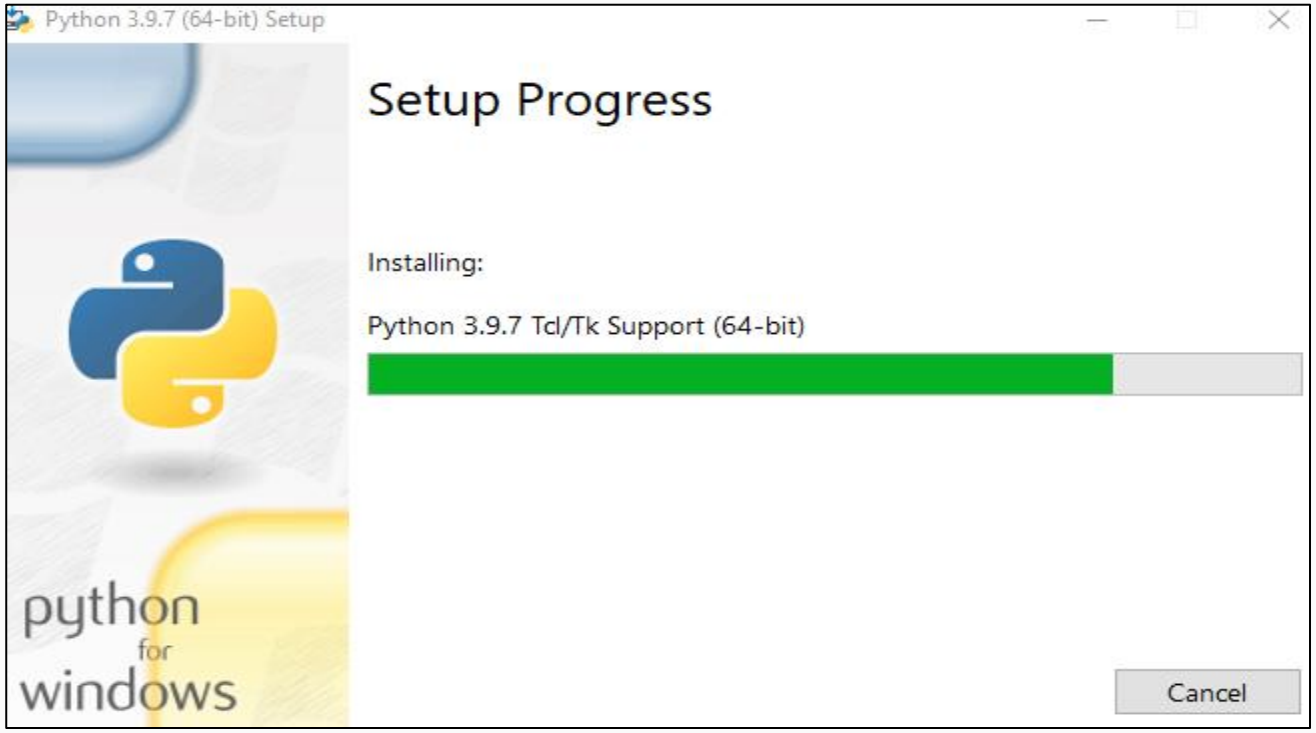
- Installation wizard will open up. Make sure that ‘Add Python 3.9 to Path’ is checked. Click on Install Now.



Meta Brains

Setting Up Python – Windows (6/8)

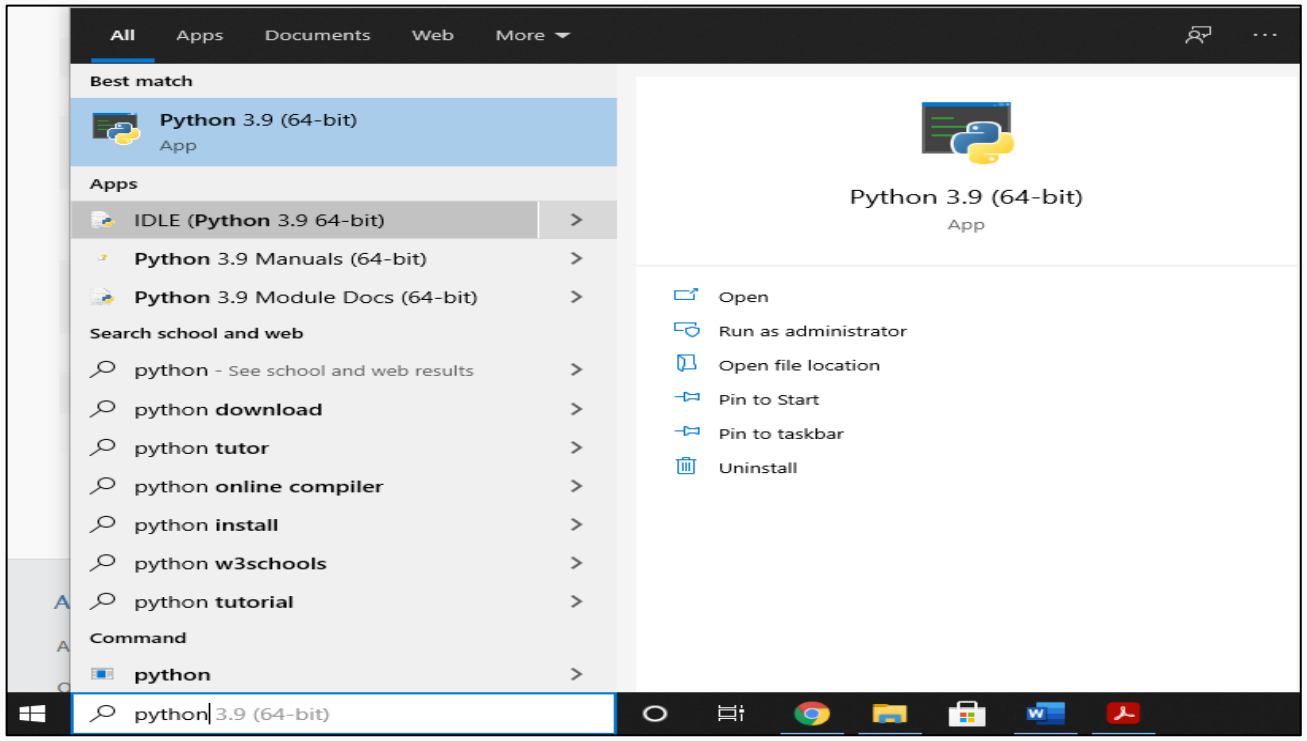
- It will take sometime to install.



Meta Brains

Setting Up Python – Windows (7/8)

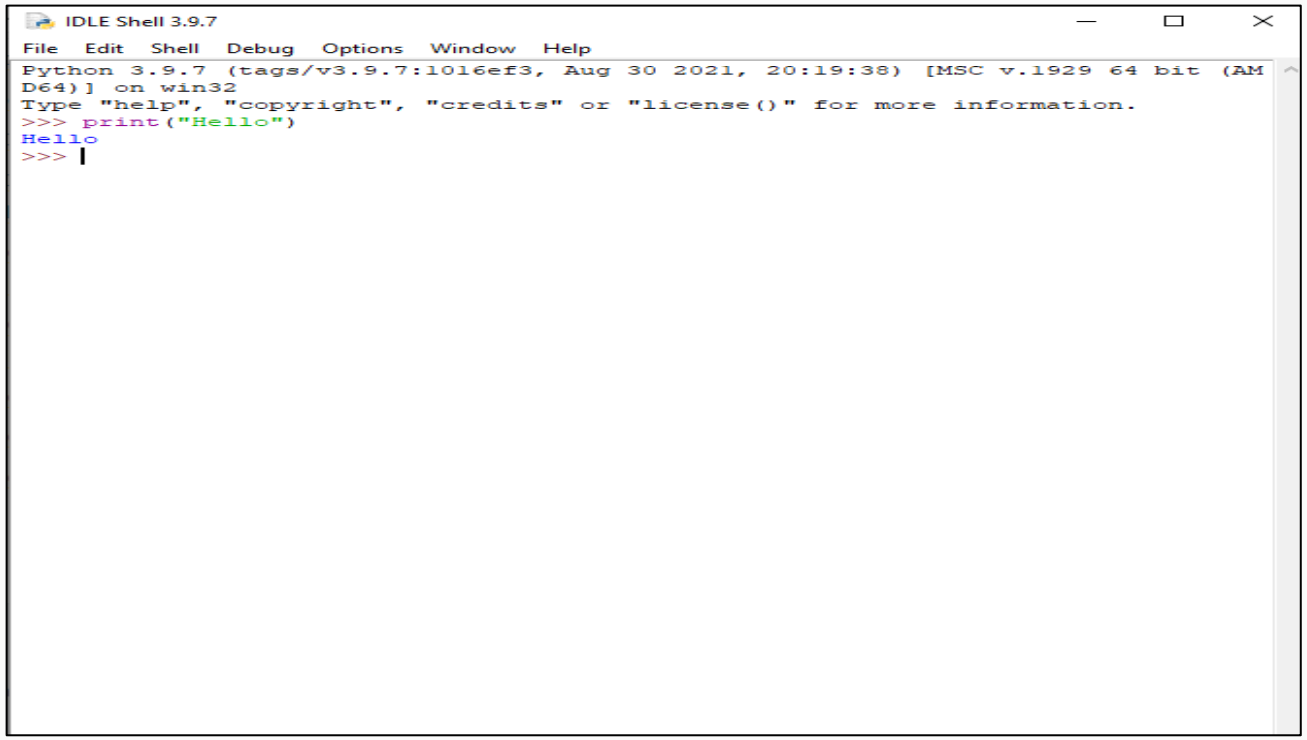
- Once installed, type python in your search bar and click on ‘IDLE (Python 3.9 64-bit)’.



Meta Brains

Setting Up Python – Windows (8/8)

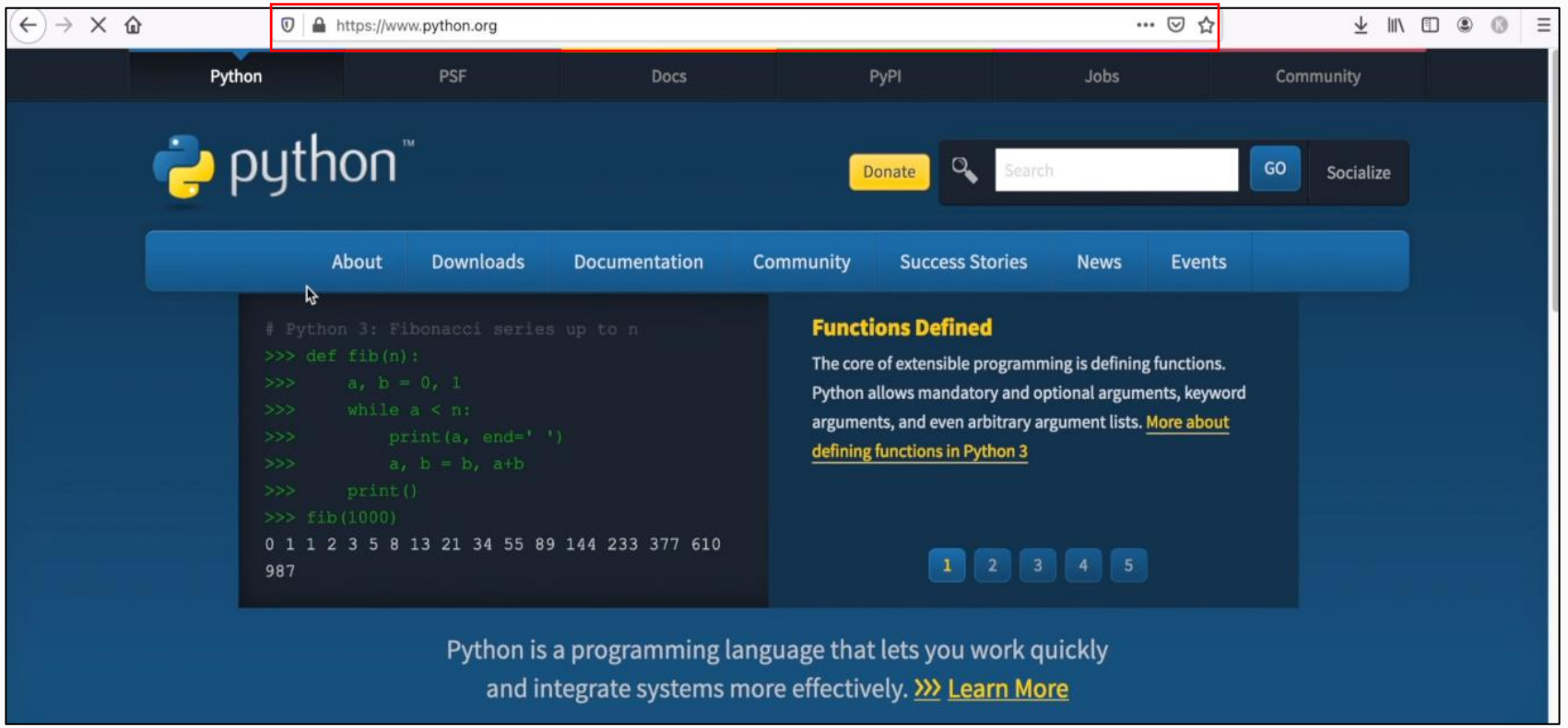
- This will open up Python IDLE. Type print(“Hello”) and press Enter to execute.



Meta Brains

Setting Up Python – Mac (1/13)

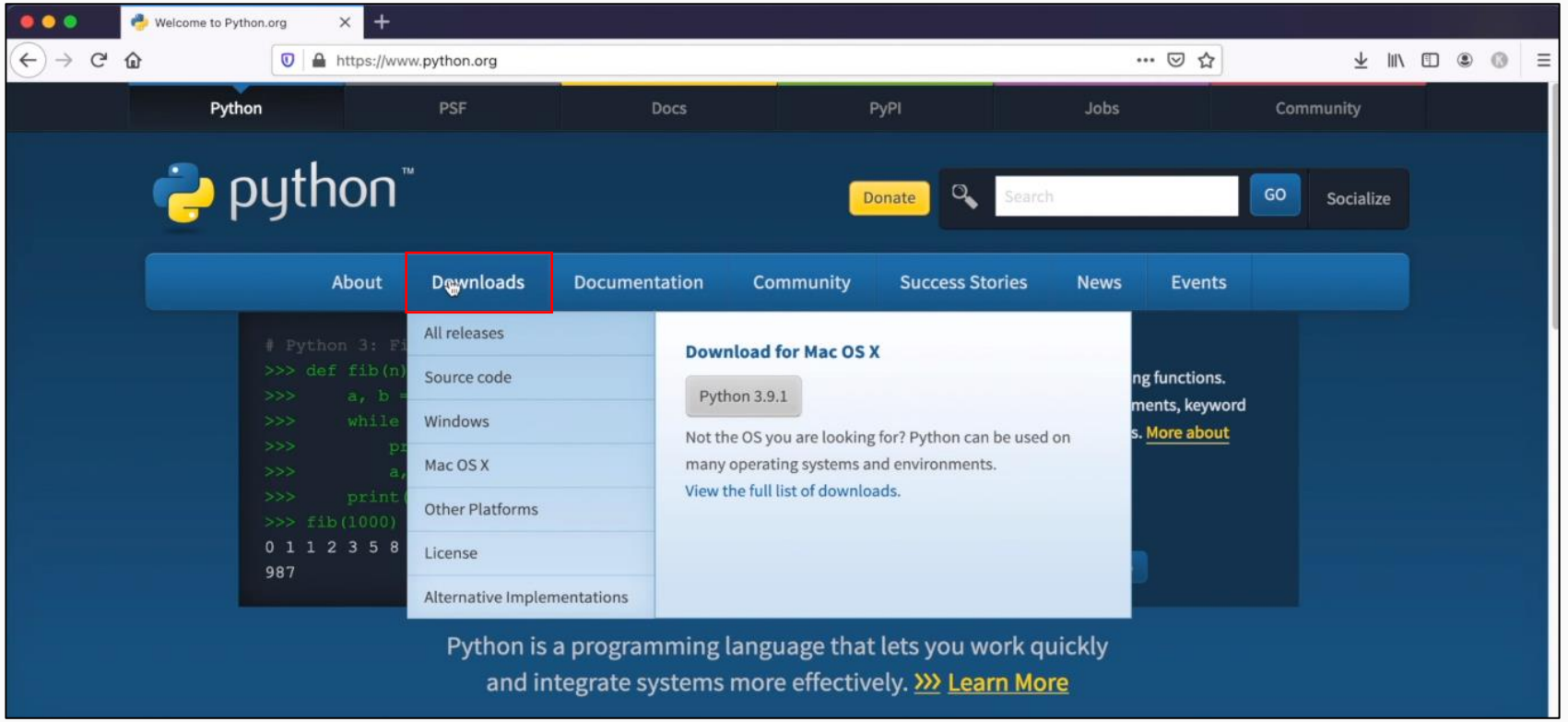
- Visit: https://www.python.org/



Meta Brains

Setting Up Python – Mac (2/13)

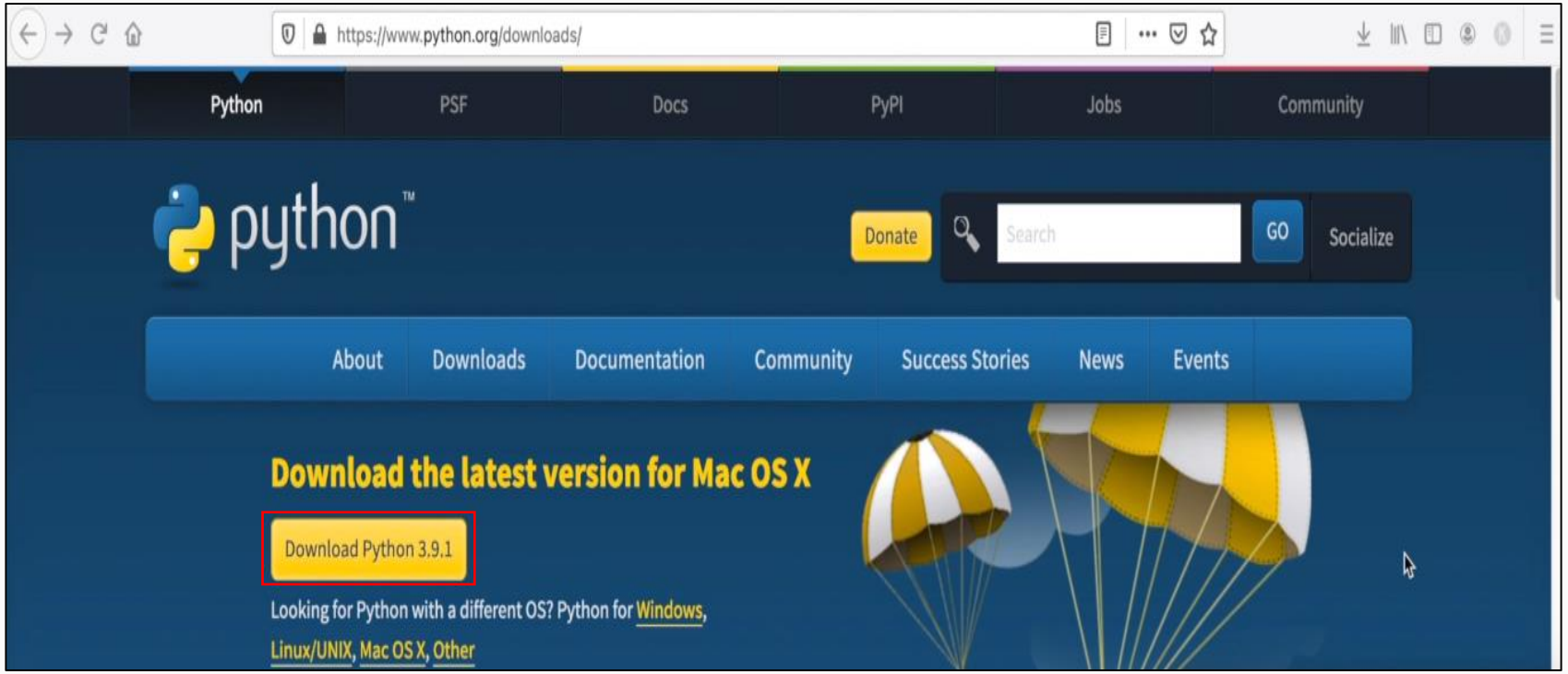
- Click on Downloads.



Meta Brains

Setting Up Python – Mac (3/13)

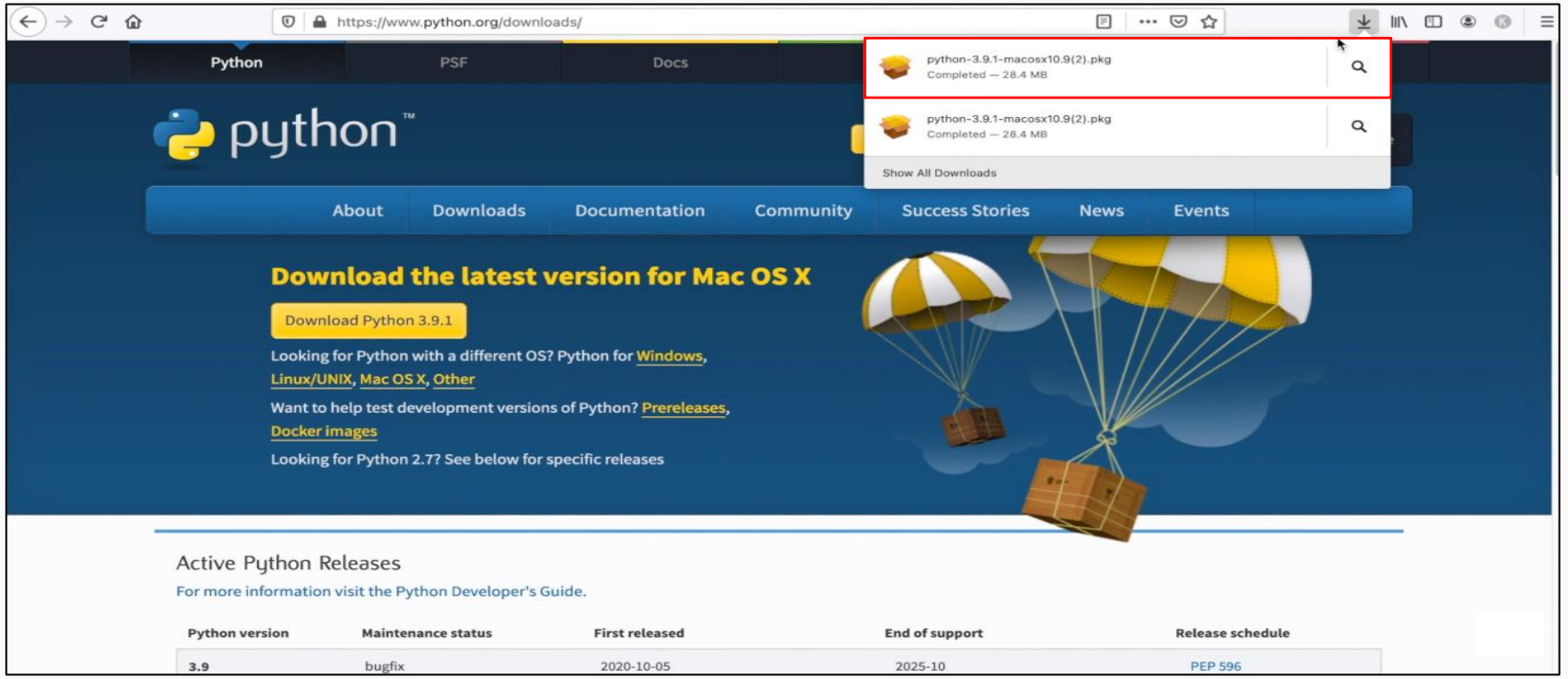
- This will show you the latest version of Python3 for your operating system. Download it.



Meta Brains

Setting Up Python – Mac (4/13)

- Once downloaded, click on the file.



Meta Brains

Setting Up Python – Mac (5/13)

- Click continue.

Meta Brains

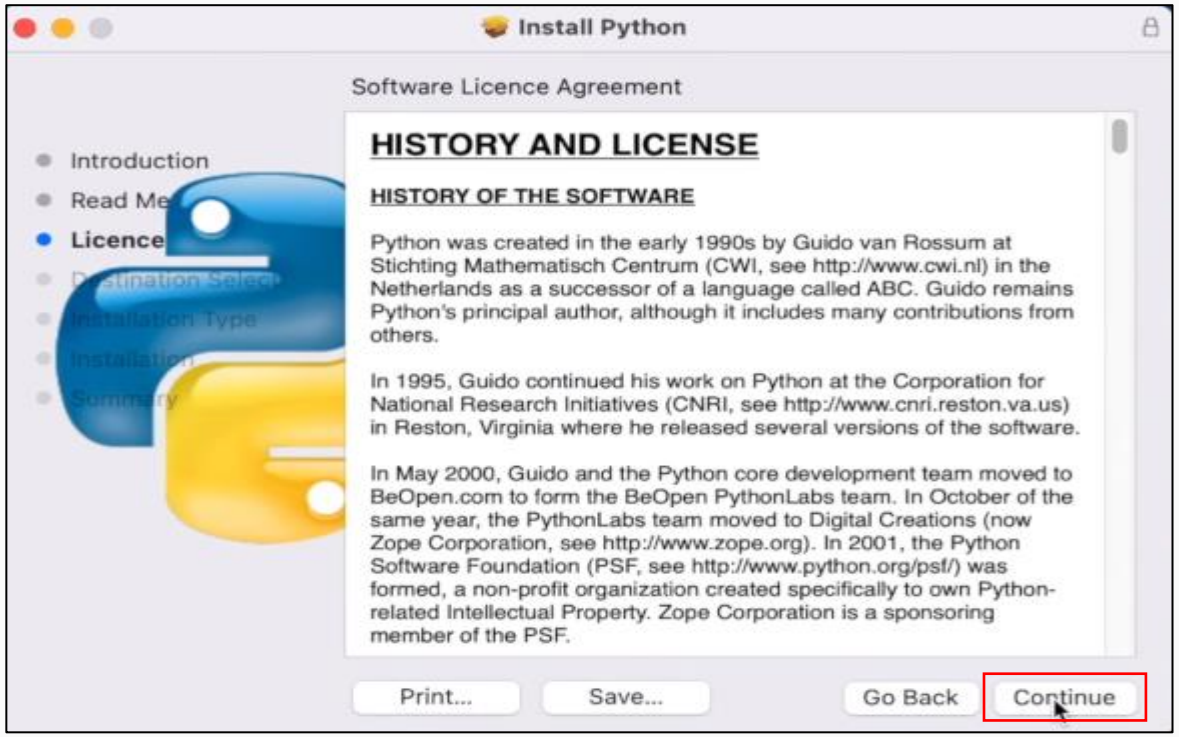
Setting Up Python – Mac (6/13)

- Click continue.

Meta Brains

Setting Up Python – Mac (7/13)

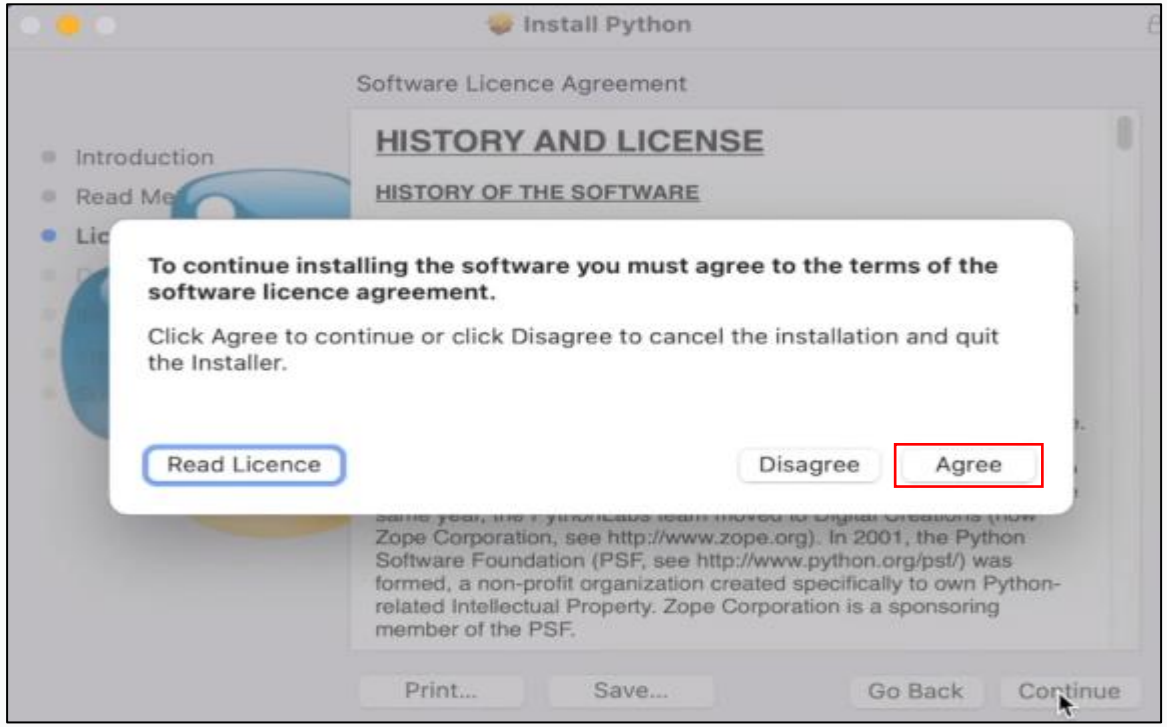
- Click continue.



Meta Brains

Setting Up Python – Mac (8/13)

- Read and accept the license agreement.



Meta Brains

Setting Up Python – Mac (9/13)

- Click continue.

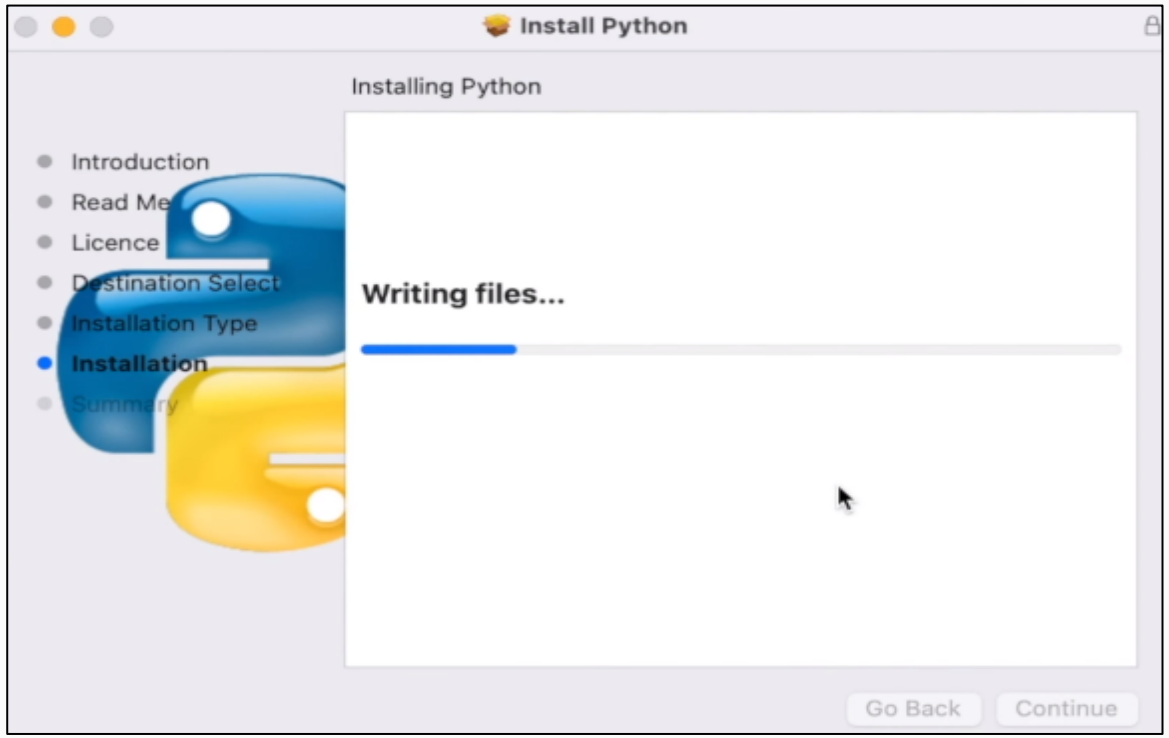
Setting Up Python – Mac (10/13)

- Click install. If prompted, enter your password.

Meta Brains

Setting Up Python – Mac (11/13)

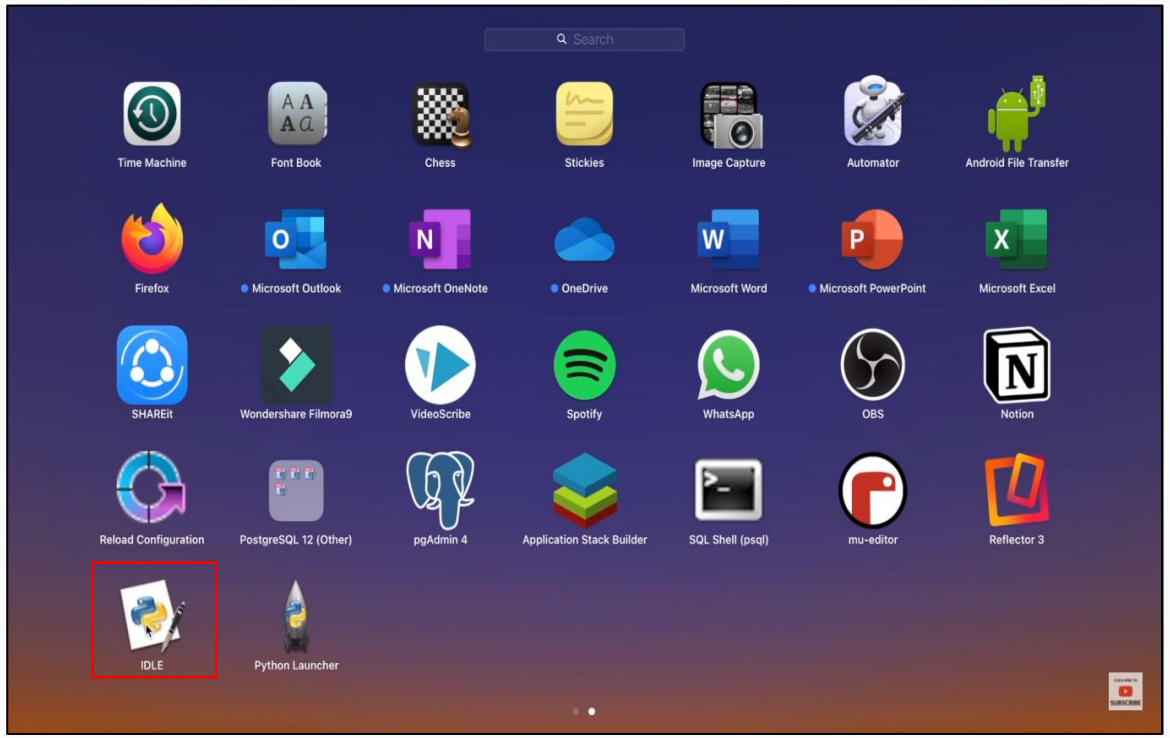
- Wait for the installation to finish.



Meta Brains

Setting Up Python – Mac (12/13)

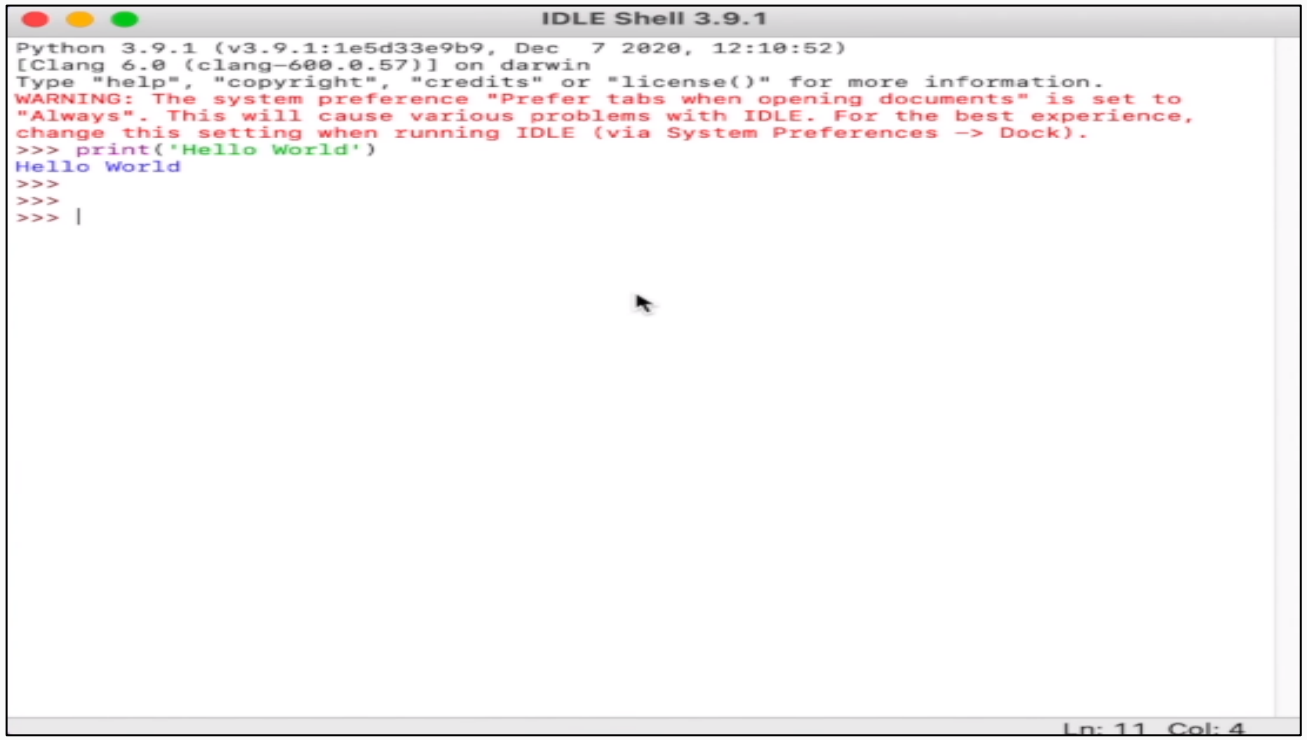
- Once installed, search for IDLE in your spotlight and click on it.



Meta Brains

Setting Up Python – Mac (13/13)

- This will open up Python IDLE. Type print(“Hello World”) and press Enter to execute.



Meta Brains

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Python libraries for Machine Learning

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Jupyter Notebook

- Jupyter Notebook is a web application that allows you to create documents that may contain code, visualizations, and narrative text.
- It is very popular and is used extensively in data science, artificial intelligence, and machine learning.
- A notebook has the .ipynb extension

Alternatives to Jupyter Notebook

- Google Colaboratory is an alternative to Jupyter Notebook and does not require downloading anything.

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Python libraries for Machine Learning

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

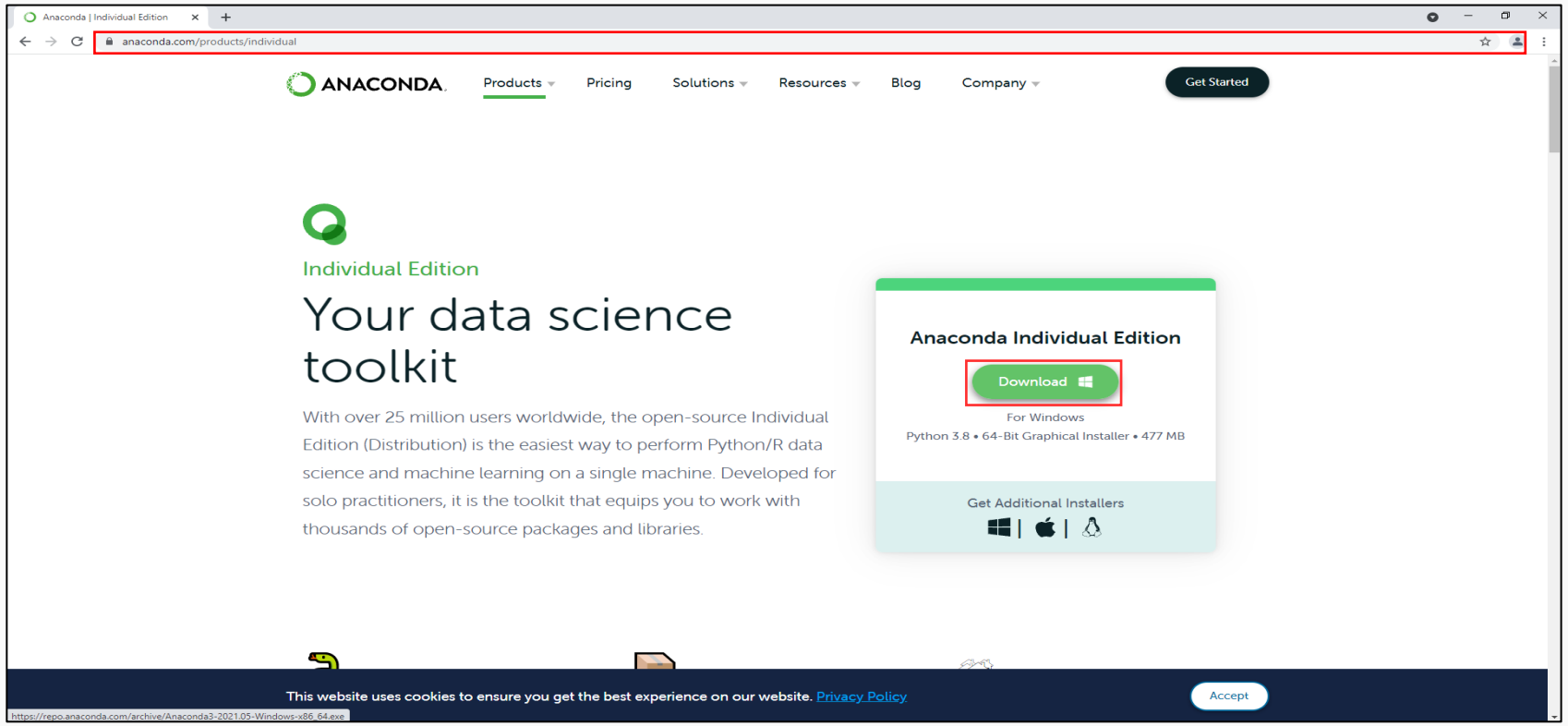
07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Anaconda for Windows (1/13)

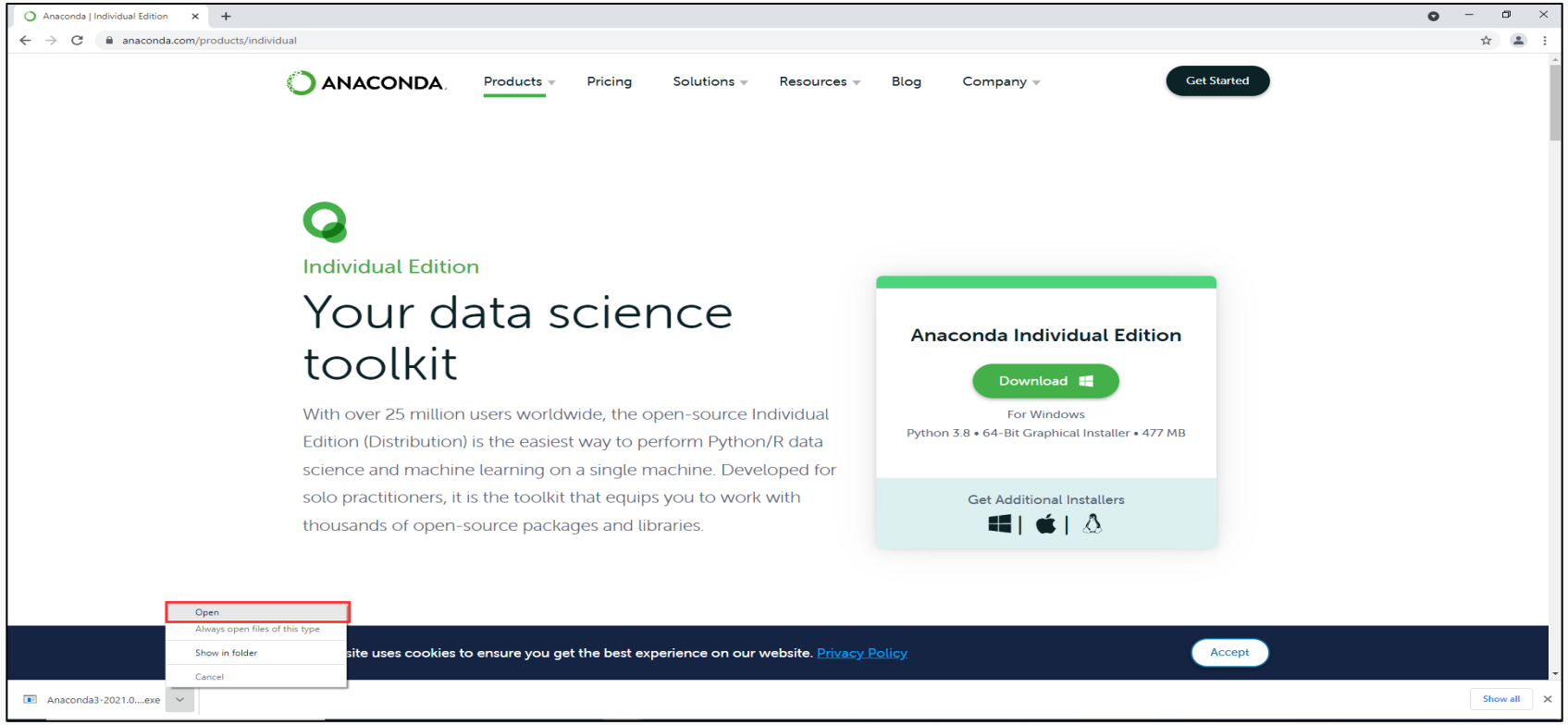
- Visit: https://www.anaconda.com/products/individual
- Click on the Download button.



