System Lab I (CSP253)

System Lab I (CSP253)

Syllabus for Semester III

Course Code : CSP253 | **Course : Systems Lab I**

L:0Hrs.,T:0Hrs.,P:4Hrs.,Per week | **Total Credits: 02**

Course Pre-requisite : CST151 (Programming for Problem Solving)

Course Objectives

1. Introduce students with basic Python programming concepts
2. Students will learn different data structures supported by python and its applications
3. Students will learn to use rich set of standard libraries supported by python to develop complex real life python applications

Course Outcomes:

On successful completion of the course, students will be able to:

1. Design Python programs using different data and control structures.
2. Design and use Python Files ,Modules and Packages to handle complex python programs
3. Develop mathematical and scientific applications in python using numpy, scipy libraries
4. Develop small applications for web scrapping using standard libraries

System Lab I (CSP253)

Syllabus

Practicals based on the following syllabus

- Python Execution model and Basic building blocks of Python Programs/Scripts/Modules
- Various keywords, Operators , control and loop constructs used in Python
- User defined Function generation in Python
- Dealing with Python files ,Modules and Packages SciPy, an Open Source Python-based library, which is used in mathematics, scientific computing, Engineering, and technical computing.
- Developing small mathematical applications using packages like Numpy, Matplotlib etc.
- Introduction of with Web scrapping and its need
- Application development to scrape the web with the help of standard libraries like Requests and bs4(Beautiful Soup).

Text Books

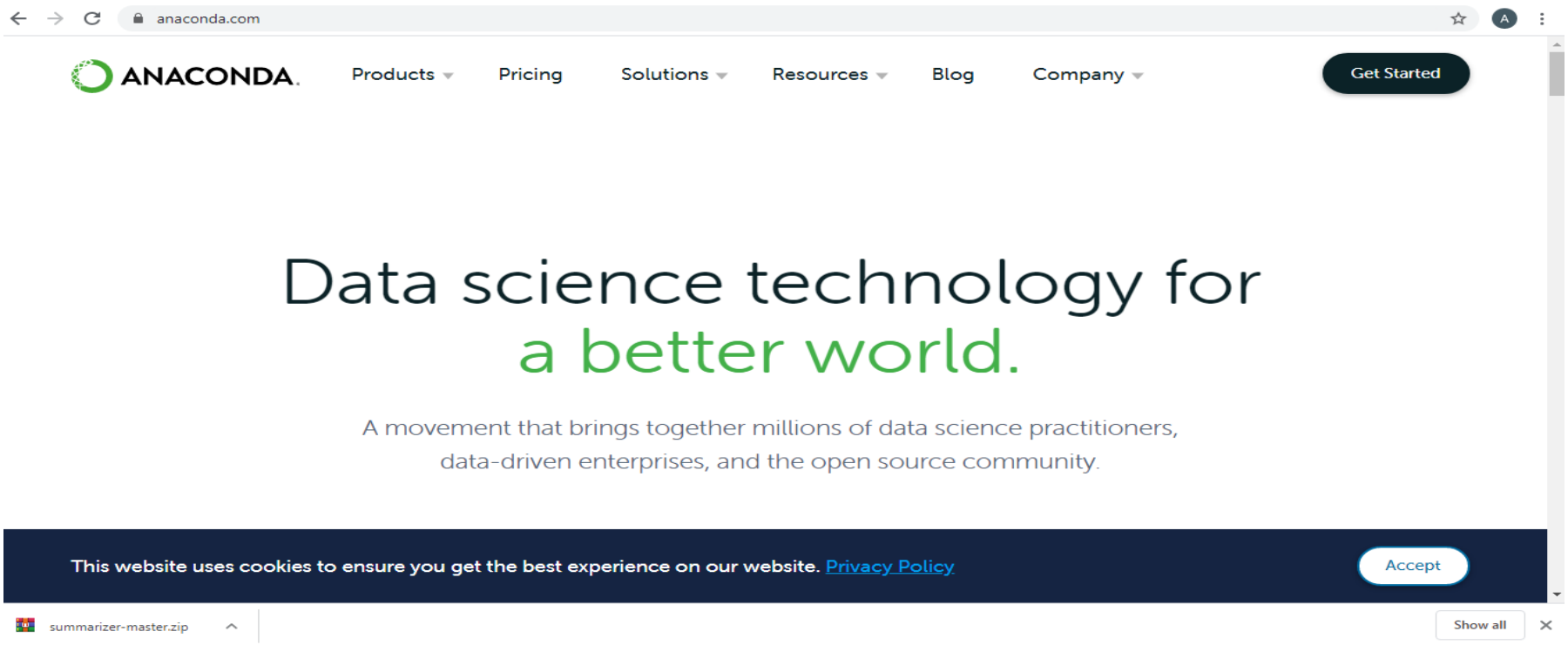
- Learning Python: Poweful object oriented programming, Mark Lutz, O'REILLY publications 5th edition
- Introduction to Computing & Problem Solving with Python Jeeva Jose and P SojanLalAscher
- Problem Solving with Algorithms and Data Structures using Python by By Brad Miller and David Ranum, 2nd addition.

