What is Python?

An all-purpose, general language that works on multiple platforms

High level and easy to learn.

More commonly used for machine learning & predictive modeling.

Open source and free to learn and use more commonly by developers.

Why Is Python So Popular?

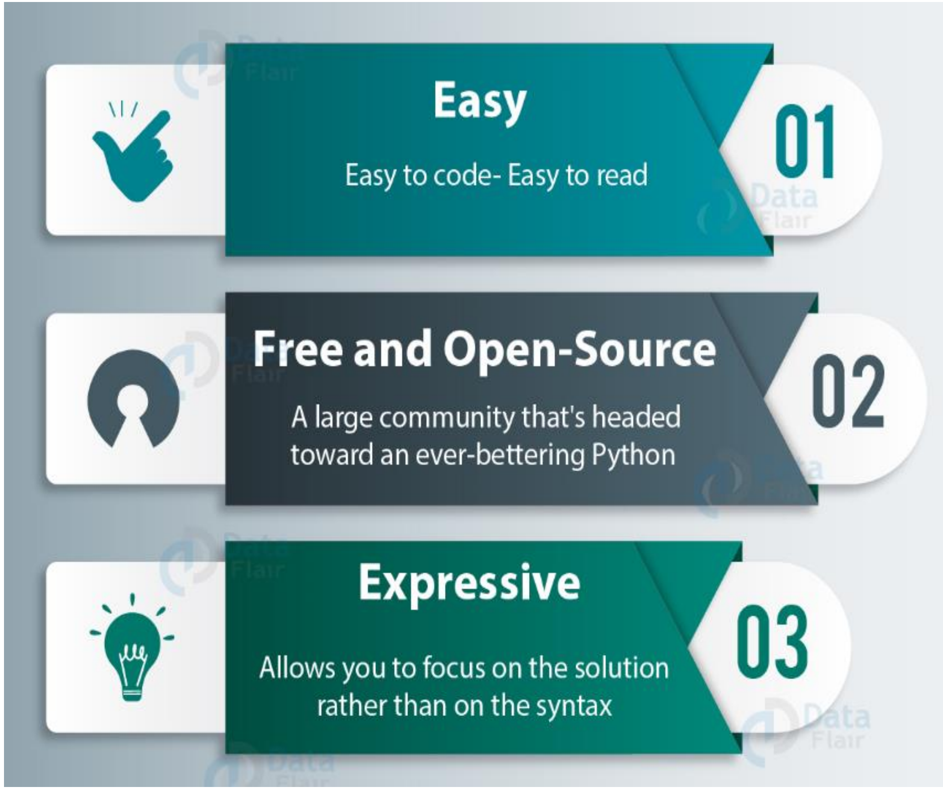
Java

- public class Main { public static void main(String[] args) { System.out.println("hello world"); } }

Python

- print(‘hello world’)