Meta Brains

Anaconda for Windows (2/13)

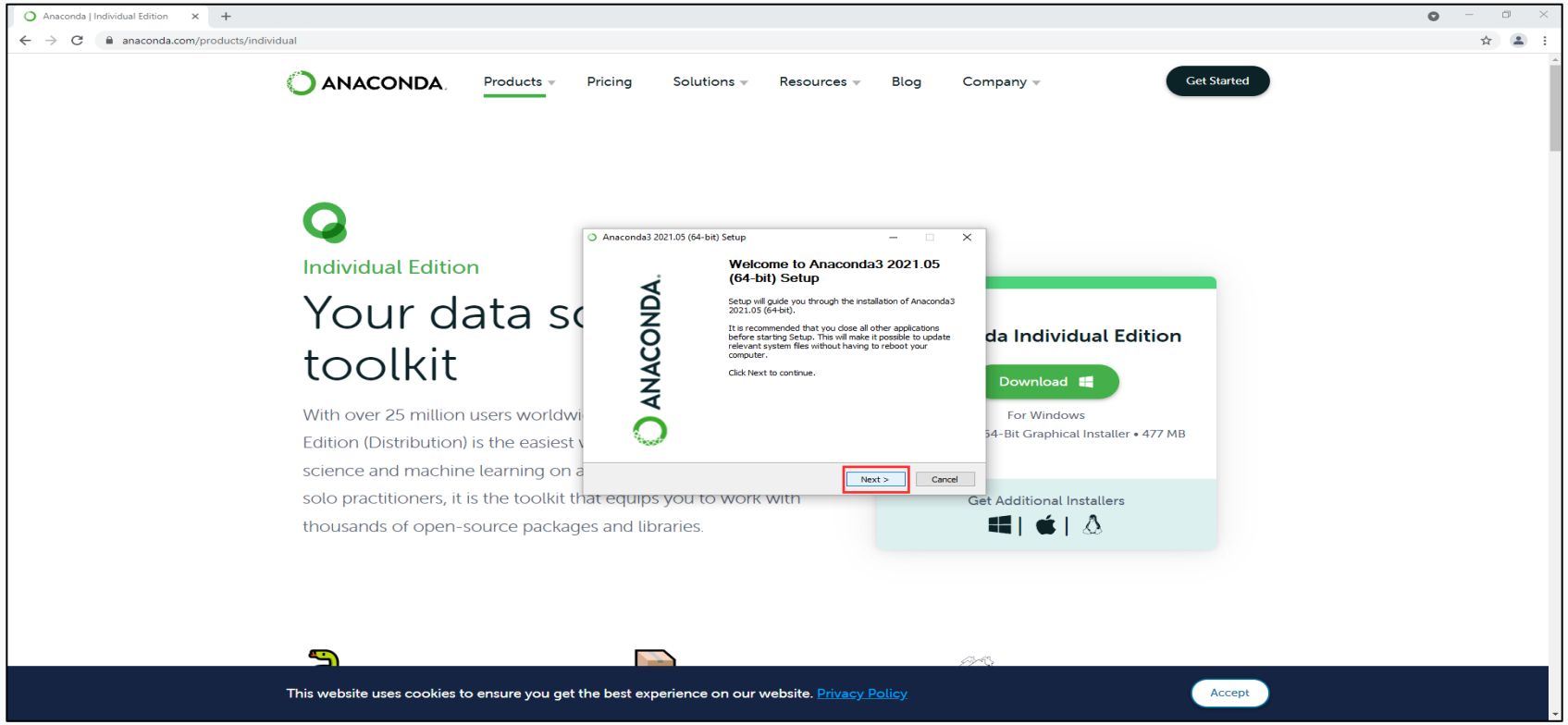
- Once Downloaded, click on Open.



Meta Brains

Anaconda for Windows (3/13)

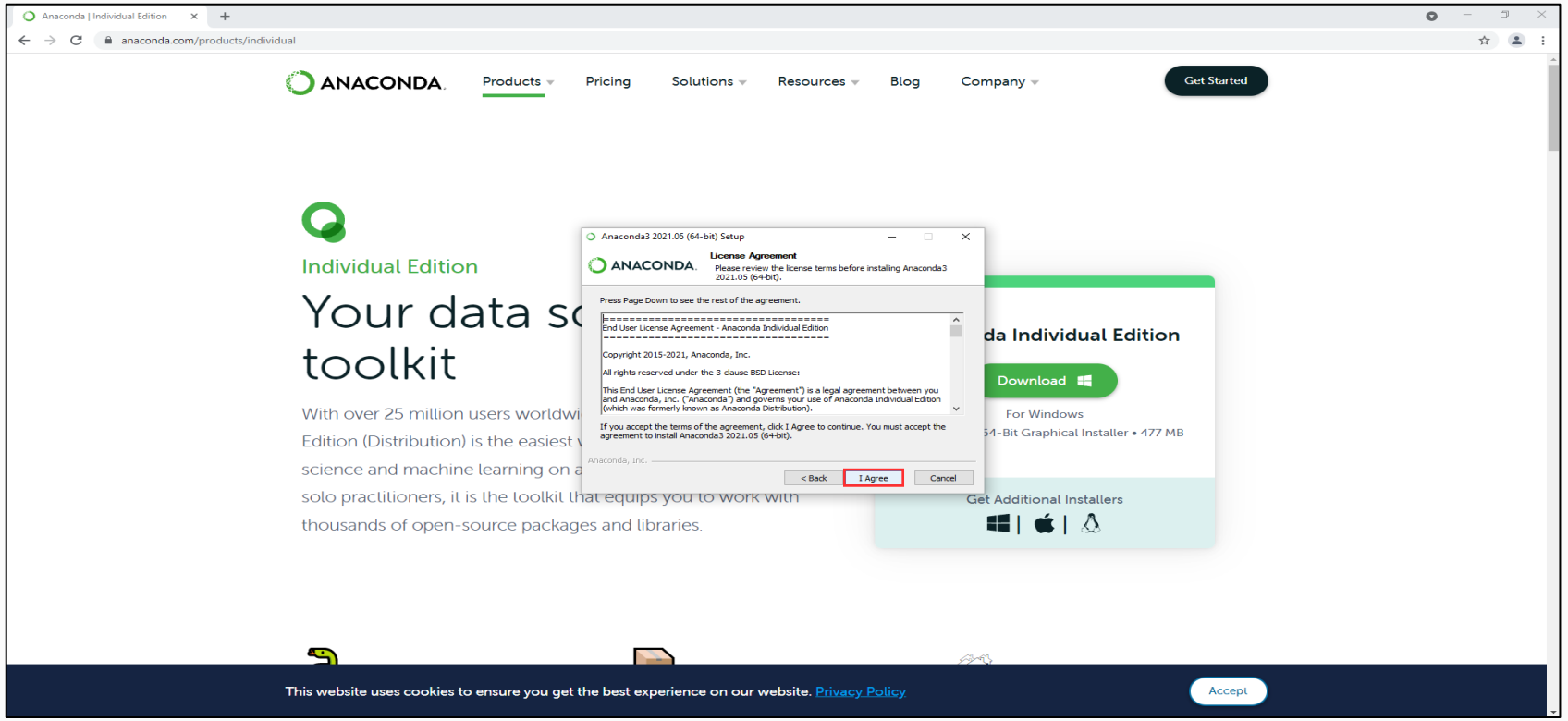
- An installation Wizard will open up.
- Click on Next.



Meta Brains

Anaconda for Windows (4/13)

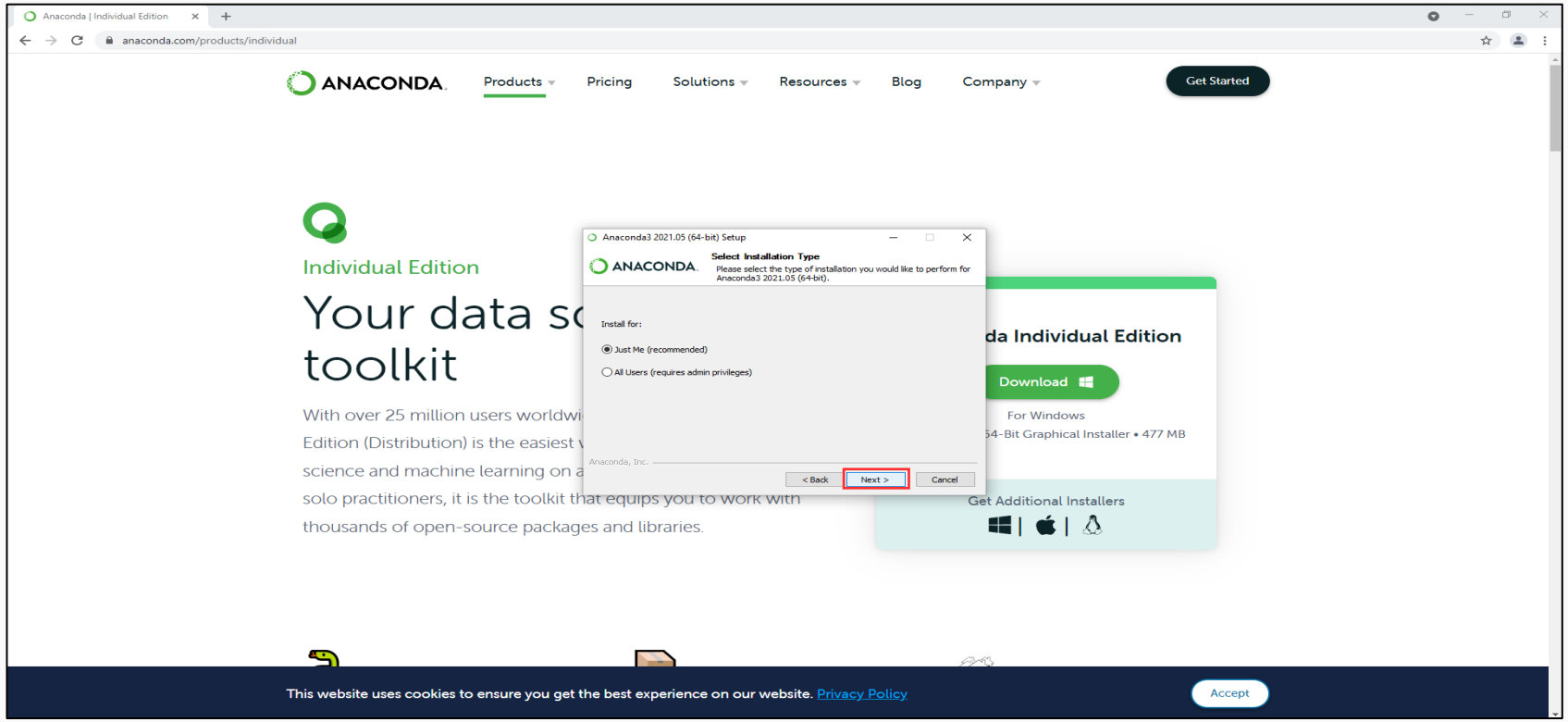
- Read the License Agreement and click I Agree.



Meta Brains

Anaconda for Windows (5/13)

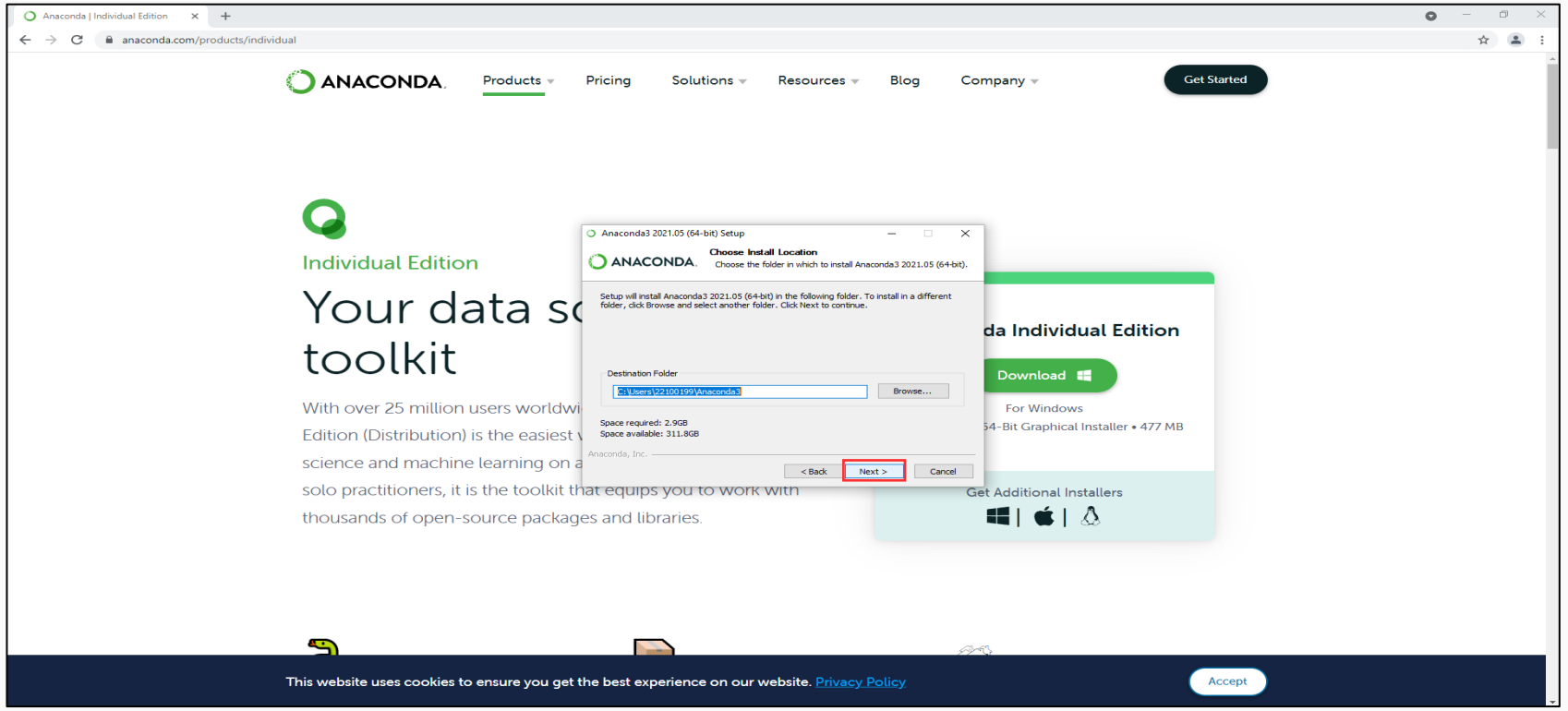
- Click Next.



Meta Brains

Anaconda for Windows (6/13)

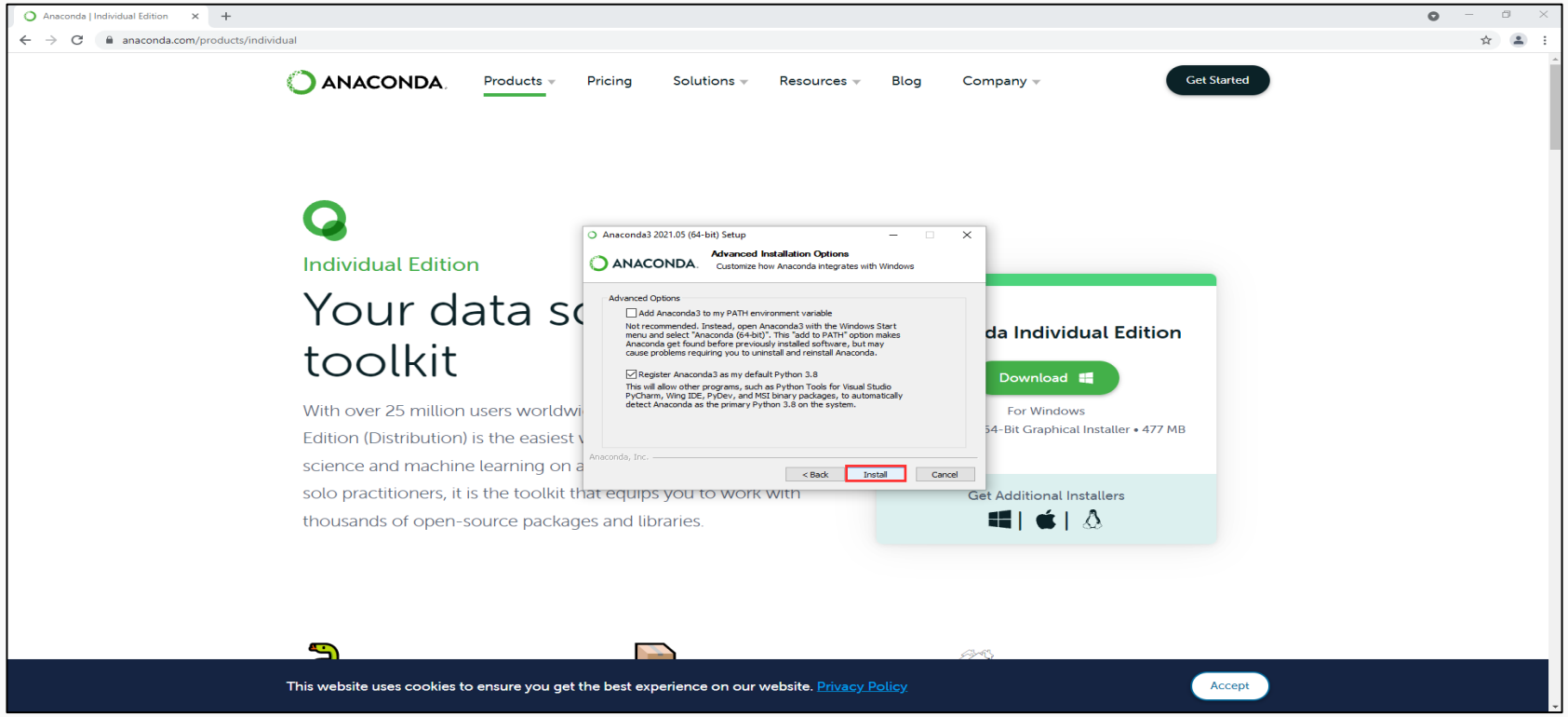
- Choose the destination and click Next.



Meta Brains

Anaconda for Windows (7/13)

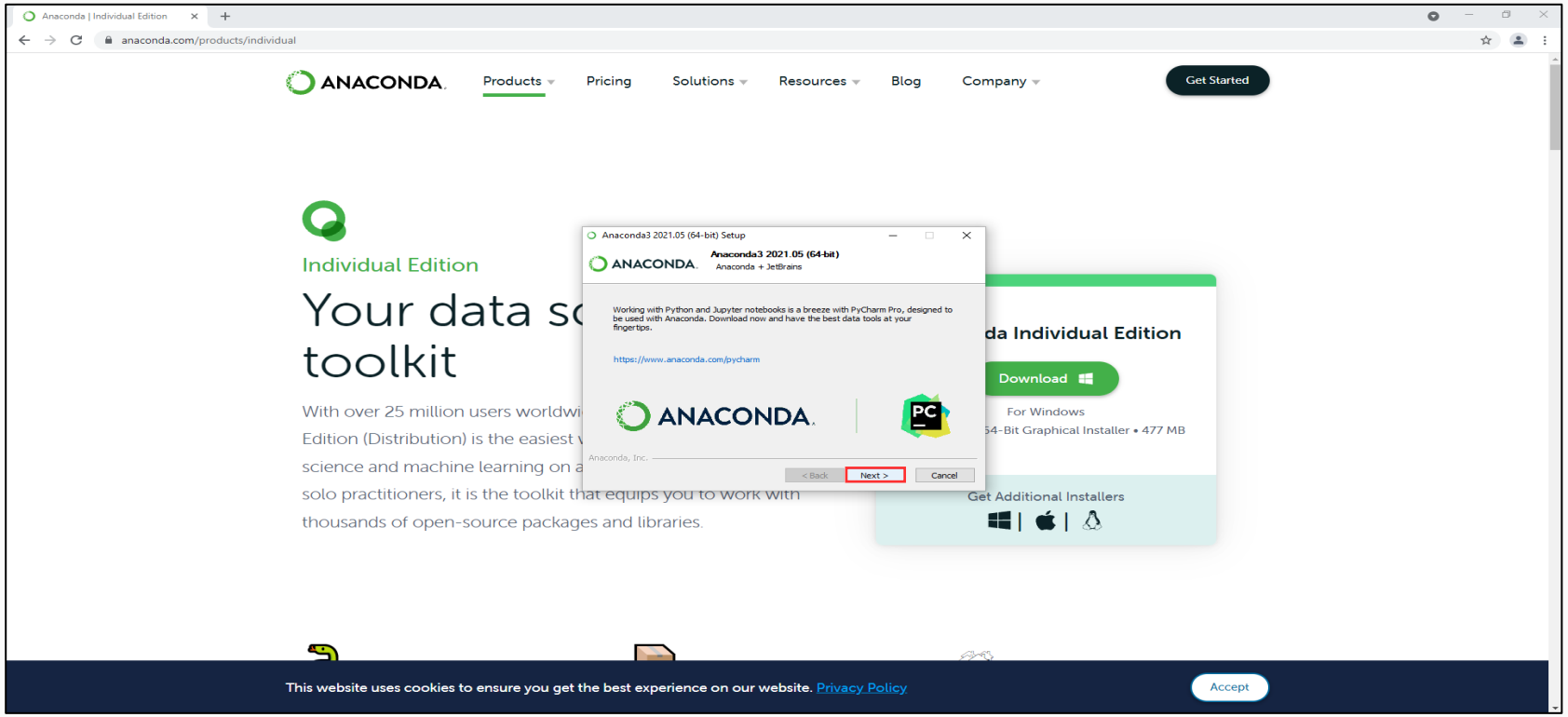
- Click Install.



Meta Brains

Anaconda for Windows (8/13)

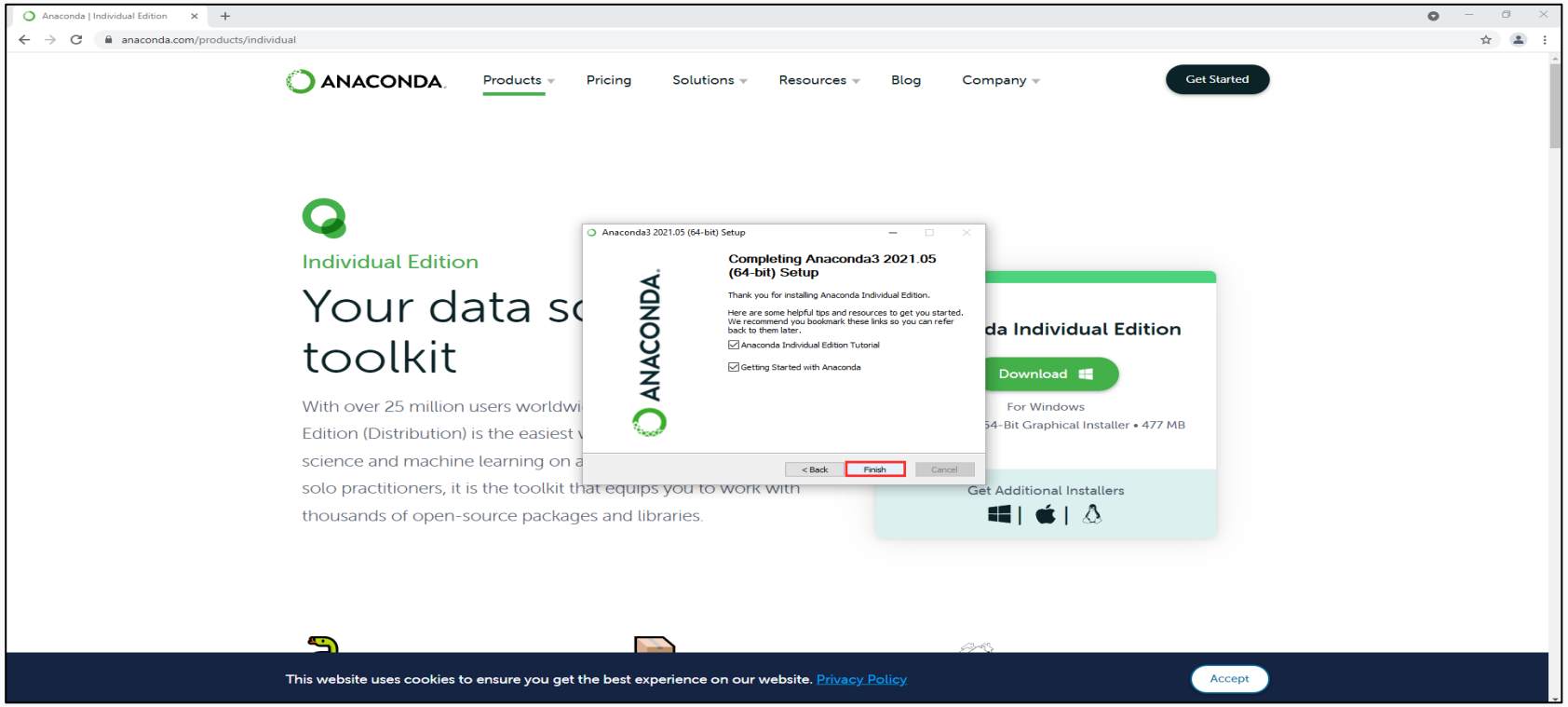
- Wait for the installation to finish and click Next.



Meta Brains

Anaconda for Windows (9/13)

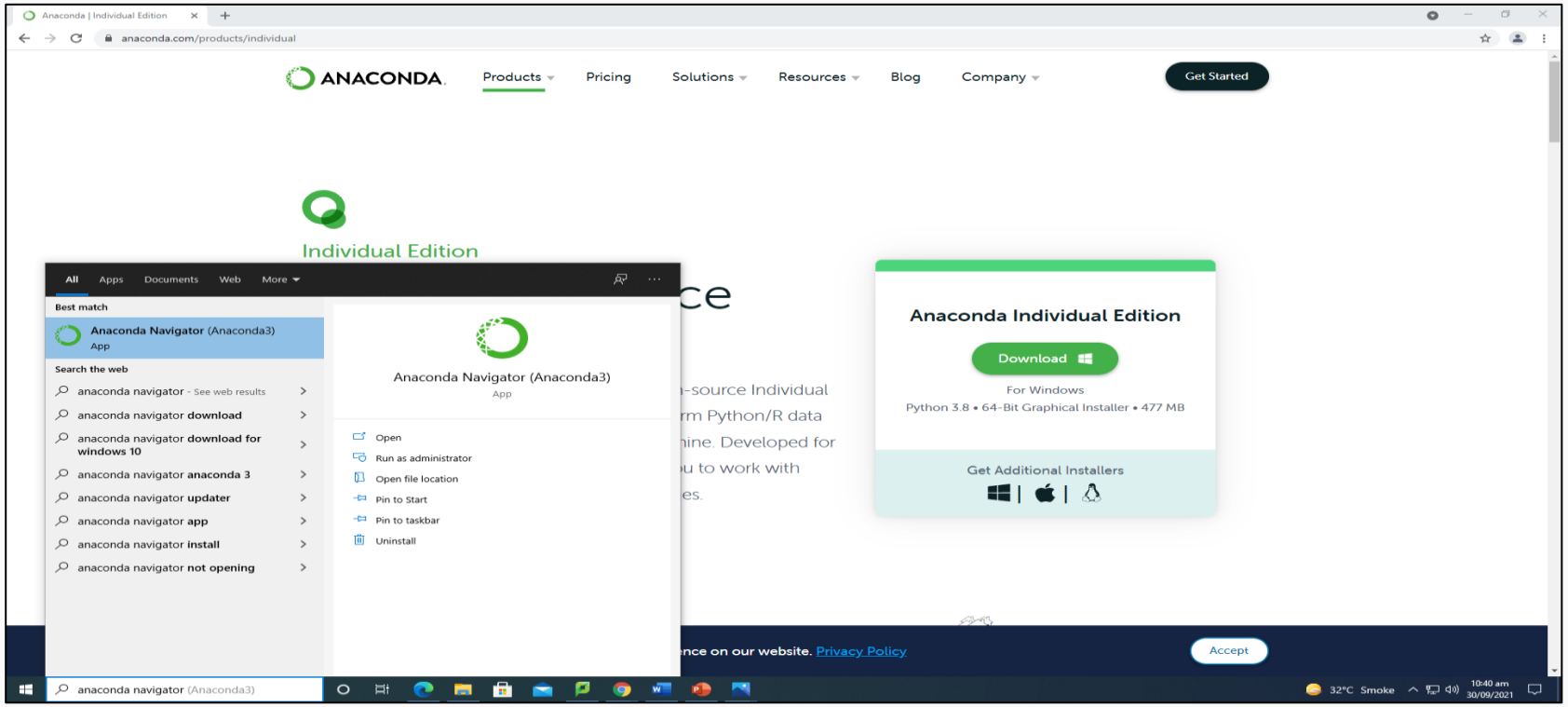
- Click Finish.



Meta Brains

Anaconda for Windows (10/13)

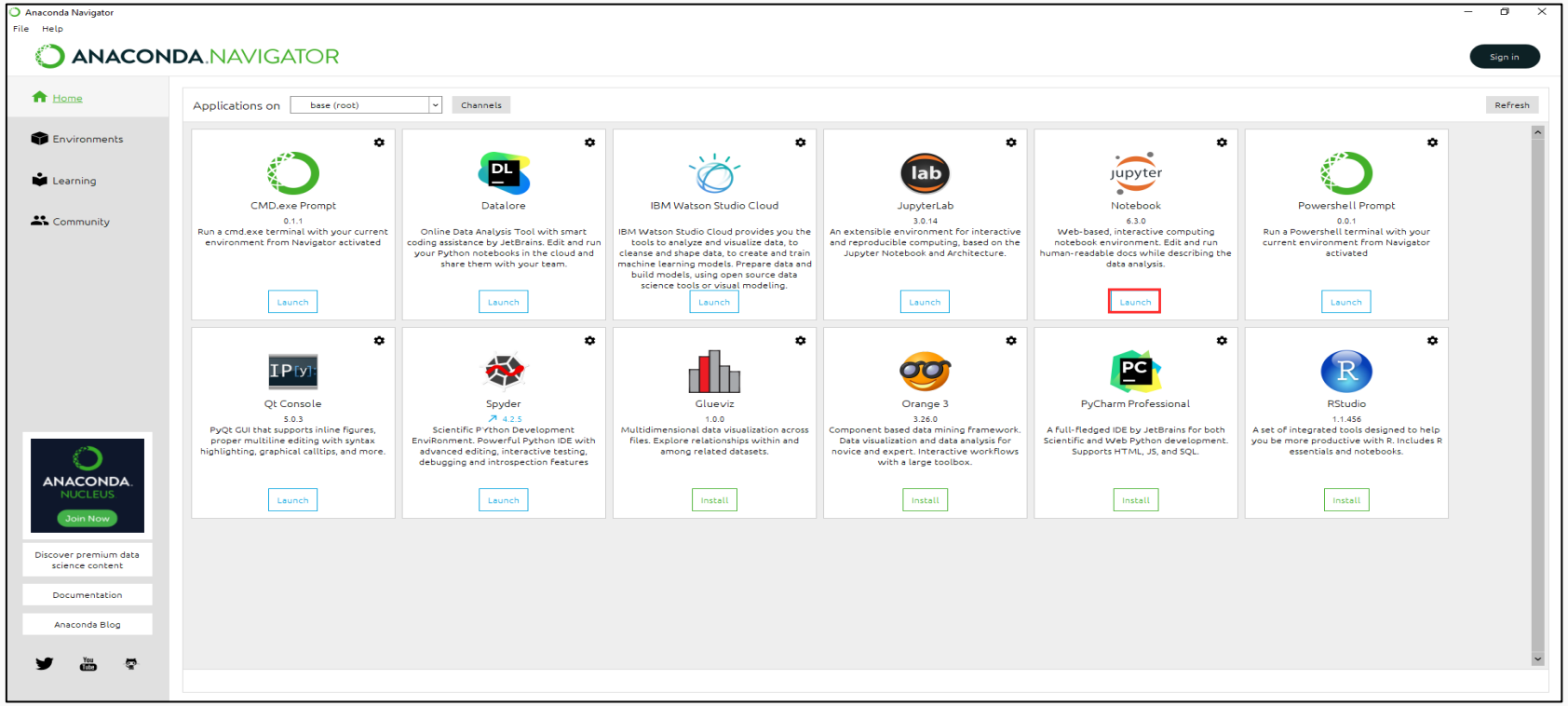
- To launch Jupyter Notebook, search for anaconda navigator and open it.



Meta Brains

Anaconda for Windows (11/13)

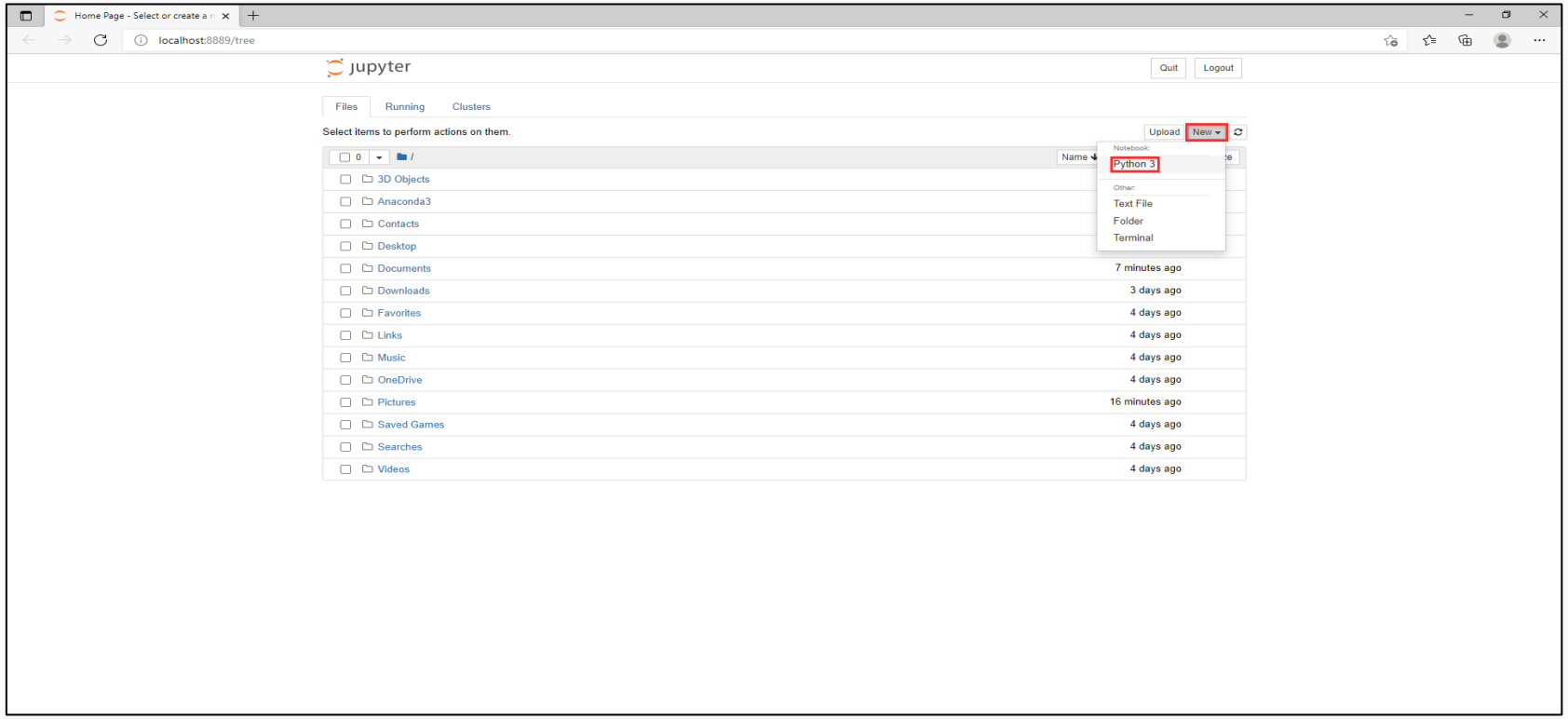
- Following screen will open up.
- Launch Jupyter Notebook from here.



Meta Brains

Anaconda for Windows (12/13)

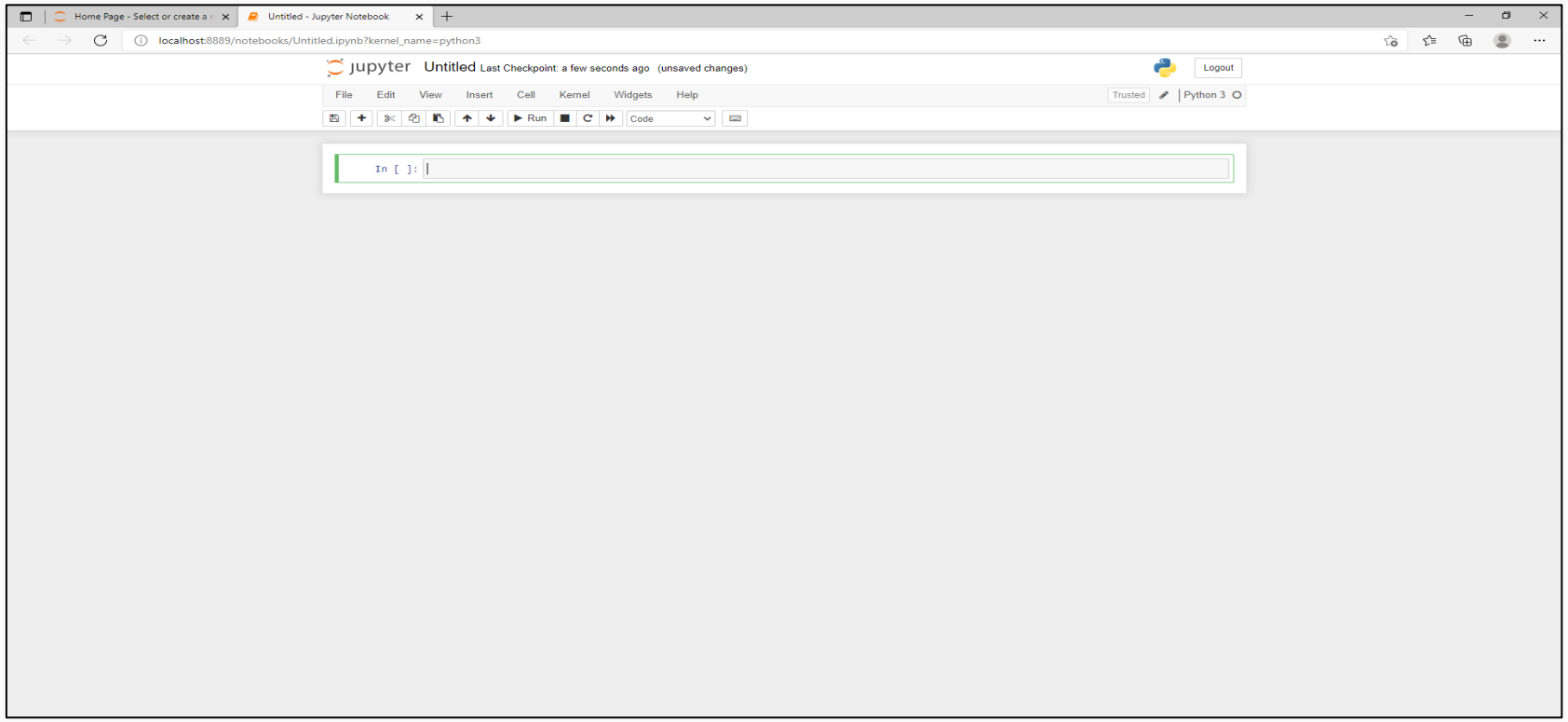
- This will launch Jupyter Notebook in your default browser.
- Click on New -> Python3 to create a new Notebook.



Meta Brains

Anaconda for Windows (13/13)

- This is how a Notebook looks like.