Reference Books

- Allen Downey ,Jeffrey Elkner, Chris Meyers,:Learning with Python, Dreamtech Press
- The Python 3 Standard Library by Example (Developer's Library) by Doug Hellmann, second edition.

How to install Python

- open anaconda.com
- click on “Get started”



anaconda.com

ANACONDA Products Pricing Solutions Resources Blog Company

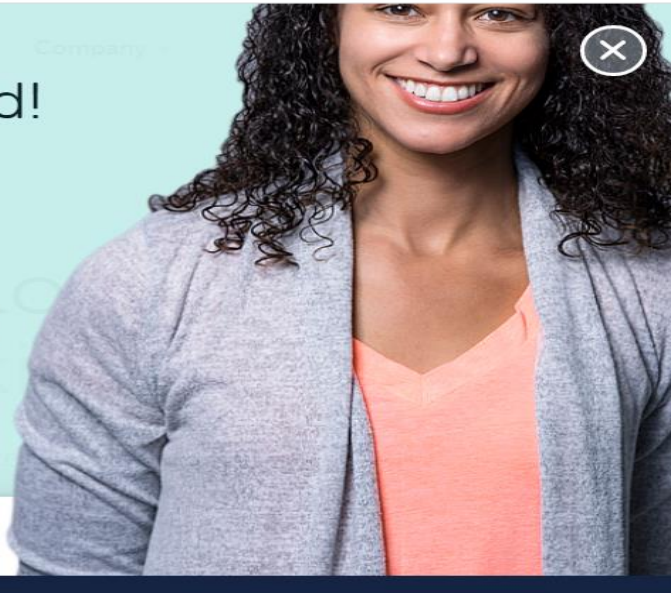
Hello! Let's get started!

See all Anaconda products

Check out the latest in data science

Request an Anaconda demo

Install Anaconda Individual Edition



Anaconda Installers

Windows

Python 3.7

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (423 MB)

Python 2.7

64-Bit Graphical Installer (413 MB)

32-Bit Graphical Installer (356 MB)

MacOS

Python 3.7

64-Bit Graphical Installer (442 MB)

64-Bit Command Line Installer (430 MB)

Python 2.7

64-Bit Graphical Installer (637 MB)

64-Bit Command Line Installer (409 MB)

Linux

Python 3.7

64-Bit (x86) Installer (522 MB)

64-Bit (Power8 and Power9) Installer (276 MB)

Python 2.7

64-Bit (x86) Installer (477 MB)

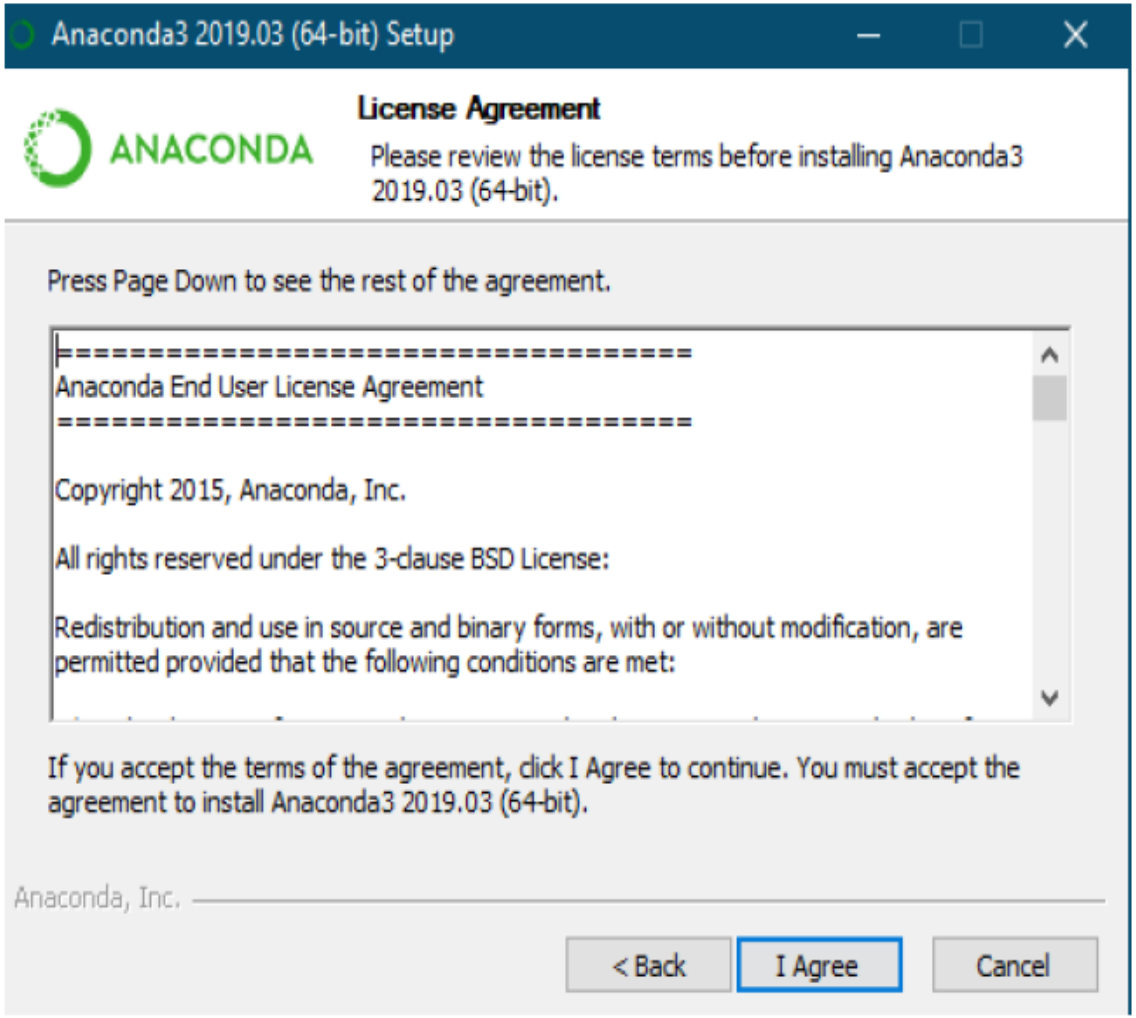
64-Bit (Power8 and Power9) Installer (295

