

Advanced Python and Data Science

Advanced Python

Jupyter Notebook

For interactively creating and presenting projects linked to AI, the Jupyter Notebook is a really powerful tool. The open source web application Jupyter Notebook allows users to create and share documents with live code, equations, visualisations, and text.

Through Anaconda, Jupyter Notebook may be installed and used most easily. The most popular Python distribution for data science, Anaconda, comes pre-installed with all the most used tools and libraries.

Introduction to Python

Guido Van Rossum, of Centrum Wiskunde & Informatica, is the inventor of the Python programming language. The language, which took its name from the 1970s BBC comedy series “Monty Python’s Flying Circus,” was made available to the general public in 1991. It can be used to programme in both an object-oriented and procedural manner. Because it offers so many features, Python is very popular.

Features of Python:

1. Easy to learn, read and maintain

Python has few keywords, simple structure and a clearly defined syntax. Python allows anyone to learn the language quickly. A program written in Python is fairly easy-to-maintain.

2. A Broad Standard library

Python has a huge bunch of libraries with plenty of built-in functions to solve a variety of problems.

3. Interactive Mode

Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

4. Portability and Compatibility

Python can run on a wide variety of operating systems and hardware platforms, and has the same interface on all platforms.

5. Extendable

We can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

6. Databases and Scalable

Python provides interfaces to all major open source and commercial databases along with a better structure and support for much larger programs than shell scripting.

Python Basics

Keywords & Identifiers:

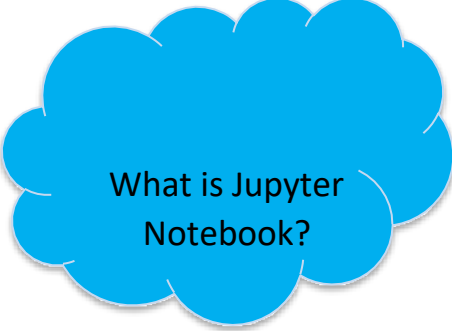
Some terms in Python have predefined meanings that the computer automatically assigns to them. These phrases are referred to as keywords. In order to avoid misunderstanding and unclear results, keywords should only be used in the default manner and cannot be changed at any point in time. The following list includes a few of the keywords:

False, class, finally, is, return, None, continue, for lambda, try, True, def, from, nonlocal, while, and, del, global, not, with, as, elif, if, or, yield, assert, else, import, pass, break, except, in, raise etc.

Recap

In this section, we will go through a quick refreshing session around Python concepts and Jupyter notebook. Along with this we will talk about newer concepts like packages, virtual environments, etc.

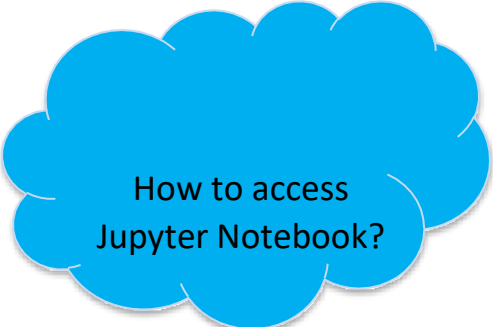
Recap 1: Jupyter Notebook



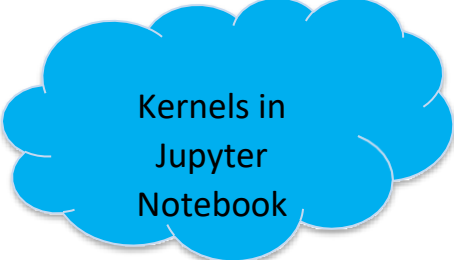
What is Jupyter Notebook?

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting AI related projects. The Jupyter project is the successor to the earlier IPython Notebook, which was first published as a prototype in 2010. Although it is possible to use many different programming languages within Jupyter Notebooks, Python remains the most commonly used language for it. In other words, we can say that the Jupyter Notebook is an open source web application that can be used to create and share documents that contain live code, equations, visualizations, and text.

The easiest way to install and start using Jupyter Notebook is through Anaconda. Anaconda is the most widely used Python distribution for data science and comes pre-loaded with all the most popular libraries and tools. With Anaconda, comes the Anaconda Navigator through which we can scroll around all the applications which come along with it. Jupyter notebook can easily be accessed using the Anaconda Prompt with the help of a local host.