Meta Brains

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Python libraries for Machine Learning

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

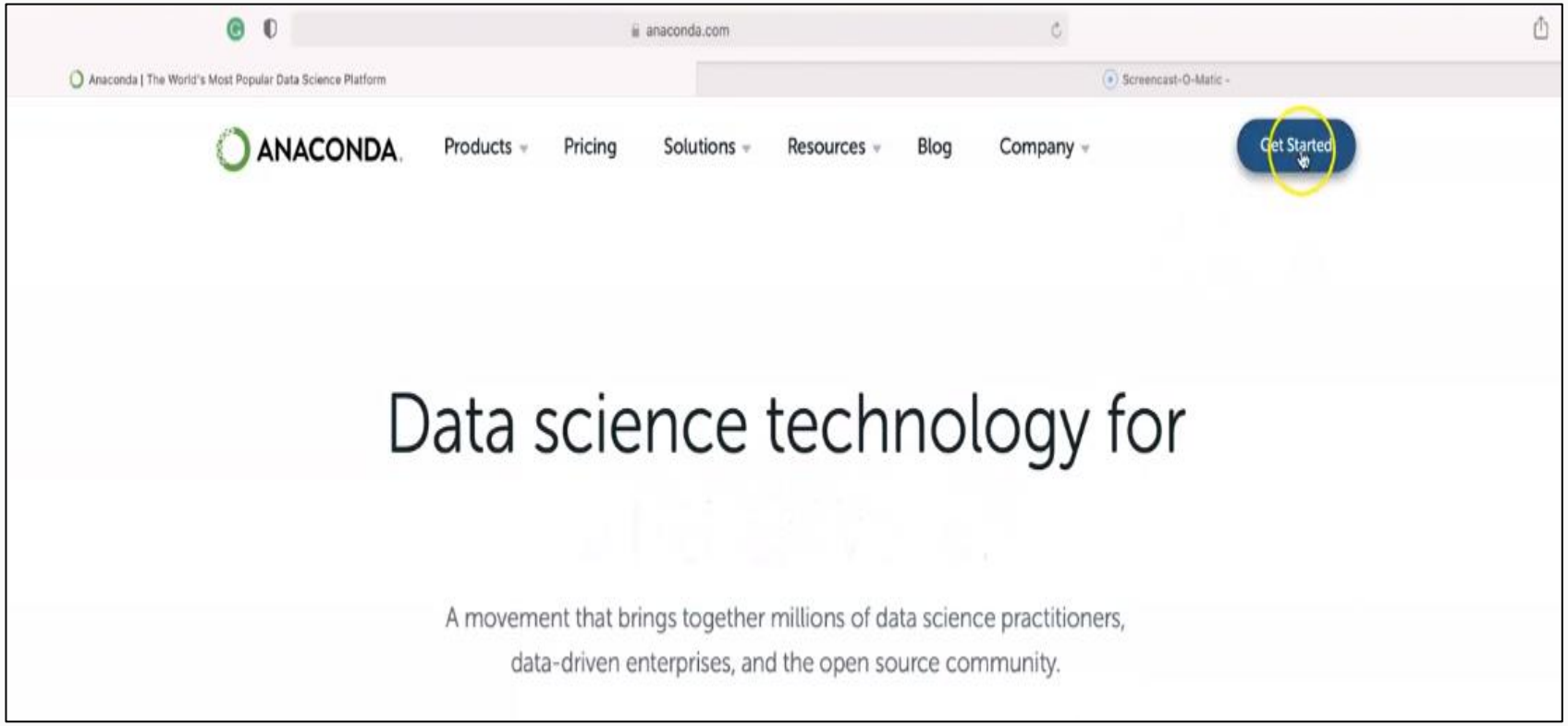
07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Anaconda for Mac (1/14)

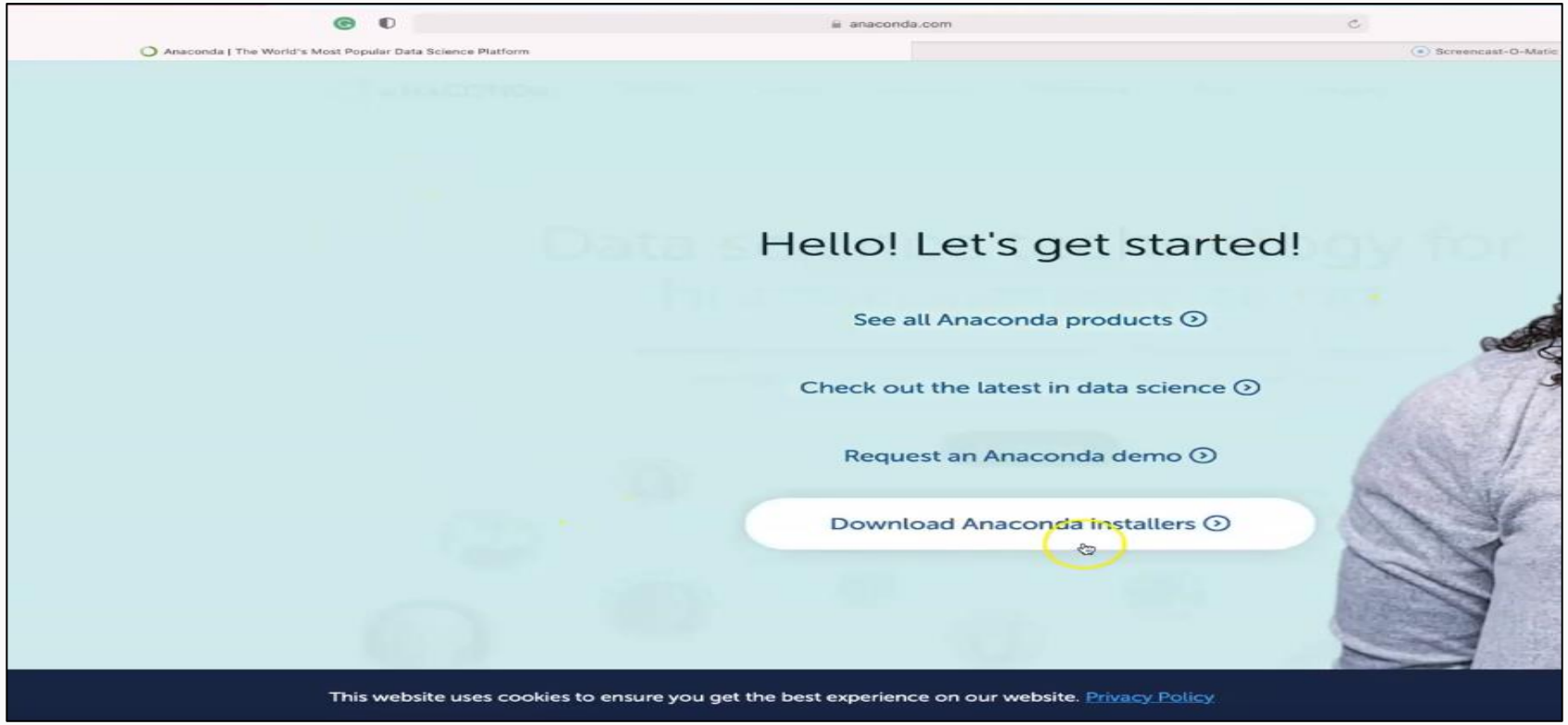
- Visit: https://www.anaconda.com/
- Click on Get Started.



Meta Brains

Anaconda for Mac (2/14)

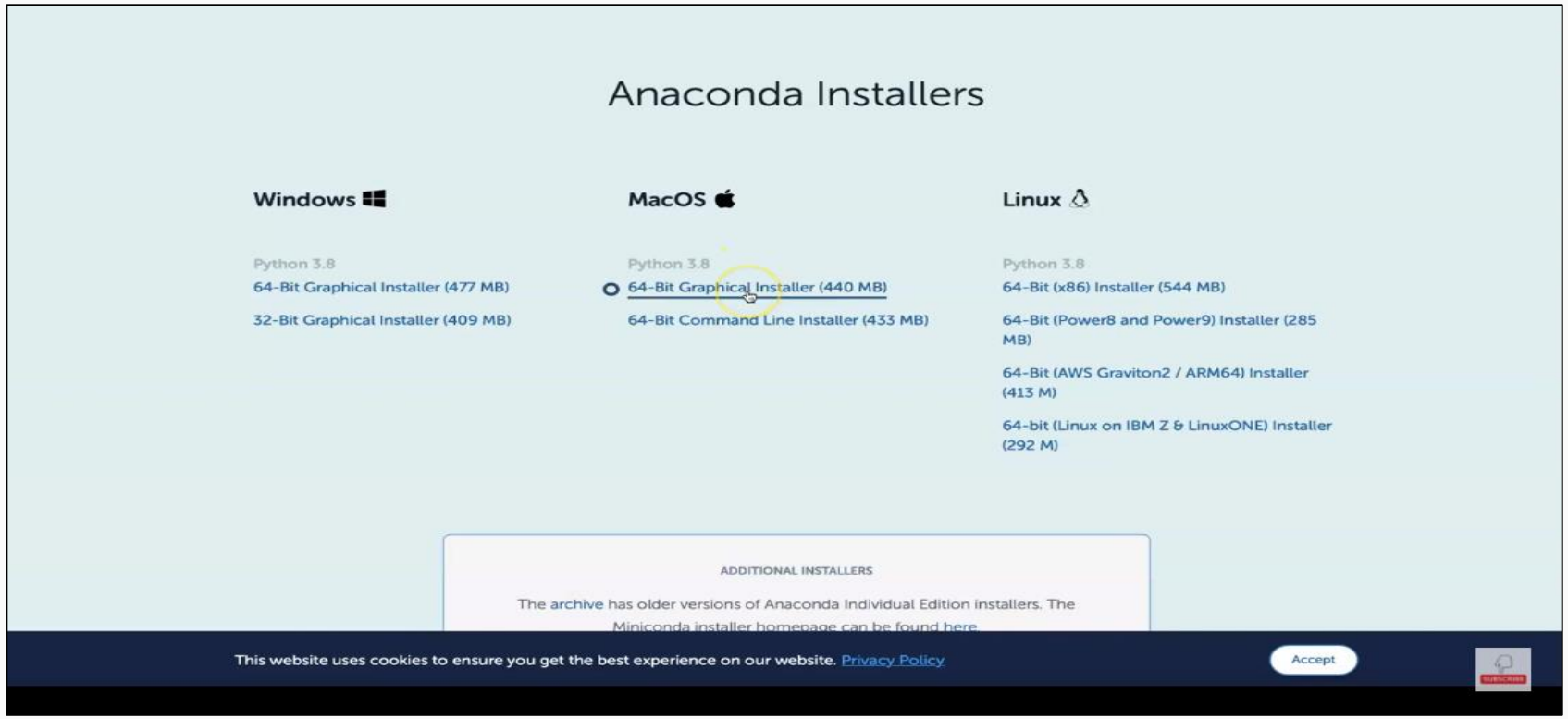
- Click Download Anaconda Installers.



Meta Brains

Anaconda for Mac (3/14)

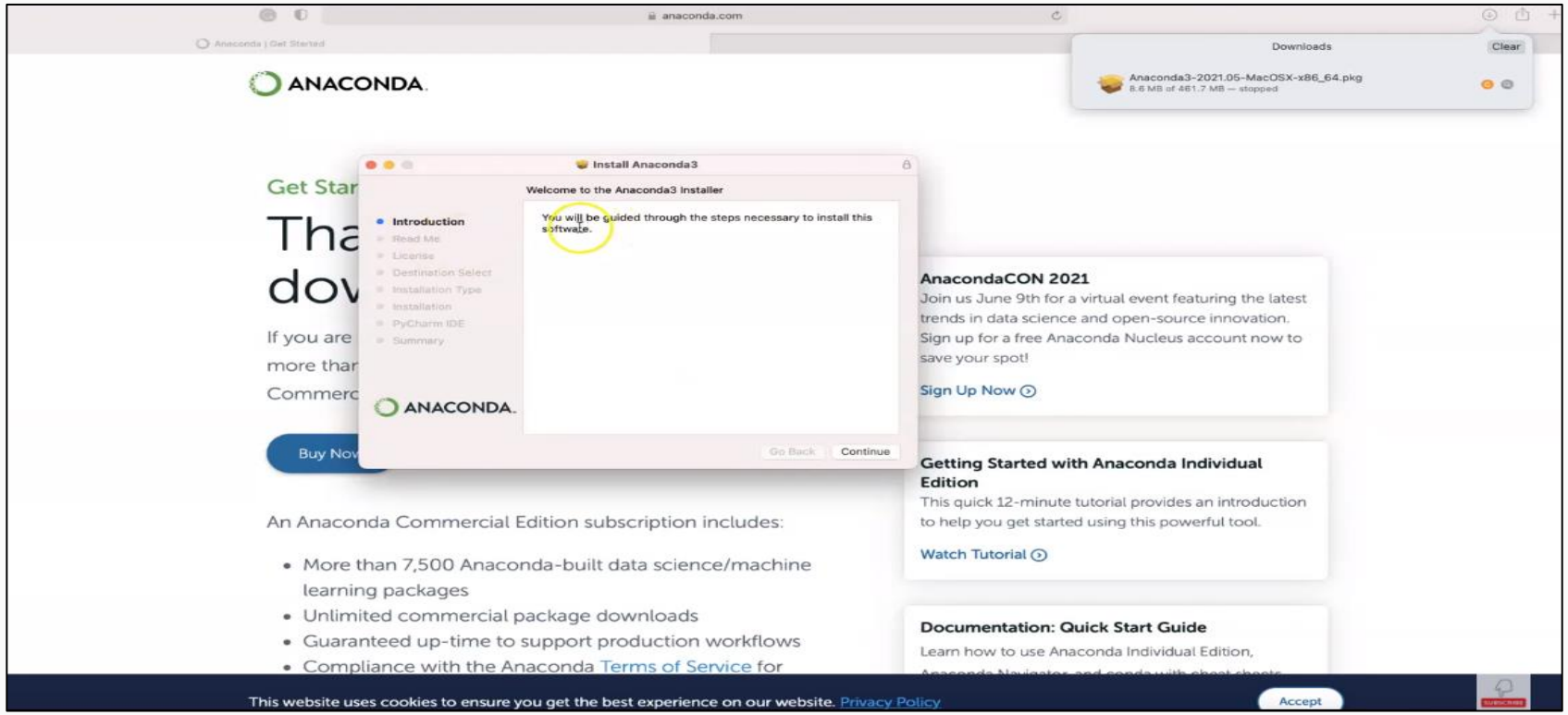
- Download the graphical installer for Mac.



Meta Brains

Anaconda for Mac (4/14)

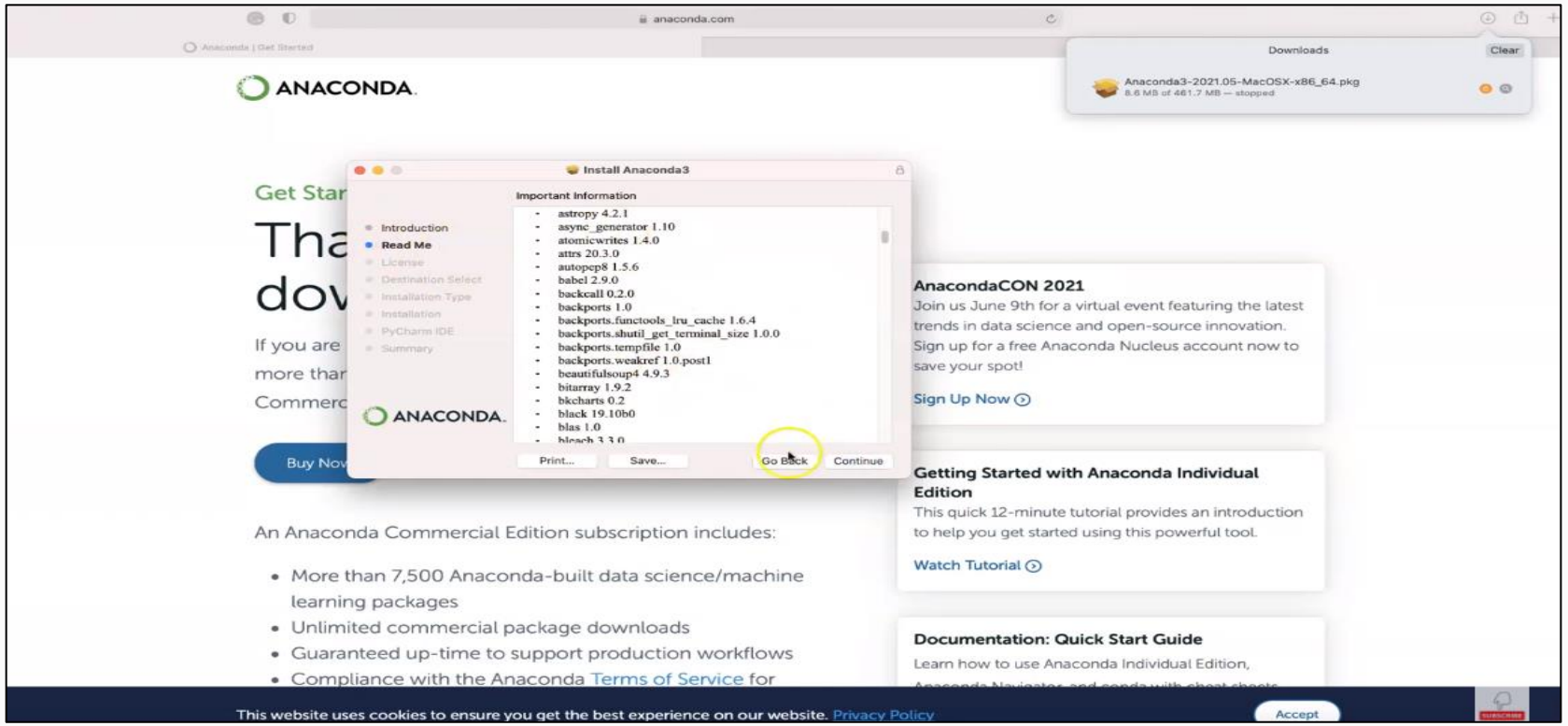
- Once download, launch the installation wizard by double clicking on the file.
- Click Continue.



Meta Brains

Anaconda for Mac (5/14)

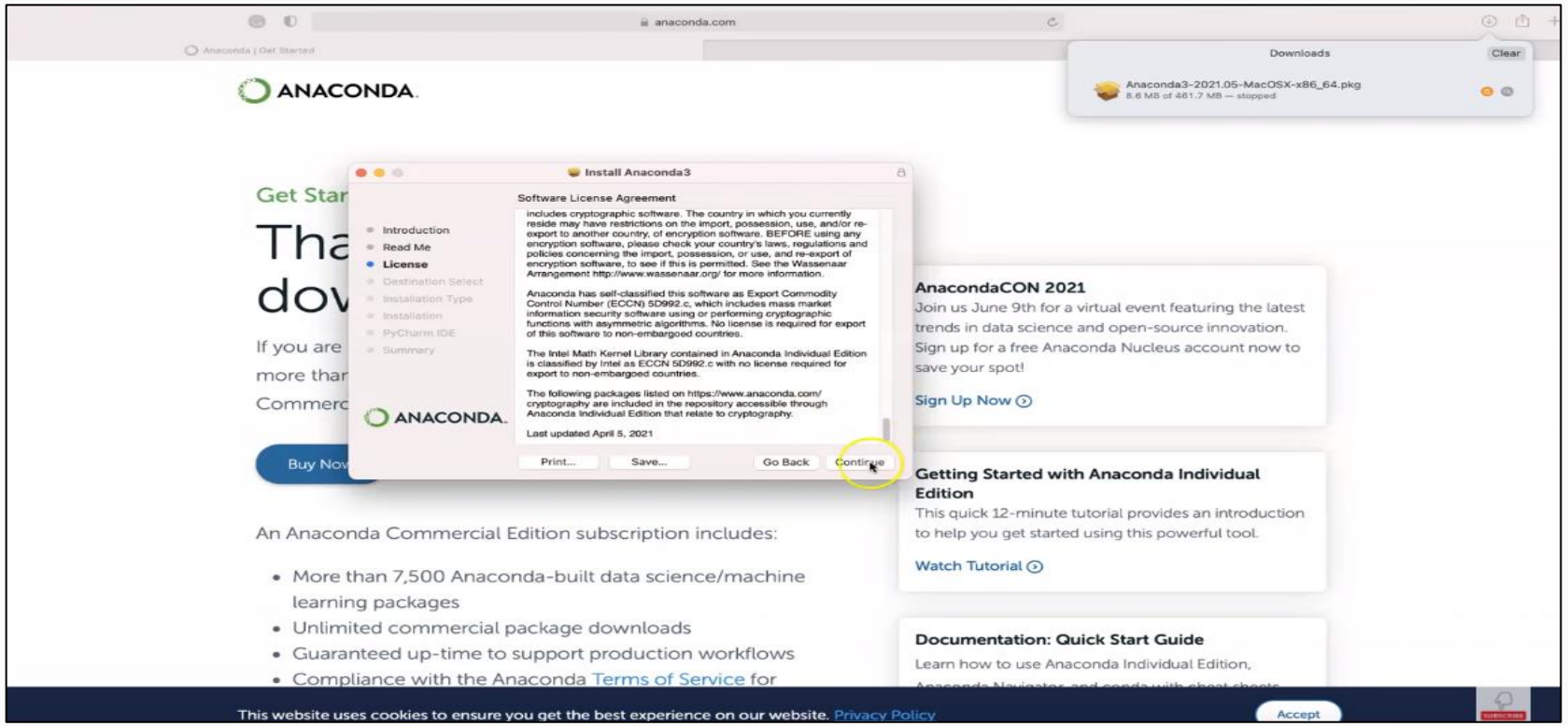
- Read the Read Me file and click continue.



Meta Brains

Anaconda for Mac (6/14)

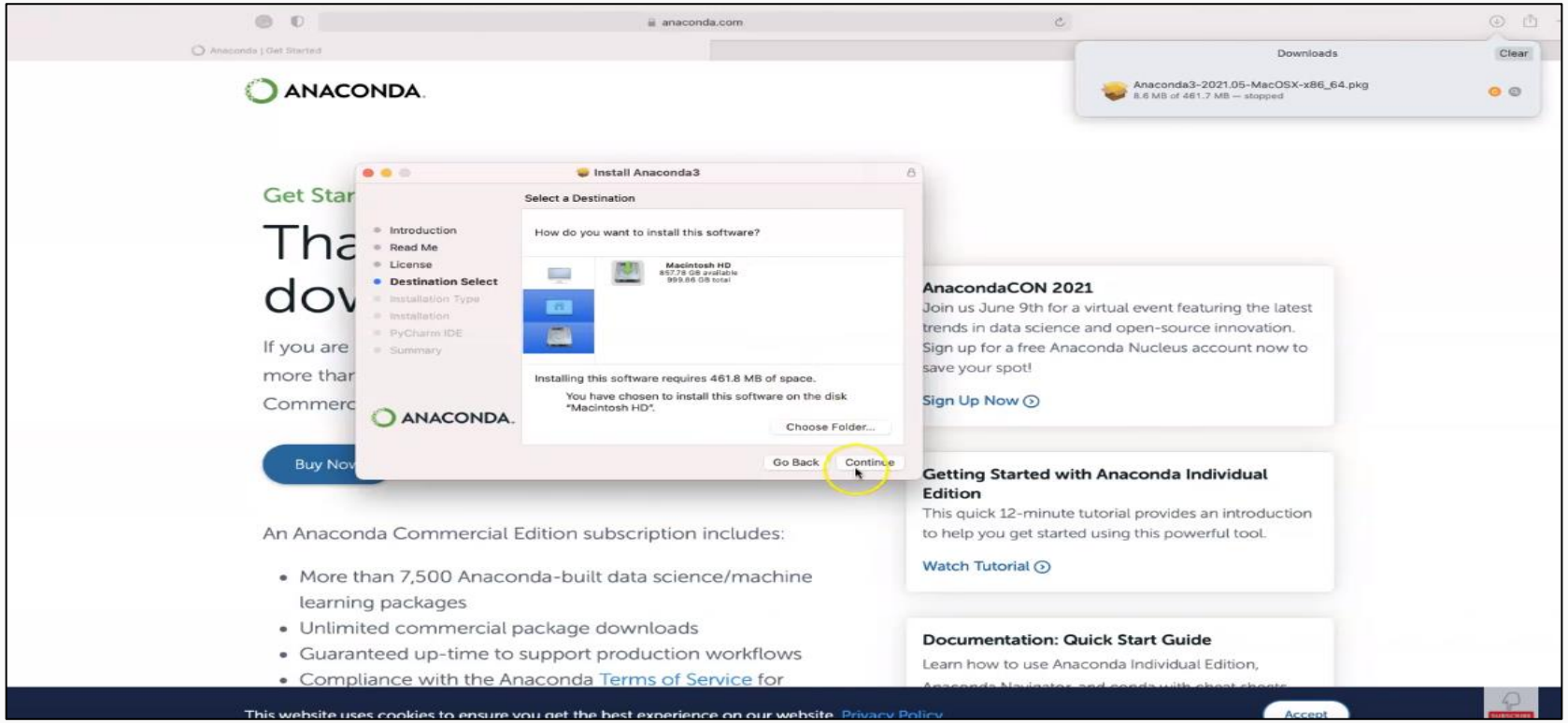
- Read the License Agreement and click continue.



Meta Brains

Anaconda for Mac (7/14)

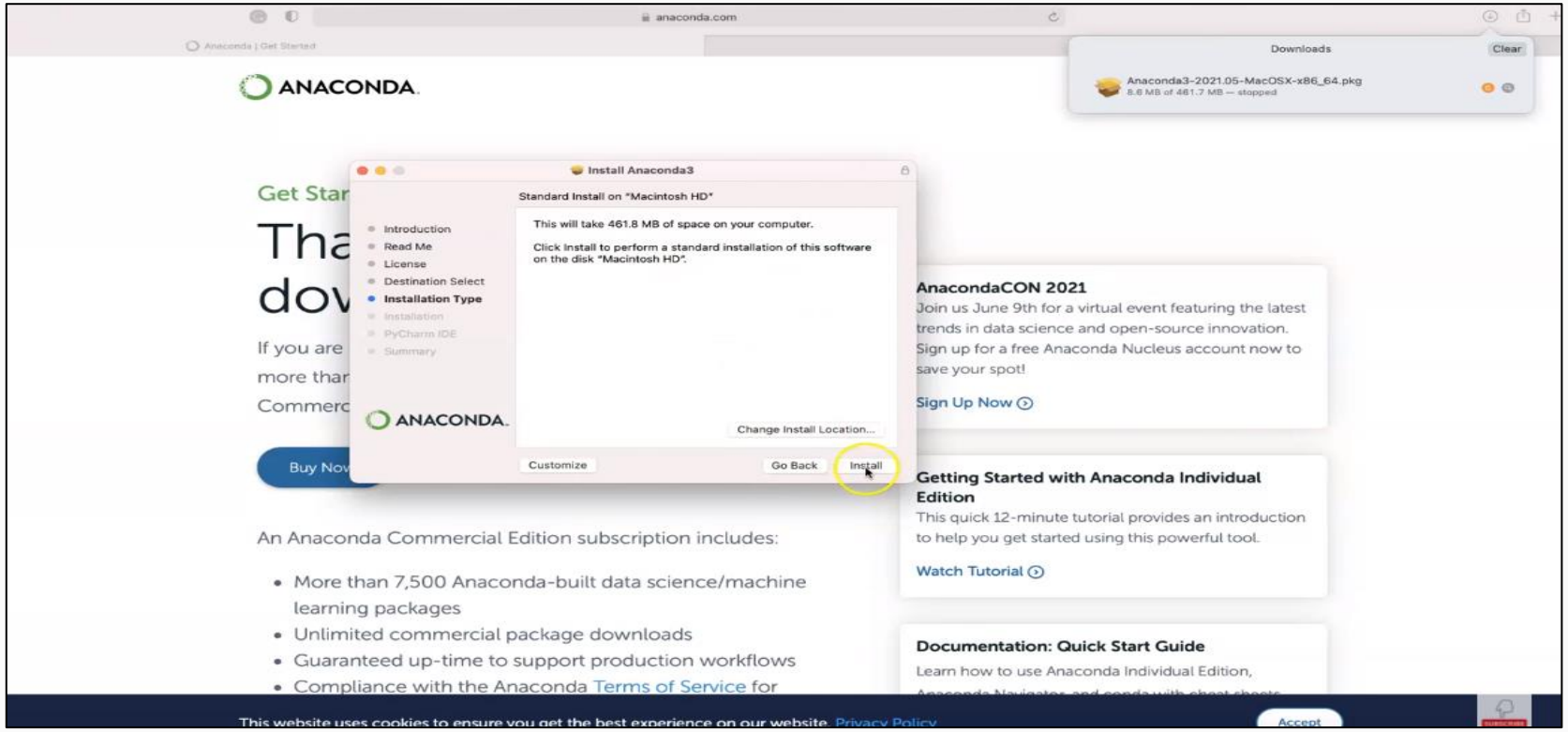
- Select the destination and click Continue.



Meta Brains

Anaconda for Mac (8/14)

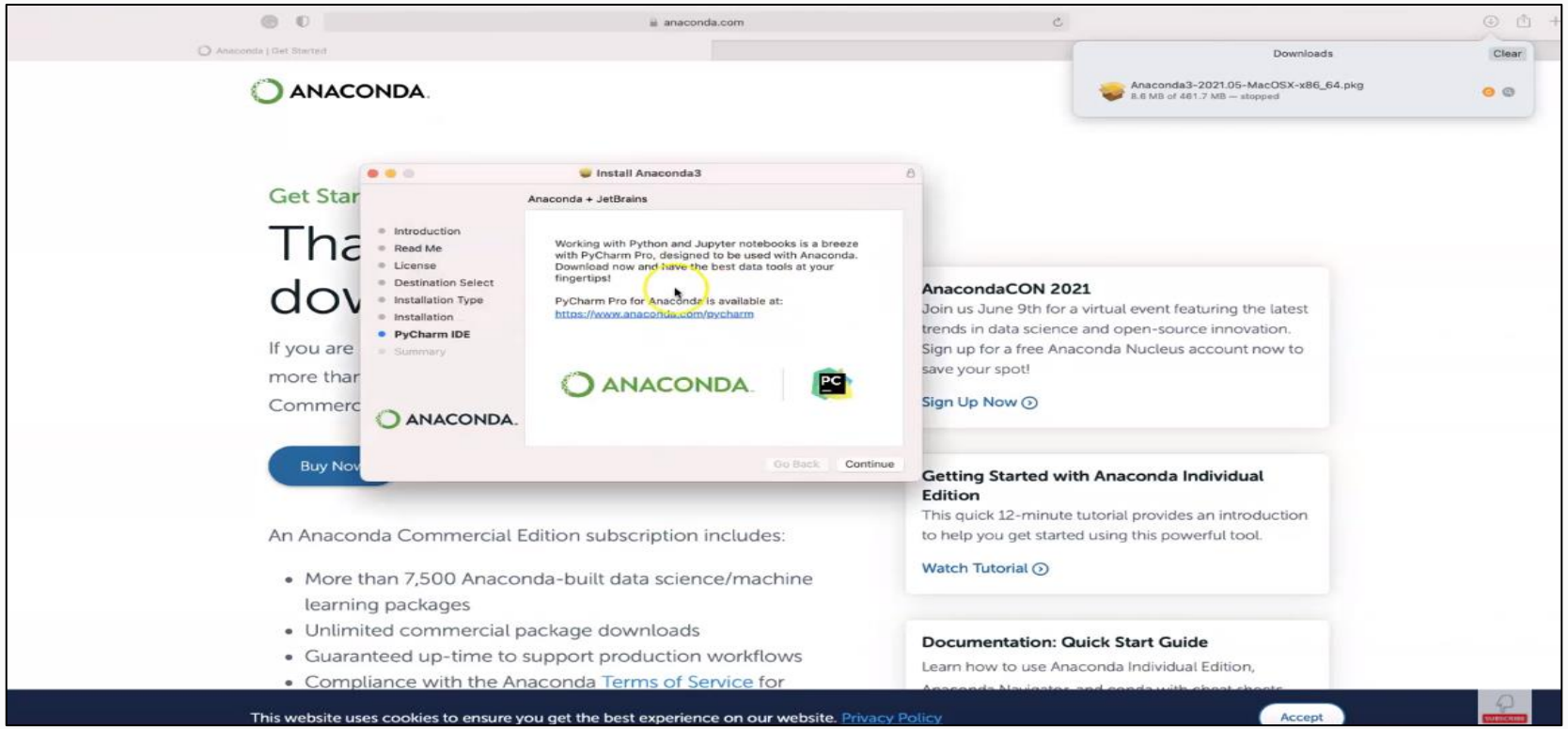
- Click Install.



Meta Brains

Anaconda for Mac (9/14)

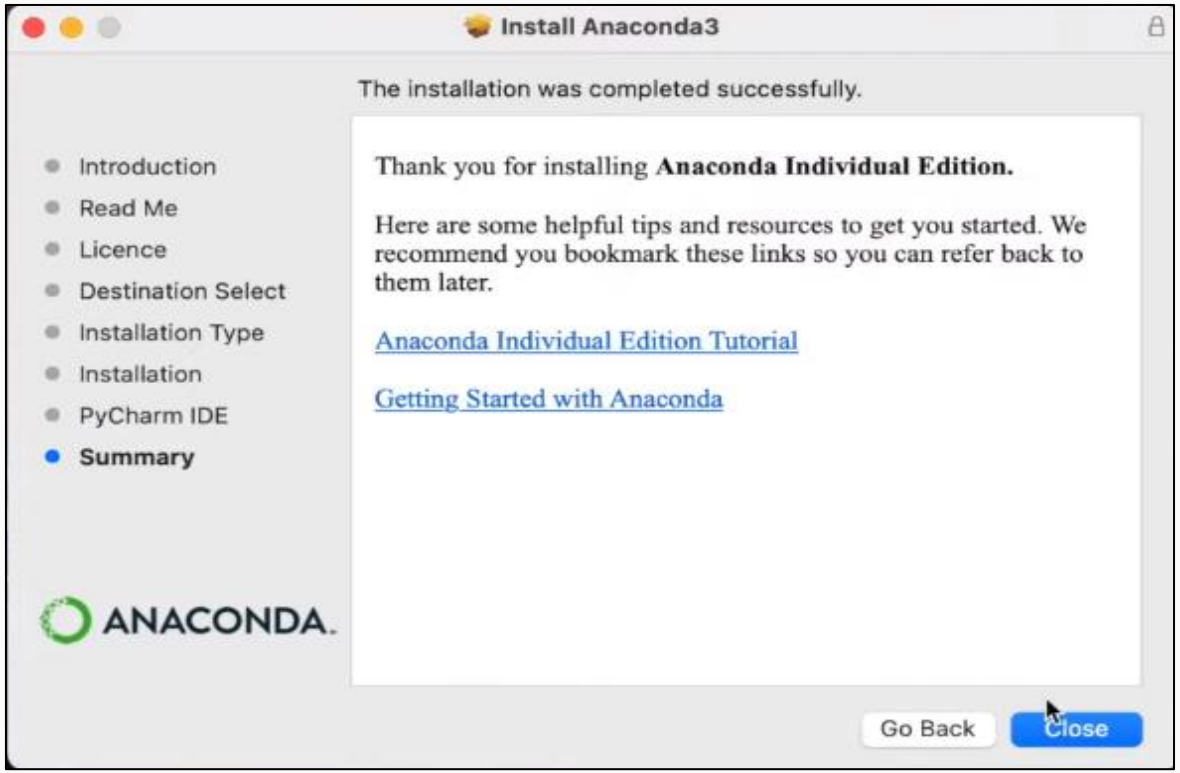
- Click Continue.



Meta Brains

Anaconda for Mac (10/14)

- Once the installation finishes, click Close.



Meta Brains

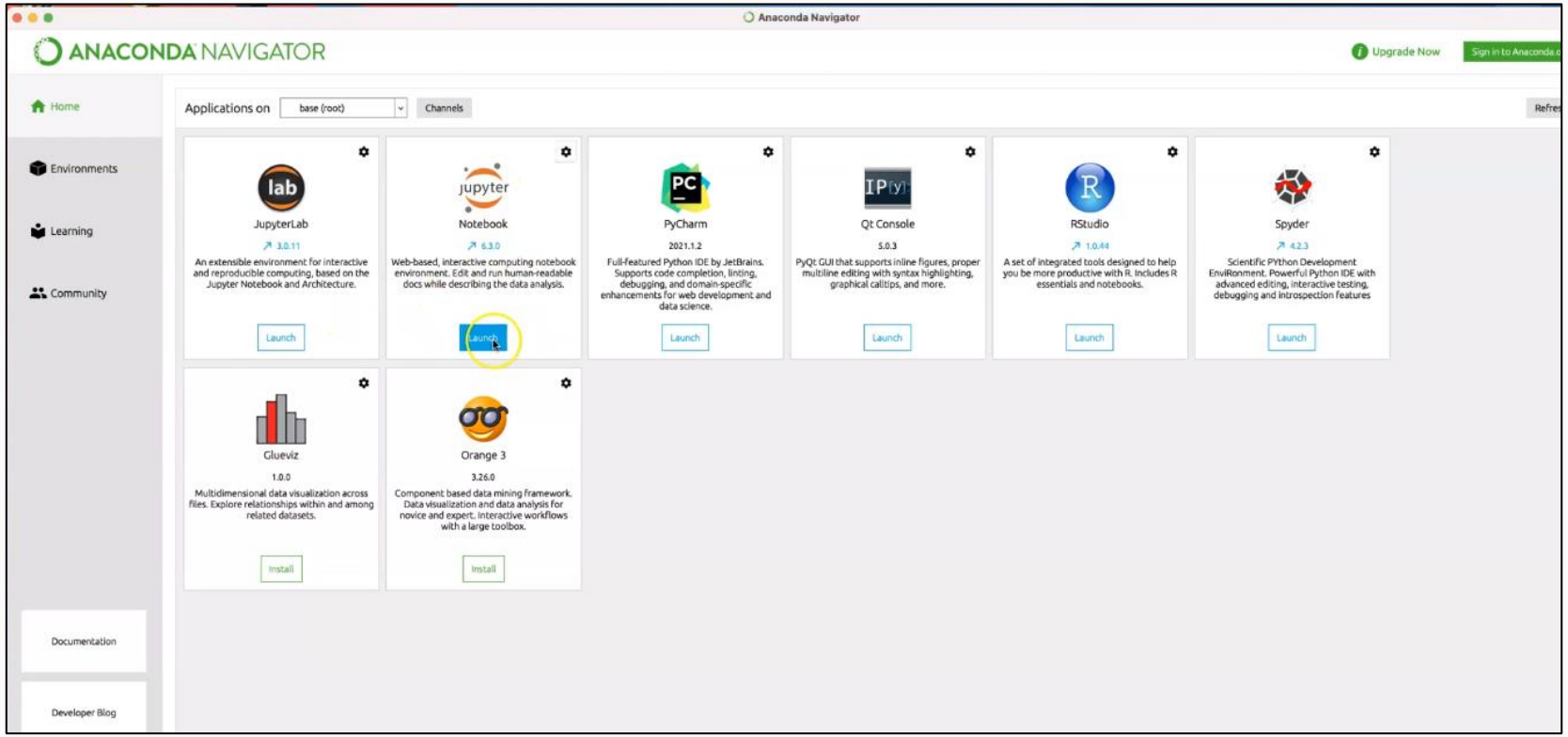
Anaconda for Mac (11/14)

- Open your spotlight and click on Anaconda Navigator to launch it.

Meta Brains

Anaconda for Mac (12/14)

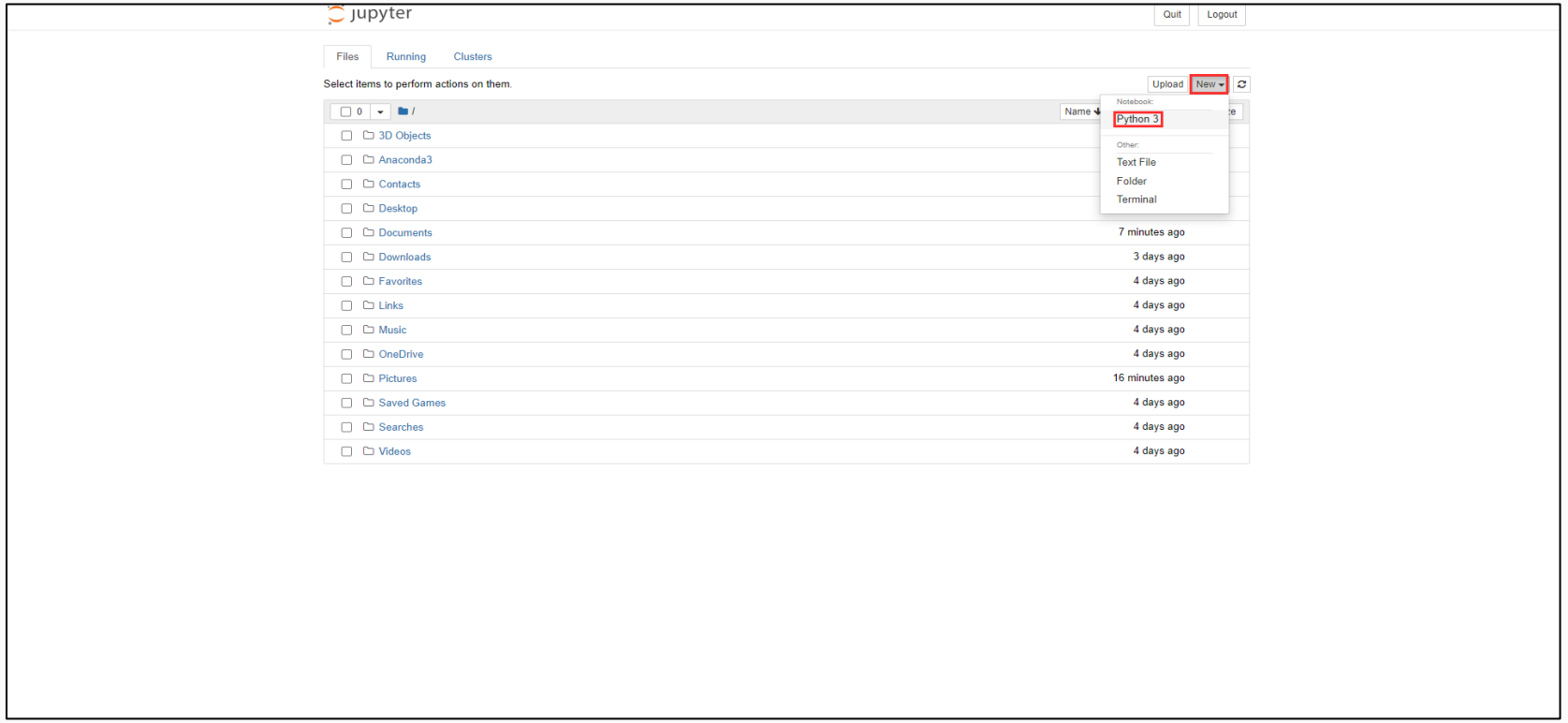
- Launch Jupyter Notebook. It will take sometime to launch.



Meta Brains

Anaconda for Mac (13/14)

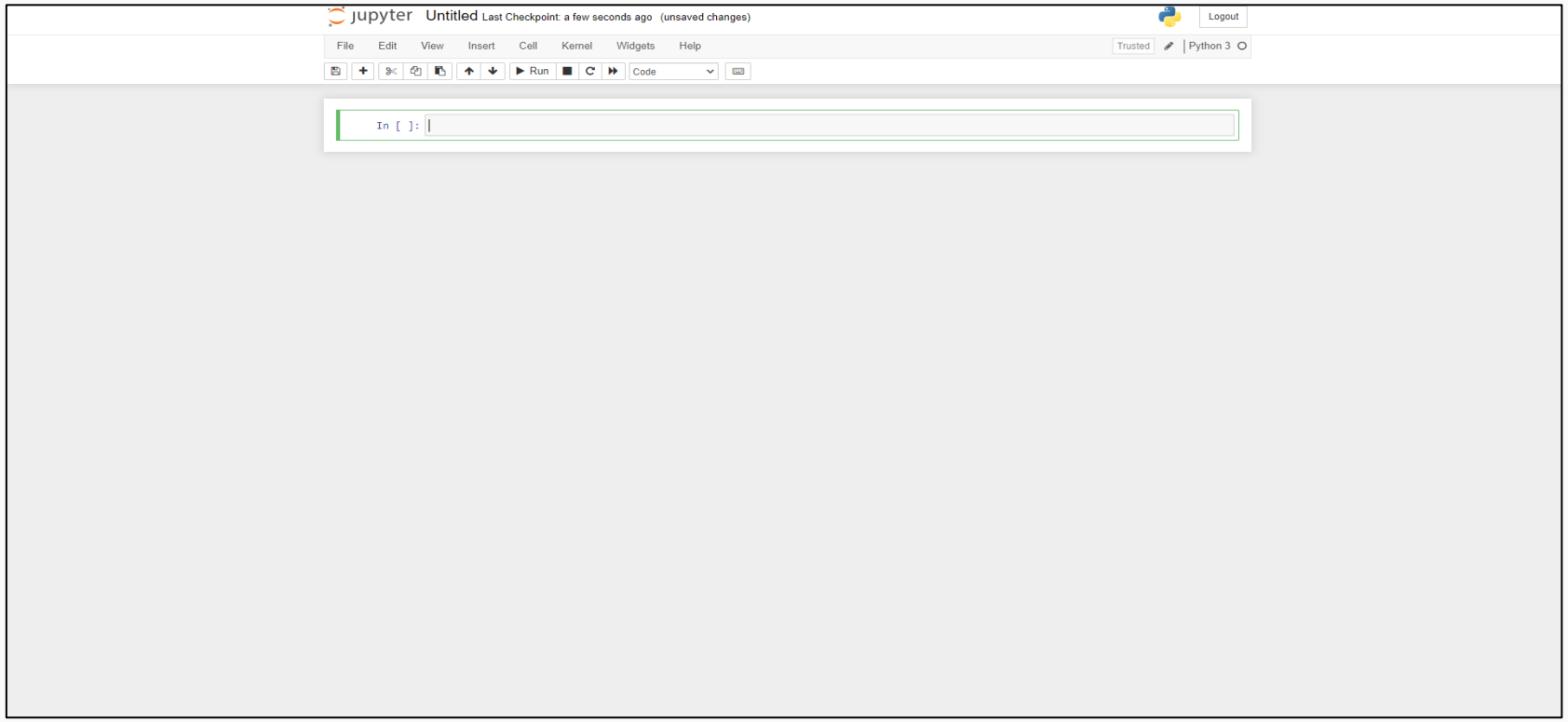
- This will launch Jupyter Notebook in your default browser.
- Click on New -> Python3 to create a new Notebook.