summarizer-master.zip

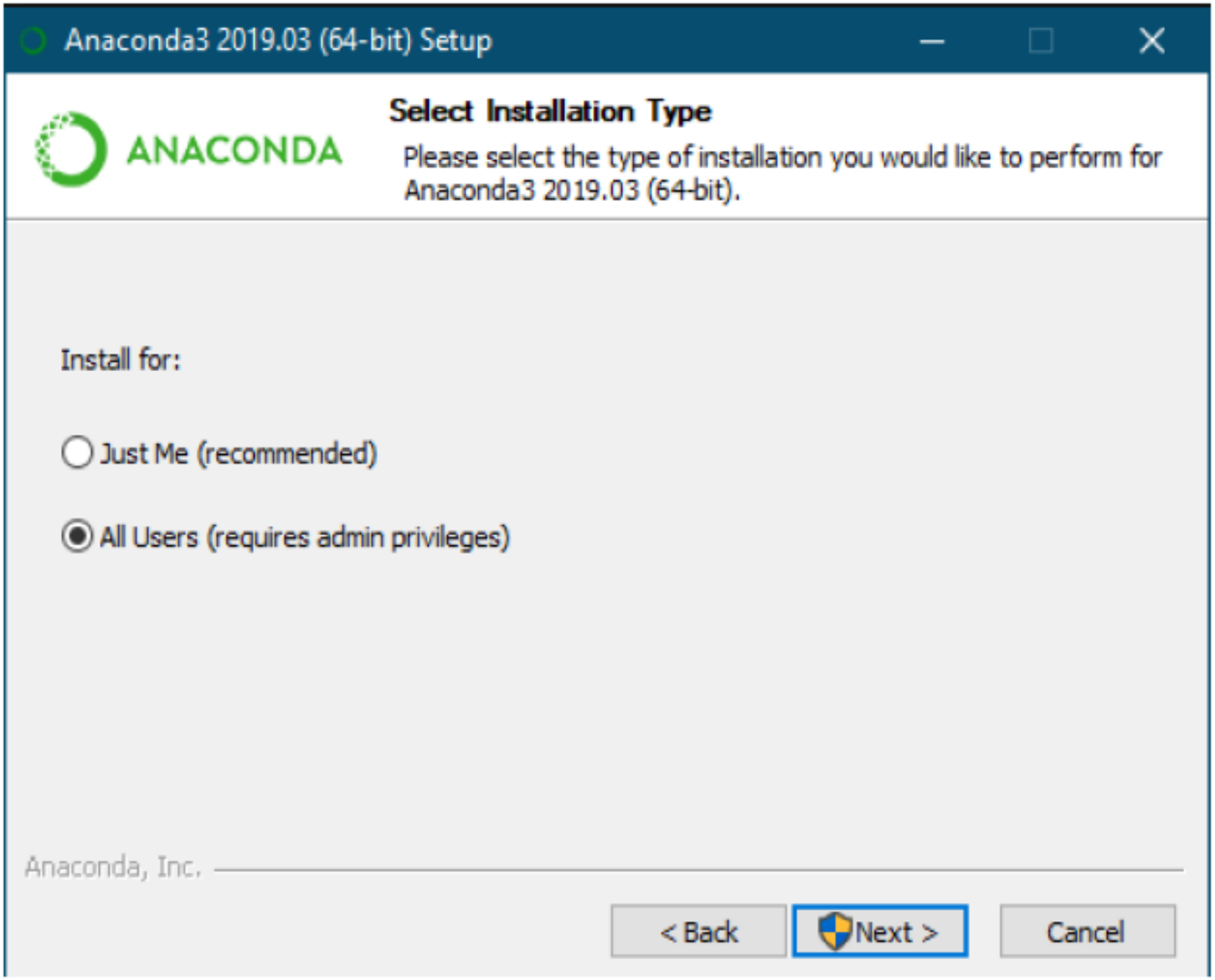
Show all

1. Open the anaconda software. It will show the installation