How to access Jupyter Notebook?



Kernels in Jupyter Notebook

To work with Jupyter Notebook, it is necessary to have a kernel on which it operates. A kernel provides programming language support in Jupyter. IPython is the default kernel for Jupyter Notebook. Therefore, whenever we need to work with Jupyter Notebook in a virtual environment, we first need to install a kernel inside the environment in which the Jupyter notebook will run.

Introduction to Virtual Environments



What?

A virtual environment is a tool that helps to keep dependencies required by different projects separated, by creating isolated Python virtual environments for them. This is one of the most important tools that most of the Python developers use.



Why?

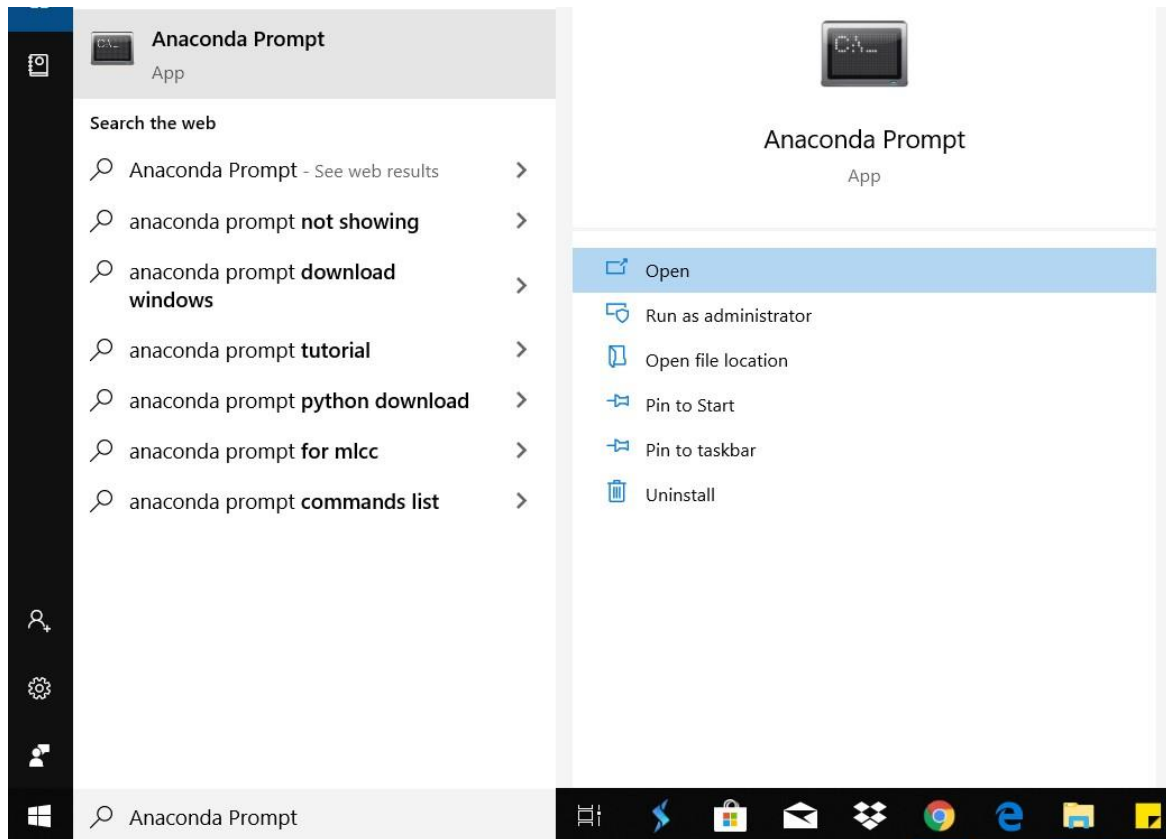
Imagine a scenario where we are working on two Python-based projects and one of them works on Python 2.7 and the other uses Python 3.7. In such situations virtual environment can be really useful to maintain dependencies of both the projects as the virtual environments will make sure that these dependencies are not conflicting with each other and no impact reaches the base environment at any point in time. Thus, different projects developed in the system might have another environment to keep their dependencies isolated from each other.