Meta Brains

Anaconda for Mac (14/14)

- This is how a Notebook looks like.



Meta Brains

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Python libraries for Machine Learning

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

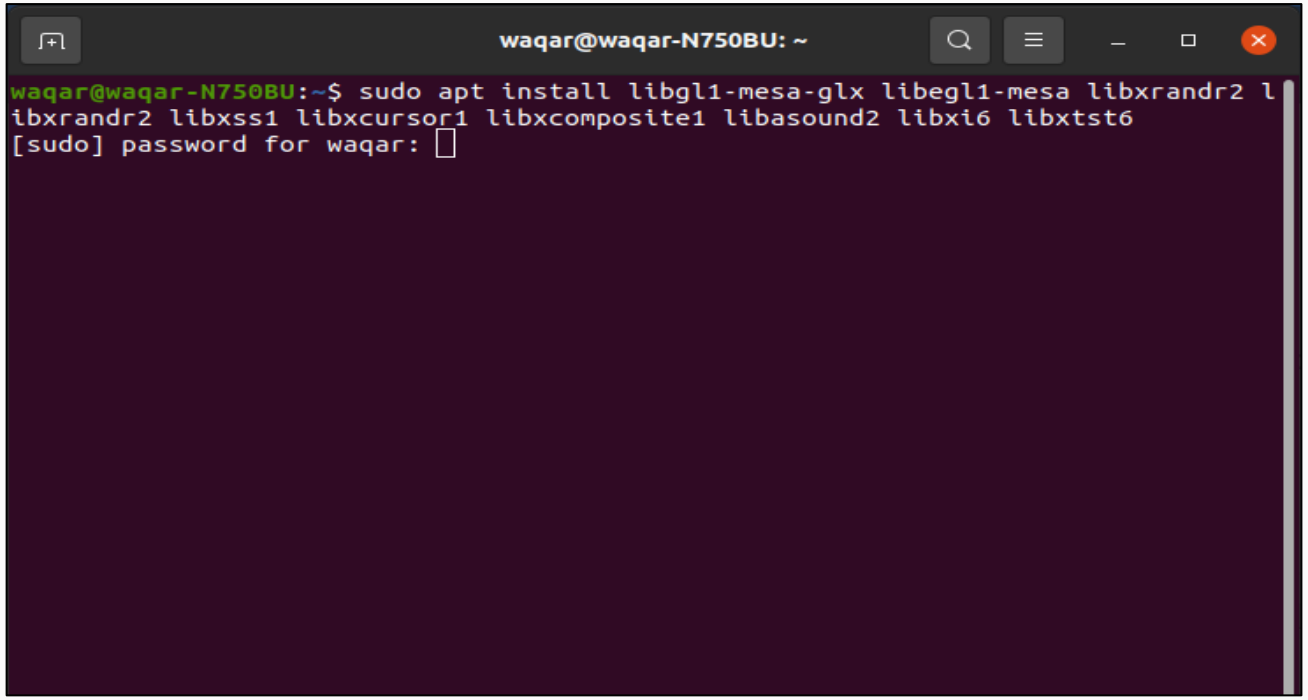
07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Anaconda for Ubuntu (1/11)

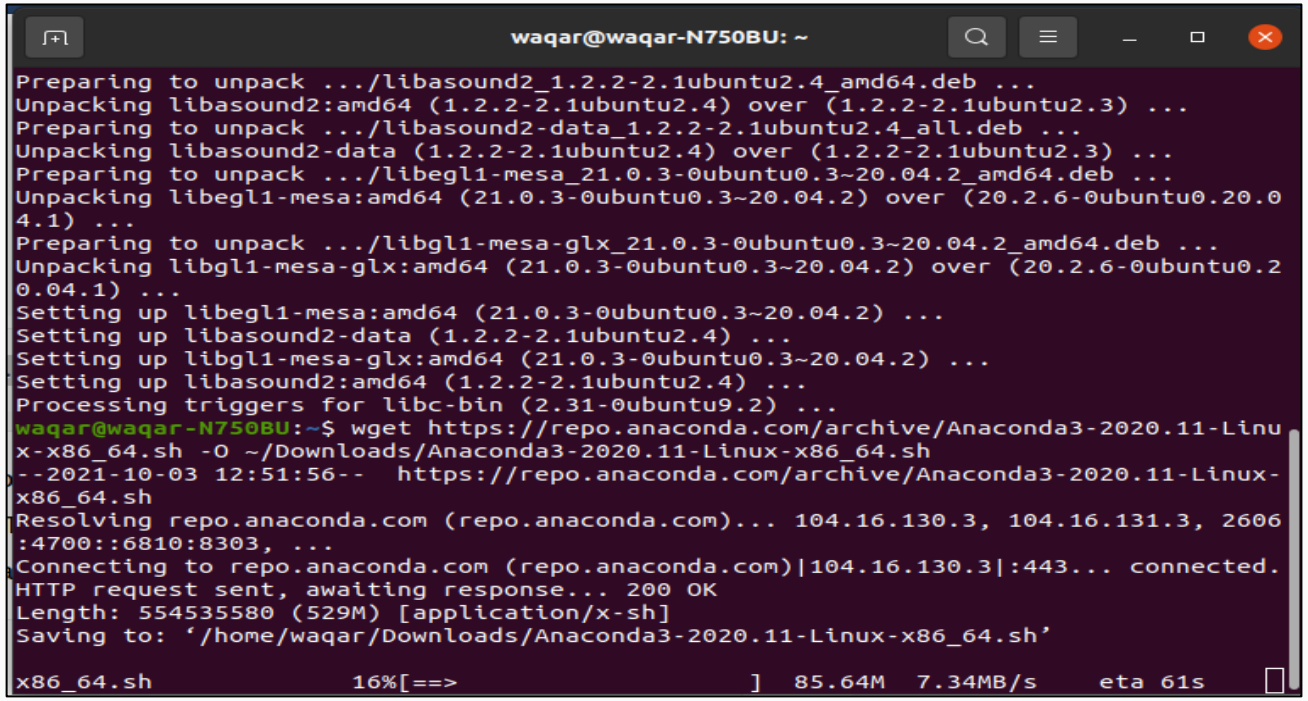
- Open your terminal, paste the following command, and hit enter
  sudo apt install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libxcomposite1 libasound2 libxi6 libxtst6
- Type your password and hit Enter.



Meta Brains

Anaconda for Ubuntu (2/11)

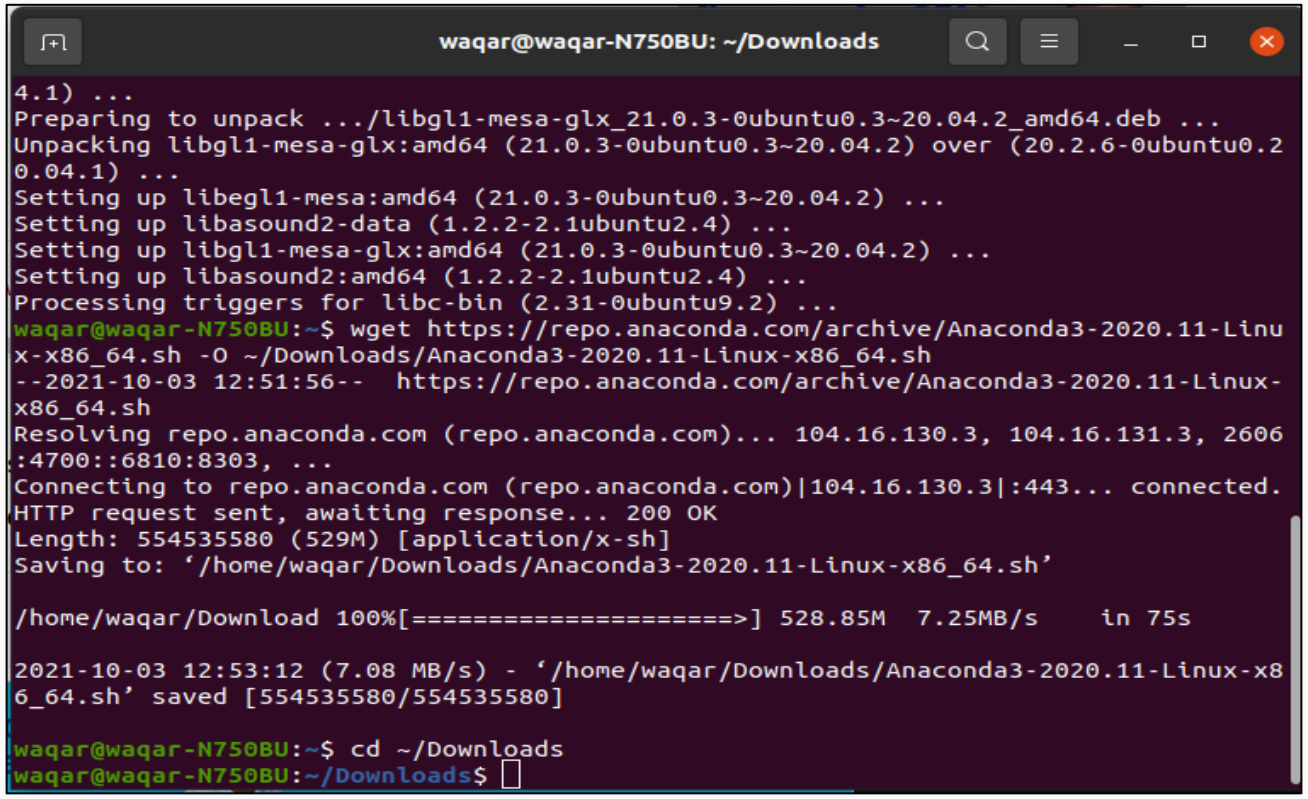
- Next give the following command and hit enter
  wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh -O ~/Downloads/Anaconda3-2020.11-Linux-x86_64.sh



Meta Brains

Anaconda for Ubuntu (3/11)

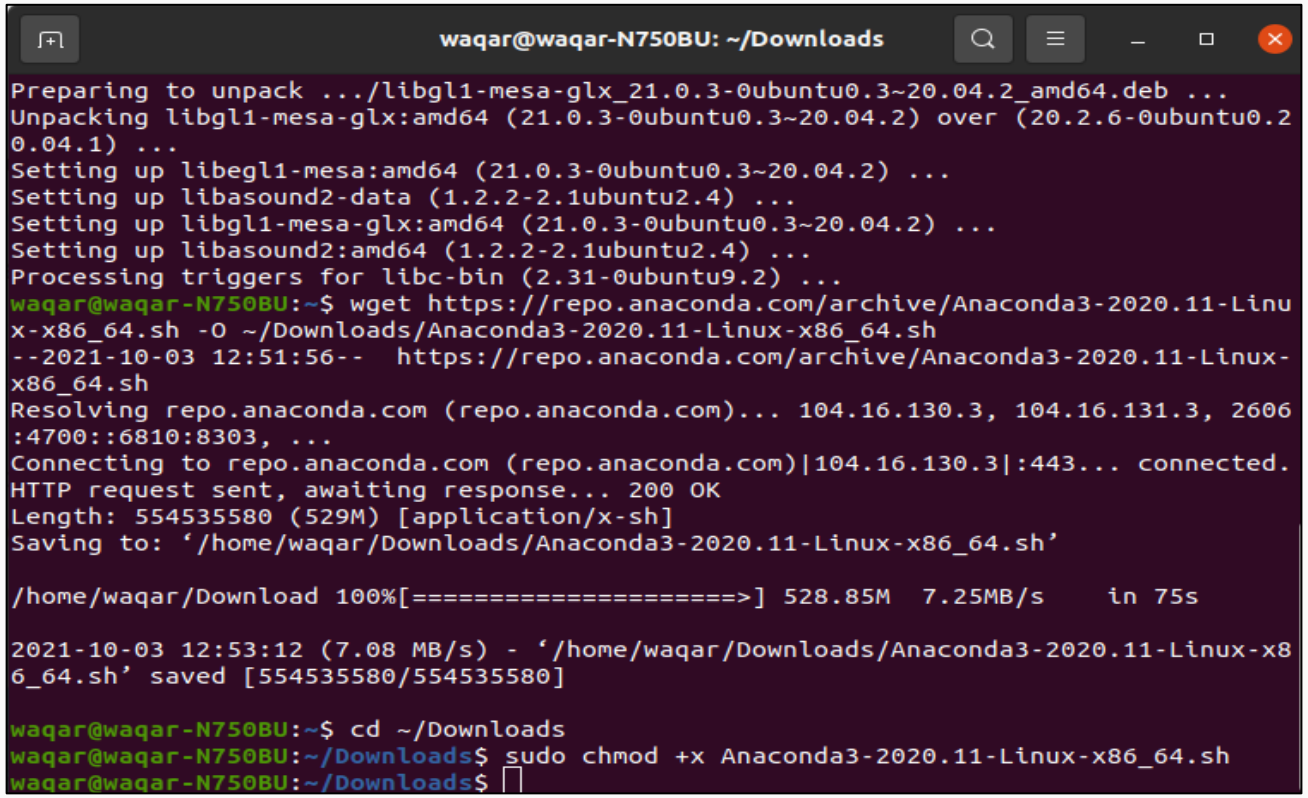
- Next give the following command and hit enter
  cd ~/Downloads



Meta Brains

Anaconda for Ubuntu (4/11)

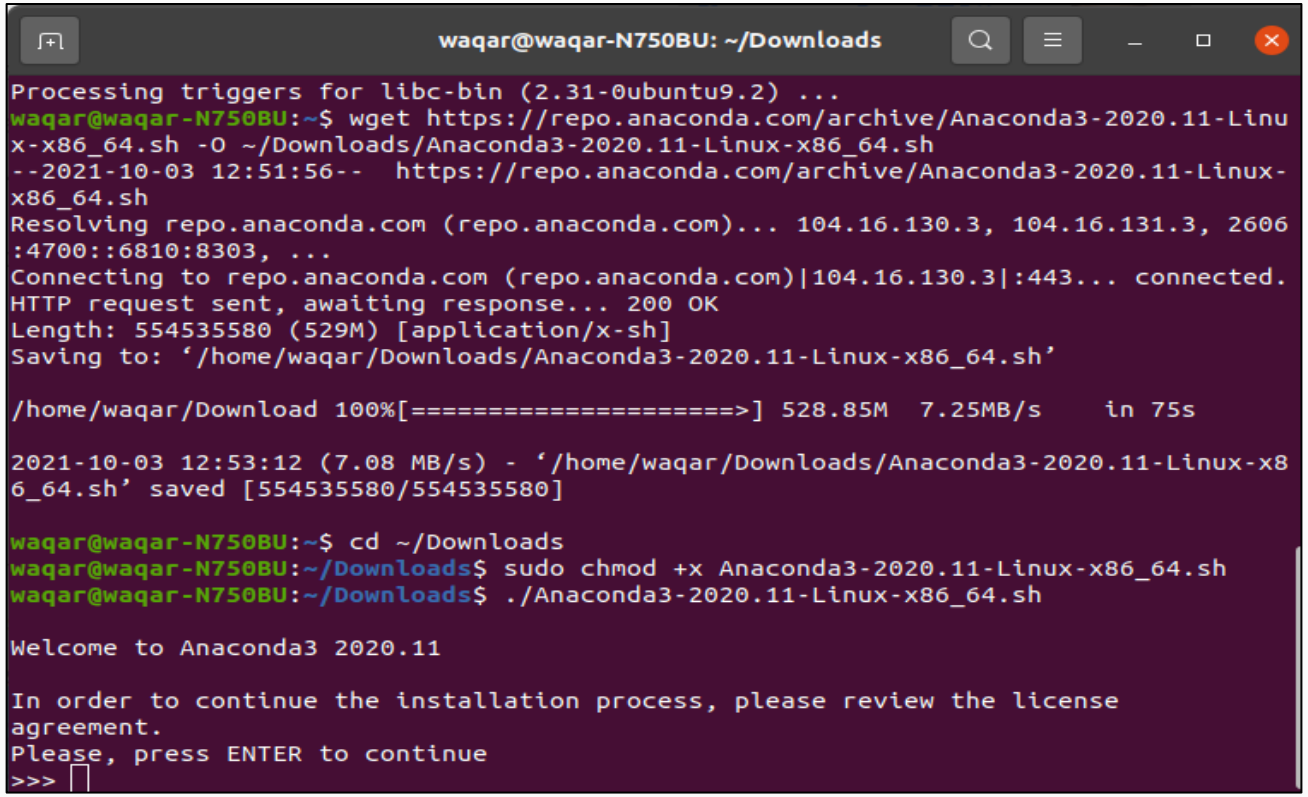
- Next give the following command and hit enter
  sudo chmod +x Anaconda3-2020.11-Linux-x86_64.sh



Meta Brains

Anaconda for Ubuntu (5/11)

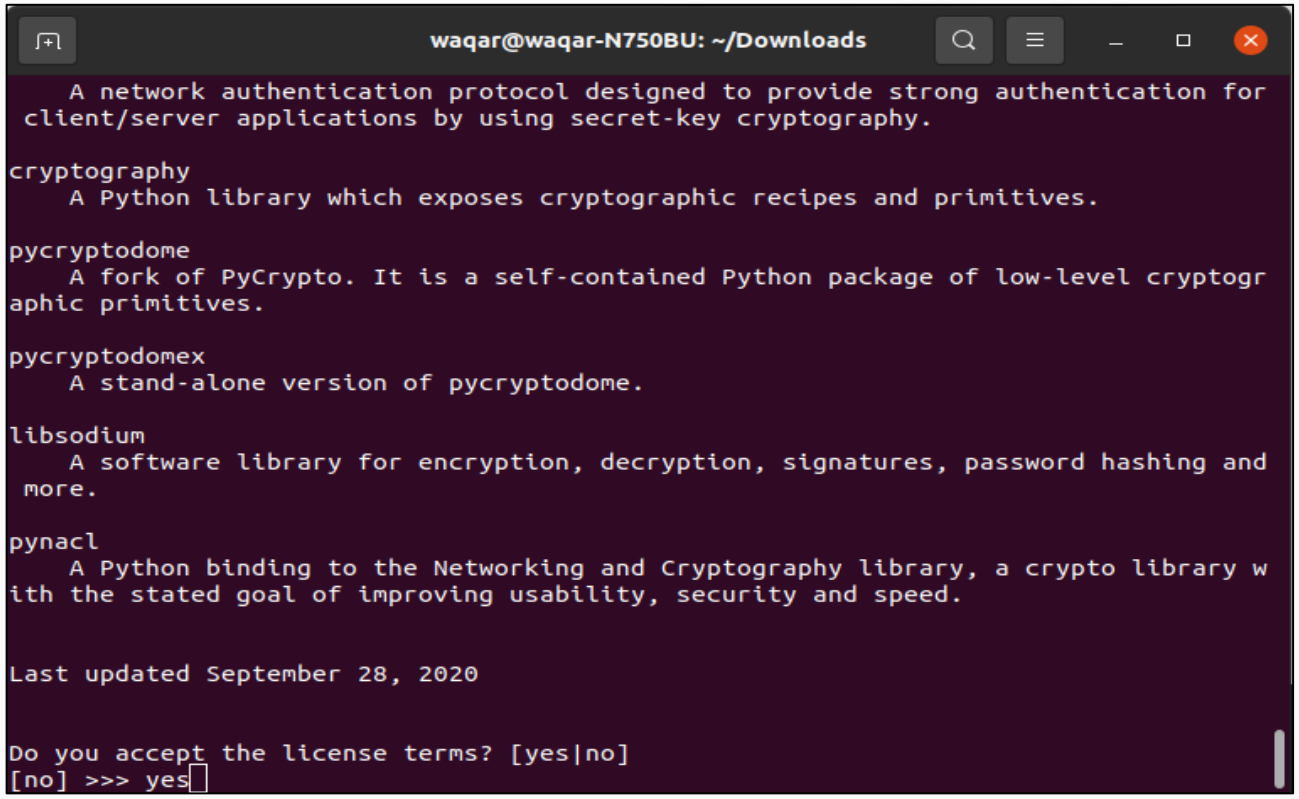
- Next give the following command and hit enter
  ./Anaconda3-2020.11-Linux-x86_64.sh
- When prompted, press enter to continue.



Meta Brains

Anaconda for Ubuntu (6/11)

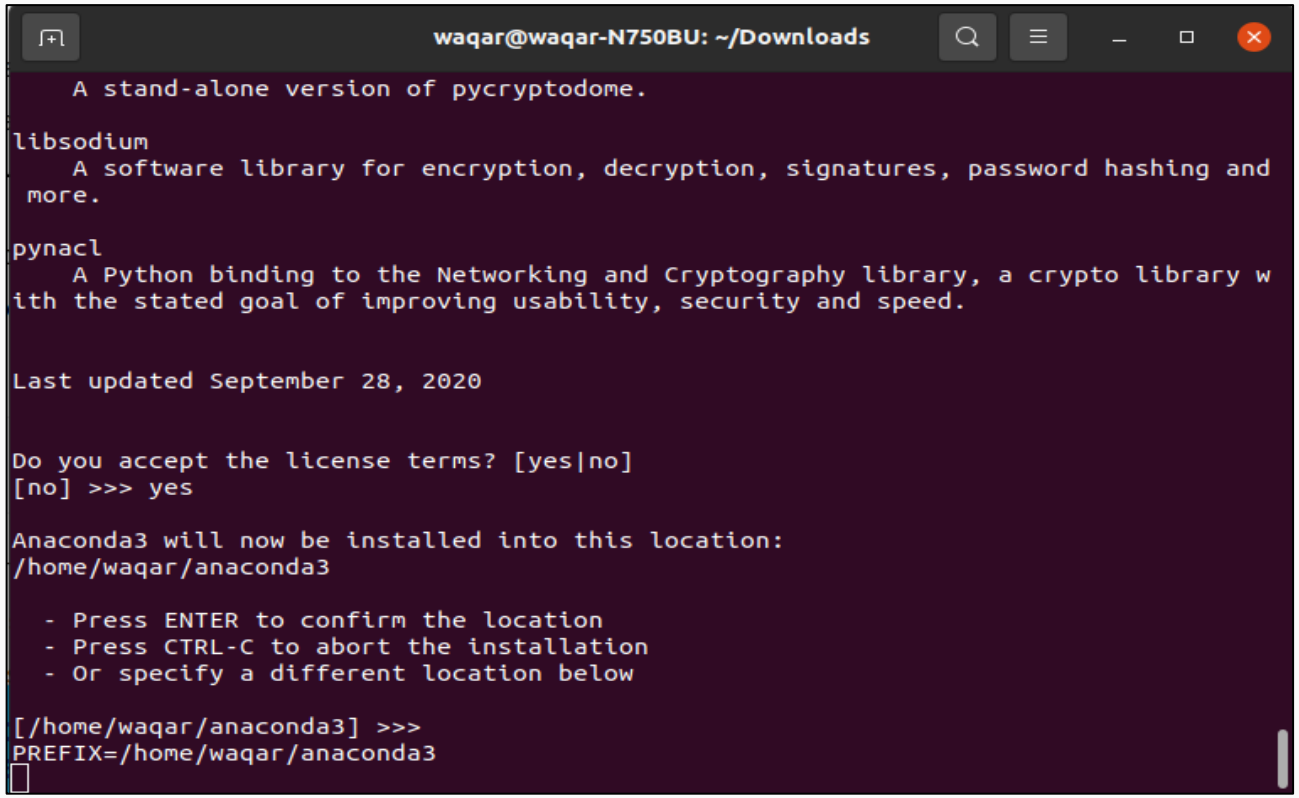
- Press the Enter key to scroll through the License Agreement.
- After reading the Agreement, type yes in the prompt and hit enter.



Meta Brains

Anaconda for Ubuntu (7/11)

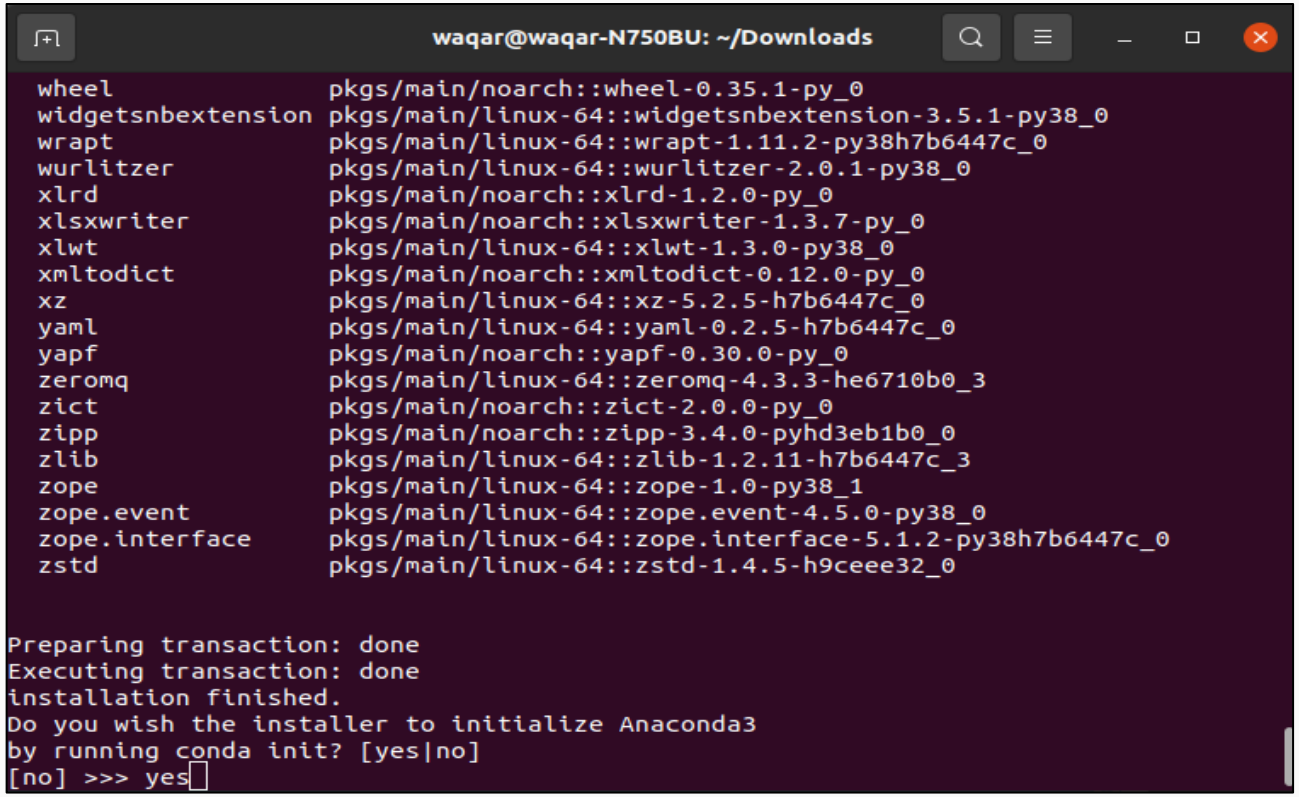
- Anaconda installer will ask you where do you want to install Anaconda. We suggest pressing Enter key to install it in the home directory.
- After sometime, Anaconda will be installed on your machine.



Meta Brains

Anaconda for Ubuntu (8/11)

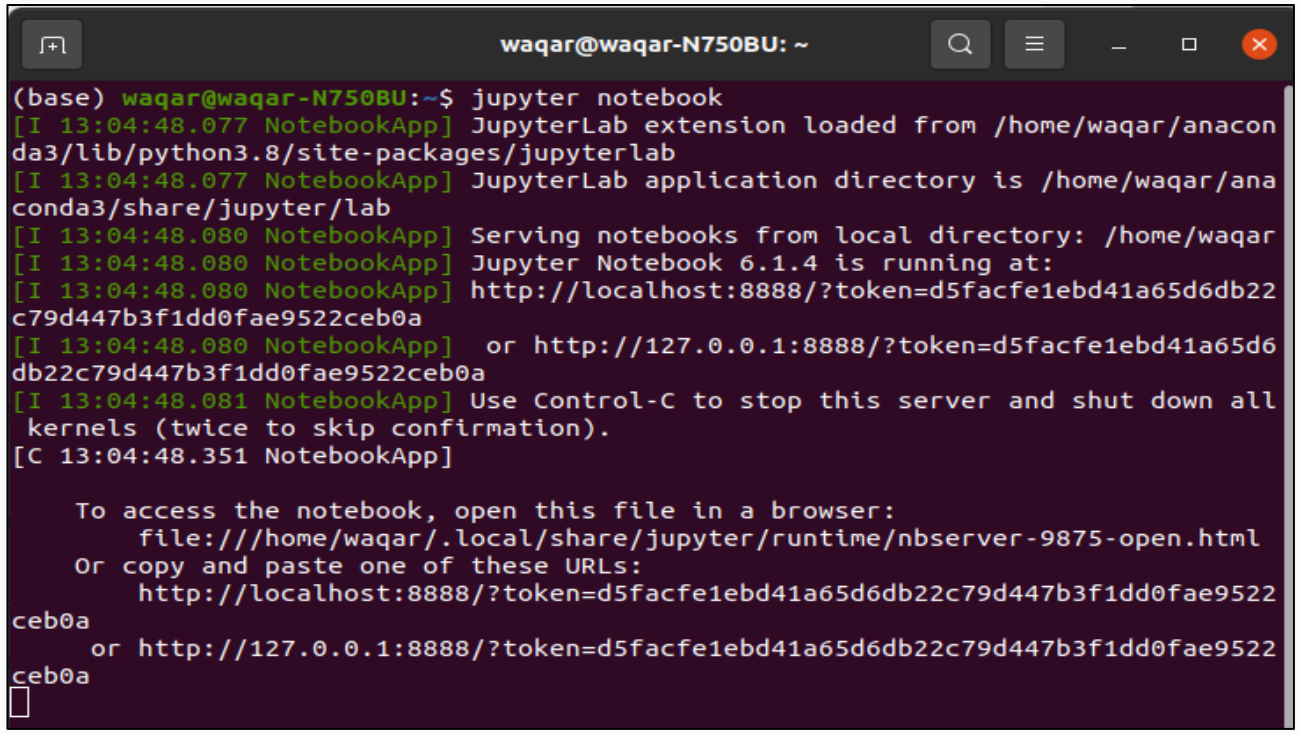
- Once installation is finished, you will be asked “Do you wish the installer to initialize Anaconda3 by running conda init?” type “yes” and hit Enter. Setup will finish the installation process.



Meta Brains

Anaconda for Ubuntu (9/11)

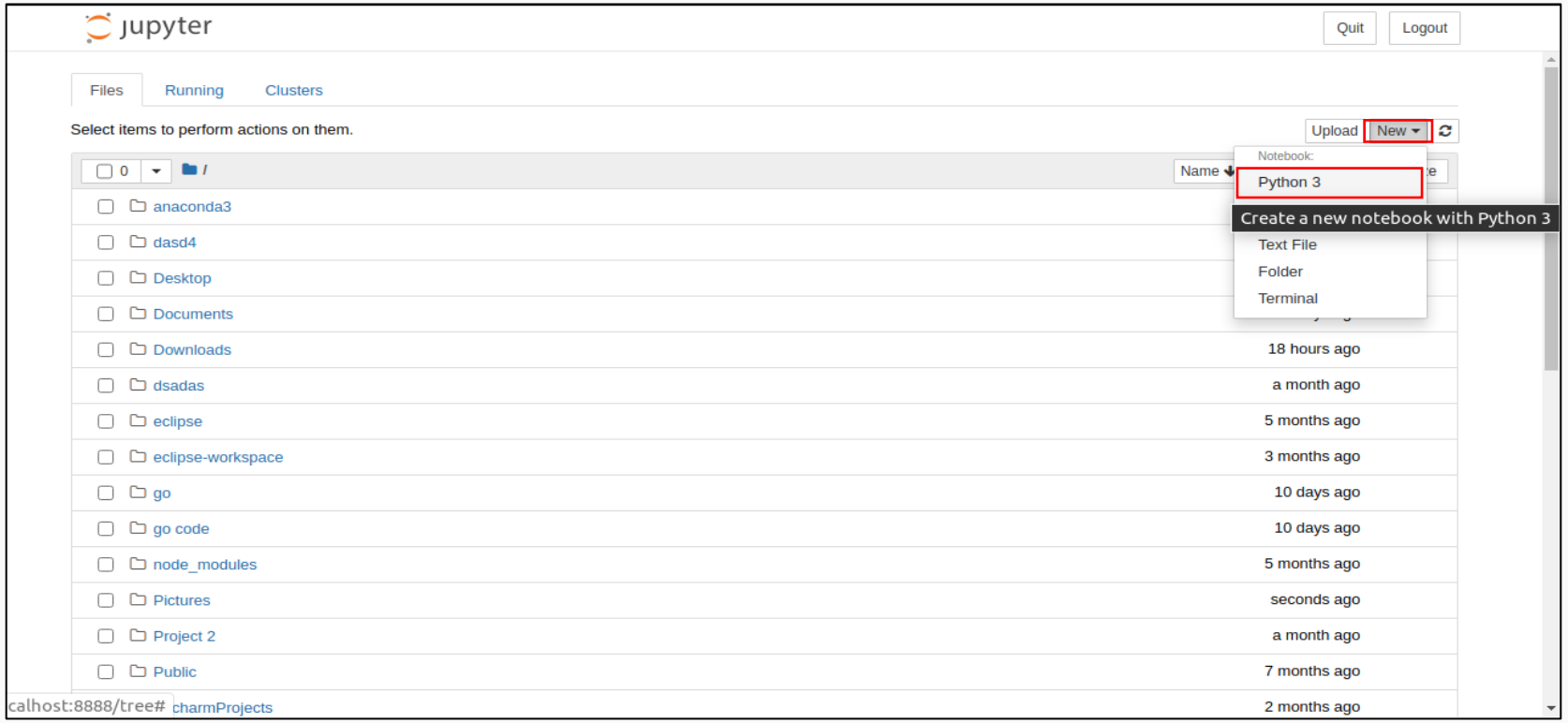
- To launch Jupyter Notebook, restart your terminal.
- You will see (base) written before your home directory name.
- Type “jupyter notebook” without the quotation marks and hit enter.
- After a few minutes, Jupyter Notebook will run on your default browser.



Meta Brains

Anaconda for Ubuntu (10/11)

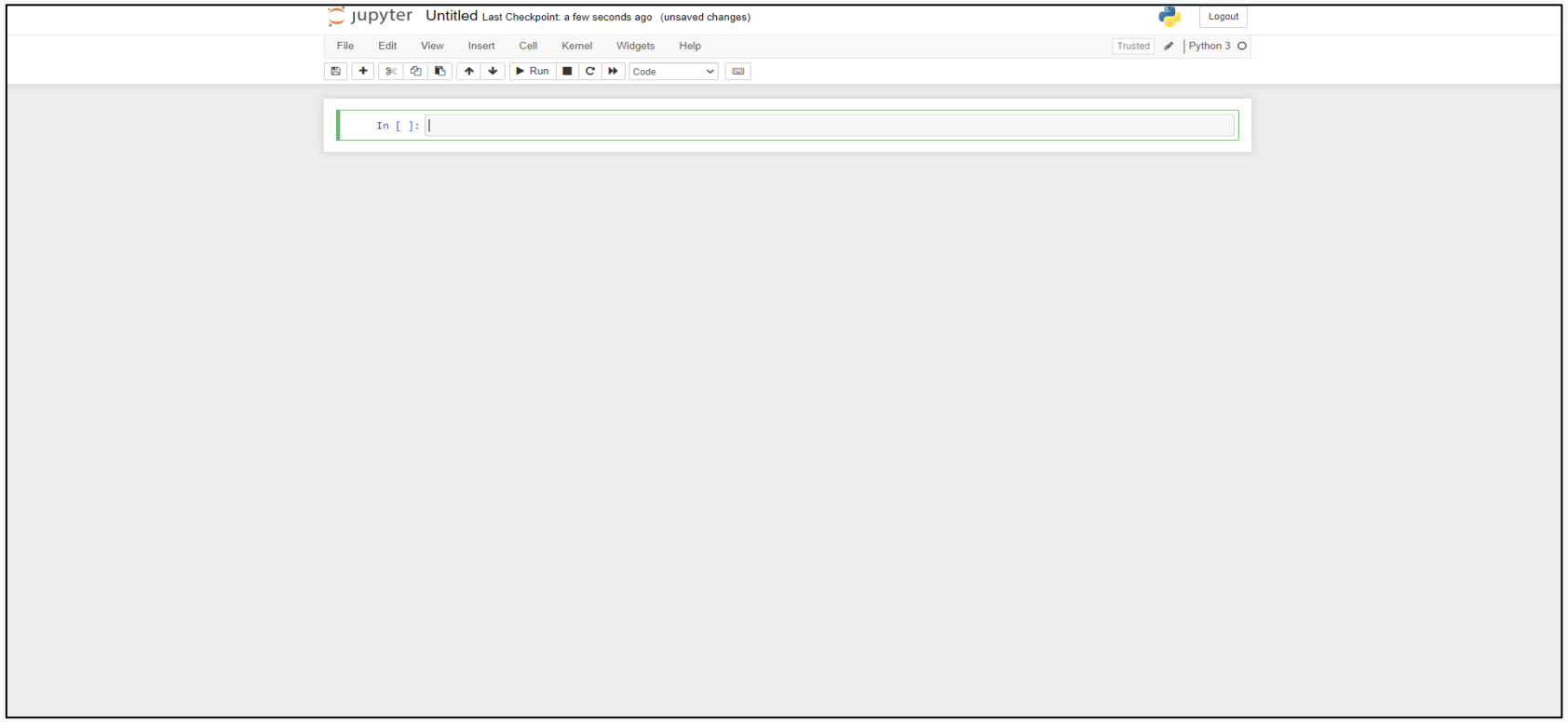
- Click on New -> Python3 to create a new Notebook.



Meta Brains

Anaconda for Ubuntu (11/11)

- This is how a Notebook looks like.



Meta Brains

- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Python libraries for Machine Learning

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

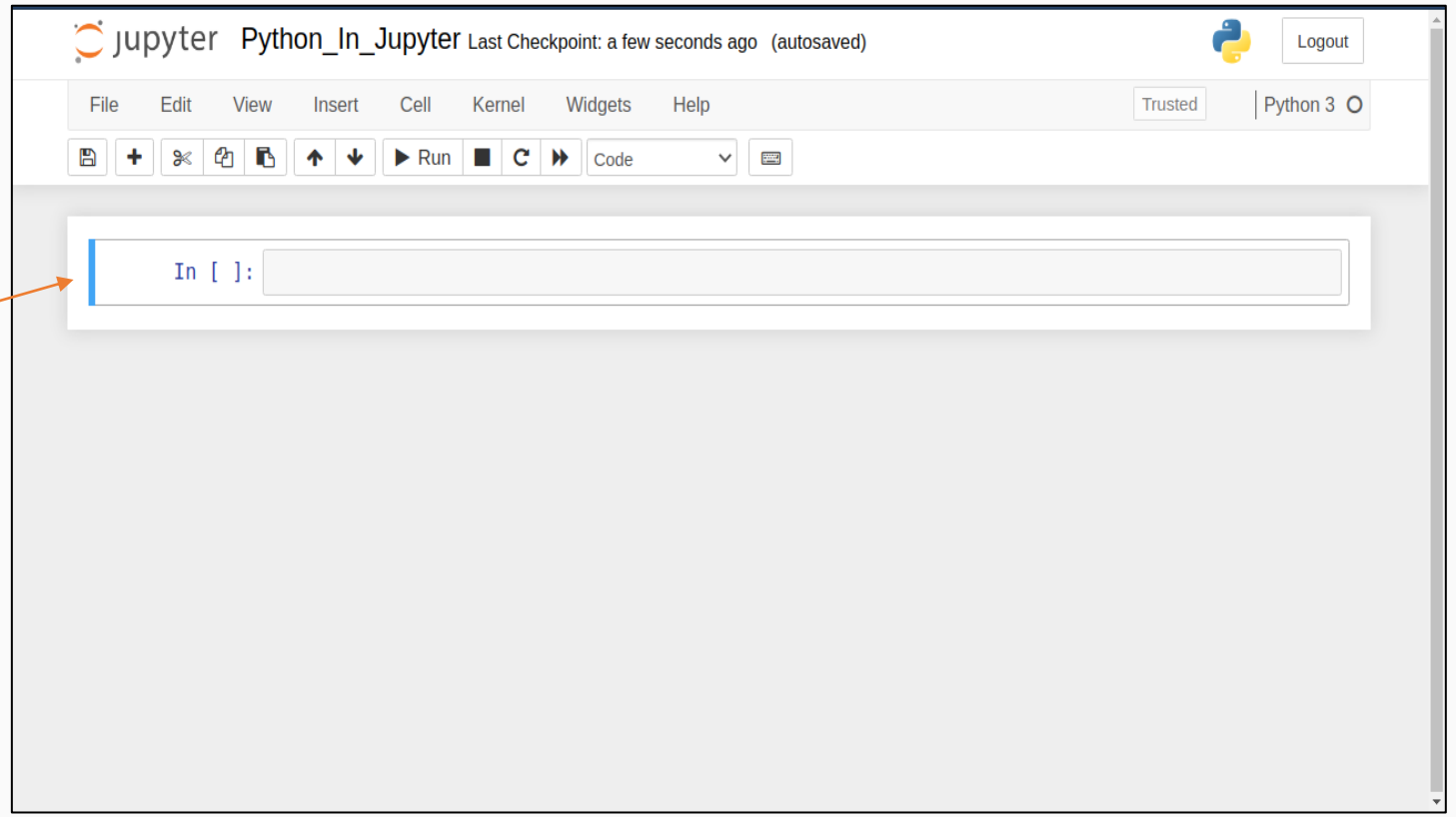
07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Implementing Python in Jupyter Notebook (1/2)

Upon launching Jupyter Notebook, this is what you will see!

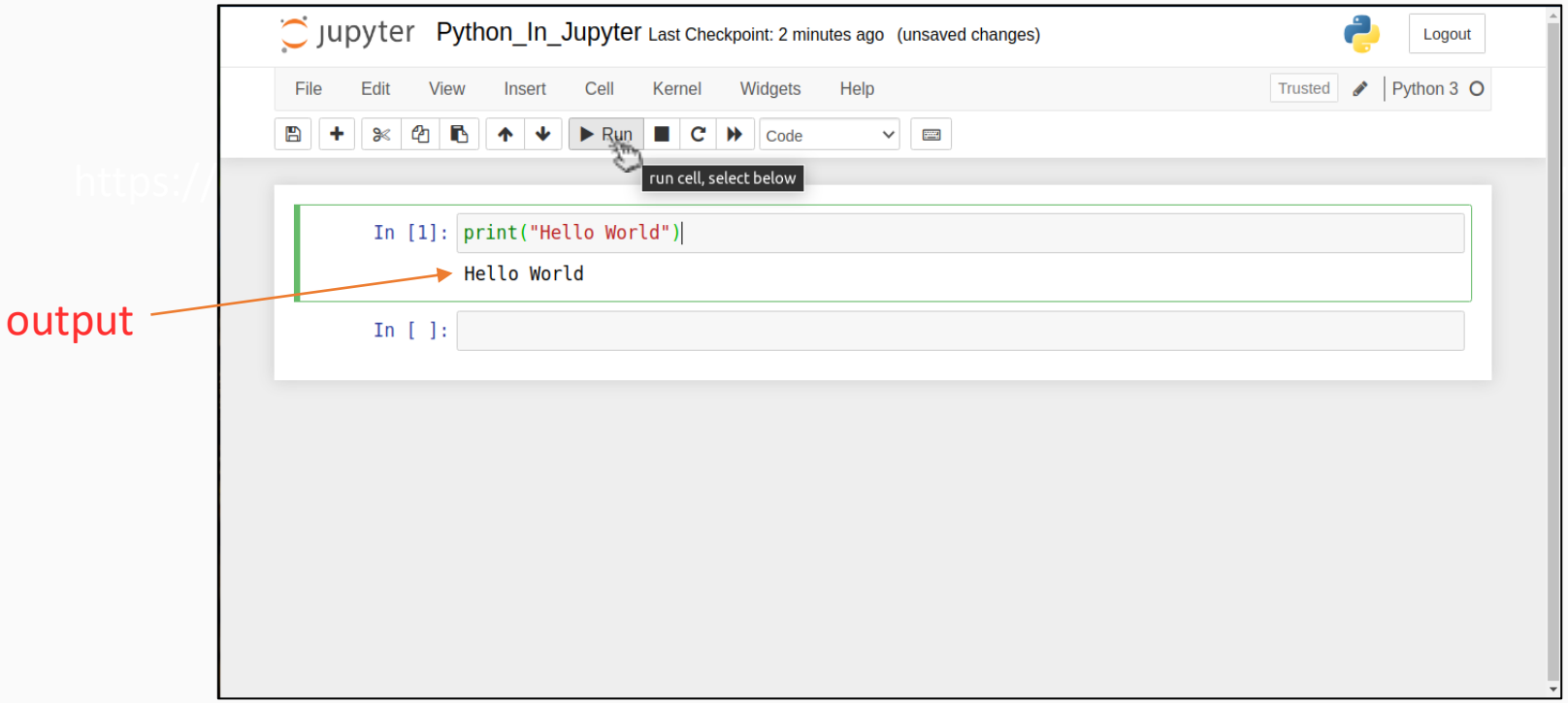


This is where we write our code, called a cell. You can have as many of them as you want.