Advantages of Python



Easy 01

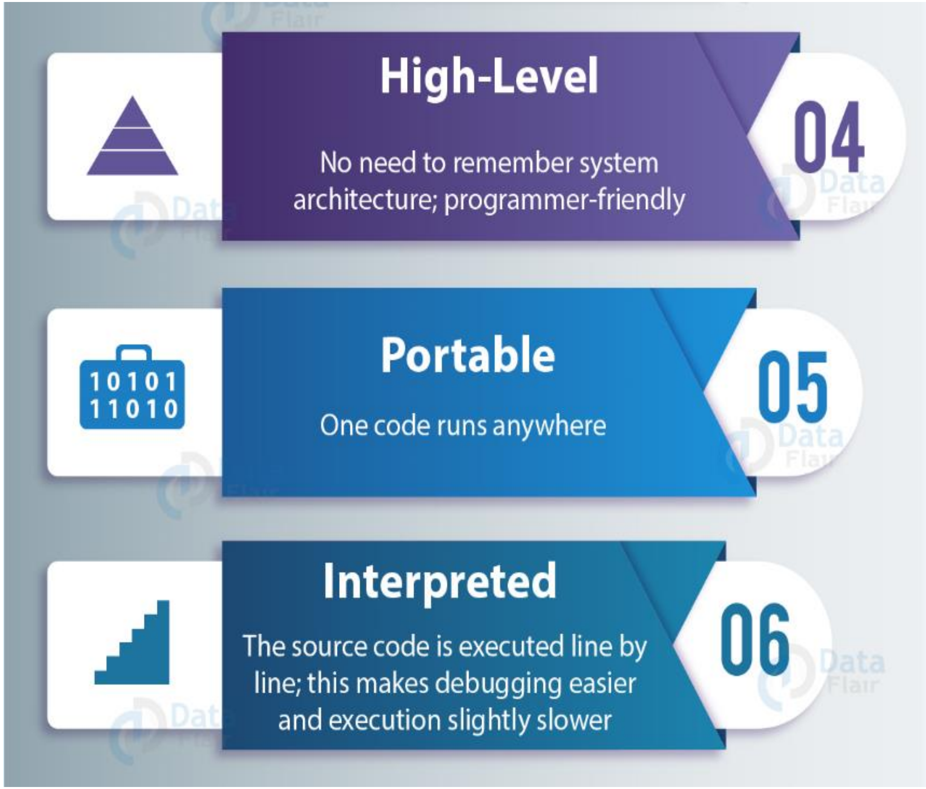
Easy to code- Easy to read

Free and Open-Source 02

A large community that's headed toward an ever-bettering Python

Expressive 03

Allows you to focus on the solution rather than on the syntax



High-Level 04

No need to remember system architecture; programmer-friendly

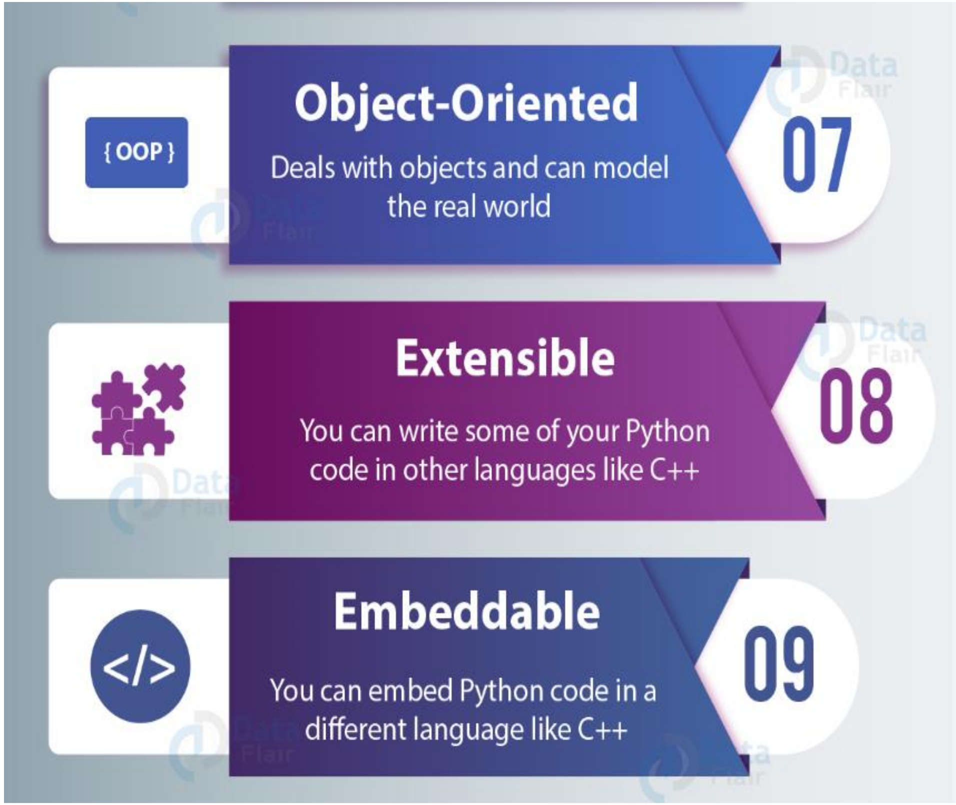
Portable 05

One code runs anywhere

Interpreted 06

The source code is executed line by line; this makes debugging easier and execution slightly slower