wizard as mentioned below. Click on Next.

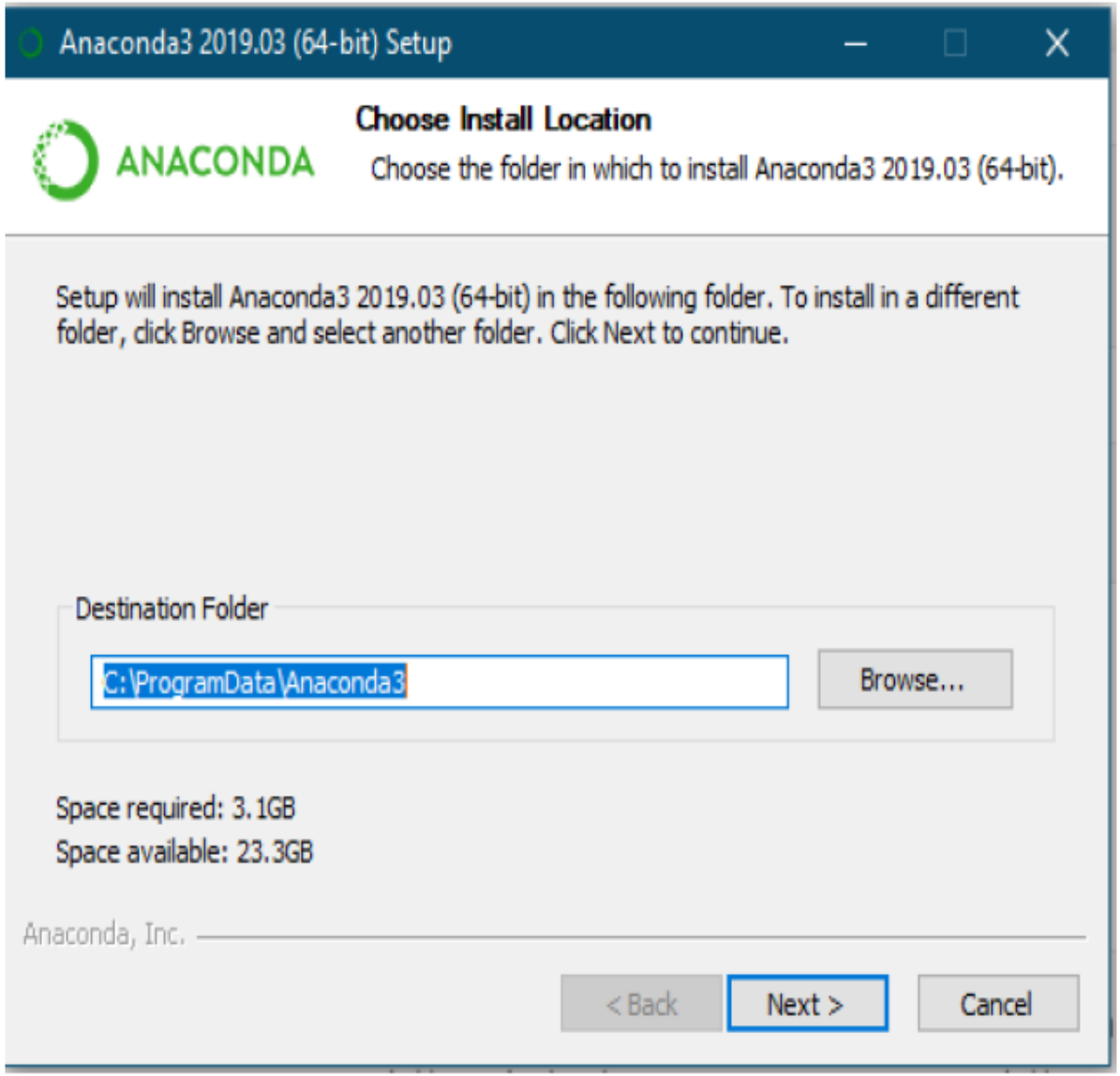
2. Click on I Agree. Move to the Next Window.