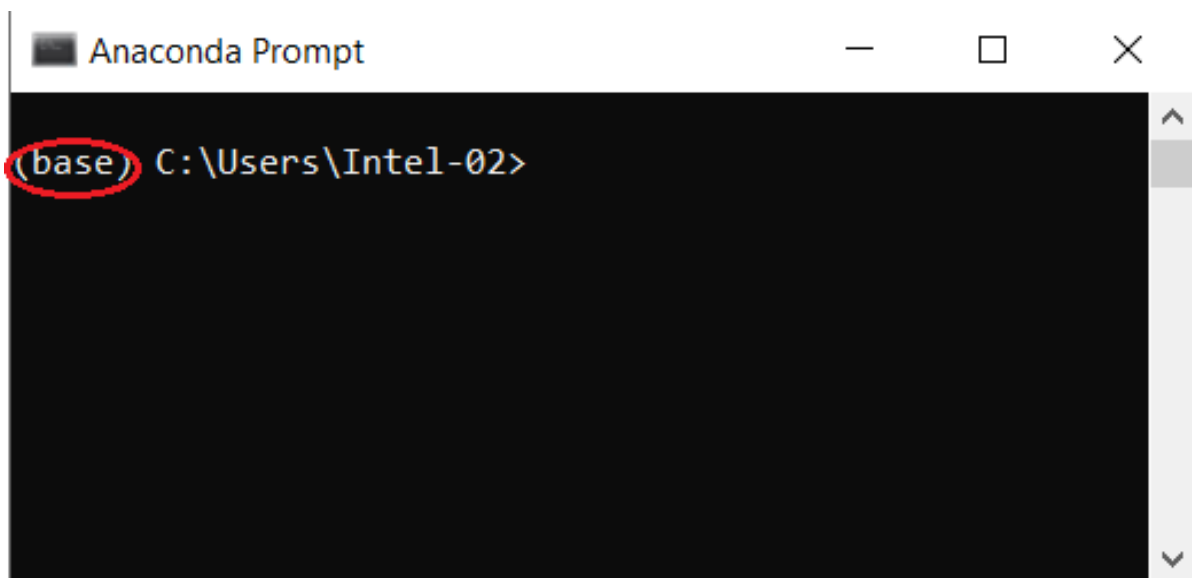
How?

Creating virtual environments is an easy task with Anaconda distribution. Steps to create one are:

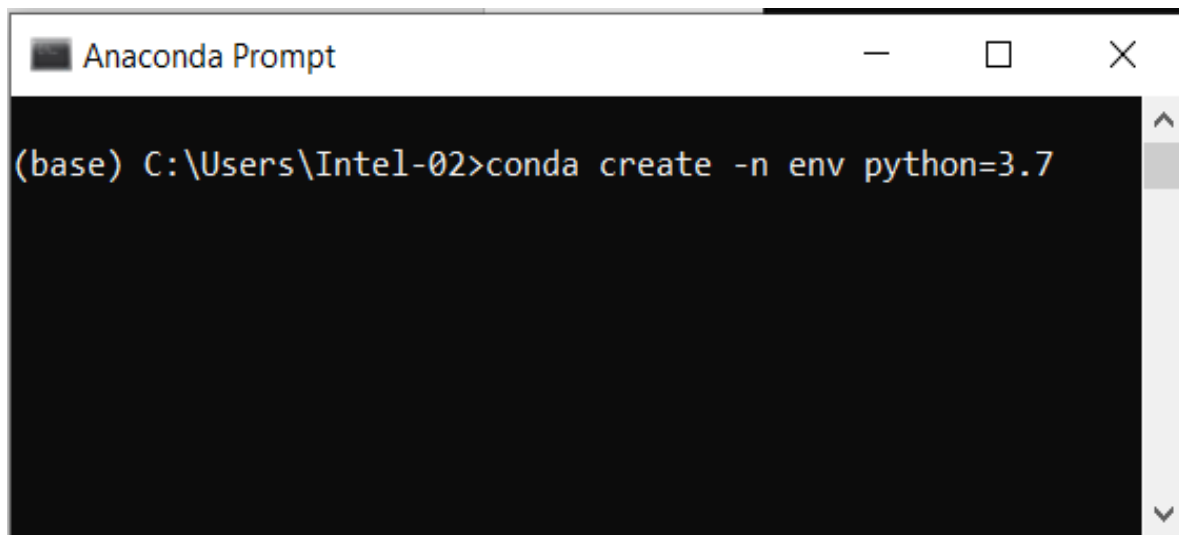
1. Open Anaconda Prompt.