Advantages of Python



{ OOP }

Object-Oriented 07

Deals with objects and can model the real world

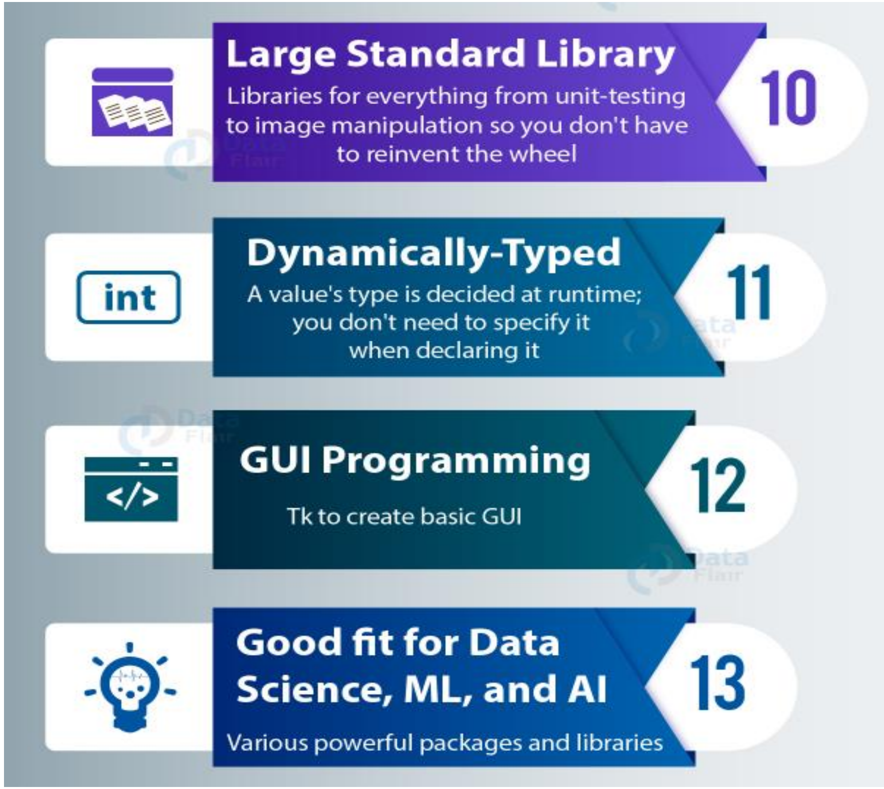
Extensible 08

You can write some of your Python code in other languages like C++

</>

Embeddable 09

You can embed Python code in a different language like C++



Large Standard Library 10

Libraries for everything from unit-testing to image manipulation so you don't have to reinvent the wheel

int

Dynamically-Typed 11

A value's type is decided at runtime; you don't need to specify it when declaring it

</>

GUI Programming 12

Tk to create basic GUI

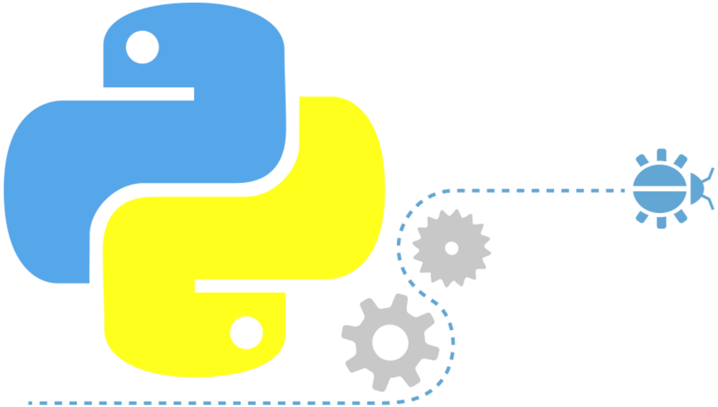
Good fit for Data Science, ML, and AI 13

Various powerful packages and libraries

Python Environment Setup and Essentials

In initial lectures of Python training, the focus will be on logic building and getting familiar with basic programming fundamentals.