Meta Brains

Implementing Python in Jupyter Notebook (2/2)

- Write your Python code inside a cell.
- Select the cell by clicking on it and then click on the Run button to execute the code.
- Output of a cell is produced right below it.



- Introduction to Machine Learning
- **Implementing ML Algorithms in Python**
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Python libraries for Machine Learning

03 Setting up Python

04 What is Jupyter?

05 Anaconda Installation for Windows OS

06 Anaconda Installation for Mac OS

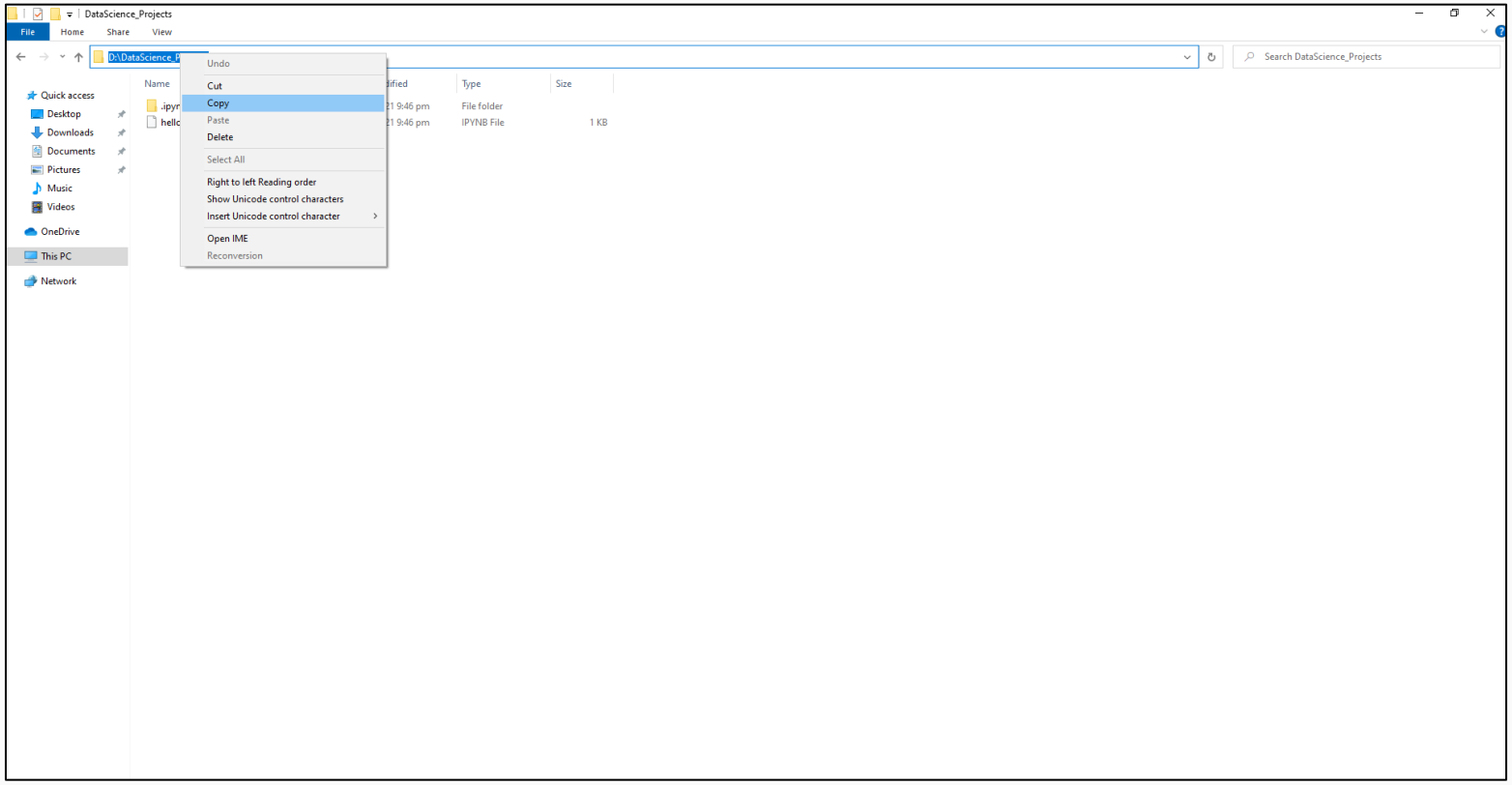
07 Anaconda Installation for Ubuntu OS

08 Implementing Python in Jupyter

09 Managing Directories in Jupyter Notebook

Managing Directories in Jupyter Notebook – Windows (1/7)

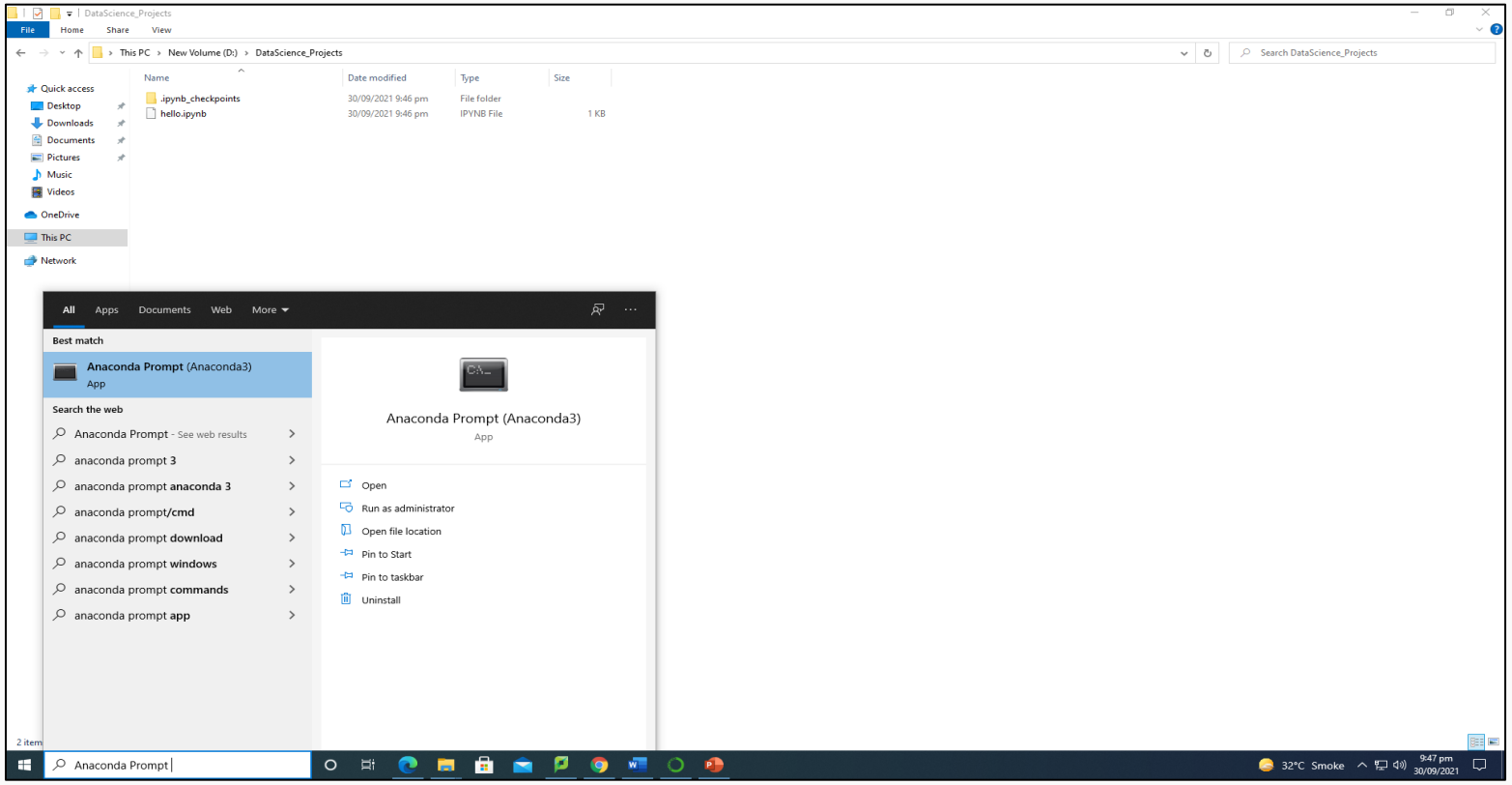
- To open Jupyter Notebook in a particular directory/folder in windows, copy the path of the directory.



Meta Brains

Managing Directories in Jupyter Notebook – Windows (2/7)

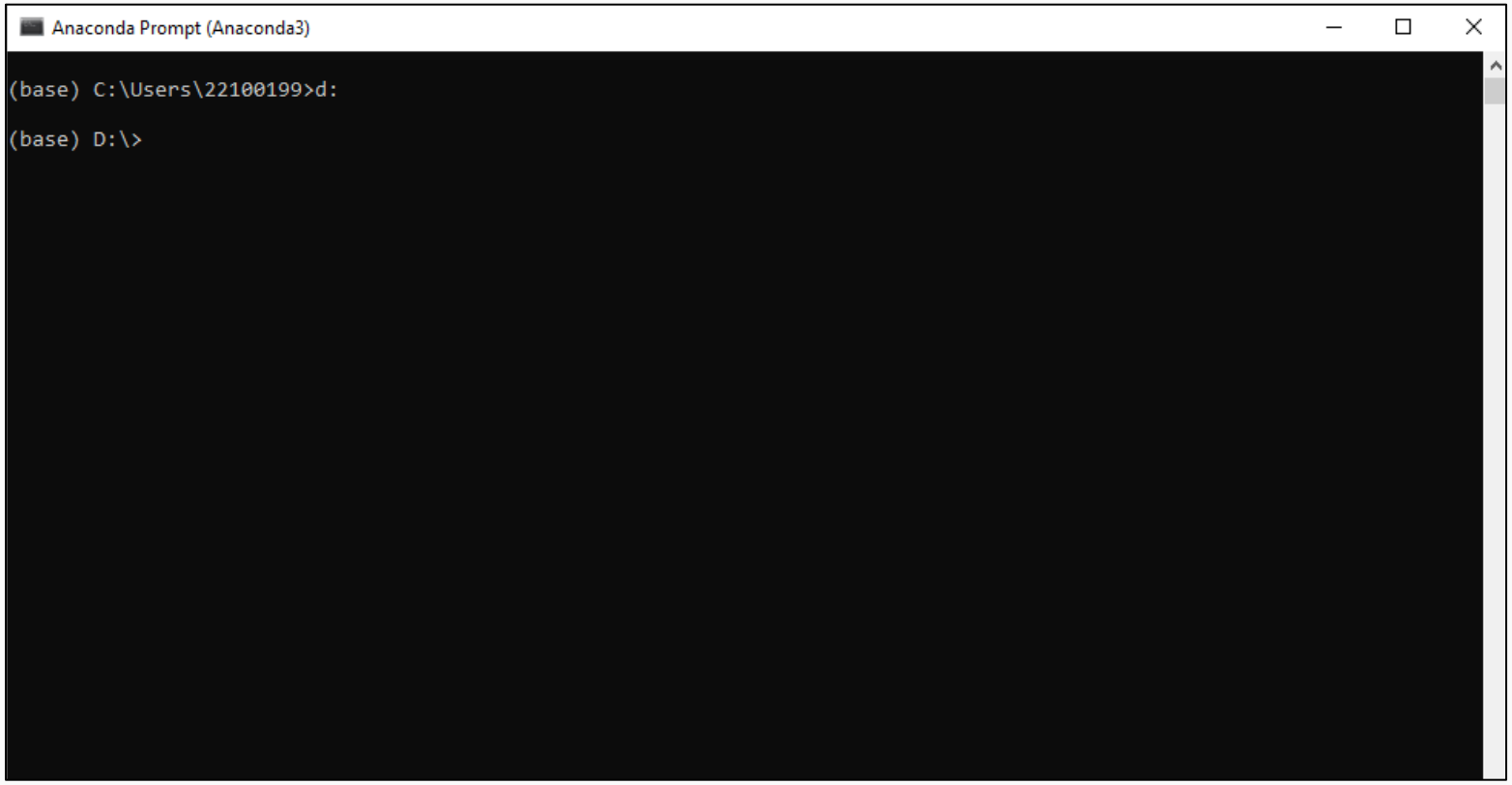
- In your search bar type Anaconda Prompt and launch it
  - Anaconda Prompt is the CLI of Anaconda



Meta Brains

Managing Directories in Jupyter Notebook – Windows (3/7)

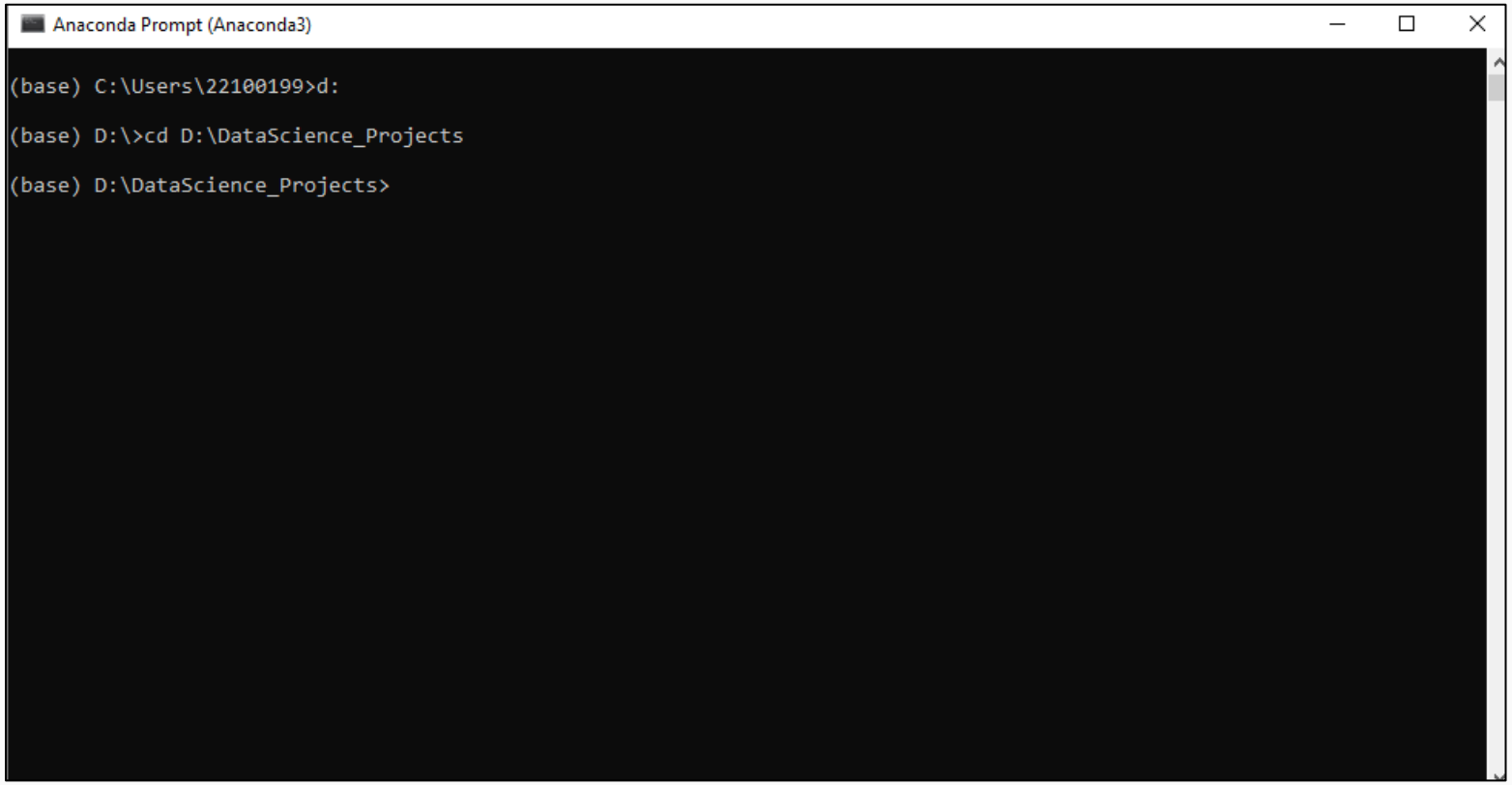
- Anaconda Prompt opens up in your C drive. Since we want to move to the D drive, we type d: and hit enter.



Meta Brains

Managing Directories in Jupyter Notebook – Windows (4/7)

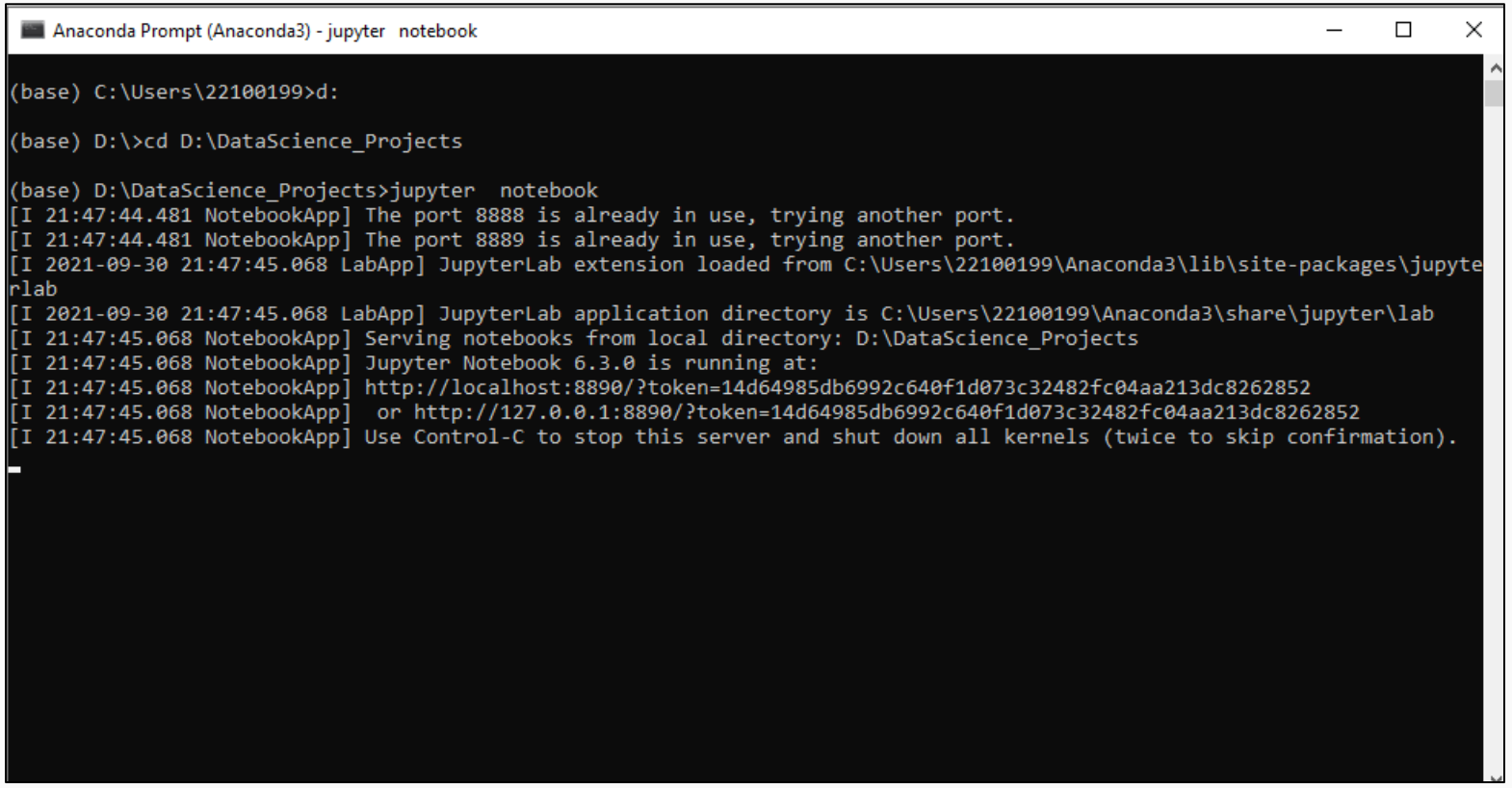
- Next, to switch to the desired directory we type **cd** and then **paste the path of the directory**.



Meta Brains

Managing Directories in Jupyter Notebook – Windows (5/7)

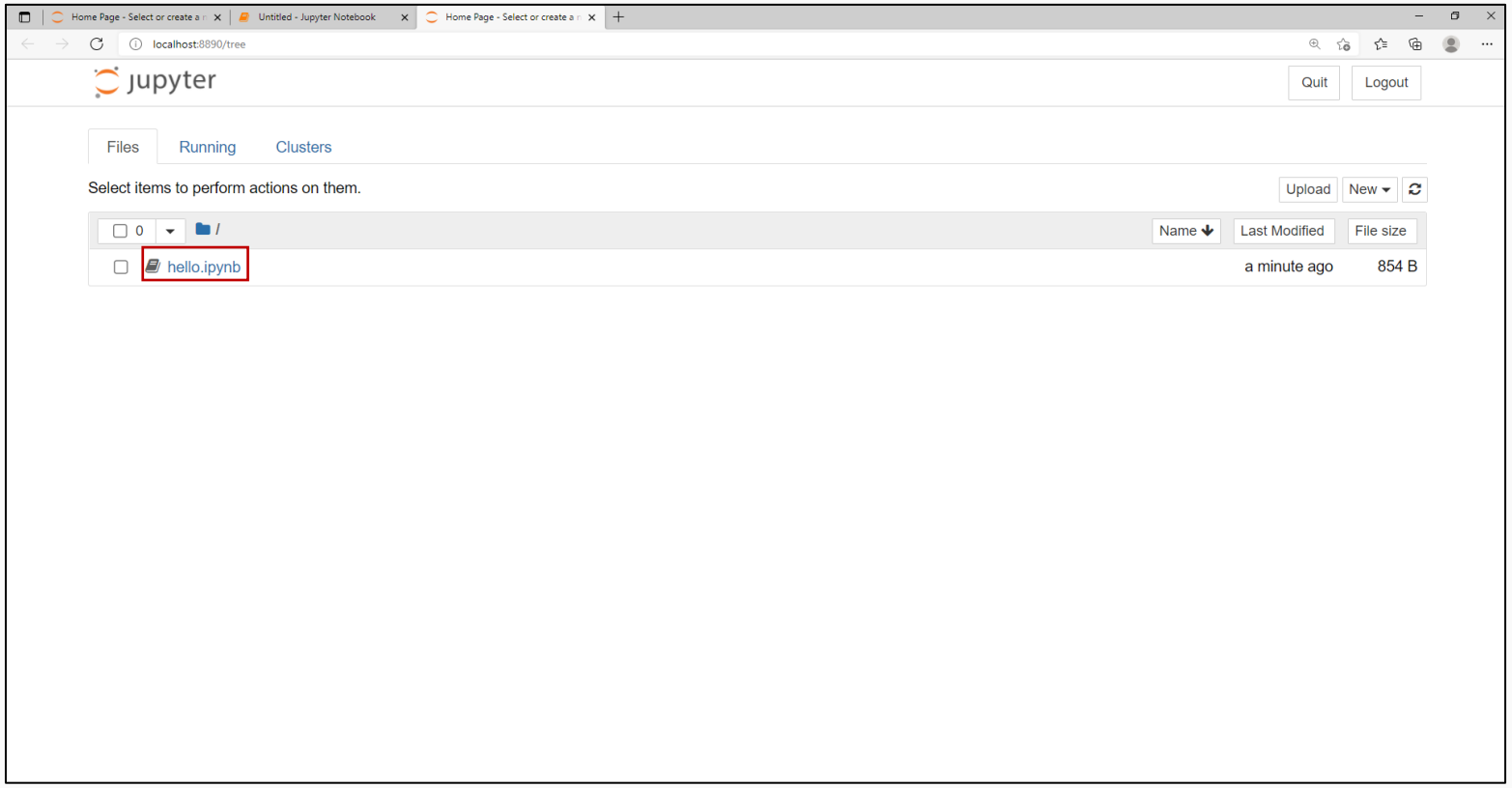
- Finally, we launch Jupyter Notebook by typing jupyter notebook.
- Jupyter Notebook takes a while to open.



Meta Brains

Managing Directories in Jupyter Notebook – Windows (6/7)

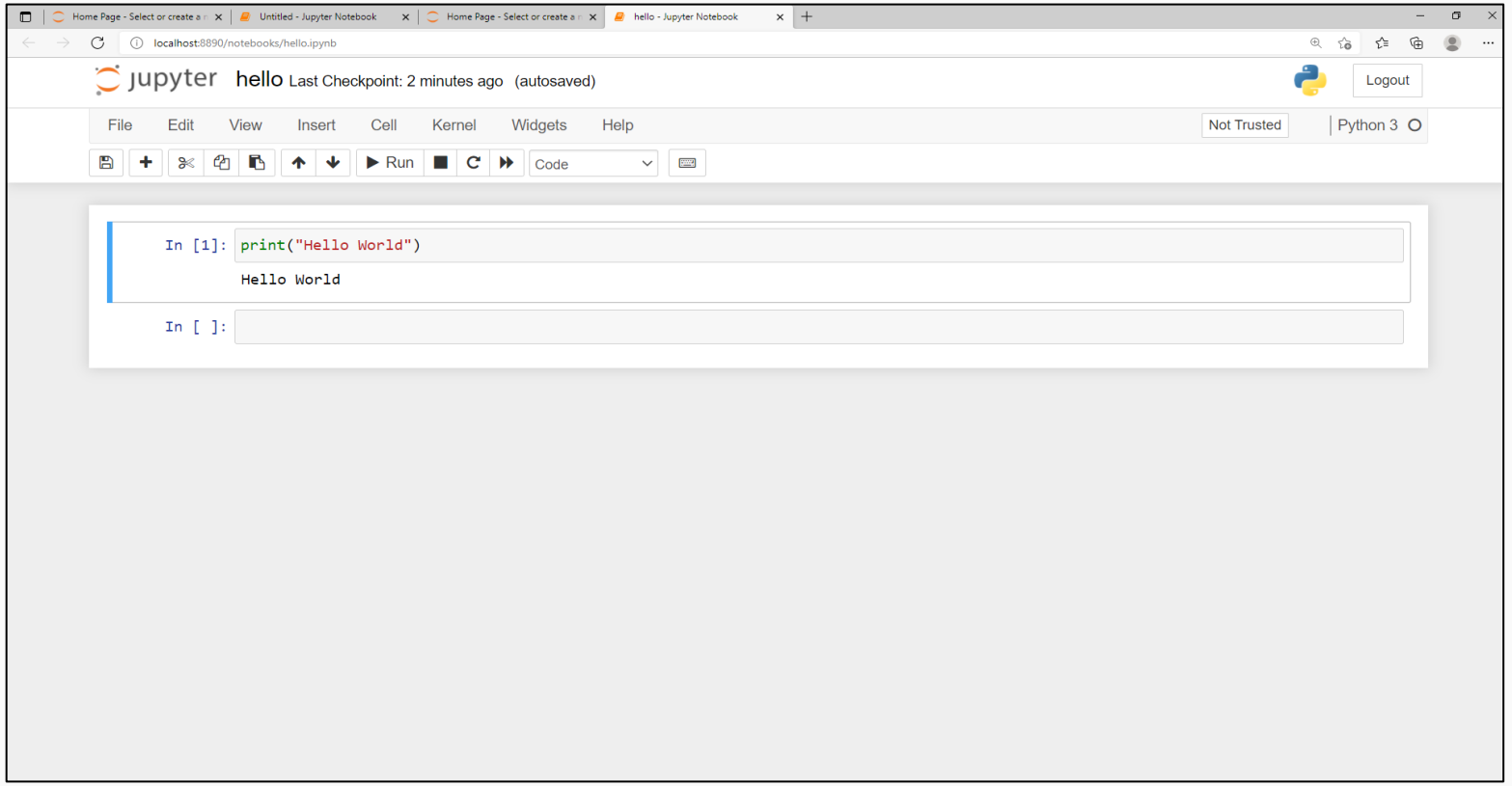
- Once it opens, we can either open an existing notebook or create a new one.
- In this case we open the existing notebook named hello.ipynb.



Meta Brains

Managing Directories in Jupyter Notebook – Windows (7/7)

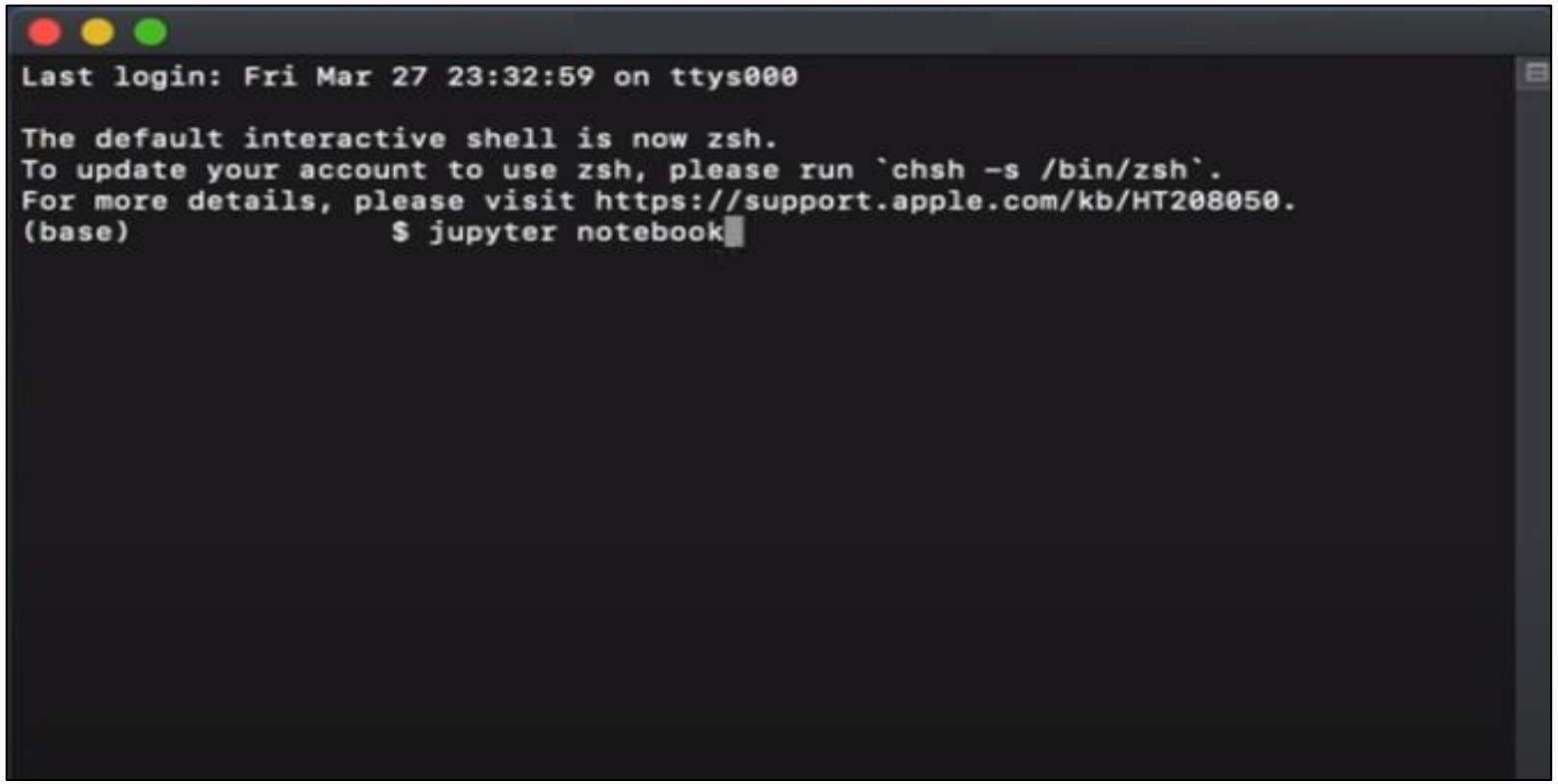
- This is how hello.ipynb looks like.



Meta Brains

Managing Directories in Jupyter Notebook – Mac (1/3)

- To open Jupyter Notebook in a particular directory/folder in Mac, open that directory in terminal.
- Type ‘jupyter notebook’ without the quotation marks and hit Enter.



Meta Brains

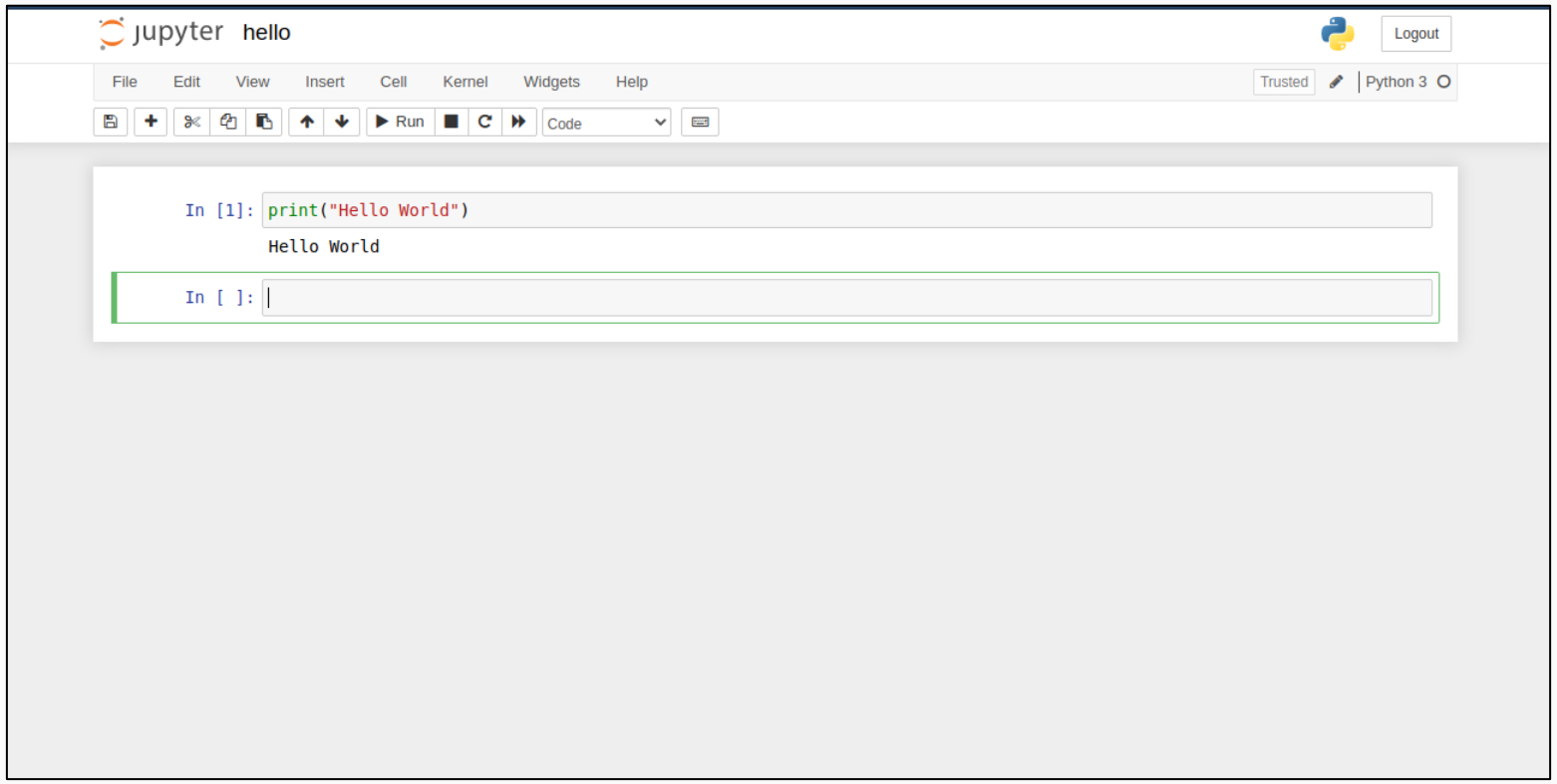
Managing Directories in Jupyter Notebook – Mac (2/3)

- Jupyter Notebook will launch in your browser.
- You can create a new Notebook or open an existing one.
- Let's open this 'hello.ipynb' file.

Meta Brains

Managing Directories in Jupyter Notebook – Mac (3/3)

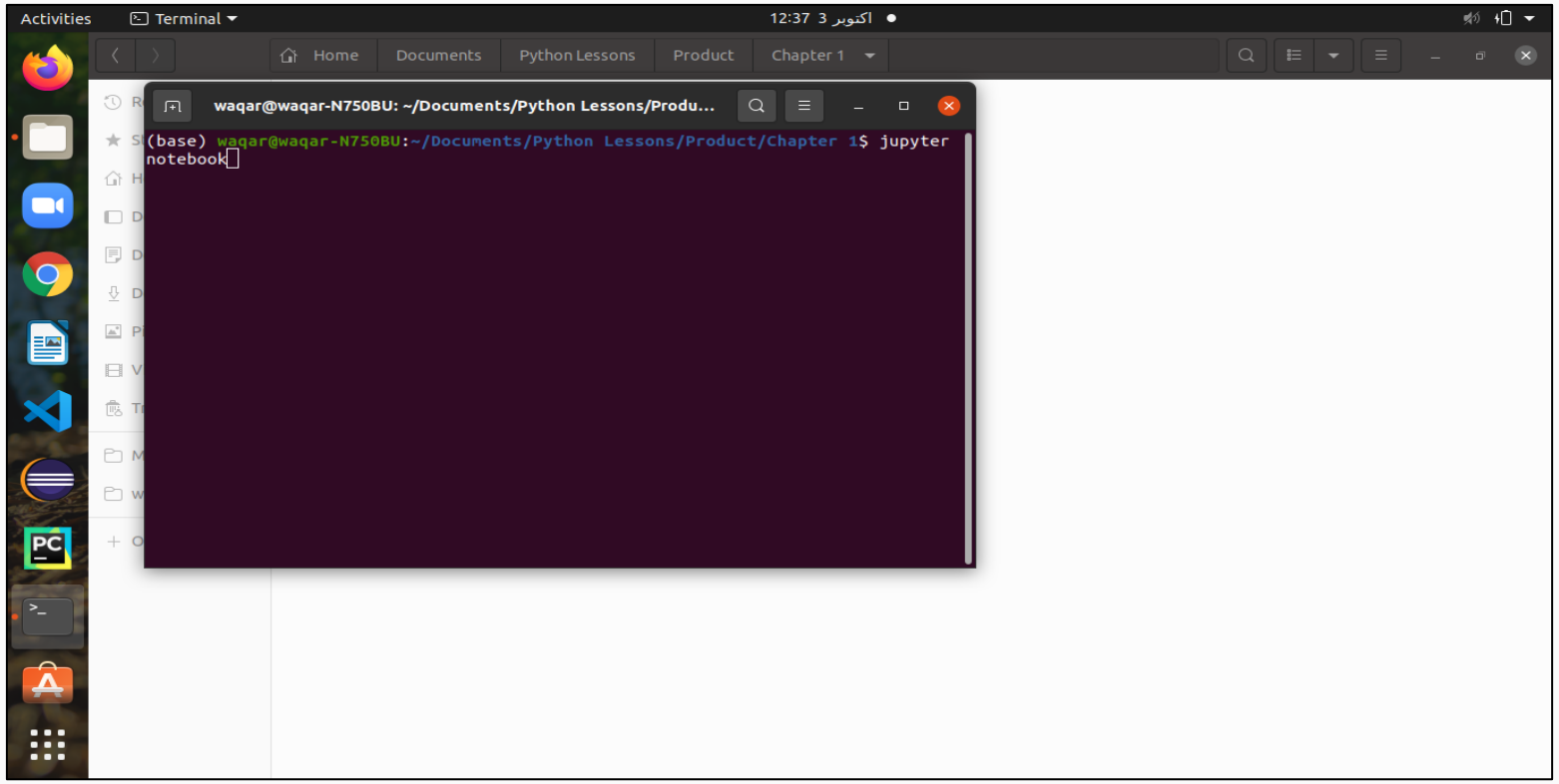
- ‘hello.ipynb’ contains a simple print statement.



Meta Brains

Managing Directories in Jupyter Notebook – Ubuntu (1/4)

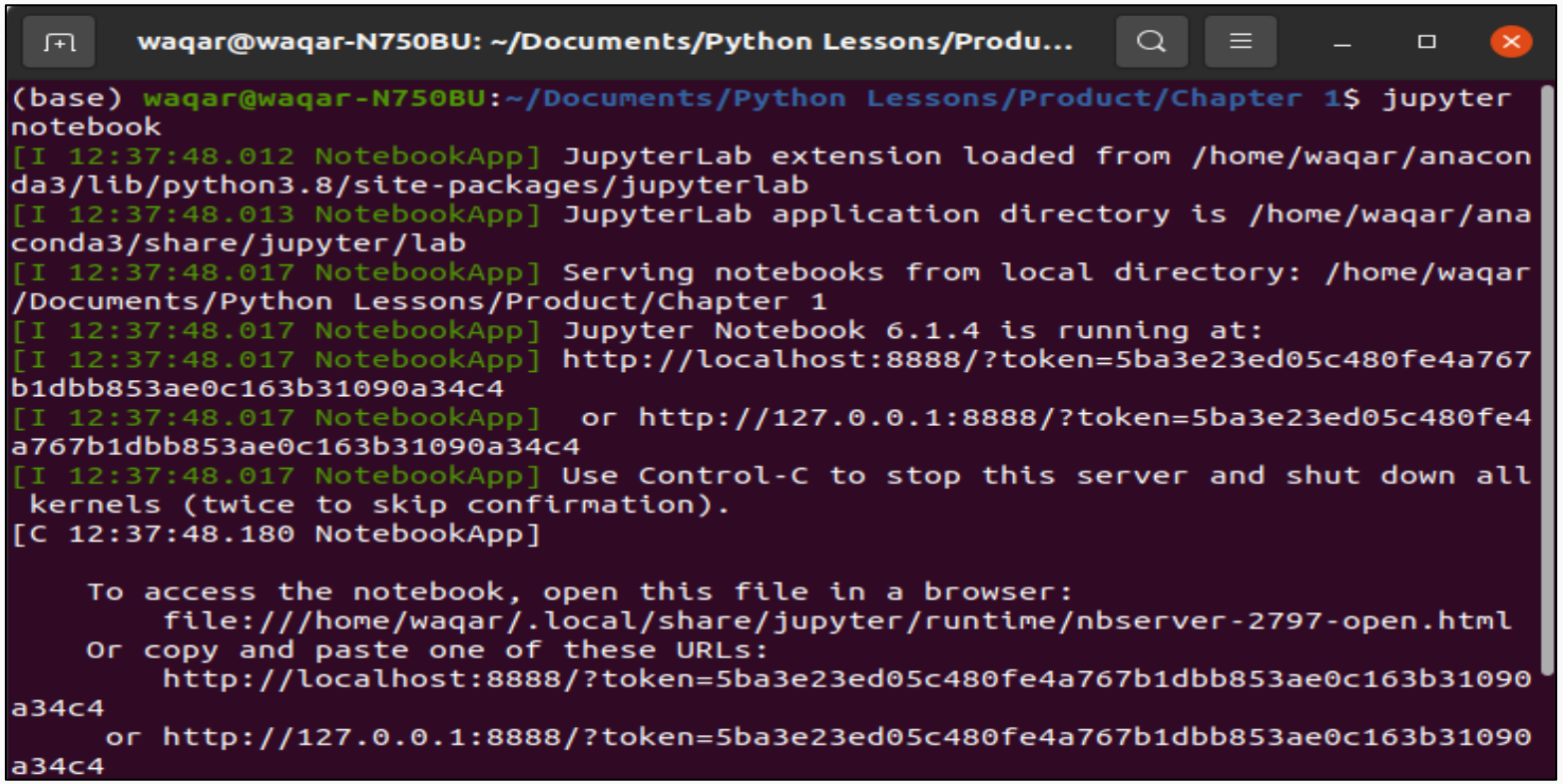
- To open Jupyter Notebook in a particular directory/folder in Ubuntu, go to that directory, right click and select ‘open in terminal’.
- Type ‘jupyter notebook’ without quotation marks and hit Enter.