2. As we open the Anaconda prompt, we can see that in the beginning of the prompt message, the term **(base)** is written. This is the default environment in which the anaconda works. Now, we can create our own virtual environment and use it so that the base does not get affected by anything that is done in the virtual environment.



3. Let us now create a virtual environment named **env**. To create the environment, write `conda create -n env python=3.7`

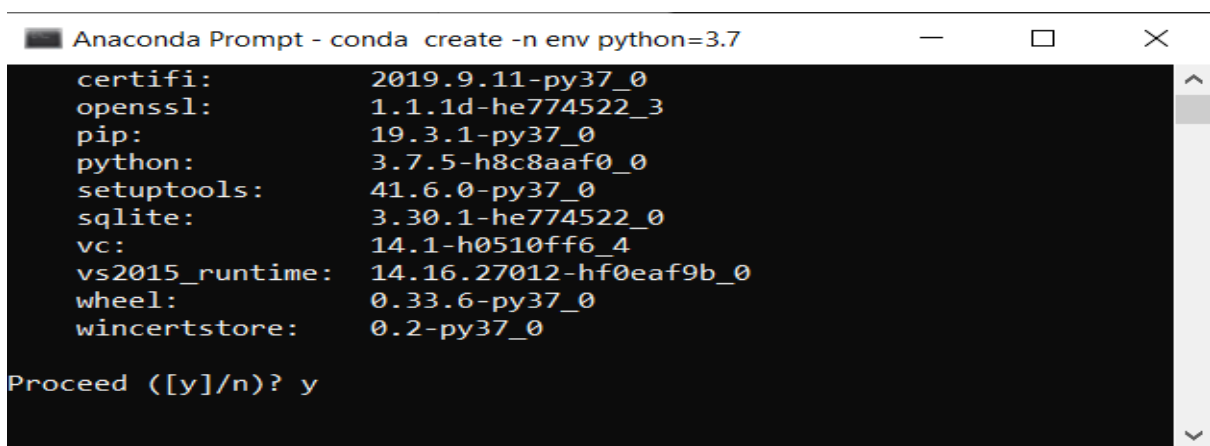


```
Anaconda Prompt

(base) C:\Users\Intel-02>conda create -n env python=3.7
```

This code will create an environment named **env** and will install Python 3.7 and other basic packages into it.

4. After some processing, the prompt will ask if we wish to proceed with installations or not. Type **Y** on it and press Enter. Once we press Enter, the packages will start getting installed in the environment.

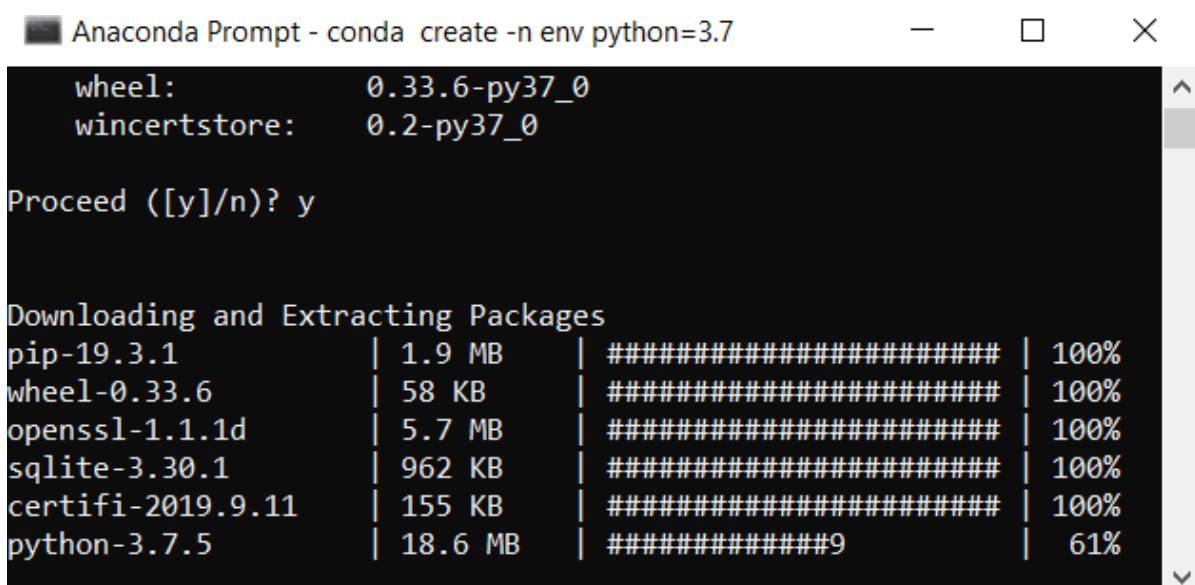


```
Anaconda Prompt - conda create -n env python=3.7

certifi:          2019.9.11-py37_0
openssl:          1.1.1d-he774522_3
pip:              19.3.1-py37_0
python:           3.7.5-h8c8aaf0_0
setuptools:       41.6.0-py37_0
sqlite:           3.30.1-he774522_0
vc:               14.1-h0510ff6_4
vs2015_runtime:   14.16.27012-hf0eaf9b_0
wheel:            0.33.6-py37_0
wincertstore:     0.2-py37_0

Proceed ([y]/n)? y
```