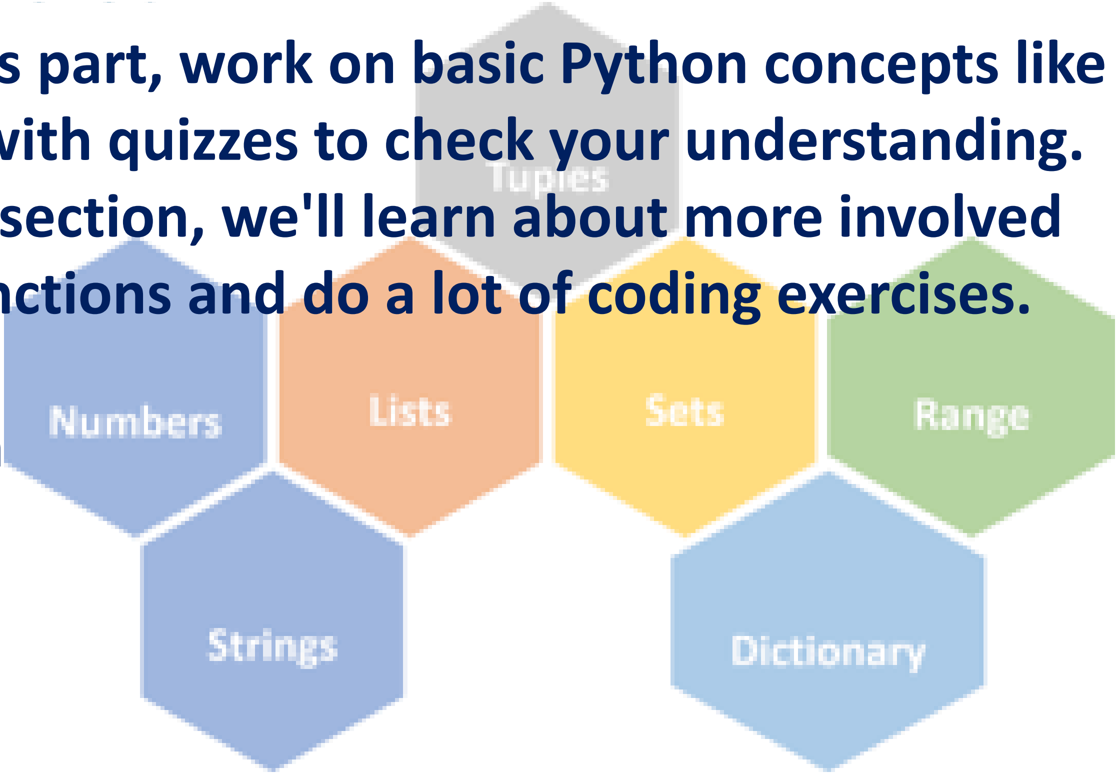
- ❑ Getting started
- ❑ Operators
- ❑ Conditions statements
- ❑ Looping in Python



Data Types in Python

After first fundamental’s part, work on basic Python concepts like strings and lists, along with quizzes to check your understanding. Towards the end of the section, we'll learn about more involved topics like loops and functions and do a lot of coding exercises.

❑ Data Types in python
❑ Strings
❑ Lists And Tuples
❑ Dictionary, Sets



Functions & Modules

The topics covered will be the backbone for the rest of the course and are also crucial for an interview perspective.

- ❑ **Functions ,**
- ❑ **Variables and their Scope**
- ❑ **Modules**
- ❑ **Object Orientation in Python (OOPs)**

Other Libraries

Here we will be learning more import concepts of File and Exception handling. This part gives you flexibility to work with any kind of files and handle any error all together.