Meta Brains

Managing Directories in Jupyter Notebook – Ubuntu (2/4)

- Wait for sometime.



Meta Brains

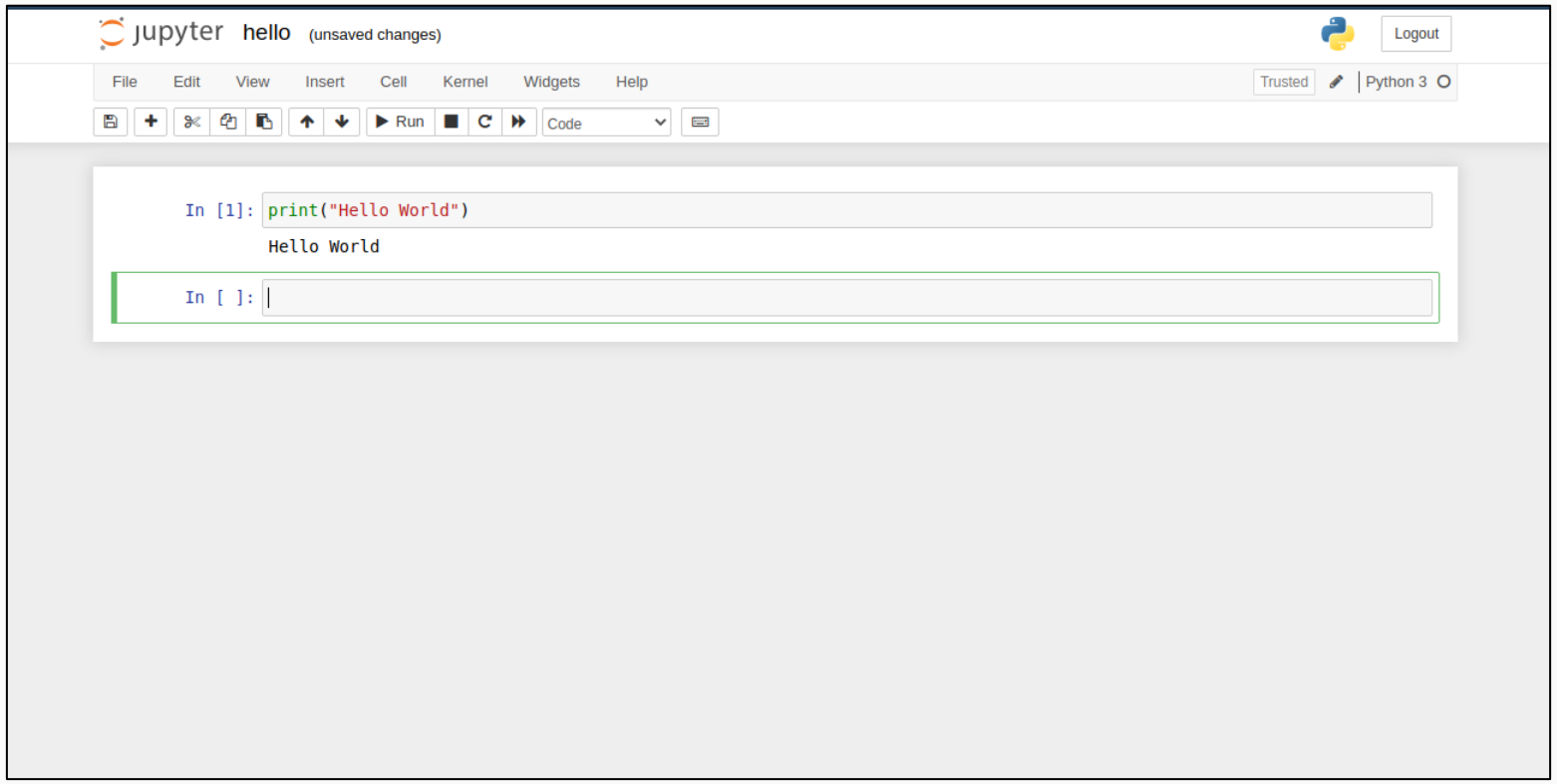
Managing Directories in Jupyter Notebook – Ubuntu (3/4)

- Jupyter Notebook will launch in your default browser.
- You can create a new Notebook or open an existing one.
- Let’s open this ‘hello.ipynb’ file.

Meta Brains

Managing Directories in Jupyter Notebook – Ubuntu (4/4)

- ‘hello.ipynb’ contains a simple print statement.



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- **Simple Linear Regression**
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to regression 01

02 How Does Linear Regression Work?

Line representation 03

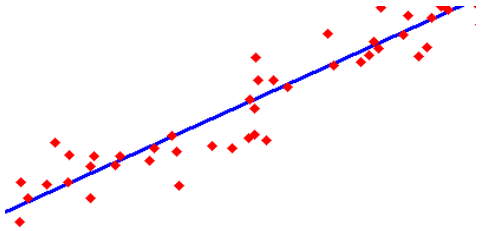
04 Implementation in python: Importing libraries & datasets

Implementation in python: Distribution of the data 05

06 Implementation in python: Creating a linear regression object

Regression

Regression is a statistical method, which predicts a relationship of a continuous outcome based on the value of one or more input variables.



Types of Regression

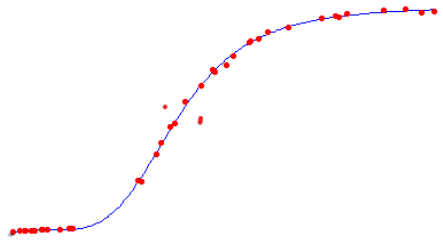
- **Linear Regression**

The relationship between input variables and output variable is linear.

1. **Simple linear Regression:** only single input variable is used to predict an output.
2. **Multiple linear regression**: more than one input variable is involved.

- **Nonlinear Regression**

Output is model by a function which is a nonlinear combination of the inputs.



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- **Simple Linear Regression**
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to regression 01

02 **How Does Linear Regression Work?**

Line representation 03

04 Implementation in python: Importing libraries & datasets

Implementation in python: Distribution of the data 05

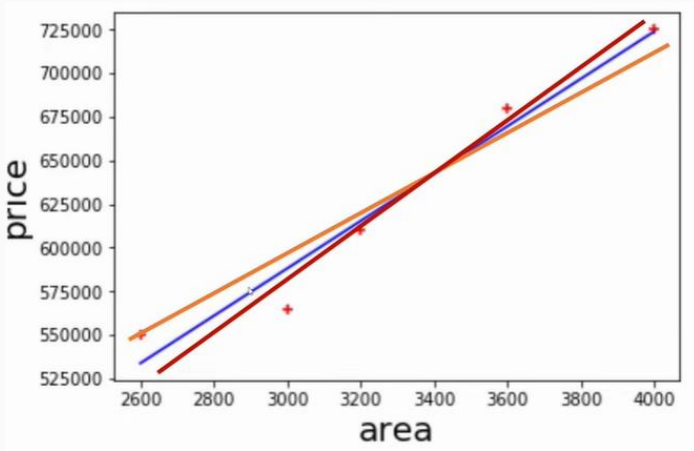
06 Implementation in python: Creating a linear regression object

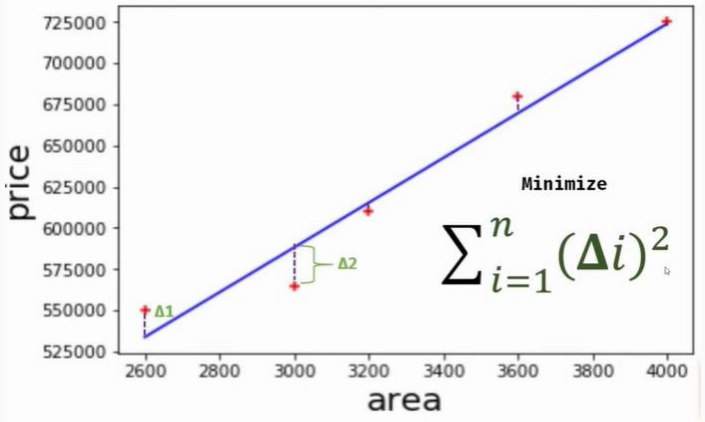
Simple Linear Regression

Working of simple Linear Regression

Let’s consider the following example to explain the working of linear regression

area	price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000





What is the price of a house having an area of 330 square feet?

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- **Simple Linear Regression**
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to regression 01

02 How Does Linear Regression Work?

Line representation **03**

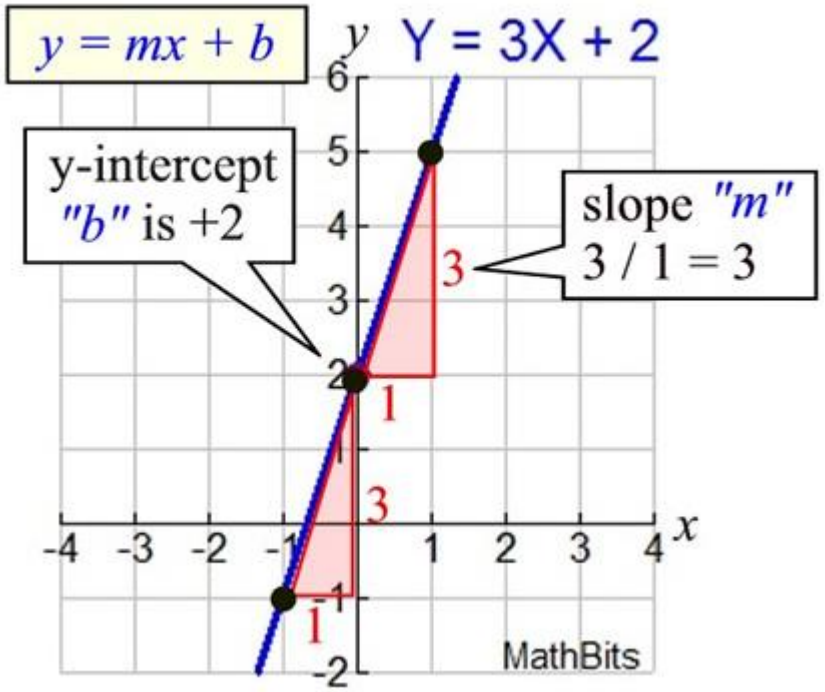
04 Implementation in python: Importing libraries & datasets

Implementation in python: Distribution of the data 05

06 Implementation in python: Creating a linear regression object

Simple Linear Regression

Line representation
Y = mx + b



Equation of line in terms of area and price

```
price = m * area + b
```

Dependent variable

Independent variable

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- **Simple Linear Regression**
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to regression

02 How Does Linear Regression Work?

03 Line representation

04 Implementation in python: Importing libraries & datasets

05 Implementation in python: Distribution of the data

06 Implementation in python: Creating a linear regression object

Simple Linear Regression

Simple Linear Regression in python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

```
df = pd.read_csv("homeprices.csv")
df
```

	Area	Price
0	2600	550000
1	3000	565000
2	3200	610000
3	3600	680000
4	4000	725000

Import the important libraries.

Import linear model from sklearn library.

Import the csv file of the data set.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- **Simple Linear Regression**
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to regression

02 How Does Linear Regression Work?

03 Line representation

04 Implementation in python: Importing libraries & datasets

05 Implementation in python: Distribution of the data

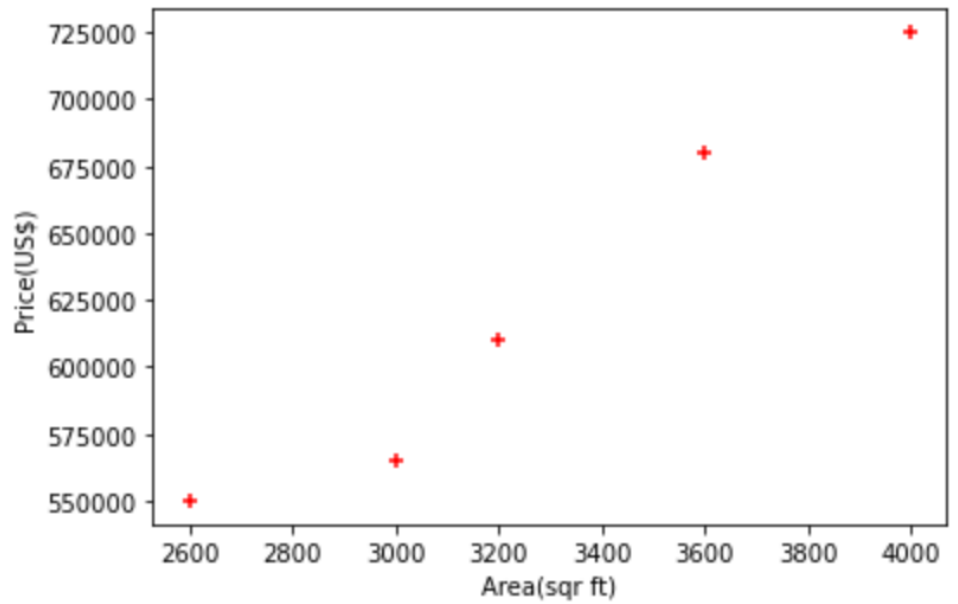
06 Implementation in python: Creating a linear regression object

Simple Linear Regression

Simple Linear Regression in python

```
plt.xlabel('Area(sqr ft)')
plt.ylabel('Price(US$)')
plt.scatter(df.Area, df.Price, color = 'red', marker='+')
```

```
<matplotlib.collections.PathCollection at 0x2a2b538cee0>
```



Set x and y labels. x label would be area and y label would be price.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- **Simple Linear Regression**
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to regression 01

02 How Does Linear Regression Work?

Line representation 03

04 Implementation in python: Importing libraries & datasets

Implementation in python: Distribution of the data 05

06 Implementation in python: Creating a linear regression object

Simple Linear Regression

Simple Linear Regression in python

```
In [4]: reg = LinearRegression()
        reg.fit(df[['Area']], df.Price)

Out[4]: LinearRegression()

In [5]: reg.predict([[3300]])

Out[5]: array([628715.75342466])

In [6]: reg.intercept_

Out[6]: 180616.43835616432

In [7]: reg.coef_

Out[7]: array([135.78767123])

In [8]: 135.788 * 3300 + 180616.438

Out[8]: 628716.838
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- **Multiple Linear Regression**
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Understanding Multiple linear regression 01

02 Implementation in python: Exploring the dataset

Implementation in python: Encoding Categorical Data 03

04 Implementation in python: Splitting data into Train and Test Sets

Implementation in python: Training the model on the Training set 05

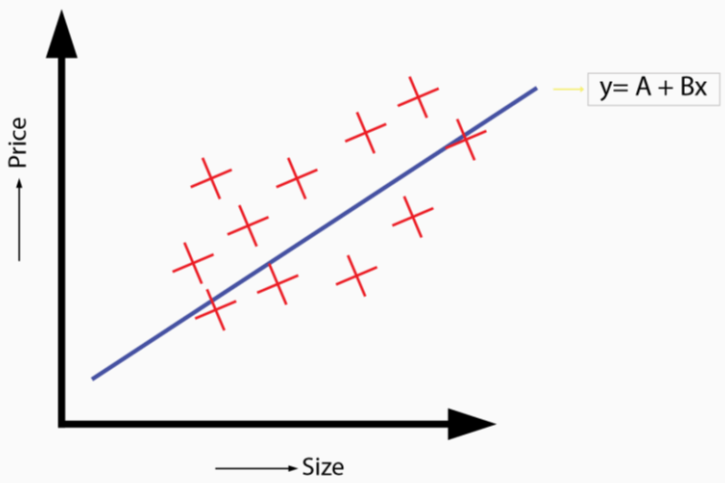
06 Implementation in python: Predicting the Test Set results

Evaluating the performance of the regression model 07

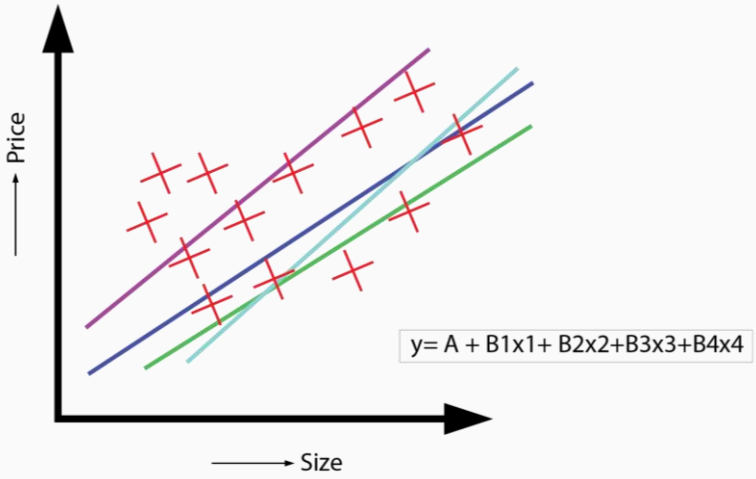
08 Root Mean Squared Error in Python

Multiple Linear Regression

Simple Linear Regression



Multiple Linear Regression



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- **Multiple Linear Regression**
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Understanding Multiple linear regression

02 **Implementation in python: Exploring the dataset**

03 Implementation in python: Encoding Categorical Data

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Training the model on the Training set

06 Implementation in python: Predicting the Test Set results

07 Evaluating the performance of the regression model

08 Root Mean Squared Error in Python

Multiple Linear Regression

Multiple Linear Regression in python

Dataset

R&D Spend	Administration	Marketing Spend	State	Profit
165349.2	136897.8	471784.1	New York	192261.83
162597.7	151377.59	443898.53	California	191792.06
153441.51	101145.55	407934.54	Florida	191050.39
144372.41	118671.85	383199.62	New York	182901.99
142107.34	91391.77	366168.42	Florida	166187.94
131876.9	99814.71	362861.36	New York	156991.12
134615.46	147198.87	127716.82	California	156122.51

Meta Brains

Multiple Linear Regression

Multiple Linear Regression in python

```
#import the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#reading the dataset
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
dataset.head(5)
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- **Multiple Linear Regression**
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Understanding Multiple linear regression

02 Implementation in python: Exploring the dataset

03 Implementation in python: Encoding Categorical Data

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Training the model on the Training set

06 Implementation in python: Predicting the Test Set results

07 Evaluating the performance of the regression model

08 Root Mean Squared Error in Python

Multiple Linear Regression

Encoding Categorical Data

X:

165349.2	136897.8	471784.1	New York
162597.7	151377.59	443898.53	California
153441.51	101145.55	407934.54	Florida
144372.41	118671.85	383199.62	New York
142107.34	91391.77	366168.42	Florida
131876.9	99814.71	362861.36	New York

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

Meta Brains

Multiple Linear Regression

Encoding Categorical Data
After Encoding, X:

0.0	0.0	1.0	165349.2	136897.8	471784.1
1.0	0.0	0.0	162597.7	151377.59	443898.53
0.0	1.0	0.0	153441.51	101145.55	407934.54
0.0	0.0	1.0	144372.41	118671.85	383199.62
0.0	1.0	0.0	142107.34	91391.77	366168.42
0.0	0.0	1.0	131876.9	99814.71	362861.36
1.0	0.0	0.0	134615.46	147198.87	127716.82

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- **Multiple Linear Regression**
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Understanding Multiple linear regression

02 Implementation in python: Exploring the dataset

03 Implementation in python: Encoding Categorical Data

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Training the model on the Training set

06 Implementation in python: Predicting the Test Set results

07 Evaluating the performance of the regression model

08 Root Mean Squared Error in Python

Multiple Linear Regression

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- **Multiple Linear Regression**
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Understanding Multiple linear regression

02 Implementation in python: Exploring the dataset

03 Implementation in python: Encoding Categorical Data

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Training the model on the Training set

06 Implementation in python: Predicting the Test Set results

07 Evaluating the performance of the regression model

08 Root Mean Squared Error in Python

Multiple Linear Regression

Training the Multiple Linear Regression model on the Training set

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- **Multiple Linear Regression**
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Understanding Multiple linear regression

02 Implementation in python: Exploring the dataset

03 Implementation in python: Encoding Categorical Data

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Training the model on the Training set

06 Implementation in python: Predicting the Test Set results

07 Evaluating the performance of the regression model

08 Root Mean Squared Error in Python

Multiple Linear Regression

Predicting the Test Set results

```
y_pred = regressor.predict(X_test)
```

```
df = pd.DataFrame({'Real Values':y_test, 'Predicted Values':y_pred})
df
```

	Real Values	Predicted Values
0	65200.33	69622.870630
1	14681.40	52846.107901
2	49490.75	62780.406794
3	96712.80	94377.300765
4	97427.84	100871.598266
5	77798.83	77589.217308
6	97483.56	99644.979787
7	132602.65	151447.661176
8	103282.38	106415.733395
9	105008.31	116820.266911

But, how to measure how well our model predicted?

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- **Multiple Linear Regression**
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Understanding Multiple linear regression

02 Implementation in python: Exploring the dataset

03 Implementation in python: Encoding Categorical Data

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Training the model on the Training set

06 Implementation in python: Predicting the Test Set results

07 Evaluating the performance of the regression model

08 Root Mean Squared Error in Python

Model Evaluation for Regression

Mean Absolute Error (MAE):

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Number of values

Actual value

Predictive value

Mean Squared Error (MSE):

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Meta Brains

Introduction to Machine Learning

Implementing ML Algorithms in Python

Simple Linear Regression

Multiple Linear Regression

Classification Algorithms: K-Nearest Neighbors

Classification Algorithms: Decision Tree

Classification Algorithms: Logistic regression

Clustering

Recommender System

Conclusion

01 Understanding Multiple linear regression

02 Implementation in python: Exploring the dataset

03 Implementation in python: Encoding Categorical Data

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Training the model on the Training set

06 Implementation in python: Predicting the Test Set results

07 Evaluating the performance of the regression model

08 Root Mean Squared Error in Python

Model Evaluation for Regression

Python Code for Root Mean Squared Error (RMSE):

```
# RMSE
from sklearn import metrics
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
14430.747692019655
```

Meta Brains

Quiz Time

Which of the following evaluation metrics can be used to evaluate a model while modeling a continuous output variable?

A) AUC-ROC
B) Accuracy
C) Logloss
D) Mean-Squared-Error

Quiz Time

Which of the following evaluation metrics can be used to evaluate a model while modeling a continuous output variable?

A) AUC-ROC
B) Accuracy
C) Logloss
D) Mean-Squared-Error

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to classification 01

02 K-Nearest Neighbors algorithm

Example of KNN 03

04 K-Nearest Neighbours (KNN) using python

Implementation in python: Importing required libraries 05

06 Implementation in python: Importing the dataset

Implementation in python: Splitting data into Train and Test Sets 07

08 Implementation in python: Feature Scaling

Implementation in python: Importing the KNN classifier 09

10 Implementation in python: Results prediction & Confusion matrix

Classification

Classification

Classification is a predictive modelling issue in machine learning where a class label is predicted for a given sample of input data.

Application of Classification

- Filtering spam emails
- Speech and handwriting recognition
- Biometric
- Fingerprint recognition etc.

Algorithms of Classification

- Neural Networks
- Regression
- K-nearest neighbor

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to classification 01

02 **K-Nearest Neighbors algorithm**

Example of KNN 03

04 K-Nearest Neighbours (KNN) using python

Implementation in python: Importing required libraries 05

06 Implementation in python: Importing the dataset

Implementation in python: Splitting data into Train and Test Sets 07

08 Implementation in python: Feature Scaling

Implementation in python: Importing the KNN classifier 09

10 Implementation in python: Results prediction & Confusion matrix

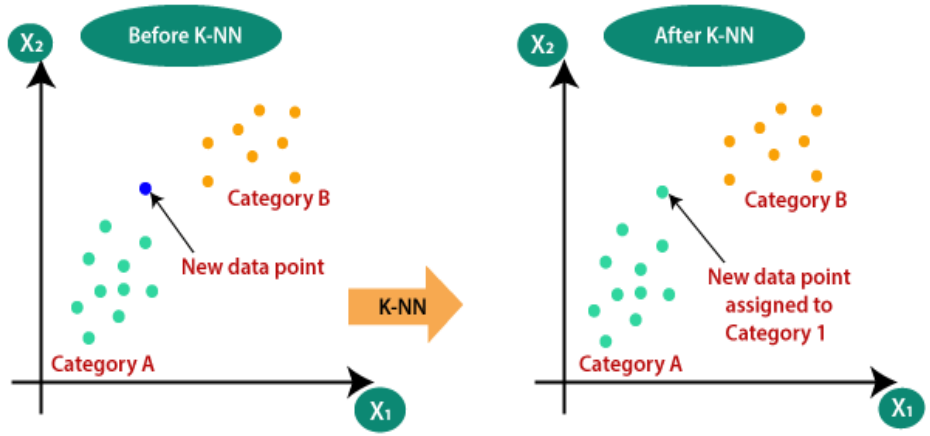
K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN)

The K-Nearest Neighbors algorithm is a classification method that utilizes a set of labelled points to learn how to identify new ones

Algorithm of KNN

- Choose a value for K.
- Determine the distance of an unknown point from all other points.
- Now in training data set choose the K which are nearest to the unknown data point.
- Now predict the response of unknown data point.



$$\text{Dis}\,(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 Implementation in python: Importing required libraries

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

10 Implementation in python: Results prediction & Confusion matrix

K-Nearest Neighbours (KNN)

K-Nearest Neighbours (KNN)

- ***Example***

KNN classifier will be use in this example to classify the input image into cat or dog.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 Implementation in python: Importing required libraries

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

10 Implementation in python: Results prediction & Confusion matrix

KNN Implementation

K-Nearest Neighbours (KNN) in python

Steps to implement KNN

- Data pre-processing like filtering
- Fitting KNN to the training set.
- Predict the test results.
- Test accuracy of the results.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 **Implementation in python: Importing required libraries**

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

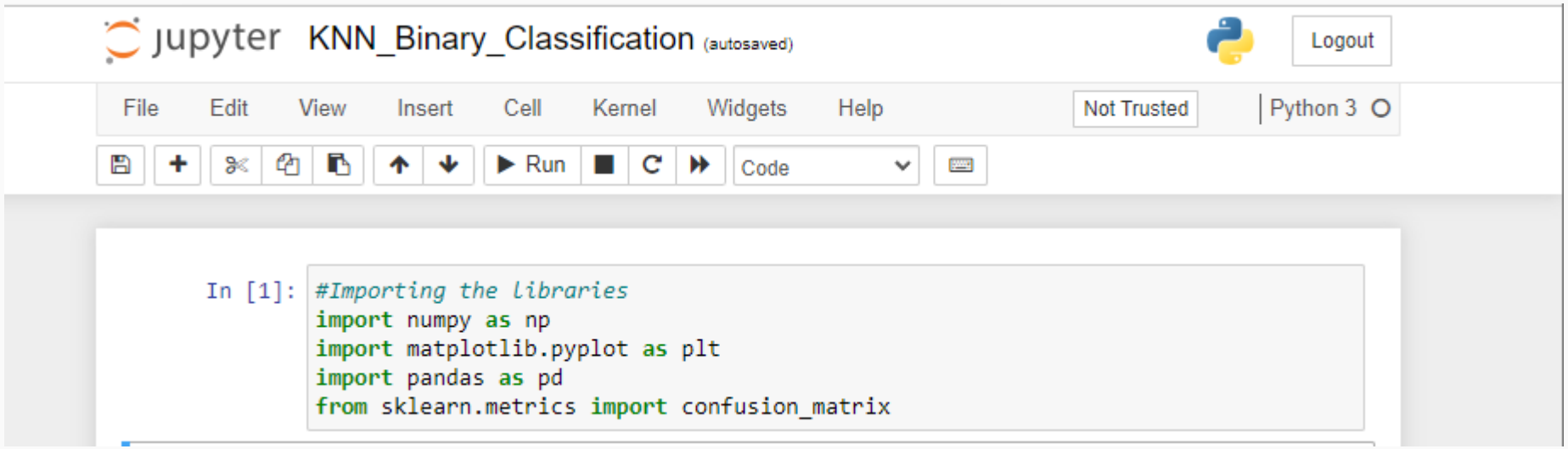
10 Implementation in python: Results prediction & Confusion matrix

KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 1

- First, we will import the required libraries.



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 Implementation in python: Importing required libraries

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

10 Implementation in python: Results prediction & Confusion matrix

KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 2

- Then we will import the given data sets.

```
# Importing the dataset
dataset = pd.read_csv('user data.csv')
X = dataset.iloc[:, 2:4].values
y = dataset.iloc[:, 4].values
```

- Extract independent and dependent variables – X & y

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 Implementation in python: Importing required libraries

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

10 Implementation in python: Results prediction & Confusion matrix

KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 3

- Split the data into testing and training data set.

```
In [3]: #Training and Testing Data (divide the data into two part)
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test =train_test_split(X,y,test_size= 0.25, random_s
```

- Import sklearn for pre-processing and scale the features.

```
In [4]: from sklearn.preprocessing import StandardScaler
        sc_X = StandardScaler()
        X_train = sc_X.fit_transform(X_train)
        X_test = sc_X.fit_transform(X_test)
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 Implementation in python: Importing required libraries

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

10 Implementation in python: Results prediction & Confusion matrix

KNN Implementation

K-Nearest Neighbours (KNN)in python

Unscaled data

```
print(X_train)
```

```
[[     44   39000]
 [     32  120000]
 [     38   50000]
 [     32  135000]
 [     52   21000]
 [     53  104000]
 [     39   42000]
 [     38   61000]
 [     36   50000]
 [     36   63000]
 [     35   25000]
 [     35   50000]
 [     42   73000]
 [     47   49000]
```

Scaled data

```
print(X_train)
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]
 [-0.30964085 -1.29261101]
 [-0.30964085 -0.5677824 ]
 [ 0.38358493  0.09905991]
 [ 0.8787462  -0.59677555]
 [ 2.06713324 -1.17663843]
 [ 1.07681071 -0.13288524]
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 Implementation in python: Importing required libraries

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

10 Implementation in python: Results prediction & Confusion matrix

KNN Implementation

K-Nearest Neighbours (KNN)in python

Implementation steps 4

- Import the KNN classifier from sklearn library

```
In [8]: from sklearn.neighbors import KNeighborsClassifier
        classifer=KNeighborsClassifier(n_neighbors=5, metric="minkowski", p=2)
        classifer.fit(X_train,y_train)

Out[8]: KNeighborsClassifier()
```

Define the different parameters:

- n_neighbors
- metric='minkowski’.
- p=2.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- **Classification Algorithms: K-Nearest Neighbors**
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to classification

02 K-Nearest Neighbors algorithm

03 Example of KNN

04 K-Nearest Neighbours (KNN) using python

05 Implementation in python: Importing required libraries

06 Implementation in python: Importing the dataset

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Feature Scaling

09 Implementation in python: Importing the KNN classifier

10 Implementation in python: Results prediction & Confusion matrix

KNN Implementation

K-Nearest Neighbours (KNN)in python

Prediction of results and confusion matrix

- Now will predict the results using predict() method
- Get the confusion matrix using y_pred, y_test values

```
y_pred = classifer.predict(X_test)

# Making the Confusion Matrix
cm = confusion_matrix(y_pred, y_test)
print(cm)
```

```
[[64  3]
 [ 4 29]]
```

Correct predictions= 64+29 = 93
Incorrect predictions= 3+4 = 7

Meta Brains

Quiz Time

Classification is-

A. Unsupervised learning
B. Reinforcement learning
C. Supervised learning
D. None

Meta Brains

Quiz Time

Classification is-

A. Unsupervised learning
B. Reinforcement learning
C. Supervised learning
D. None

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to decision trees 01

02 What is Entropy?

Exploring the dataset 03

04 Decision tree structure

Implementation in python: Importing libraries & datasets 05

06 Implementation in python: Encoding Categorical Data

Implementation in python: Splitting data into Train and Test Sets 07

08 Implementation in python: Results prediction & Accuracy

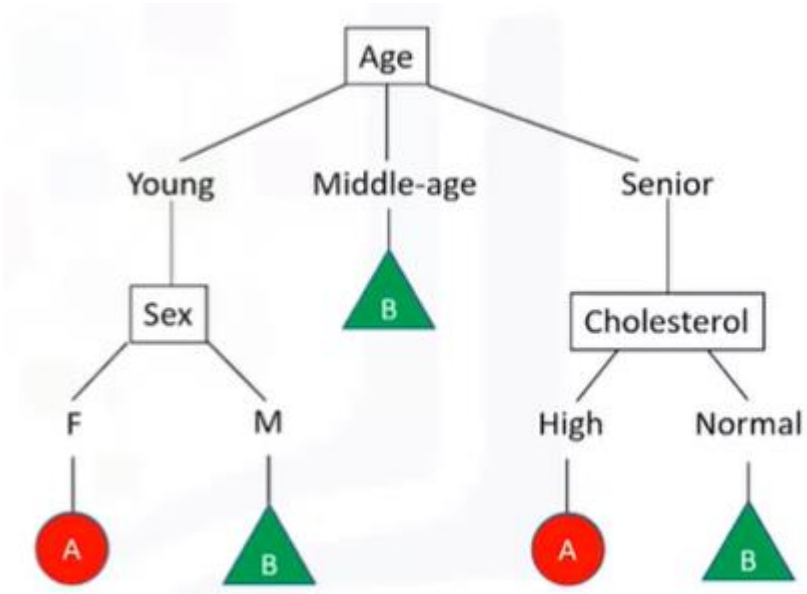
Decision Tree

What is decision tree?

Decision trees are a kind of Supervised Machine Learning in which data is continually segmented based on a parameter.

Learning algorithm of decision tree

- Choose a feature from the given data set.
- Determine the importance of feature in segmenting of data.
- Data segmenting depend upon the best attribute.
- Now repeat the above steps.



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to decision trees 01

02 **What is Entropy?**

Exploring the dataset 03

04 Decision tree structure

Implementation in python: Importing libraries & datasets 05

06 Implementation in python: Encoding Categorical Data

Implementation in python: Splitting data into Train and Test Sets 07

08 Implementation in python: Results prediction & Accuracy

Decision Tree

Entropy

It is the measure of the amount of uncertainty or randomness in data.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

P(x) is a probability function of the event x to take place

Lowest value: 0 (no randomness)
Highest vale: 1 (high randomness)

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to decision trees

02 What is Entropy?

03 Exploring the dataset

04 Decision tree structure

05 Implementation in python: Importing libraries & datasets

06 Implementation in python: Encoding Categorical Data

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Results prediction & Accuracy

Decision Tree Implementation

Building a Decision Tree in python

Data set

```
In [12]: data.head(10)
```

Out[12]:

	company	job	degree	salary_more_then_100k
0	company	job	degree	salary_more_then_100k
1	google	sale exective	bachelaors	0
2	google	sale exective	masters	0
3	google	business manager	bachelaors	1
4	google	business manager	masters	1
5	google	computer programmer	bachelaors	0
6	google	computer programmer	masters	1
7	abc pharma	sale exective	masters	0
8	abc pharma	computer programmer	bachelaors	0
9	abc pharma	business manager	bachelaors	0

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to decision trees

02 What is Entropy?

03 Exploring the dataset

04 Decision tree structure

05 Implementation in python: Importing libraries & datasets

06 Implementation in python: Encoding Categorical Data

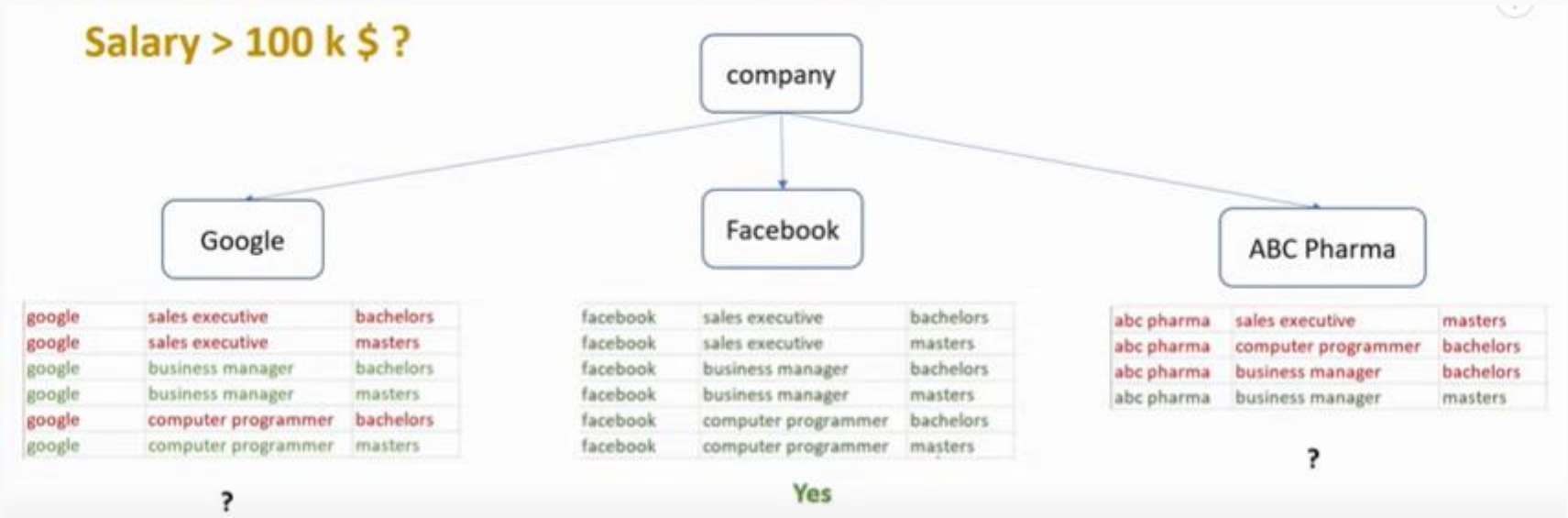
07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Results prediction & Accuracy

Decision Tree Implementation

Building a Decision Tree in python

Decision tree



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to decision trees 01

02 What is Entropy?

Exploring the dataset 03

04 Decision tree structure

Implementation in python: Importing libraries & datasets **05**

06 Implementation in python: Encoding Categorical Data

Implementation in python: Splitting data into Train and Test Sets 07

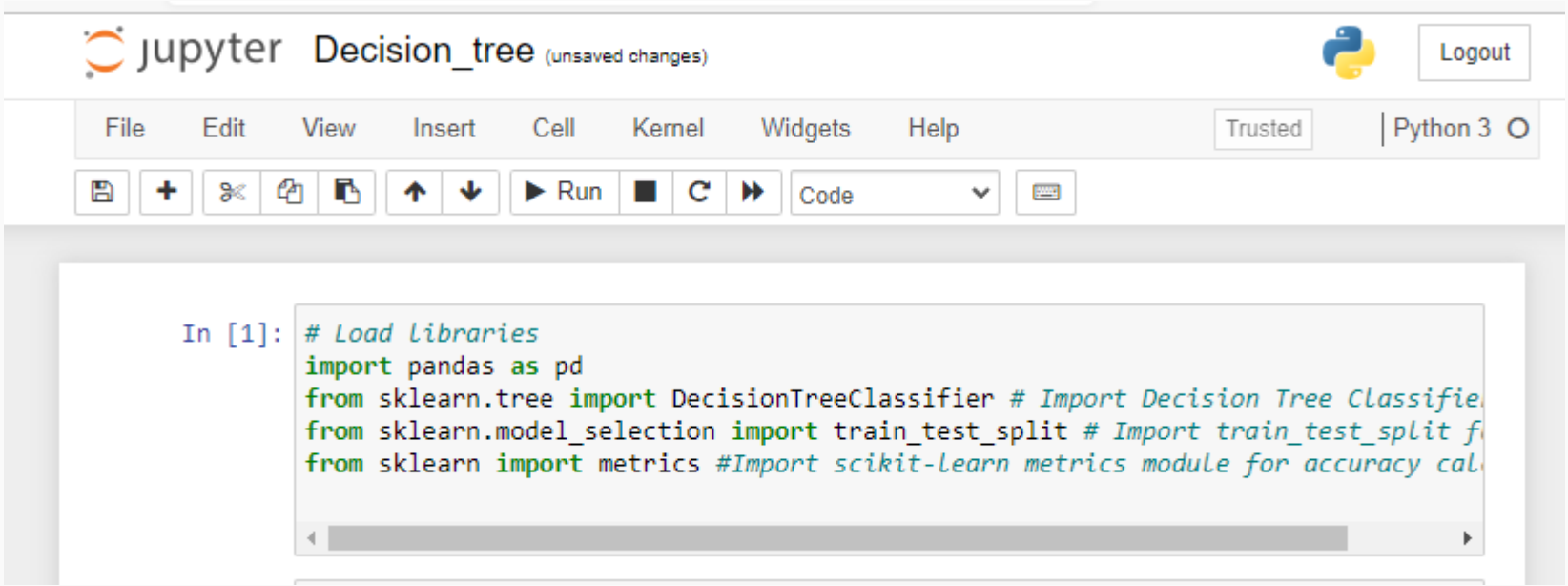
08 Implementation in python: Results prediction & Accuracy

Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 1

- *Import the required libraries*



Meta Brains

Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 2

- *Define the data variables and import the data file*

```
In [2]: #Working with the dataset here

        col_names = ['company', 'job', 'degree', 'salary_more_then_100k']# load dataset
        data = pd.read_csv("salaries.csv", header=None, names=col_names)
```

Meta Brains

Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 3

- Split the data into features (Inputs - X) and Target variable (output y)

```
In [19]: #Split the dataset in features and target variable
         feature_cols = ['company','job','degree']
         X = data[feature_cols]
         y = data['salary_more_then_100k']
```

- Target varible => Salary_more_then100k
- Defining Features for salary => company, job, degree

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to decision trees

02 What is Entropy?

03 Exploring the dataset

04 Decision tree structure

05 Implementation in python: Importing libraries & datasets

06 Implementation in python: Encoding Categorical Data

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Results prediction & Accuracy

Meta Brains

Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 4

- Encode the categorical features into numerical values
- Use ordinal encoder

```
In [4]: # Import label encoder
        from sklearn import preprocessing
        # label_encoder object knows how to understand word labels.
        label_encoder = preprocessing.LabelEncoder()
        # Encode labels in column
        data['company']= label_encoder.fit_transform(data['company'])
        data['job']= label_encoder.fit_transform(data['job'])
        data['degree']= label_encoder.fit_transform(data['degree'])
        print(data.head())
```

```
   company  job  degree  salary_more_then_100k
0        1    2       1  salary_more_then_100k
1        3    3       0                      0
2        3    3       2                      0
3        3    0       0                      1
4        3    0       2                      1
```

Meta Brains

Decision Tree Implementation

Building a Decision Tree in python

```
print(X)
```

	company	job	degree
0	1	2	1
1	3	3	0
2	3	3	2
3	3	0	0
4	3	0	2
5	3	1	0
6	3	1	2
7	0	3	2
8	0	1	0
9	0	0	0
10	0	0	2
11	2	3	0
12	2	3	2
13	2	0	0
14	2	0	2
15	2	1	0
16	2	1	2

```
In [20]: print(y)
```

```
0     salary_more_then_100k
1                         0
2                         0
3                         1
4                         1
5                         0
6                         1
7                         0
8                         0
9                         0
10                        1
11                        1
12                        1
13                        1
14                        1
15                        1
16                        1
Name: salary_more_then_100k, dtype: object
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

Introduction to decision trees 01

02 What is Entropy?

Exploring the dataset 03

04 Decision tree structure

Implementation in python: Importing libraries & datasets 05

06 Implementation in python: Encoding Categorical Data

Implementation in python: Splitting data into Train and Test Sets **07**

08 Implementation in python: Results prediction & Accuracy

Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 5

- Split the data into training and testing sets

```
In [7]: X_train, X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=100)
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- **Classification Algorithms: Decision Tree**
- Classification Algorithms: Logistic regression
- Clustering
- Recommender System
- Conclusion

01 Introduction to decision trees

02 What is Entropy?

03 Exploring the dataset

04 Decision tree structure

05 Implementation in python: Importing libraries & datasets

06 Implementation in python: Encoding Categorical Data

07 Implementation in python: Splitting data into Train and Test Sets

08 Implementation in python: Results prediction & Accuracy

Decision Tree Implementation

Building a Decision Tree in python

Implementation steps 6

- ***Creating Decision Tree***
- ***Use entropy to train the model***

```
In [24]: # Create Decision Tree classifier object using entropy
         clf_entropy = DecisionTreeClassifier(criterion="entropy", max_depth=3)


         # Train Decision Tree Classifier
         clf_entropy = clf_entropy.fit(X_train,y_train)
```

- ***Predict the results & Find the accuracy of the model***

```
In [8]: # Create Decision Tree classifier object using entropy
        clf_entropy = DecisionTreeClassifier(criterion="entropy", max_depth=8)

        # Train Decision Tree Classifier
        clf_entropy = clf_entropy.fit(X_train,y_train)

        #Predict the response for test dataset
        y_pred = clf_entropy.predict(X_test)


        print("Accuracy:", metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.75
```

Meta Brains

Quiz Time

Decision Trees can be used for Classification Tasks.

a) True
b) False

Quiz Time

Decision Trees can be used for Classification Tasks.

a) True
b) False

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion

Introduction 01

02 Implementation steps

Implementation in python: Importing libraries & datasets 03

04 Implementation in python: Splitting data into Train and Test Sets

Implementation in python: Pre-processing 05

06 Implementation in python: Training the model

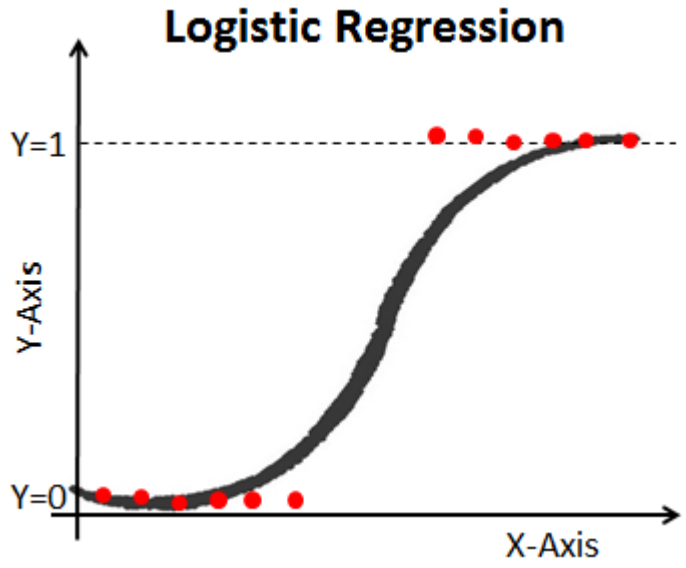
Implementation in python: Results prediction & Confusion matrix 07

08 Logistic Regression vs Linear Regression

Logistic Regression

What is logistic Regression?