5. Depending upon the internet speed, the downloading of packages might take varied time. The processing screen will look like this: Once all the packages are downloaded and installed, we will get a message like this:



```
Anaconda Prompt - conda create -n env python=3.7

wheel:            0.33.6-py37_0
wincertstore:     0.2-py37_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
pip-19.3.1         | 1.9 MB | ##### | 100%
wheel-0.33.6       | 58 KB | ##### | 100%
openssl-1.1.1d     | 5.7 MB | ##### | 100%
sqlite-3.30.1      | 962 KB | ##### | 100%
certifi-2019.9.11  | 155 KB | ##### | 100%
python-3.7.5       | 18.6 MB | #####9 | 61%
```

```
Anaconda Prompt
ca-certificates-2019 | 163 KB | ##### | 100%
setuptools-41.6.0    | 687 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate env
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:\Users\Intel-02>
```

6. This shows that our environment called **env** has been successfully created. Once an environment has been successfully created, we can access it by writing the following:

conda activate env

```
Anaconda Prompt

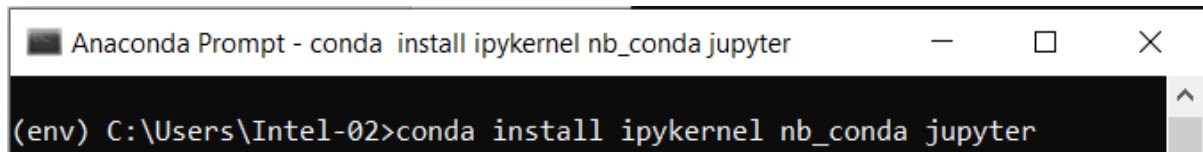
(base) C:\Users\Intel-02>conda activate env
(env) C:\Users\Intel-02>
```

This would activate the virtual environment and we can see the term written in brackets has changed from **(base)** to **(env)**. Now our virtual environment is ready to be used.

But, to open and work with Jupyter Notebooks in this environment, we need to install the packages which help in working with Jupyter Notebook. These packages get installed by default in the base environment when Anaconda gets installed.

To install Jupyter Notebook dependencies, we need to activate our virtual environment **env** and write:

```
conda install ipykernel nb_conda jupyter
```



```
Anaconda Prompt - conda install ipykernel nb_conda jupyter
(env) C:\Users\Intel-02>conda install ipykernel nb_conda jupyter
```

It will again ask if we wish to proceed with the installations, type **Y** to begin the installations. Once the installations are complete, we can start working with Jupyter notebooks in this environment.

Recap 2: Introduction to Python

In class 9, we were introduced to Python as the programming language which will be used for working around AI. Let us recall the basics of Python.

What?

Python is a programming language which was created by Guido Van Rossum in Centrum Wiskunde & Informatica. The language was publicly released in 1991 and it got its name from a BBC comedy series from 1970s – ‘Monty Python’s Flying Circus’. It can be used to follow both procedural approach and object-oriented approach of programming. Python has a lot of functionalities which makes it so popular to use.