- ❑ File Handling
- ❑ Pickle & OS Module
- ❑ Regular expression
- ❑ DateTime Module
- ❑ Exception Handling

Installation of Python on Windows / Linux

1. For Windows

   - Download Python from www.python.org

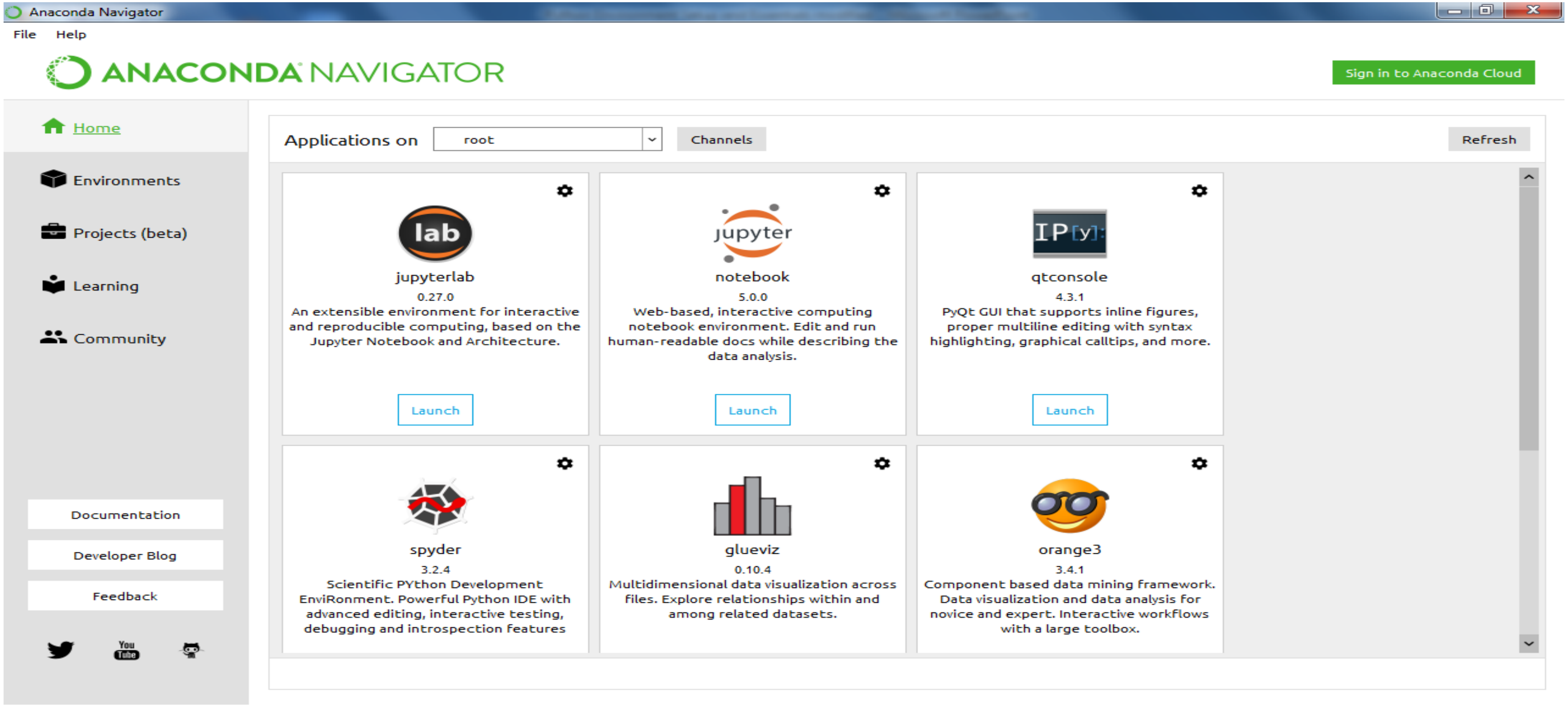
2. For Linux

   - Python is already install on Linux when Operating System is getting installed

Anaconda Installation

- Download and install Anaconda (windows version) from https://www.anaconda.com/download/

Anaconda Navigator



Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Projects (beta)