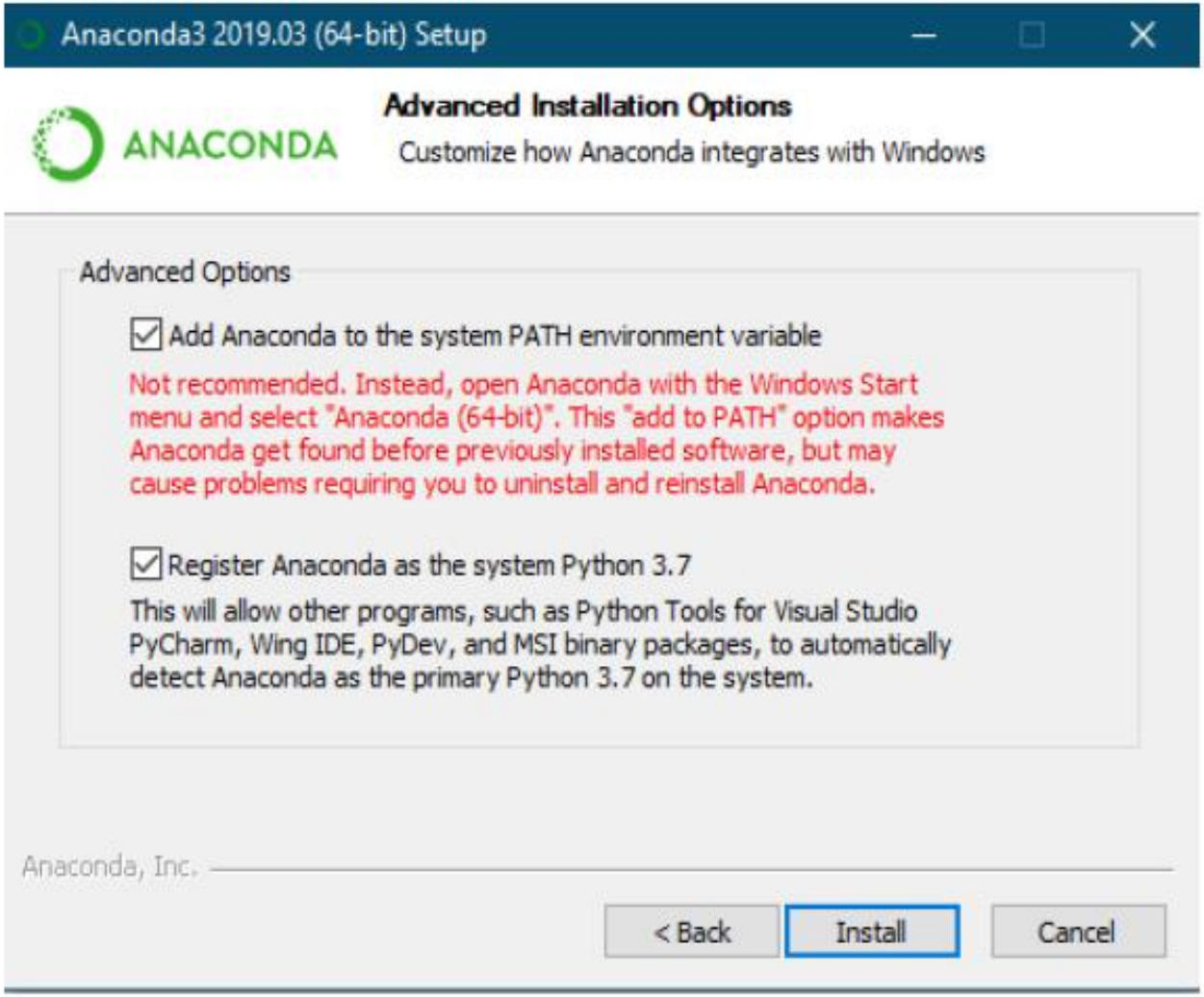
3. Select All Users. (**Note:** By Default Just Me will be selected change it to All Users). Give Admin Permissions. Click on Next & move to the Next Window.



4. It will Open a new Window showing path of Ananconda Folder. Keep it as Default. Click on Next.



5. Select Both Advanced Options as shown. & Click on Install.



6. Wait for Installation to finish complete. Check if Installation completed by Searching " **Jupyter Notebook**" on Windows Start Search Option. If opens successfully. Congratulations it worked.