Why?

Artificial intelligence is the trending technology of the future. We can see so many applications around us. If we as individuals would also like to develop an AI application, we will need to know a programming language. There are various programming languages like Lisp, Prolog, C++, Java and Python, which can be used for developing applications of AI. Out of these, Python gains a maximum popularity because of the following reasons:

Easy to learn, read and maintain

Python has few keywords, simple structure and a clearly defined syntax. Python allows anyone to learn the language quickly. A program written in Python is fairly easy-to-maintain.

A Broad Standard library

Python has a huge bunch of libraries with plenty of built-in functions to solve a variety of problems.

Interactive Mode

Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portability and Compatibility

Python can run on a wide variety of operating systems and hardware platforms, and has the same interface on all platforms.

Extendable

We can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Python provides interfaces to all major open source and commercial databases along with a better structure and support for much larger programs than shell scripting.

Applications of Python

There exist a wide variety of applications when it comes to Python. Some of the applications are:



Recap 3: Python Basics

In class 9, as Python was introduced, we also discussed about some basic Python syntaxes which can help us in writing codes in Python language. Let us brush up all the concepts once and see how we can use them in coding.

1. Printing Statements

We can use Python to display outputs for any code we write. To print any statement, we use **print()** function in Python.

2. Python Statements and Comments

Instructions written in the source code to execute are known as statements. These are the lines of code which we write for the computer to work upon. For example, if we wish to print the addition of two numbers, say 5 and 10, we would simply write:

```
print(5+10)
```

This is a Python statement as the computer would go through it and do the needful (which in this case would be to calculate 5+10 and print it on the output screen)

On the other hand, there exist some statements which do not get executed by the computer. These lines of code are skipped by the machine. They are known as comments. Comments are the statements which are incorporated in the code to give a better understanding of code statements to the user. To write a comment in Python, one can use # and then write anything after it. For example:

```
# This is a comment and will not be read by the machine.
```

```
print(5+10) # This is a statement and the machine will print the summation.
```

Here, we can see that the first line is a comment as it starts with #. In the second line, we have an executable statement followed by a comment which is written to explain the code. In this way, we can add comments into our code so that anyone can understand the gist of it.

3. Keywords & Identifiers

In Python, there exist some words which are pre-defined and carry a specific meaning for the machine by default. These words are known as keywords. Keywords cannot be changed at any point in time and should not be used any other way except the default one, otherwise they create confusion and might result in ambiguous outputs. Some of the Keywords are mentioned below:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Note that keywords are case-sensitive.

An identifier is any word which is variable. Identifiers can be declared by the user as per their convenience of use and can vary according to the way the user wants. These words are not defined and can be used in any way. **Keywords cannot be used as identifiers.** Some examples of keywords can be: **count, interest, x, ai_learning, Test**, etc. Identifiers are also case-sensitive hence an identifier named as **Test** would be different from an identifier named **test**.

4. Variables & Datatypes

A variable is a named location used to store data in the memory. It is helpful to think of variables as a container that holds data which can be changed later throughout programming. Just like in Mathematics, in Python too we can use variables to store values in it. The difference here is, that in Python, the variables not only store numerical values, but can also contain different types of data.

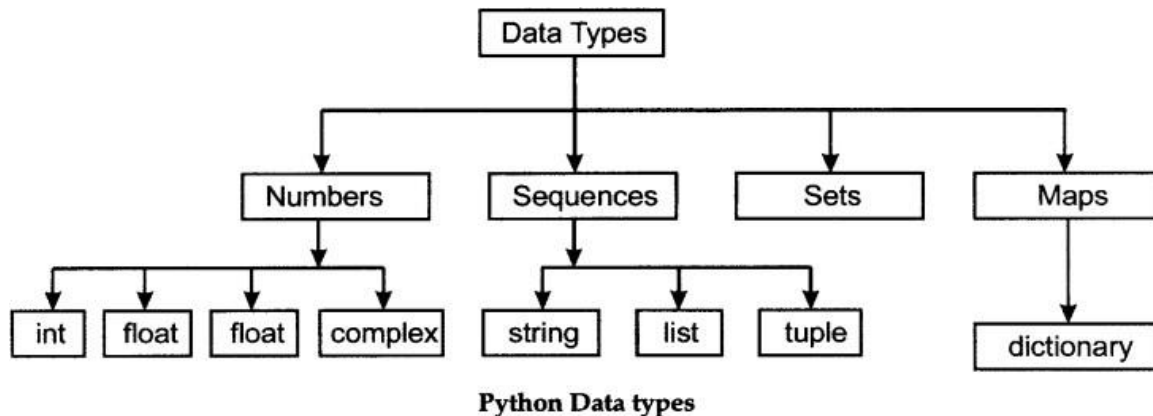
For example:

`X = 10` # X variable contains numerical data

`Letters = 'XYZ'` # Letters variable contains alphabetic data
`number = 13.95` #

number variable contains a decimal value
`word = 'k'` # word variable contains a character

All of these variables contain different types of data in them. The type of data is defined by the term datatype in Python. There can be various types of data which are used in Python programming. Hence, the machine identifies the type of variable according to the value which is stored inside it. Various datatypes in Python can be:



5. Python inputs

In Python, not only can we display the output to the user, but we can also collect data from the user and can pass it on to the Python script for further processing. To collect the data from the user at the time of execution, **input()** function is used. While using the input function, the datatype of the expected input is required to be mentioned so that the machine does not interpret the received data in an incorrect manner as the data taken as input from the user is considered to be a string (sequence of characters) by default.

For example:

`Str = input(<String>)` # Python expects the input to be of string datatype

`Number = int(input(<string>))` # Input string gets converted to an integer value before assignment

`Value = float(input(<String>))` # Input string gets converted to a decimal value before assignment

6. Python Operators

Operators are special symbols which represent computation. They are applied on operand(s), which can be values or variables. Some operators can behave differently on different data types. Operators when applied on operands form an expression. Operators are categorized as Arithmetic, Relational, Logical and Assignment. Value and variables when used with operators are known as operands.

a. Arithmetic Operators

Operator	Meaning	Expression	Result
+	Addition	10 + 20	30
-	Subtraction	30 - 10	20
*	Multiplication	30 * 100	300
/	Division	30 / 10	20.0
//	Integer Division	25 // 10	2
%	Remainder	25 % 10	5
**	Raised to power	3 ** 2	9

b. Conditional Operators

Operator	Meaning	Expression	Result
>	Greater Than	20 > 10	True
		15 > 25	False
<	Less Than	20 < 45	True
		20 < 10	False
==	Equal To	5 == 5	True
		5 == 6	False
!=	Not Equal to	67 != 45	True
		35 != 35	False
>=	Greater than or Equal to	45 >= 45	True
		23 >= 34	False
<=	Less than or equal to	13 <= 24	True
		13 <= 12	False

c. Logical Operators

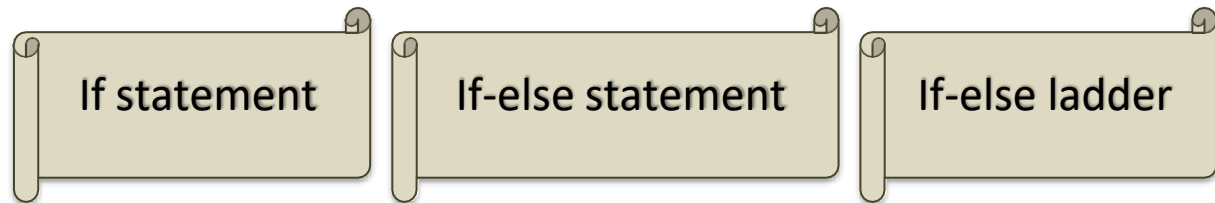
Operator	Meaning	Expression	Result
and	And operator	True and True	True
		True and False	False
or	Or operator	True or False	True
		False or False	False
not	Not Operator	not False	True
		not True	False

d. Assignment Operators

Operator	Expression	Equivalent to
=	X=5	X = 5
+=	X +=5	X = X + 5
-=	X -= 5	X = X - 5
*=	X *= 5	X = X * 5
/=	X /= 5	X = X / 5