The logistic model is used to predict the likelihood of a specific class or occurrence.



Conditions to use logistic regression

- If the information is binary.
- If you require probabilistic outcomes.
- When a linear decision boundary is required.

Applications

- To forecast survival in wounded patients in the medical profession.
- To estimate the likelihood of a person suffering a heart attack.
- Predicting the likelihood of a procedure or product failing.
- Predicting a homeowner's possibility of defaulting on a lender.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Implementation steps

03 Implementation in python: Importing libraries & datasets

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Pre-processing

06 Implementation in python: Training the model

07 Implementation in python: Results prediction & Confusion matrix

08 Logistic Regression vs Linear Regression

Logistic Regression Implementation

Logistic Regression in python

- ***Steps to implement Logistic Regression***
  - Data pre-processing like filtering
  - Fitting KNN to the training set.
  - Predict the test results.
  - Test accuracy of the results.

User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Implementation steps

03 Implementation in python: Importing libraries & datasets

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Pre-processing

06 Implementation in python: Training the model

07 Implementation in python: Results prediction & Confusion matrix

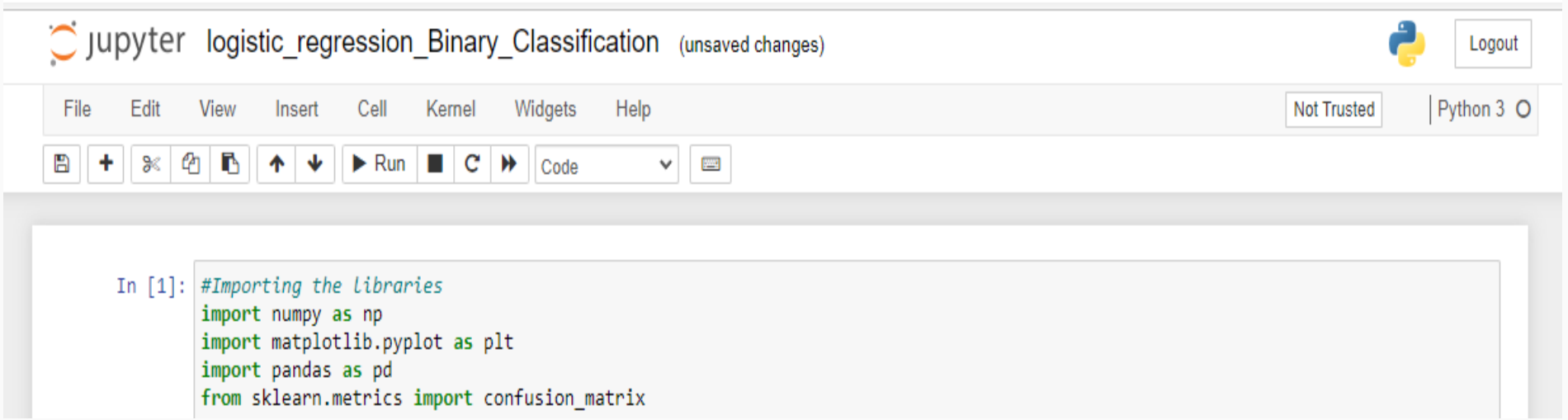
08 Logistic Regression vs Linear Regression

Logistic Regression Implementation

Logistic Regression in python

Implementation step 1

- First, we will import the important libraries.



Meta Brains

Logistic Regression Implementation

Logistic Regression in python

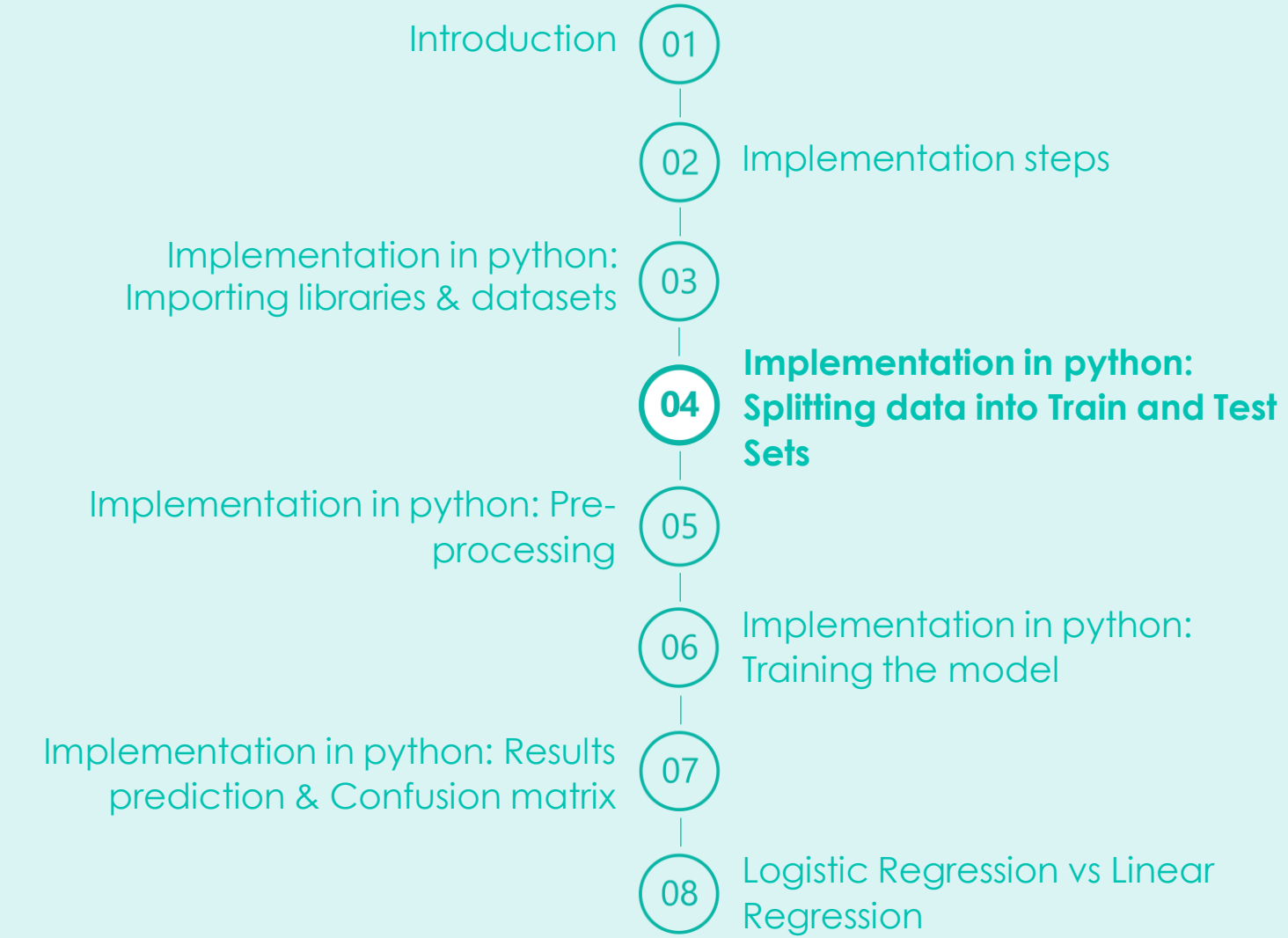
Implementation step 2

- we will import the given data sets.
- Extract independent and dependent variables.

```
In [3]: # Importing the dataset
        dataset = pd.read_csv('user data.csv')
        X = dataset.iloc[:, [2,4]].values
        y = dataset.iloc[:, 4].values
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion



Logistic Regression Implementation

Logistic Regression in python

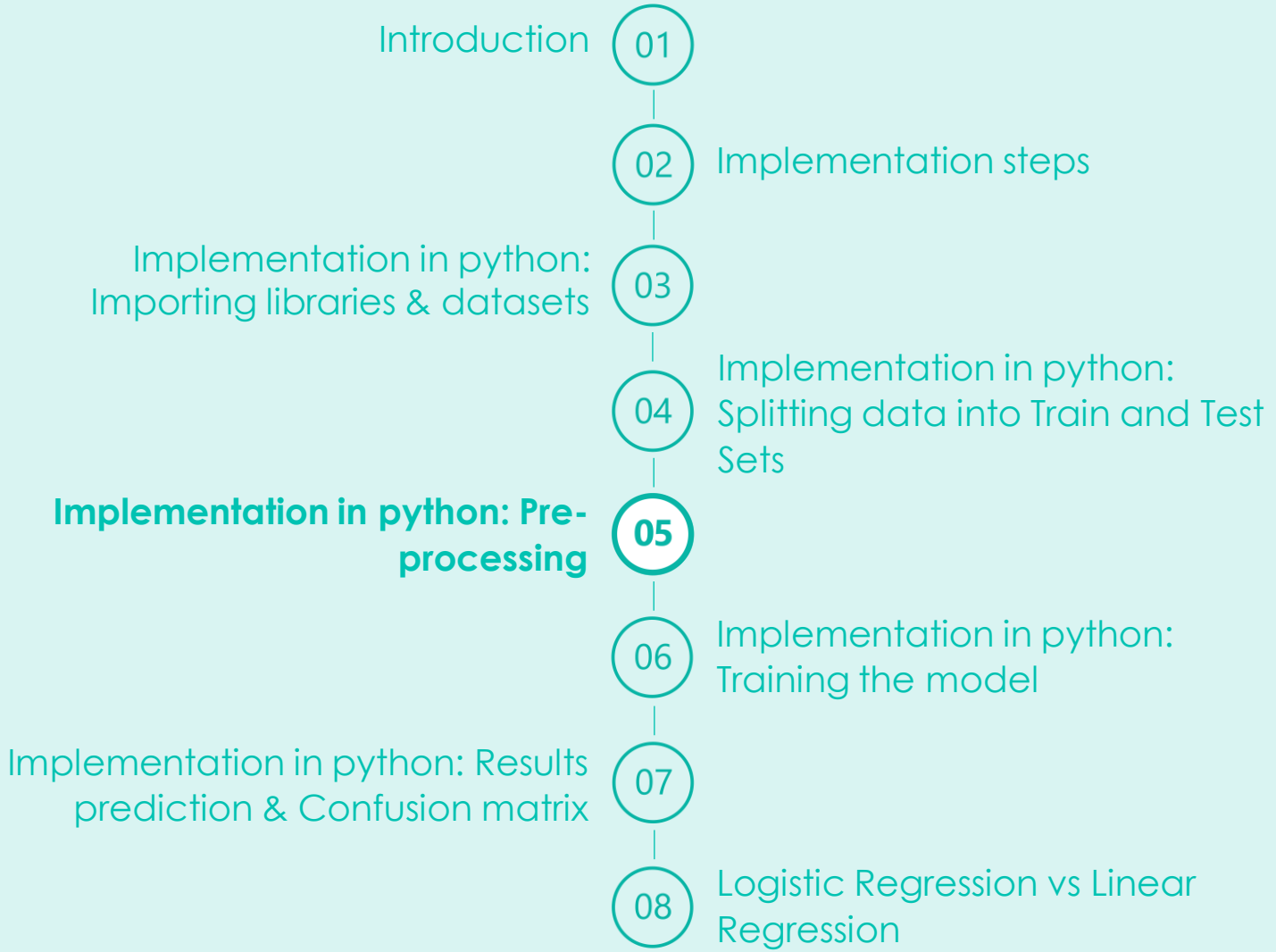
Implementation step 3

- Split the data into testing and training data set.
- Import sklearn for pre-processing and scale the features.

```
In [4]: #Training and Testing Data (divide the data into two part)
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test =train_test_split(X,y,test_size= 0.25, random_state=0)
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion



Logistic Regression Implementation

Logistic Regression in python

Implementation step 3

- Import sklearn for pre-processing and scale the features.

```
In [5]: from sklearn.preprocessing import StandardScaler
        sc_X = StandardScaler()
        X_train = sc_X.fit_transform(X_train)
        X_test = sc_X.fit_transform(X_test)
```

Meta Brains

Logistic Regression Implementation

Logistic Regression in python

Scaled data

```
In [11]: print(X)
```

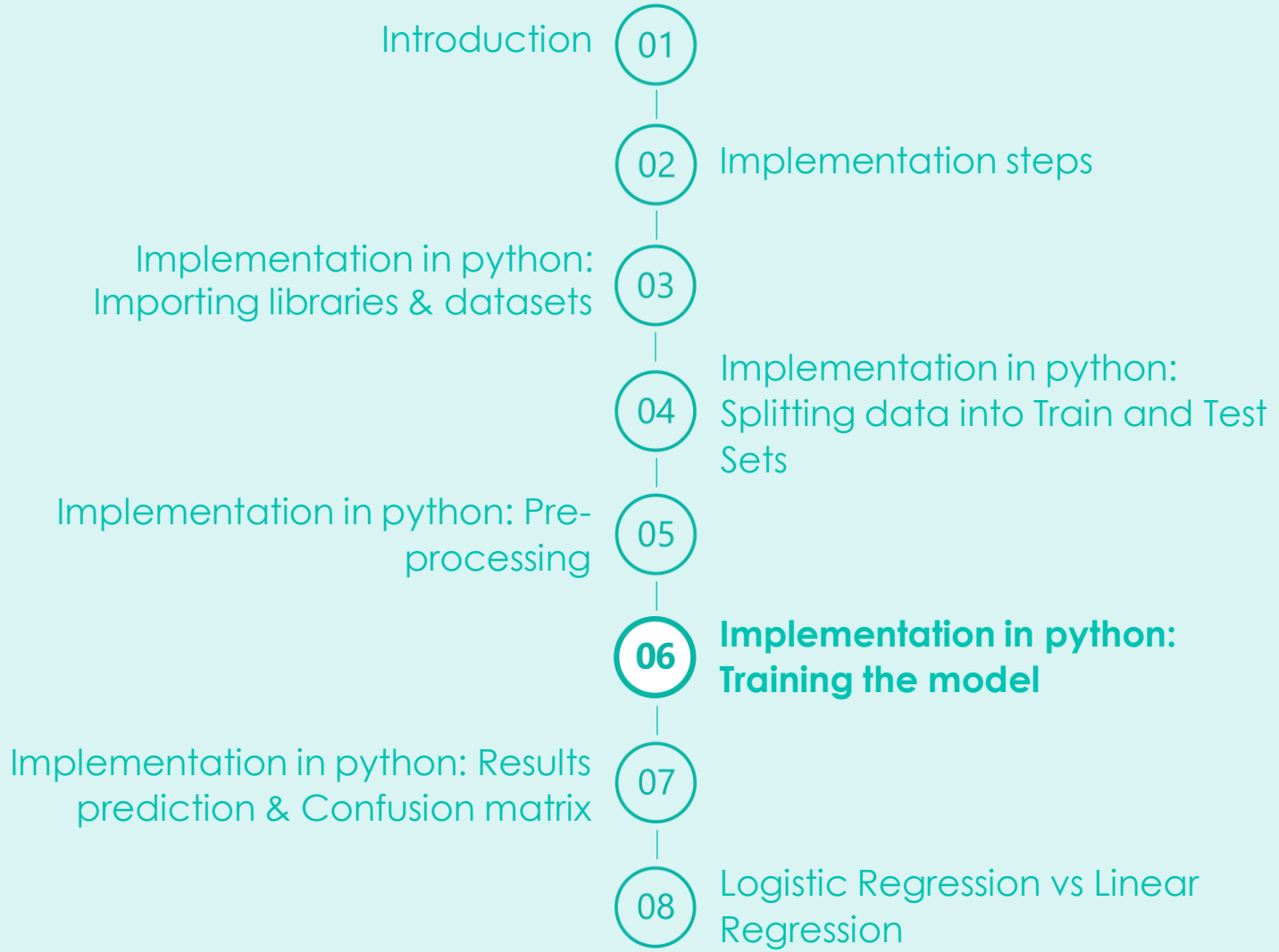
```
[[19  0]
 [35  0]
 [26  0]
 [27  0]
 [19  0]
 [27  0]
 [27  0]
 [32  1]
 [25  0]
 [35  0]
 [26  0]
 [26  0]
 [20  0]
 [32  0]
 [18  0]
 [29  0]
 [47  1]
 [45  1]
 [46  1]
```

```
In [9]: print(X_train)
```

```
[[ 0.58164944 -0.76635604]
 [-0.60673761  1.30487651]
 [-0.01254409 -0.76635604]
 [-0.60673761  1.30487651]
 [ 1.37390747  1.30487651]
 [ 1.47293972  1.30487651]
 [ 0.08648817 -0.76635604]
 [-0.01254409 -0.76635604]
 [-0.21060859 -0.76635604]
 [-0.21060859 -0.76635604]
 [-0.30964085 -0.76635604]
 [-0.30964085 -0.76635604]
 [ 0.38358493  1.30487651]
 [ 0.8787462   1.30487651]
 [ 2.06713324  1.30487651]
 [ 1.07681071 -0.76635604]
 [ 0.68068169  1.30487651]
 [-0.70576986 -0.76635604]
 [ 0.77971394 -0.76635604]
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion



Logistic Regression Implementation

Logistic Regression in python

Implementation step 4

- ***Fitting Logistic Regression classifier to training data***
- Import sklearn and get our classifier – Logistic Regression

```
In [12]: from sklearn.linear_model import LogisticRegression
         classifer=LogisticRegression(random_state=0)
         classifer.fit(X_train,y_train)

Out[12]: LogisticRegression(random_state=0)
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Implementation steps

03 Implementation in python: Importing libraries & datasets

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Pre-processing

06 Implementation in python: Training the model

07 Implementation in python: Results prediction & Confusion matrix

08 Logistic Regression vs Linear Regression

Logistic Regression Implementation

Logistic Regression in python

Prediction of results and confusion matrix

Now will predict the results and confusion matrix.

```
y_pred = classifer.predict(X_test)
print(y_pred)
```

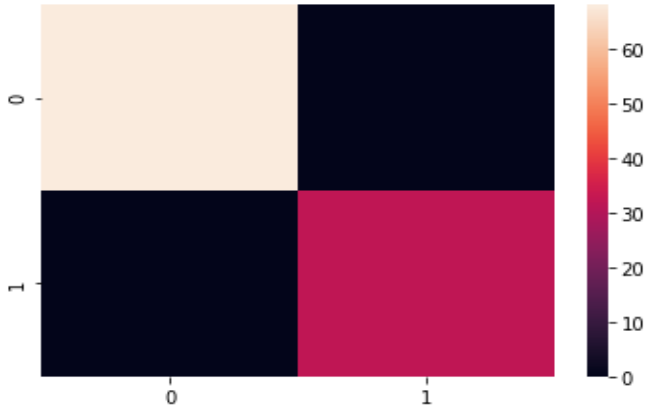
```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

Correct predictions= 68+32 = 100
Incorrect predictions= 0

```
In [14]: # Making the Confusion Matrix
         import seaborn as sns

         cm = confusion_matrix(y_pred, y_test)
         print(cm)
         sns.heatmap(cm)
```

```
[[68  0]
 [ 0 32]]
```

```
Out[14]: <AxesSubplot:>
```



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- **Classification Algorithms: Logistic regression**
- Clustering
- Recommender System
- Conclusion

01 Introduction

02 Implementation steps

03 Implementation in python: Importing libraries & datasets

04 Implementation in python: Splitting data into Train and Test Sets

05 Implementation in python: Pre-processing

06 Implementation in python: Training the model

07 Implementation in python: Results prediction & Confusion matrix

08 Logistic Regression vs Linear Regression

Logistic Regression Vs Linear Regression

Logistic Regression

- We estimate the values of categorical variables using logistic regression.
- We discover the S-curve in Logistic Regression and use it to categorize the samples.
- It is not necessary to have a linear connection between the dependent and independent variables in logistic regression.
- There should be no collinearity between the independent variables in logistic regression.

Linear Regression

- We estimate the outcome of continuous variables using linear regression.
- In linear regression, we look for the best fit line, which allows us to predict the outcome with ease.
- It is necessary for the connection between the dependent and independent variables to be linear.
- There is a possibility of collinearity between the independent factors in linear regression.

Meta Brains

Quiz Time

Is Logistic regression mainly used for Regression?

A) TRUE
B) FALSE

Meta Brains

Quiz Time

Is Logistic regression mainly used for Regression?

A) TRUE
B) FALSE

Solution: B

Logistic regression is a classification algorithm, don't confuse with the name regression.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

Introduction to clustering 01

02 Use cases

K-Means Clustering Algorithm 03

04 Elbow method

Steps of the Elbow method 05

06 Implementation in python

Hierarchical clustering 07

08 Density-based clustering

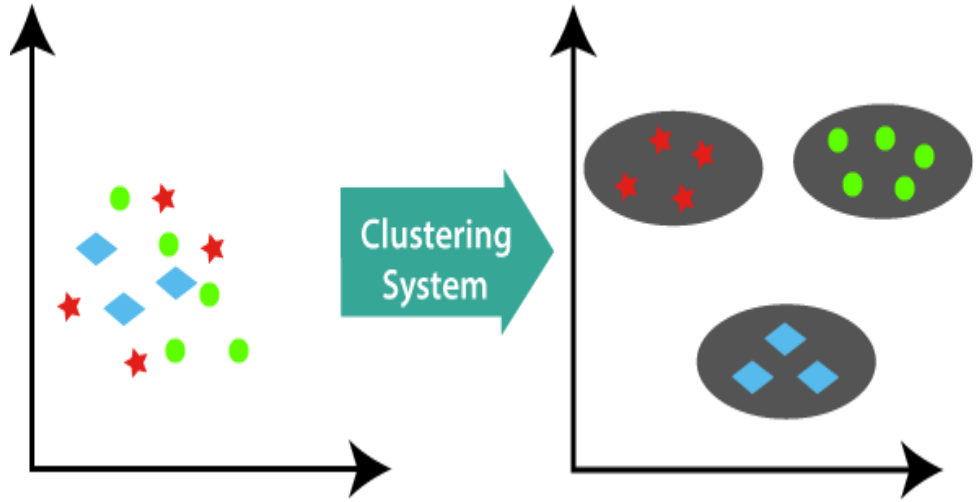
Implementation of k-means clustering in python 09

10 Importing the dataset

Clustering

What is Clustering?

Clustering is an unsupervised learning; it is a process of partitioning a set of data into a meaningful sub-classes.



Applications of Clustering

- Pattern recognition
- Spatial data analysis
- Image processing
- Document classification
- Identification of similar entities
- Finding pattern of weather behavior

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

01 Introduction to clustering

02 **Use cases**

03 K-Means Clustering Algorithm

04 Elbow method

05 Steps of the Elbow method

06 Implementation in python

07 Hierarchical clustering

08 Density-based clustering

09 Implementation of k-means clustering in python

10 Importing the dataset

Clustering

Why do we need Clustering?

Clustering can be used for the following tasks:

- Analyzing data from the research
- Creating a summary
- Detecting noise
- Duplicate detection
- Procedures for pre-processing

Algorithms of Clustering

- K-Means Clustering
- Hierarchical Clustering
- Density-based Clustering

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

Introduction to clustering 01

02 Use cases

K-Means Clustering Algorithm **03**

04 Elbow method

Steps of the Elbow method 05

06 Implementation in python

Hierarchical clustering 07

08 Density-based clustering

Implementation of k-means clustering in python 09

10 Importing the dataset

Clustering

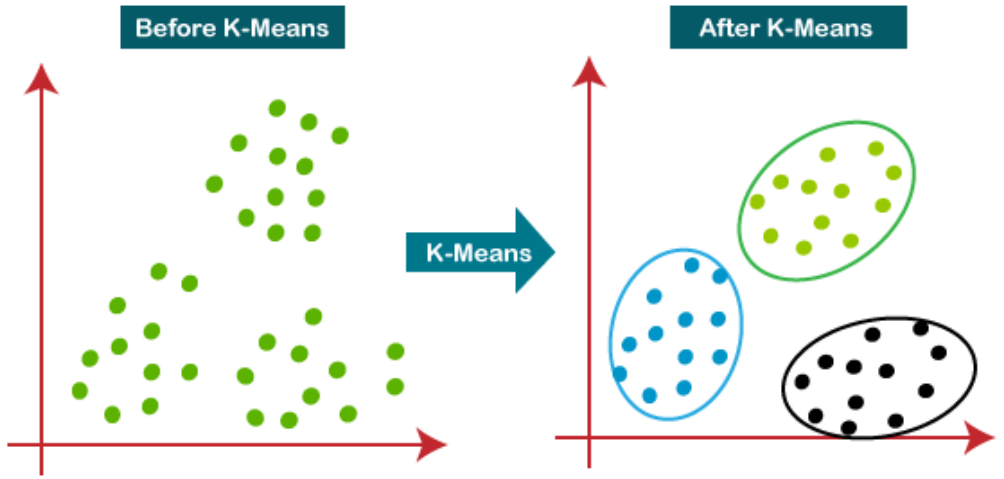
K-Means Clustering

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

Algorithms of K-Clustering

Following is an algorithm of K-Means Clustering:

- Partition clustering
- Without any cluster-internal structure, K-Means divides data into non-overlapping groups.
- Within a cluster, examples are very similar.
- Various clusters have quite different examples.



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

Introduction to clustering 01

02 Use cases

K-Means Clustering Algorithm 03

04 Elbow method

Steps of the Elbow method 05

06 Implementation in python

Hierarchical clustering 07

08 Density-based clustering

Implementation of k-means clustering in python 09

10 Importing the dataset

Clustering

Determination of K in K-means clustering

The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task.

Elbow method

This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS.

$$WCSS= \sum_{P_i \text{ in Cluster1}} distance(P_i\ C_1)^2 + \sum_{P_i \text{ in Cluster2}} distance(P_i\ C_2)^2 + \sum_{P_i \text{ in CLuster3}} distance(P_i\ C_3)^2$$

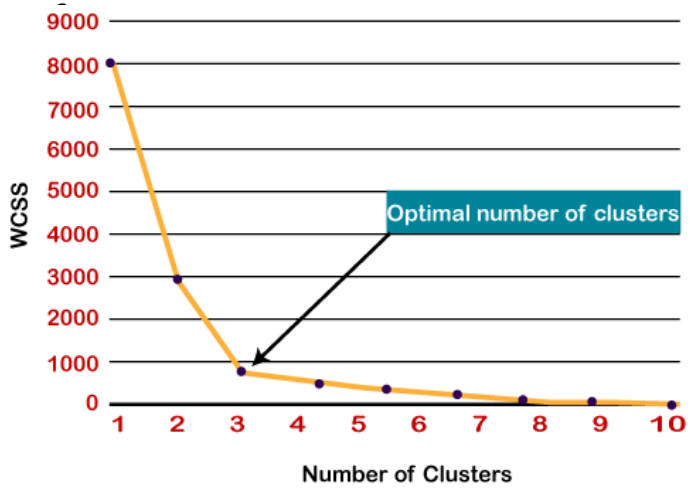
Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

Clustering

Steps of Elbow method

•It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
•For each value of K, calculates the WCSS value.
•Plots a curve between calculated WCSS values and the number of clusters K.
•The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

01 Introduction to clustering

02 Use cases

03 K-Means Clustering Algorithm

04 Elbow method

05 Steps of the Elbow method

06 Implementation in python

07 Hierarchical clustering

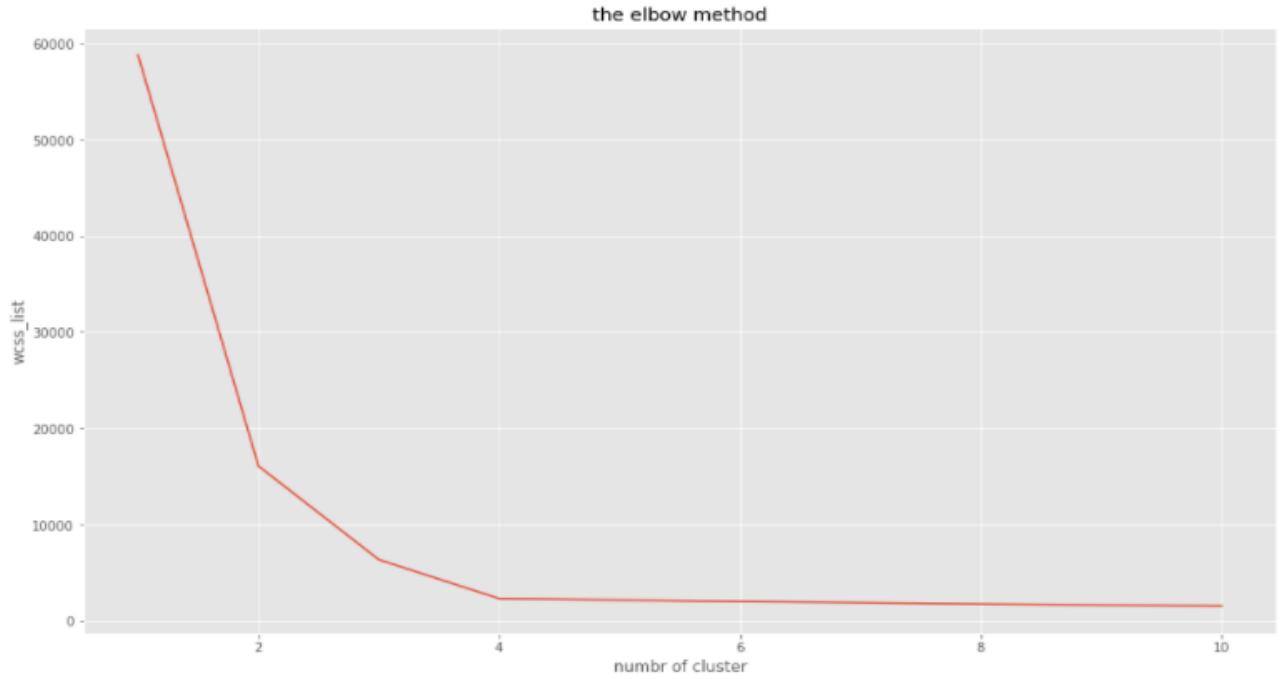
08 Density-based clustering

09 Implementation of k-means clustering in python

10 Importing the dataset

The Elbow method

In [66]:
```
# Finding the optimal number of clusters using elbow method
from sklearn.cluster import KMeans
wcss_list =[] # Intializing the list for the values of WCSS

# 10 iterations
for i in range(1,11):
    kmeans=KMeans(n_clusters=i, init='k-means++',random_state =42)
    kmeans.fit(X)
    wcss_list.append(kmeans.inertia_)

plt.plot(range(1,11),wcss_list)
plt.title("the elbow method")
plt.xlabel("numbr of cluster")
plt.ylabel("wcss_list")
plt.show()
```



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

01 Introduction to clustering
02 Use cases
03 K-Means Clustering Algorithm
04 Elbow method
05 Steps of the Elbow method
06 Implementation in python
07 Hierarchical clustering
08 Density-based clustering
09 Implementation of k-means clustering in python
10 Importing the dataset

Clustering

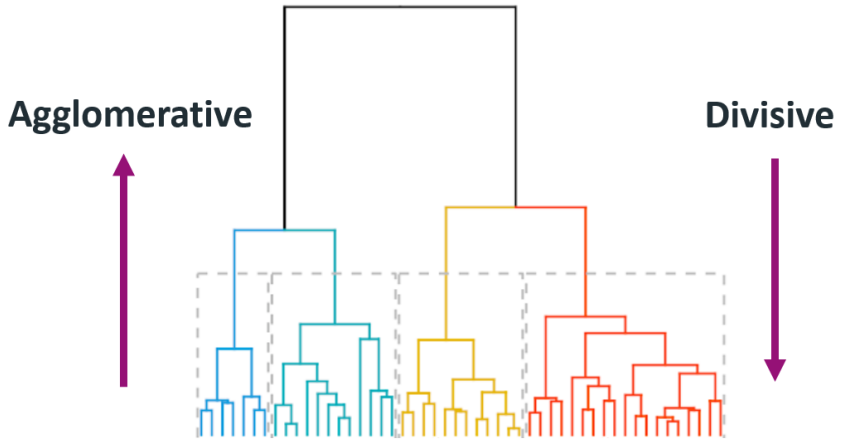
What is Hierarchical Clustering?

It is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly like each other.

Types of Hierarchical Clustering

There are two types of Hierarchical clustering:

- **Divisive clustering** is top to down approach in which observations starts from large cluster and splits into smaller ones.
- **Agglomerative clustering** is a bottom to top approach in which observations start from many clusters and merged as one cluster in the end.



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

01 Introduction to clustering

02 Use cases

03 K-Means Clustering Algorithm

04 Elbow method

05 Steps of the Elbow method

06 Implementation in python

07 Hierarchical clustering

08 Density-based clustering

09 Implementation of k-means clustering in python

10 Importing the dataset

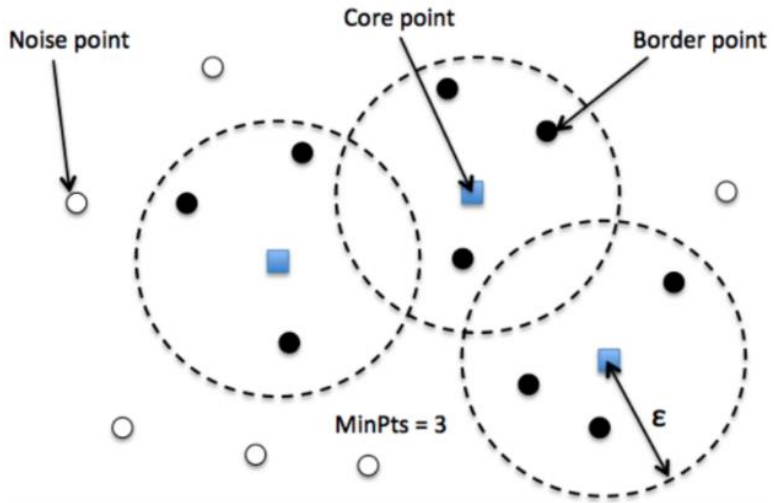
Clustering

What is Density-based Clustering?

Density-based clustering locates regions of high density and separates outliers. So, the regions of high density get separated from the regions of low density.

What is DBSCAN?

The most powerful attribute of DBSCAN algorithm is that it can find out any arbitrary shape cluster without getting effected by noise.



Advantages

- It can discover arbitrarily shaped cluster.
- Find cluster surrounded by different cluster.
- Robust towards noise detection.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

01 Introduction to clustering

02 Use cases

03 K-Means Clustering Algorithm

04 Elbow method

05 Steps of the Elbow method

06 Implementation in python

07 Hierarchical clustering

08 Density-based clustering

09 Implementation of k-means clustering in python

10 Importing the dataset

Clustering

K-means clustering in python

Implementation step 1

- Import the required dataset

```
In [18]: %matplotlib inline
         from copy import deepcopy
         import numpy as np
         import pandas as pd
         from matplotlib import pyplot as plt
         plt.rcParams['figure.figsize'] = (16, 9)
         plt.style.use('ggplot')
```

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- **Clustering**
- Recommender System
- Conclusion

01 Introduction to clustering

02 Use cases

03 K-Means Clustering Algorithm

04 Elbow method

05 Steps of the Elbow method

06 Implementation in python

07 Hierarchical clustering

08 Density-based clustering

09 Implementation of k-means clustering in python

10 Importing the dataset

Clustering

K-means clustering in python

Implementation step 2

- Import the dataset

```
In [19]: # Importing the dataset
         data = pd.read_csv('mallCustomerData.txt', sep=",")
         print(data.shape)
         data.head(10)
```

```
(200, 5)
```

Out[19]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

Meta Brains

Clustering

K-means clustering in python

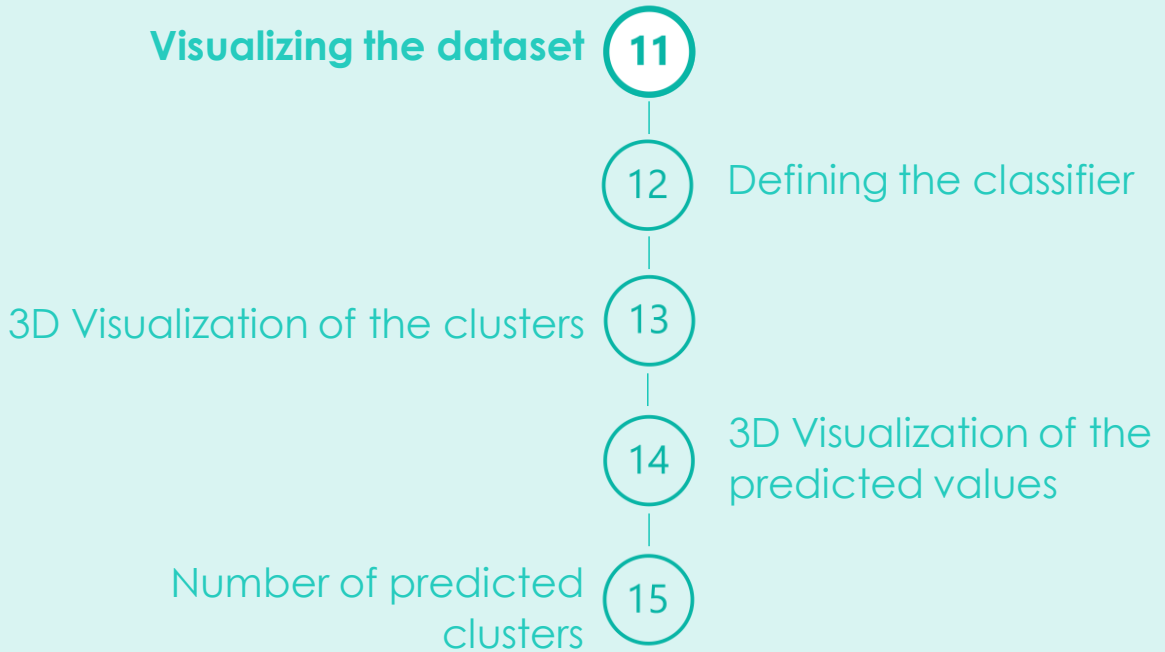
Implementation step 3

- *HotEncoding of Gender values*

```
In [5]: print(data['Gender'].value_counts())

Female    112
Male       88
Name: Gender, dtype: int64
```

```
In [6]: data = pd.get_dummies(data,columns=['Gender'])
print(data)

     CustomerID  Age  Annual Income (k$)  Spending Score (1-100)  \
0             1   19                  15                      39
1             2   21                  15                      81
2             3   20                  16                       6
3             4   23                  16                      77
4             5   31                  17                      40
..          ...  ...                 ...                     ...
195         196   35                 120                      79
196         197   45                 126                      28
197         198   32                 126                      74
198         199   32                 137                      18
199         200   30                 137                      83

     Gender_Female  Gender_Male
0                0            1
1                0            1
2                1            0
3                1            0
4                1            0
..             ...          ...
195              1            0
196              1            0
197              0            1
198              0            1
199              0            1