Anaconda Navigator

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Learning

Community

Documentation

Developer Blog

Applications on base (root) Channels Refresh

JupyterLab
0.35.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
5.7.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Spyder
3.3.3
Scientific Python Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install

Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install

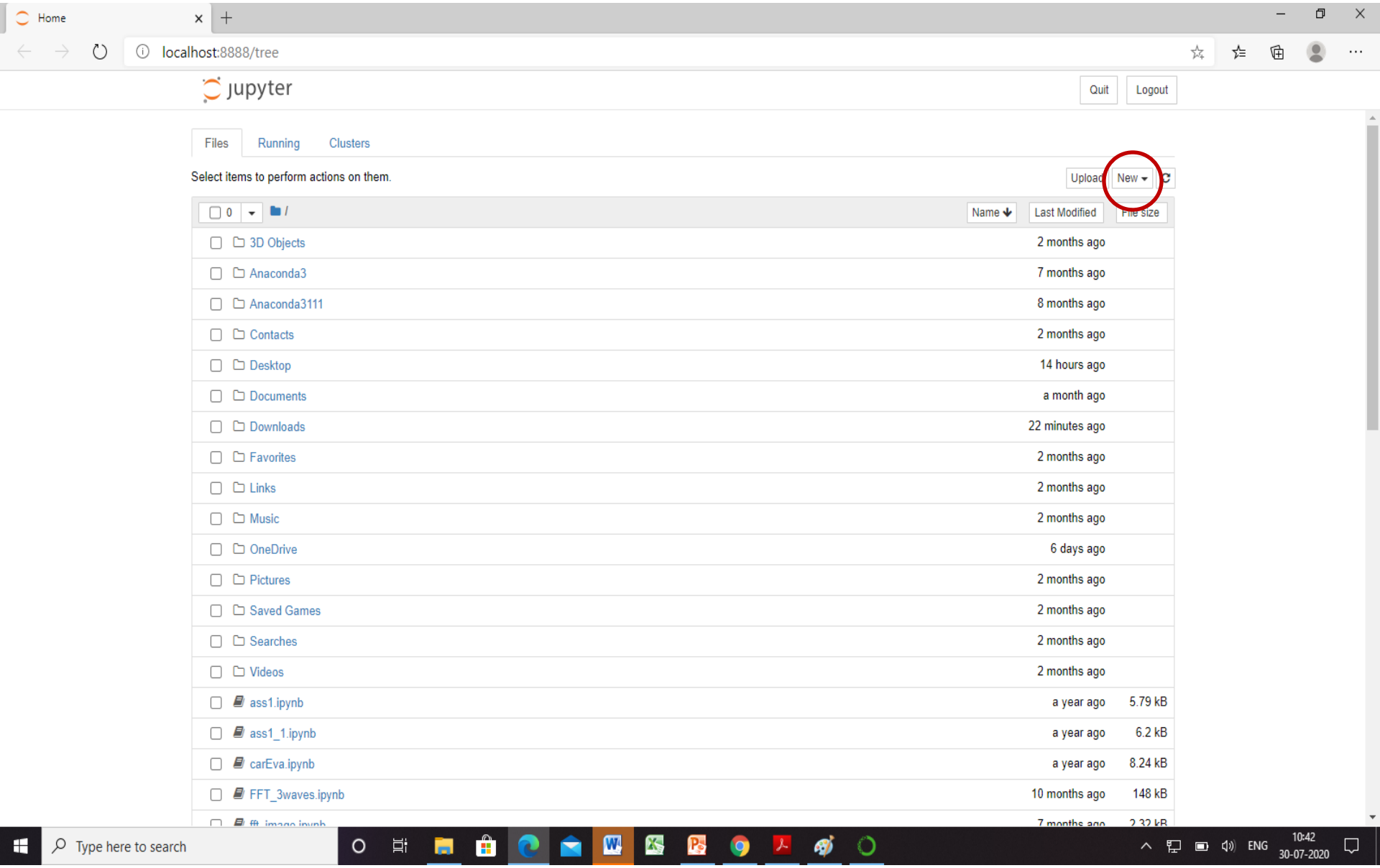
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install

VS Code
1.47.3
Streamlined code editor with support for development operations like debugging, task running and version control.
Install