7. Conditional Statements

While coding in Python, a lot of times we need to take decisions. For example, if a person needs to create a calculator with the help of a Python code, he/she needs to take in 2 numbers from the user and then ask the user about which function he/she wishes to operate. Now, according to the user's choice, the selection of function would change. In this case, we need the machine to understand what should happen when. This is where conditional statements help. Conditional statements help the machine in taking a decision according to the condition which gets fulfilled. There exist different types of conditional statements in Python. Some of them are:

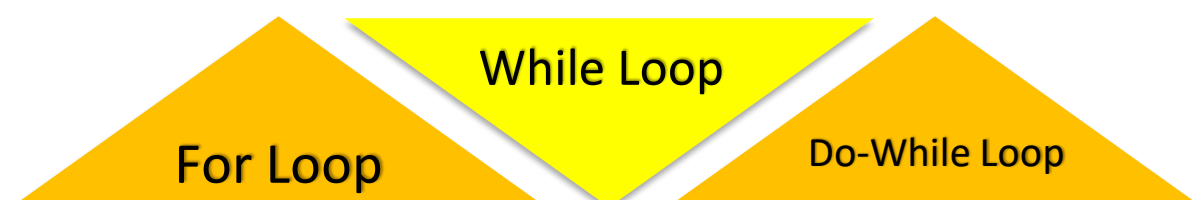


According to the number of conditions and their dependency on each other, the relevant type of conditional statement is used.

8. Looping

A lot of times, it happens that a task needs to be executed multiple number of times. For example, we need to print hello 10 times on the output screen. One way of doing this is writing 10 print statements. But this is time and space consuming. The other way, which is more efficient, is to use loop statements. The loop statements help in iterating statements or a group of statements as many times as it is asked for. In this case, we will simply write a loop which would start counting from 1 to 10. At every count, it will print hello once on the screen and as soon as it reaches 10, the loop will stop executing. All this can be done by just one loop statement.

Various types of looping mechanisms are available in Python. Some of them are:



These were some of the basic concepts for writing a code in Python. We can explore these concepts further by going through the experiential Jupyter notebook for this chapter. In that notebook, we will get to explore Python basic concepts and we can also work around them to develop better understanding around it.

Python Packages

A package is nothing but a space where we can find codes or functions or modules of similar type. There are various packages readily available to use for free (perks of Python being an open-sourced language) for various purposes.

To use any package in Python, we need to install it. Installing Python packages is easy. Steps for package installation are:

1. Open Anaconda Navigator and activate your working environment.
2. Let us assume we wish to install the numpy package. To install this package, simply write:

`conda install numpy`

3. It will ask us to type Y if we wish to proceed with the installations. As soon as we type Y, the installations will start and our package will be installed in our selected environment.
4. We can also install multiple packages all at once by mentioning all of them in one line. For example, if we wish to install numpy, pandas and matplotlib package in our working environment. For this, simply write:

`conda install numpy pandas matplotlib`

This code will install these three packages altogether in our environment.

Now, once the packages are installed, we can start using them by importing them in the file where they are required. As soon as we open our Jupyter Notebook, include the package in the notebook by writing the import command. Importing a package can be done in various ways:

`import numpy`

Meaning: Import numpy in the file to use its functionalities in the file to which it has been imported.

`import numpy as np`

Meaning: Import numpy and refer to it as np wherever it is used.

`from numpy import array`

Meaning: import only one functionality (array) from the whole numpy package. While this gives faster processing, it limits the package's usability.

`from numpy import array as arr`

Meaning: Import only one functionality (array) from the whole numpy package and refer to it as arr wherever it is used. Some of the readily available packages are:

NumPy

- A package created to work around numerical arrays in python.
- Handy when it comes to working with large numerical databases and calculations around it.

OpenCV

- An image processing package which can explicitly work around images and can be used for image manipulation and processing like cropping, resizing, editing, etc.

Matplotlib

- A package which helps in plotting the analytical (numerical) data in graphical form.
- It helps the user in visualizing the data thereby helping them in understanding it better.

NLTK

- NLTK stands for Natural Language Tool Kit and it helps in tasks related to textual data.
- It is one of the most commonly used package for Natural Language Processing.

Pandas

- A package which helps in handling 2-dimensional data tables in python.
- It is useful when we need to work with data from excel sheets and other databases.

To develop a better understanding around these packages, let us go through the Jupyter Notebook of package exploration and see how these packages can be used in Python.