[200 rows x 6 columns]
```

Meta Brains



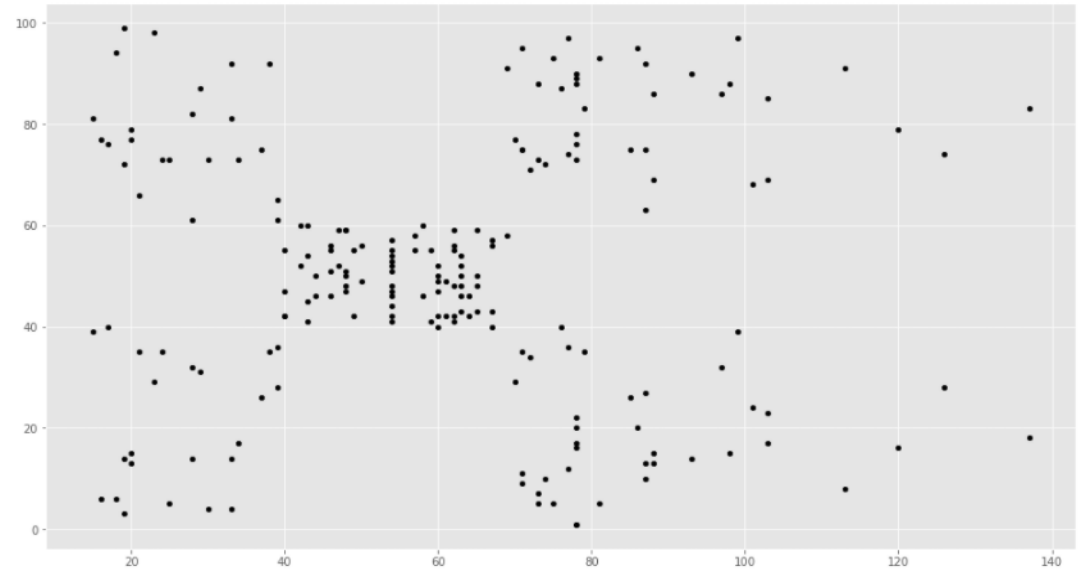
Clustering

K-means clustering in python

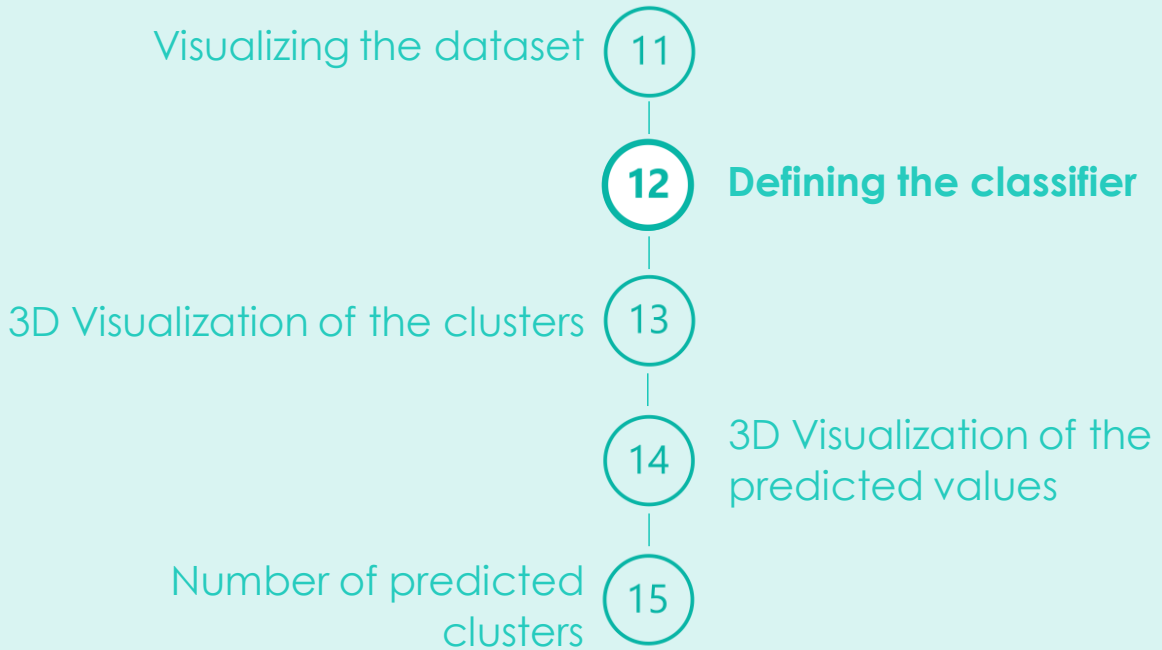
Implementation step 4

- *Visualize the dataset*

```
In [20]: # Getting the values and plotting it
         f1 = data['Annual Income (k$)'].values
         f2 = data['Spending Score (1-100)'].values

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

         X = np.array(list(zip(f1, f2)))
         #X=[[1, 1], [1, 2], [3, 3],[4,4]]
         plt.scatter(f1, f2, c='black', s=20)

         CustomerID
         Gender
         Age
         Annual Income (k$)
         Spending Score (1-100)

Out[20]: <matplotlib.collections.PathCollection at 0x1460b208be0>
```



Meta Brains



Clustering

K-means clustering in python

Implementation step 5

- *Defining the Classifier*
- *Fit the kmeans*
- *Print the values of cluters coordinates*

```
In [50]: from sklearn.cluster import KMeans

         # Number of clusters
         kmeans = KMeans(n_clusters=3)
         # Fitting the input data
         kmeans = kmeans.fit(X)
         # Getting the cluster labels
         labels = kmeans.predict(X)
         # Centroid values
         C = kmeans.cluster_centers_
```

```
In [57]: # Values of the clusters
         print(C)

[[-2.59662966  8.17293022 -3.18861476]
 [ 9.90577958 -1.78901987  4.83348761]
 [-2.67717112 -0.81623897 -7.81162806]
 [-5.19053091  0.683535   -2.16596538]]
```

Meta Brains

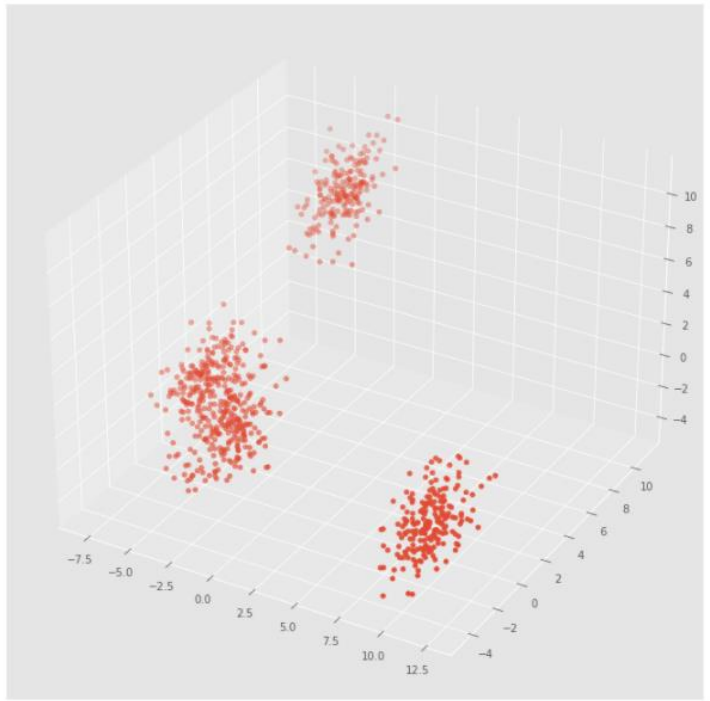
Clustering

K-means clustering in python

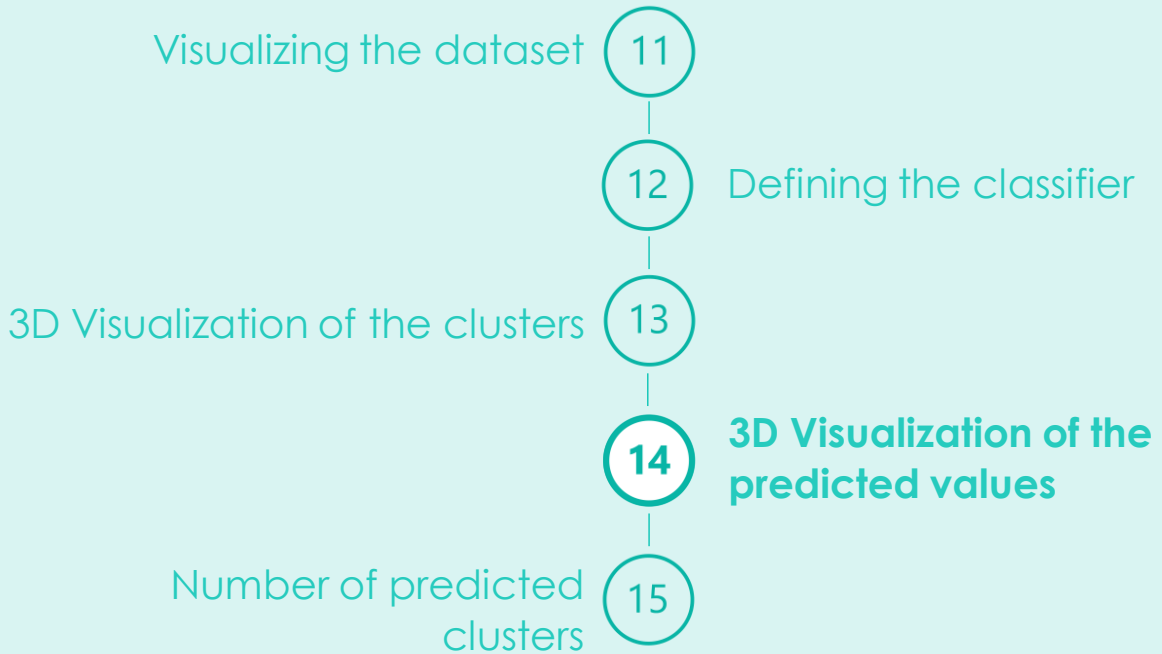
Implementation step 6

- A 3D plot of our dataset

```
In [41]: fig = plt.figure()
         ax = Axes3D(fig)
         ax.scatter(X[:, 0], X[:, 1], X[:, 1])
Out[41]: <mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1460cc20cd0>
```



Meta Brains



Clustering

K-means clustering in python

Implementation step 7

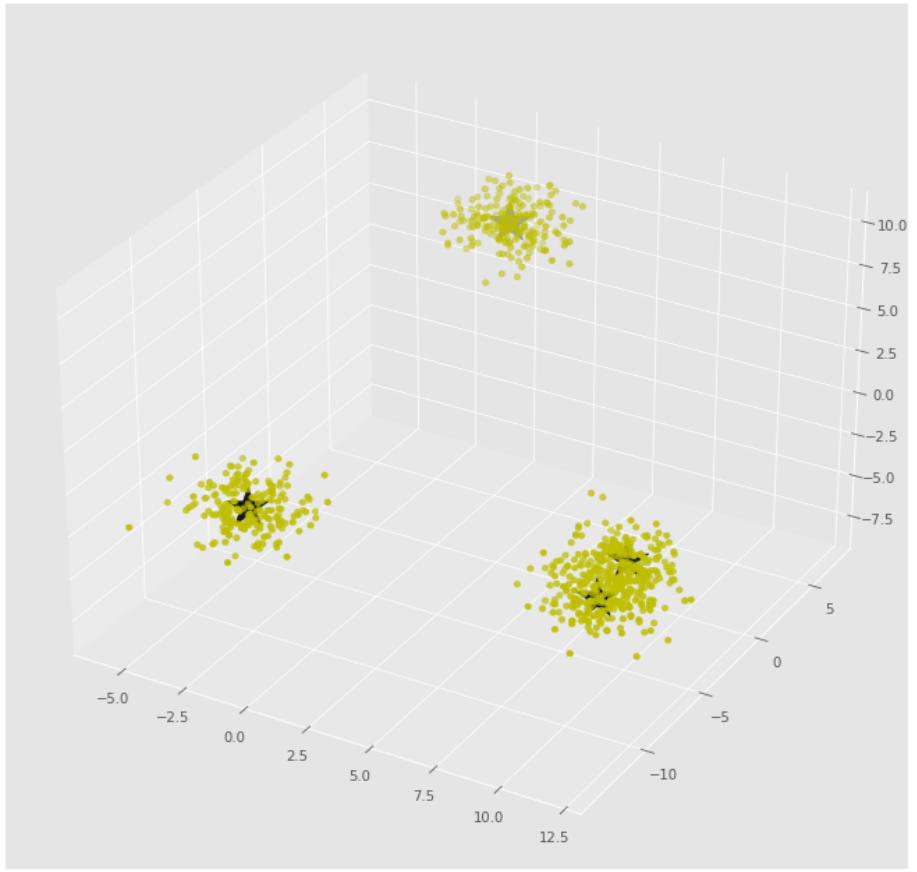
- *3D plot of the prediction Clusters*

```
In [57]: # Values of the clusters
         print(C)
```

```
[[-2.59662966  8.17293022 -3.18861476]
 [ 9.90577958 -1.78901987  4.83348761]
 [-2.67717112 -0.81623897 -7.81162806]
 [-5.19053091  0.683535   -2.16596538]]
```

```
fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c='y')
ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker='*', c='#050505', s=1000)
```

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x1498114b5b0>
```



Meta Brains

Clustering

K-means clustering in python

Implementation step 7
Checking the predicted Values cluster number

```
In [67]: # Initializing KMeans
         kmeans = KMeans(n_clusters=4)
         # Fitting with inputs
         kmeans = kmeans.fit(X)
         # Predicting the clusters
         labels = kmeans.predict(X)
         # Getting the cluster centers
         C = kmeans.cluster_centers_
         print(labels)
```

```
print(labels)

[3 1 3 0 2 1 1 1 2 2 2 1 3 0 2 2 2 1 3 0 3 0 1 3 0 3 0 0 1 0 2 3 3 2 2 1 0
 2 2 3 0 1 3 0 3 3 0 1 3 2 0 2 0 0 2 0 2 1 1 2 1 3 0 1 2 1 2 0 0 3 3 2 0 0
 2 3 0 2 0 3 0 3 3 0 3 3 0 2 3 1 1 1 3 0 2 2 3 3 0 3 1 1 0 1 3 1 1 0 3 2 1
 1 1 0 2 1 0 0 3 2 0 2 1 0 3 2 0 0 3 1 3 3 1 0 2 2 2 0 2 1 1 3 2 3 1 3 0 3
 0 2 1 2 0 2 2 1 2 0 1 1 0 2 0 0 3 2 2 2 2 3 3 2 3 2 3 2 1 2 2 2 3 1 3 1 3
 3 2 3 1 1 0 3 0 2 1 0 2 0 2 1 3 2 1 2 1 3 3 0 3 1 1 2 3 3 0 0 3 1 2 0 2 2
 2 0 3 0 3 2 0 1 0 0 2 0 2 0 3 1 2 2 0 0 3 3 0 1 1 3 0 0 3 3 2 0 0 0 0 0 1
 1 2 1 3 3 3 2 2 0 1 0 2 0 0 0 1 3 3 3 0 2 1 3 3 1 0 2 3 3 1 1 1 0 0 1 0 0
 2 2 0 3 3 0 1 3 1 2 0 3 3 3 2 1 3 0 2 3 3 3 1 0 1 2 1 0 2 3 0 0 0 2 3 0 3
 0 1 1 2 1 3 1 2 1 2 2 0 1 1 3 3 2 3 2 0 3 3 2 1 1 1 2 0 1 0 1 1 2 1 1 0 2
 3 3 3 2 3 0 1 0 1 1 2 2 3 3 2 3 1 0 0 1 0 0 0 3 1 2 1 3 3 2 0 0 1 3 1 1 2
 1 1 2 3 1 3 2 0 3 3 0 2 1 3 3 3 1 2 3 1 3 2 2 3 2 0 2 0 1 2 1 3 1 0 1 3 3
 2 3 2 0 1 2 2 1 3 0 0 2 2 2 1 0 3 2 1 2 0 1 1 0 3 1 1 3 0 1 2 1 0 2 1 0 1
 3 3 2 1 0 0 3 2 3 3 1 2 3 0 0 0 1 2 0 2 3 3 2 3 1 1 0 1 1 0 0 0 1 3 2 0 1
 1 2 1 3 2 2 2 2 0 0 2 1 3 0 3 0 1 1 2 2 2 0 0 2 2 2 1 1 3 2 3 1 0 0 3 2 3
 1 2 0 1 3 3 2 2 0 2 2 3 2 0 0 0 3 3 1 1 1 2 0 2 1 2 1 3 3 3 0 2 1 2 3 3 3
 0 1 0 0 3 3 3 0 1 1 1 3 2 3 2 3 2 0 3 1 2 0 3 2 0 3 0 0 0 1 1 2 2 0 1 2 1
 3 1 0 0 0 2 2 0 1 3 2 1 1 1 0 3 3 1 1 1 0 0 0 1 0 2 1 1 2 2 1 2 3 3 1 0 3
 0 2 1 1 1 0 1 0 3 2 1 3 2 2 0 0 3 2 3 3 2 2 0 3 1 1 1 0 0 2 3 0 0 3 0 2 1
 3 1 2 2 2 1 3 2 2 2 0 1 2 3 1 1 1 2 1 2 0 3 3 3 1 3 1 0 0 3 1 3 0 0 3 3 3
 3 2 0 0 2 3 1 2 1 2 1 2 2 1 2 0 0 1 0 2 2 1 3 0 1 0 1 3 2 3 0 1 1 2 3 2 2
 3 1 3 1 0 3 3 0 3 3 2 0 0 0 3 0 1 1 0 1 1 2 2]
```

Meta Brains

Quiz Time

Which of the following is required by K-means clustering?

A) defined distance metric
B) number of clusters
C) initial guess as to cluster centroids
D) all the mentioned

Quiz Time

Which of the following is required by K-means clustering?

A) defined distance metric
B) number of clusters
C) initial guess as to cluster centroids
D) all the mentioned

Answer: D
Explanation: because K-means clustering follows partitioning approach.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

Introduction 01

02 Collaborative Filtering in Recommender Systems

Content-based Recommender System 03

04 Implementation in python: Importing libraries & datasets

Merging datasets into one dataframe 05

06 Sorting by title and rating

Histogram showing number of ratings 07

08 Frequency distribution

Jointplot of the ratings and number of ratings 09

10 Data pre-processing

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

11 Sorting the most-rated movies

12 Grab the ratings for two movies

13 Correlation between the most-rated movies

14 Sorting the data by correlation

15 Filtering out movies

16 Sorting values

17 Repeating the process for another movie

Recommender system

What is Recommender System?

Recommender systems are algorithms aimed at suggesting relevant items to users for example items being movies to watch, text to read, products to buy or anything else depending on industries.

Applications

The applications of Recommender systems include movies, music, books, websites, documents, television programs.

Benefits

- Increase customer engagement.
- Increase customer satisfaction by delivering relevant content.
- Reduce the time of content searching.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

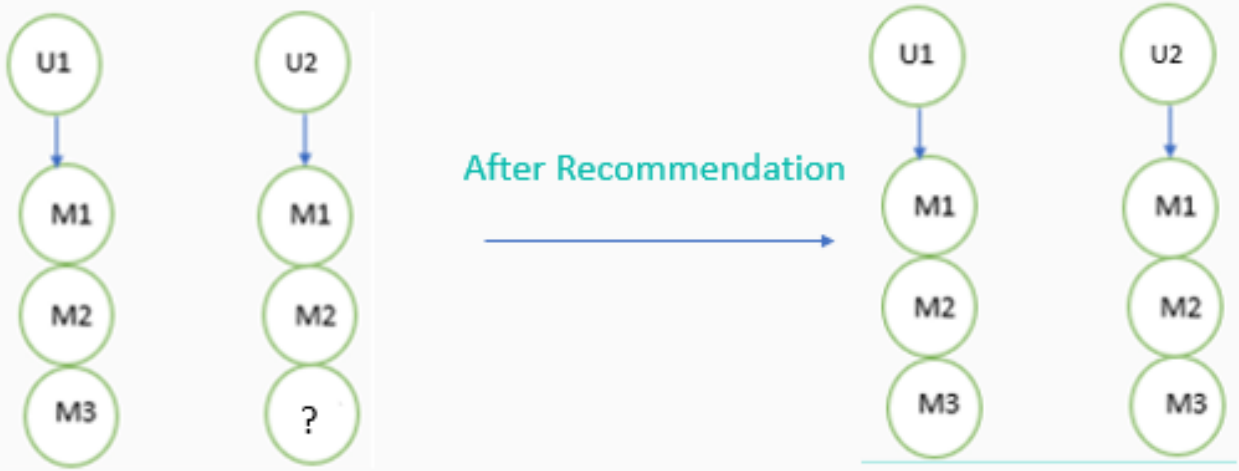
09 Jointplot of the ratings and number of ratings

10 Data pre-processing

Recommender system

Collaborative filtering Recommendation system

The collaborative filtering method predicts the interests of a user on a product by collecting preferences information from many other users.



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

Introduction 01

02 Collaborative Filtering in Recommender Systems

Content-based Recommender System **03**

04 Implementation in python: Importing libraries & datasets

Merging datasets into one dataframe 05

06 Sorting by title and rating

Histogram showing number of ratings 07

08 Frequency distribution

Jointplot of the ratings and number of ratings 09

10 Data pre-processing

Recommender system

Content-based Recommender System

Content-based filtering method utilizes product features to recommend other products like what the user likes, based on other users' previous actions or explicit feedback such as rating on products.

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

09 Jointplot of the ratings and number of ratings

10 Data pre-processing

Recommender system

Recommender System in python

Implementation step 1

- Import the libraries
- Get the datasets

In [1]:
```
import numpy as np
import pandas as pd

#for visiulization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')
%matplotlib inline
```

In [2]:
```
column_names = ['user_id', 'item_id', 'rating', 'timestamp']
df = pd.read_csv('u.data', sep='\t', names=column_names)
```

In [3]:
```
df.head()
```

Out[3]:

	user_id	item_id	rating	timestamp
0	0	50	5	881250949
1	0	172	5	881250949
2	0	133	1	881250949
3	196	242	3	881250949
4	186	302	3	891717742

In [4]:
```
movie_titles = pd.read_csv("Movie_Id_Titles")
movie_titles.head()
```

Out[4]:

	item_id	title
0	1	Toy Story (1995)
1	2	GoldenEye (1995)
2	3	Four Rooms (1995)
3	4	Get Shorty (1995)
4	5	Copycat (1995)

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

09 Jointplot of the ratings and number of ratings

10 Data pre-processing

Recommender system

Recommender System in python

Implementation step 2

- *Merge the two tables*
- *Using item_id*

```
In [5]: df = pd.merge(df,movie_titles,on='item_id')
        df.head()
```

Out[5]:

	user_id	item_id	rating	timestamp	title
0	0	50	5	881250949	Star Wars (1977)
1	290	50	5	880473582	Star Wars (1977)
2	79	50	4	891271545	Star Wars (1977)
3	2	50	5	888552084	Star Wars (1977)
4	8	50	5	879362124	Star Wars (1977)

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

09 Jointplot of the ratings and number of ratings

10 Data pre-processing

Recommender system

Recommender System in python

Implementation step 3

- *Sort the rows by group of Title, and rating*

  Find :-

  - *Count*
  - *Mean*

```
In [8]: df.groupby('title')['rating'].count().sort_values(ascending=False).head()

Out[8]: title
        Star Wars (1977)             584
        Contact (1997)               509
        Fargo (1996)                 508
        Return of the Jedi (1983)    507
        Liar Liar (1997)             485
        Name: rating, dtype: int64
```

```
In [7]: df.groupby('title')['rating'].mean().sort_values(ascending=False).head()

Out[7]: title
        They Made Me a Criminal (1939)                5.0
        Marlene Dietrich: Shadow and Light (1996)     5.0
        Saint of Fort Washington, The (1993)          5.0
        Someone Else's America (1995)                 5.0
        Star Kid (1997)                               5.0
        Name: rating, dtype: float64
```

Meta Brains

Recommender system

Recommender System in python

Implementation step 4

- *Sort the rows by group of titles and rating*

```
In [9]: ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
        ratings.head()
```

Out[9]:

title	rating
'Til There Was You (1997)	2.333333
1-900 (1994)	2.600000
101 Dalmatians (1996)	2.908257
12 Angry Men (1957)	4.344000
187 (1997)	3.024390

- *Create a new column And Put number of ratings*

```
In [10]: ratings['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
         ratings.head()
```

Out[10]:

title	rating	num of ratings
'Til There Was You (1997)	2.333333	9
1-900 (1994)	2.600000	5
101 Dalmatians (1996)	2.908257	109
12 Angry Men (1957)	4.344000	125
187 (1997)	3.024390	41

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

09 Jointplot of the ratings and number of ratings

10 Data pre-processing

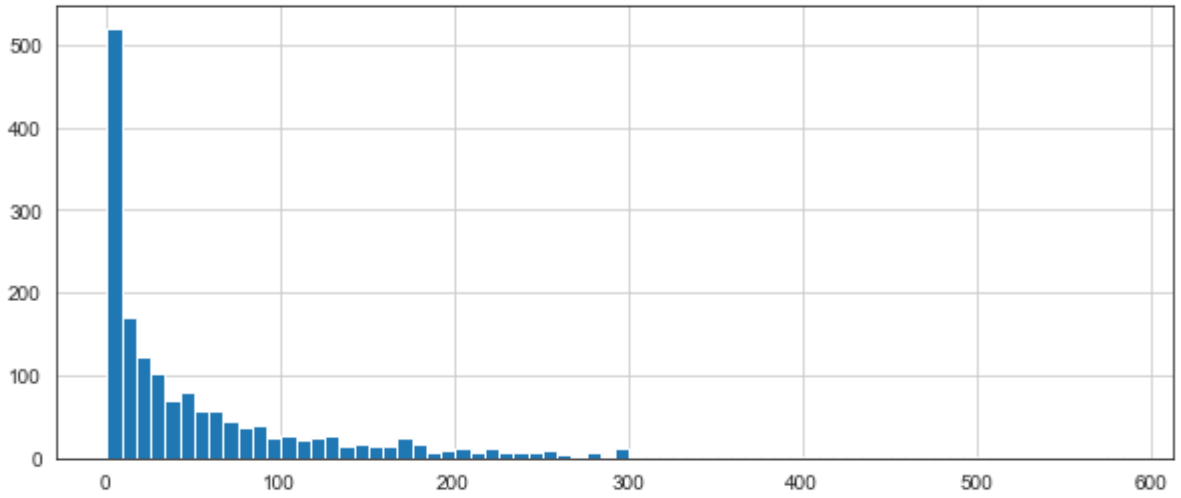
Recommender system

Recommender System in python

Implementation step 5

- *Plot the Newly created column*

```
In [11]: plt.figure(figsize=(10,4))
         ratings['num of ratings'].hist(bins=70)

Out[11]: <AxesSubplot:>
```



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

09 Jointplot of the ratings and number of ratings

10 Data pre-processing

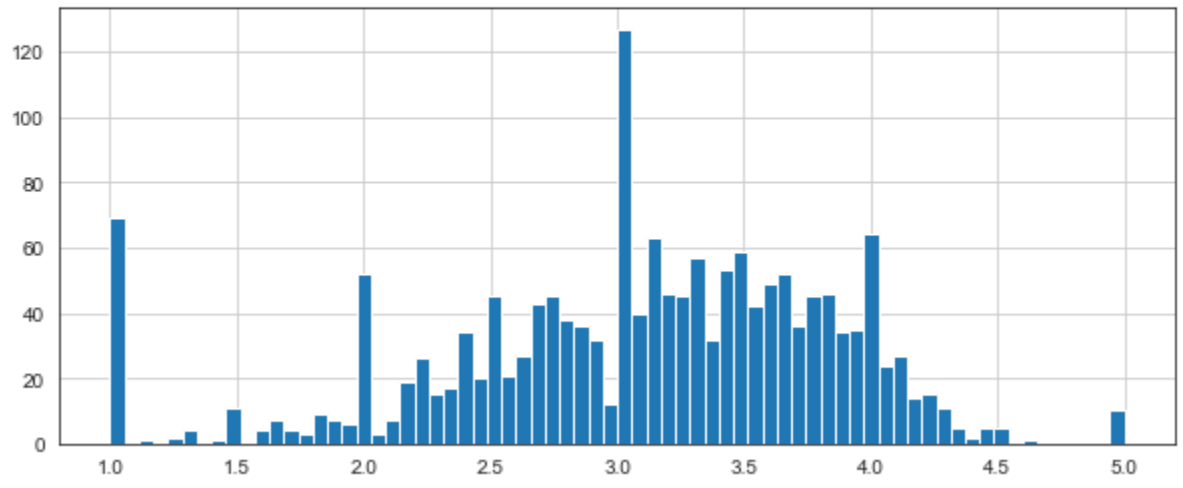
Recommender system

Recommender System in python

Implementation step 6

- *Histogram of the ratings*

```
In [12]: plt.figure(figsize=(10,4))
         ratings['rating'].hist(bins=70)

Out[12]: <AxesSubplot:>
```



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

09 Jointplot of the ratings and number of ratings

10 Data pre-processing

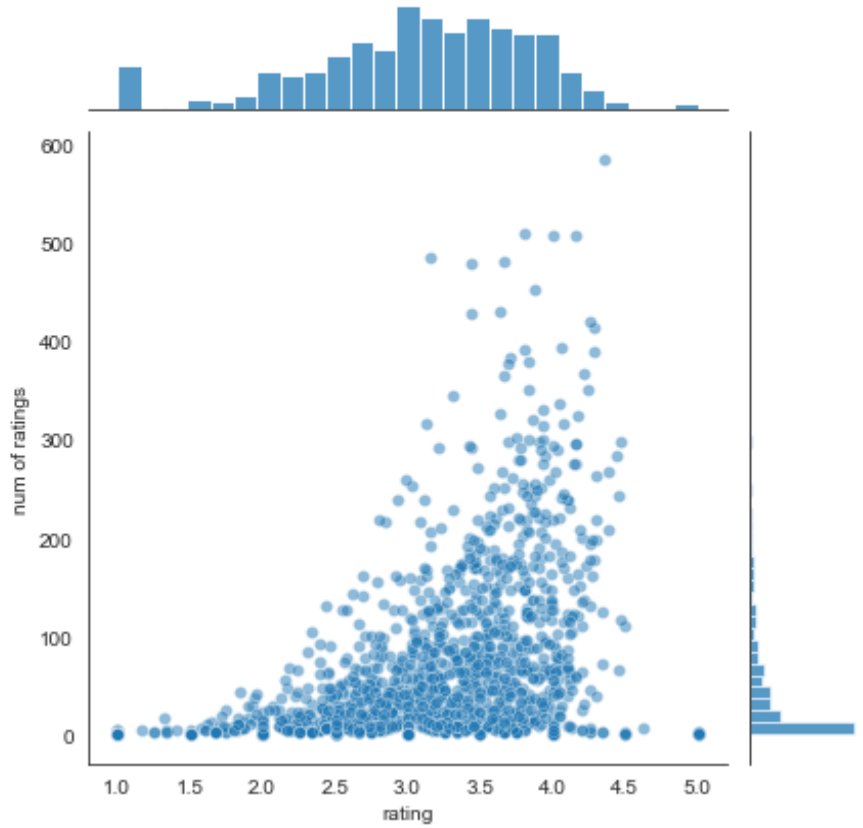
Recommender system

Recommender System in python

Implementation step 7

- *A combined plot of number of Ratings and number of ratings*

```
In [13]: sns.jointplot(x='rating',y='num of ratings',data=ratings,alpha=0.5)
Out[13]: <seaborn.axisgrid.JointGrid at 0x1d21cb69d00>
```



Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion

01 Introduction

02 Collaborative Filtering in Recommender Systems

03 Content-based Recommender System

04 Implementation in python: Importing libraries & datasets

05 Merging datasets into one dataframe

06 Sorting by title and rating

07 Histogram showing number of ratings

08 Frequency distribution

09 Jointplot of the ratings and number of ratings

10 Data pre-processing

Recommender system

Recommender System in python

Implementation step 8

Importing the dataset movies and pre-processing it

```
In [14]: moviemat = df.pivot_table(index='user_id',columns='title',values='rating')
         moviemat.head()
```

Out[14]:

title	'Til There Was You (1997)	1-900 (1994)	101 Dalmatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odyssey (1968)	3 Ninjas: High Noon At Mega Mountain (1998)	39 Steps, The (1935)	...	Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	Y Poisc Handl The (
user_id																		
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN	...	NaN	NaN	NaN	5.0	3.0	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	

5 rows × 1664 columns

Meta Brains

Sorting the most-rated movies 11

Recommender system

Recommender System in python

Implementation step 9

- Sorting the most rated movies

```
In [15]: ratings.sort_values('num of ratings',ascending=False).head(10)
```

Out[15]:

title	rating	num of ratings
Star Wars (1977)	4.359589	584
Contact (1997)	3.803536	509
Fargo (1996)	4.155512	508
Return of the Jedi (1983)	4.007890	507
Liar Liar (1997)	3.156701	485
English Patient, The (1996)	3.656965	481
Scream (1996)	3.441423	478
Toy Story (1995)	3.878319	452
Air Force One (1997)	3.631090	431
Independence Day (ID4) (1996)	3.438228	429

Meta Brains

Recommender system

Recommender System in python

Implementation step 10

- *Grab the Users ratings for two Movies*

- *Star wars (1997)*
- *Liar Liar (1997)*

```
In [17]: starwars_user_ratings = moviemat['Star Wars (1977)']
         liarliar_user_ratings = moviemat['Liar Liar (1997)']
         starwars_user_ratings.head()

Out[17]: user_id
         0    5.0
         1    5.0
         2    5.0
         3    NaN
         4    5.0
         Name: Star Wars (1977), dtype: float64
```

Meta Brains

Recommender system

Recommender System in python

Implementation step 11

- *Use the corrwith() method to get the correlation between our most rated movies.*

```
In [18]: similar_to_starwars = moviemat.corrwith(starwars_user_ratings)
         similar_to_liarliar = moviemat.corrwith(liarliar_user_ratings)
```

```
C:\Users\jawad\anaconda3\lib\site-packages\numpy\lib\function_base.py:2634: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
C:\Users\jawad\anaconda3\lib\site-packages\numpy\lib\function_base.py:2493: RuntimeWarning: divide by zero encountered in true_
divide
  c *= np.true_divide(1, fact)
```

Error !

Meta Brains

Recommender system

Recommender System in python

Implementation step 12

Error !

Lets remove it

```
In [19]: corr_starwars = pd.DataFrame(similar_to_starwars,columns=['Correlation'])
         corr_starwars.dropna(inplace=True)
         corr_starwars.head()
```

Out[19]:

title	Correlation
'Til There Was You (1997)	0.872872
1-900 (1994)	-0.645497
101 Dalmatians (1996)	0.211132
12 Angry Men (1957)	0.184289
187 (1997)	0.027398

Clean all the NaN and Use DataFrame instead of Series

Meta Brains

Recommender system

Recommender System in python

Implementation step 13

Sorting by Correlation

```
In [20]: corr_starwars.sort_values('Correlation',ascending=False).head(10)
```

Out[20]:

title	Correlation
Commandments (1997)	1.0
Cosi (1996)	1.0
No Escape (1994)	1.0
Stripes (1981)	1.0
Man of the Year (1995)	1.0
Hollow Reed (1996)	1.0
Beans of Egypt, Maine, The (1994)	1.0
Good Man in Africa, A (1994)	1.0
Old Lady Who Walked in the Sea, The (Vieille qui marchait dans la mer, La) (1991)	1.0
Outlaw, The (1943)	1.0

Meta Brains

Recommender system

Recommender System in python

Implementation step 14

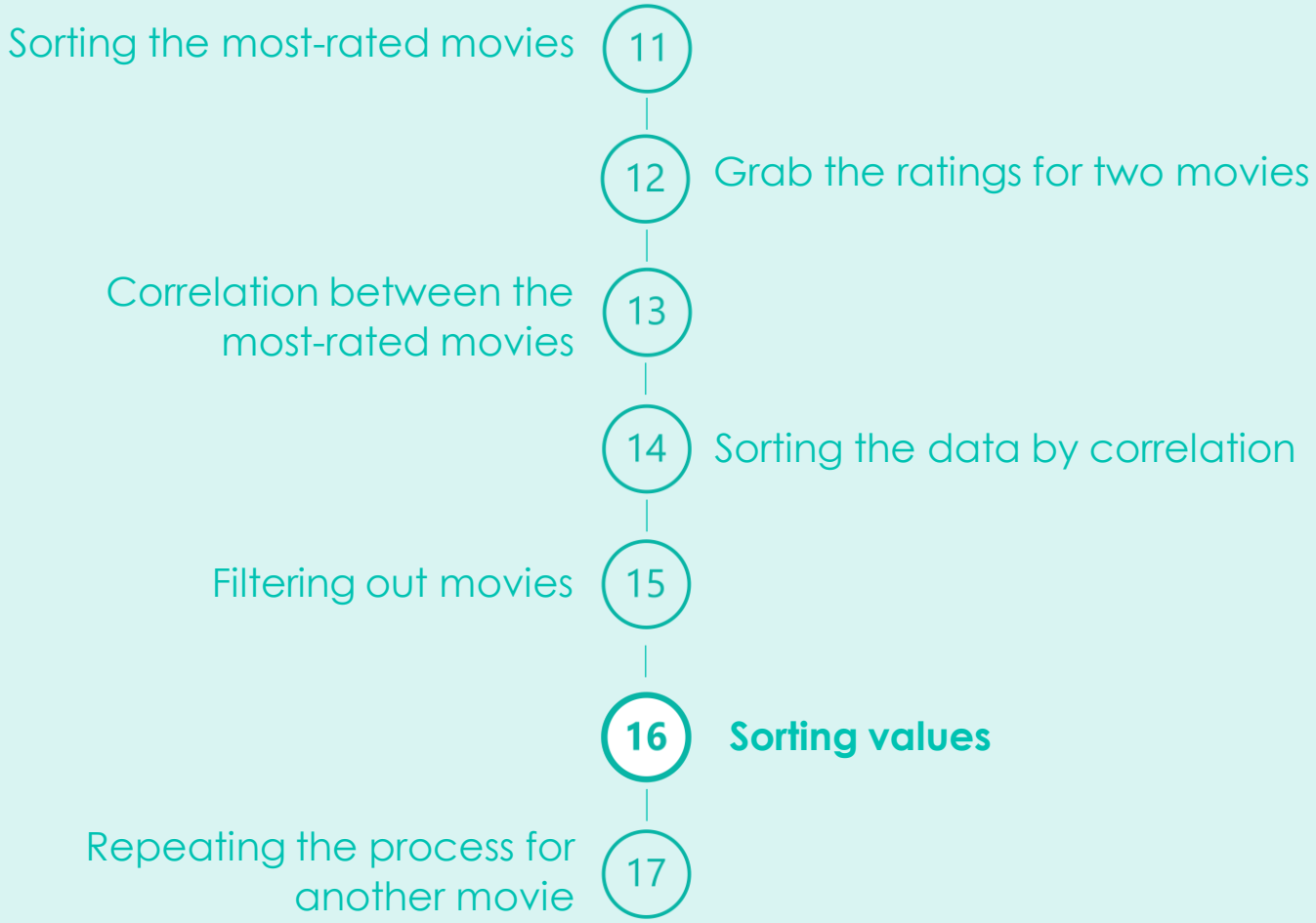
Filter out the movies having less than 100 reviews

```
In [21]: corr_starwars = corr_starwars.join(ratings['num of ratings'])
         corr_starwars.head()
```

Out[21]:

title	Correlation	num of ratings
'Til There Was You (1997)	0.872872	9
1-900 (1994)	-0.645497	5
101 Dalmatians (1996)	0.211132	109
12 Angry Men (1957)	0.184289	125
187 (1997)	0.027398	41

Meta Brains



Recommender system

Recommender System in python

Implementation step 15

Finally Filter out the titles to make sense

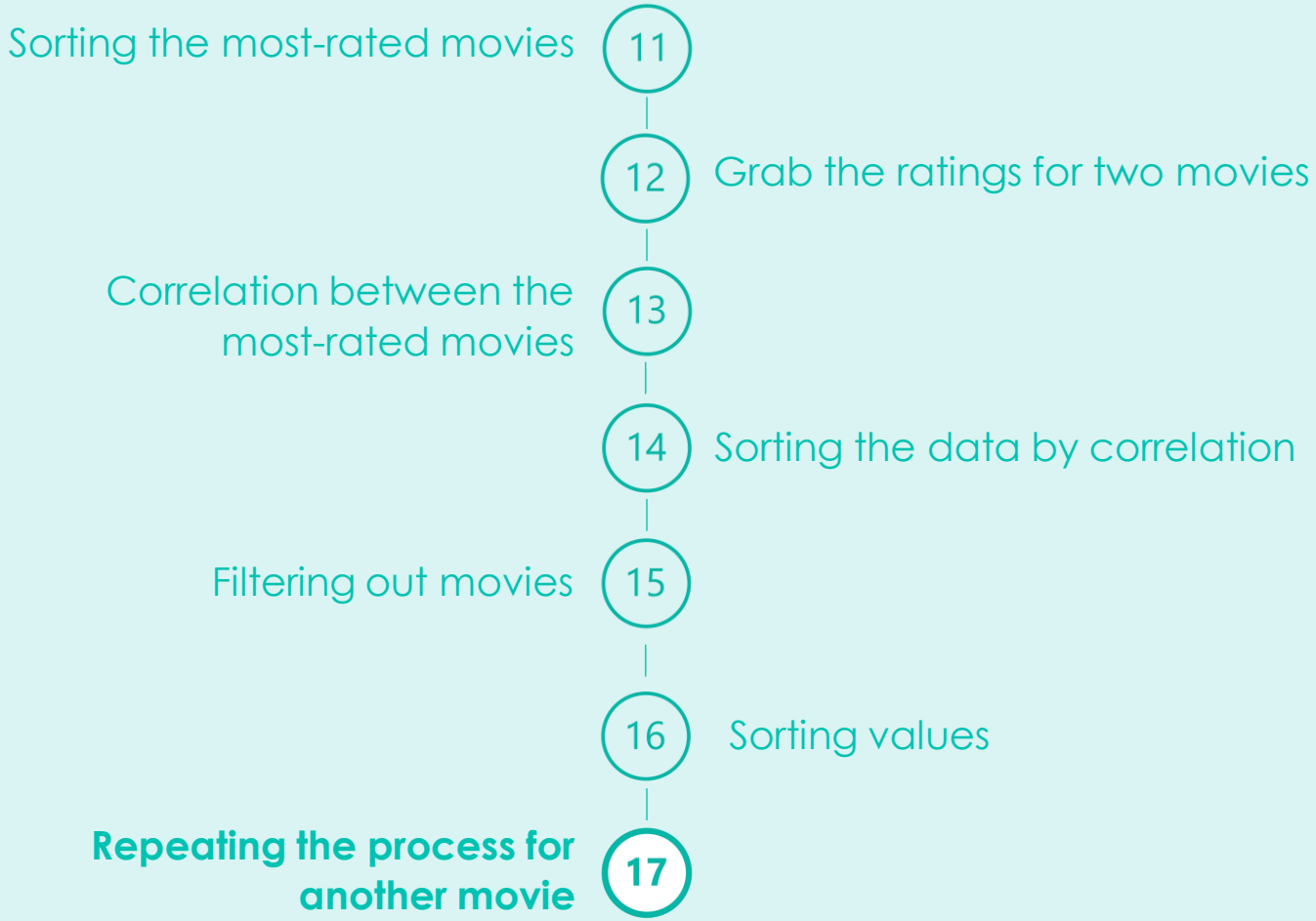
```
In [22]: corr_starwars[corr_starwars['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

Out[22]:

title	Correlation	num of ratings
Star Wars (1977)	1.000000	584
Empire Strikes Back, The (1980)	0.748353	368
Return of the Jedi (1983)	0.672556	507
Raiders of the Lost Ark (1981)	0.536117	420
Austin Powers: International Man of Mystery (1997)	0.377433	130

Meta Brains

- Introduction to Machine Learning
- Implementing ML Algorithms in Python
- Simple Linear Regression
- Multiple Linear Regression
- Classification Algorithms: K-Nearest Neighbors
- Classification Algorithms: Decision Tree
- Classification Algorithms: Logistic regression
- Clustering
- **Recommender System**
- Conclusion



Recommender system

Recommender System in python

Implementation step 16

Doing same for the movie Liar Liar (1997) (Implementation step 11-15)

In [23]:
```
corr_liarliar = pd.DataFrame(similar_to_liarliar,columns=['Correlation'])
corr_liarliar.dropna(inplace=True)
corr_liarliar = corr_liarliar.join(ratings['num of ratings'])
corr_liarliar[corr_liarliar['num of ratings']>100].sort_values('Correlation',ascending=False).head()
```

Out[23]:

title	Correlation	num of ratings
Liar Liar (1997)	1.000000	485
Batman Forever (1995)	0.516968	114
Mask, The (1994)	0.484650	129
Down Periscope (1996)	0.472681	101
Con Air (1997)	0.469828	137

Meta Brains

Quiz Time

Why are recommendation engines becoming popular?

A) Users have less time, more options and face an information overload
B) It is mandatory to have recommendation engine as per telecom rules
C) It is better to recommend than ask user to search on cell phones
D) Users don't know what they want

Quiz Time

Why are recommendation engines becoming popular?

A) Users have less time, more options and face an information overload

B) It is mandatory to have recommendation engine as per telecom rules

C) It is better to recommend than ask user to search on cell phones

D) Users don't know what they want

Meta Brains

Congratulations!

Thank You!

Meta Brains