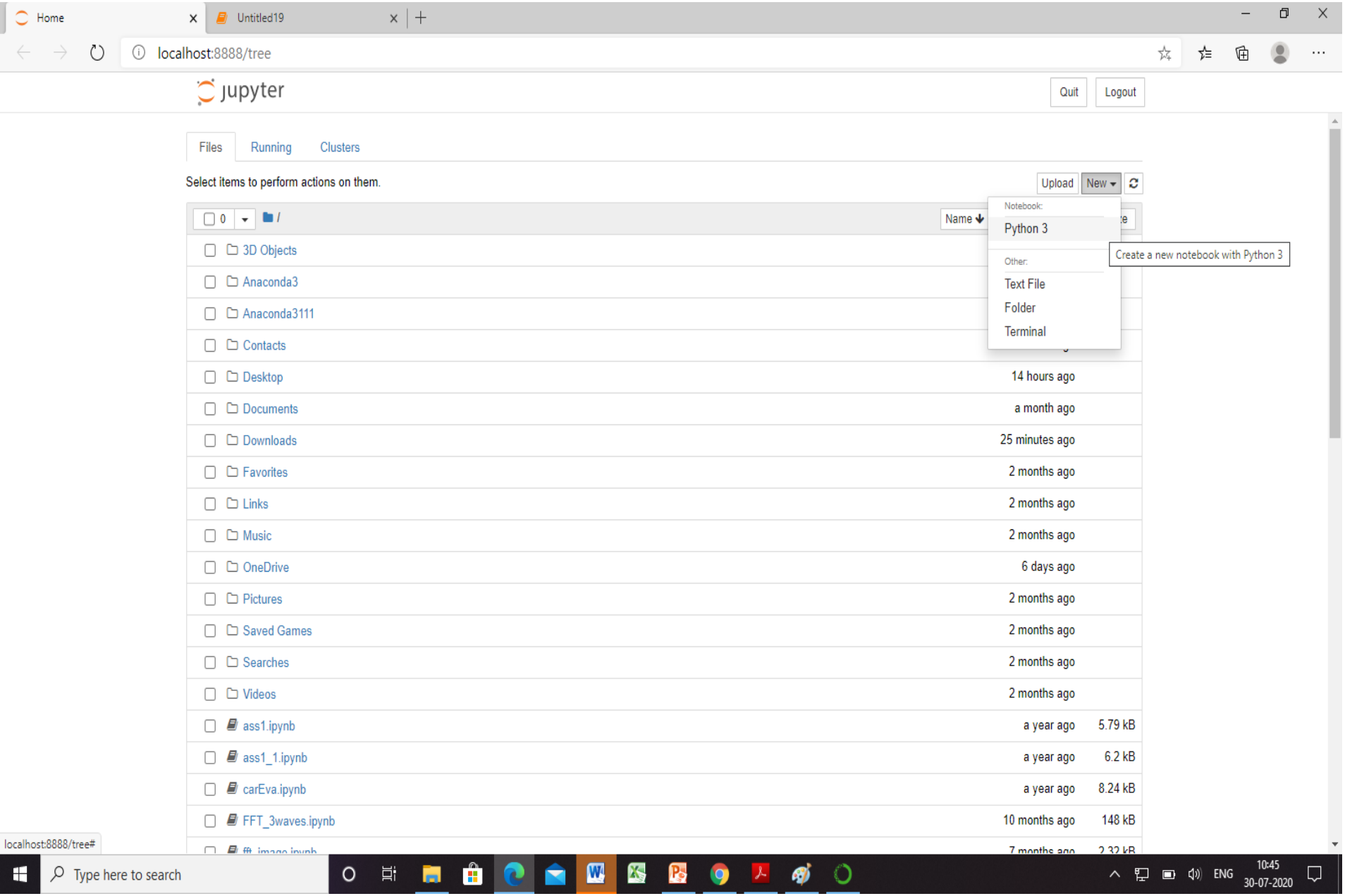
Type here to search

ENG 10:38 30-07-2020

After click on the install and launch



After click on the install and launch



Home | Untitled19

localhost:8888/notebooks/Untitled19.ipynb?kernel_name=python3

jupyter Untitled19 Last Checkpoint: 3 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [1]:
```
print ("Hello")
```

```
Hello
```

In [ ]:

Type here to search

ENG 10:48 30-07-2020

jupyter Untitled19 Last Checkpoint: 3 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [1]:
```
print ("Hello")
```

```
Hello
```

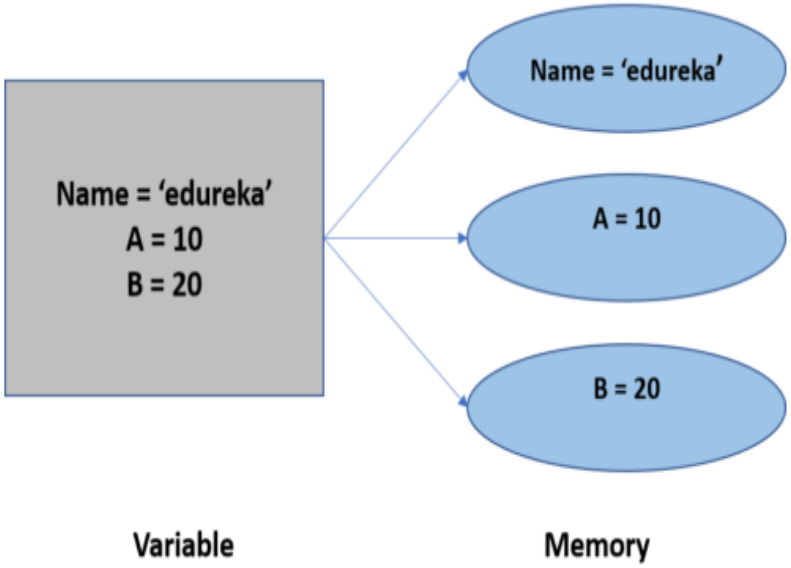
In [ ]:

Variables In Python

In a programming language, a variable is a memory location where you store a value. A Variable in python is created as soon as a value is assigned to it.

There are a certain rules that we have to keep in mind while declaring a variable:

- The variable name **cannot** start with a **number**. It can only start with a character or an underscore.
- Variables in python are case sensitive.
- They can only contain alpha-numeric characters and underscores.
- No special characters are allowed.



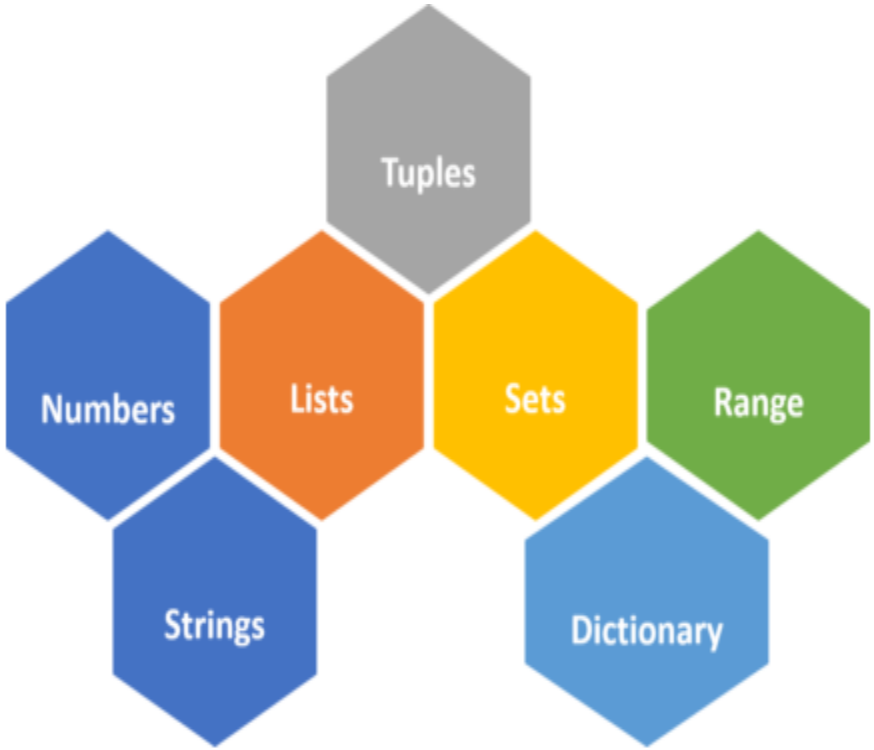
Data Types In Python

According to the properties they possess, there are mainly six data types in python. Although there is one more data type range which is often used while working with loops in python.

Numerical Data Types

Numerical data type holds numerical value. In numerical data there are 4 sub types as well. Following are the sub-types of numerical data type:

- Integers **(x = 100)**
- Float **(x = 10.25)**
- Complex Numbers **(x = 10 + 5j)**
- Boolean **(x = True)**



Python Indentation

- Indentation refers to the spaces at the beginning of a code line.
- Python uses indentation to indicate a block of code.

Syntax Error:

```
if 5 > 2:
print("Five is greater than two!")
```

```
if 5 > 2:
    print("Five is greater than two!")
```

Typecasting In Python

- Casting in python is therefore done using constructor functions:
- **int()** - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)
- **float()** - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- **str()** - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Operators In Python

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Operators In Python

Arithmetic operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment operators

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Comparison operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Identity operators

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Membership operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Bitwise operators

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

User input In Python

```
X=input(“Enter no”)
Print(x)
```

```
username = input("Enter username:")
print("Username is: " + username)
```

If, if-else, nested if In Python

```
# if
if b > a:
   print("b is greater than a")
# One line
if a > b: print("a is greater than b")
```

```
# if-else
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
# One line
print("A") if a > b else print("B")
```

```
# nested if
if x > 10:
  print("Above ten,")
  if x > 20:
     print("and also above 20!")
  else:
     print("but not above 20.")
```

while, for In Python

```
# while loop Syntax


i = 1
while i < 6:
    print(i)
    i += 1
```

```
# for loop Syntax
for x in range(6):
    print(x)


for x in range(2, 6):
    print(x)
```

range(start, end(not including), increment by)

```
for x in range(2, 30, 3):
    print(x)
```

Thank you