Learning

Community

Applications on root

Channels

Refresh

jupyterlab
0.27.0
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

notebook
5.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

spyder
3.2.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

glueviz
0.10.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

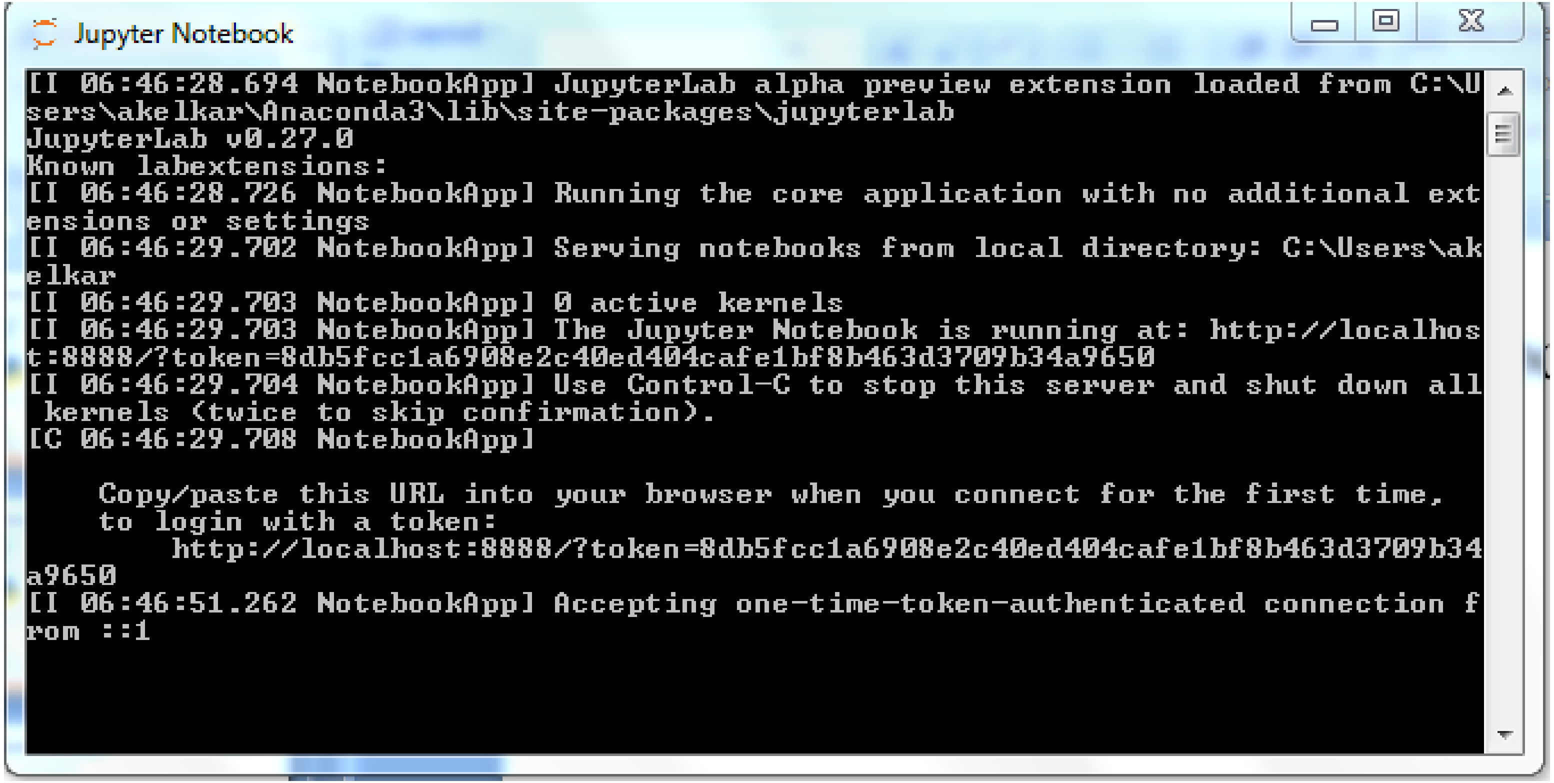
orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Documentation

Developer Blog

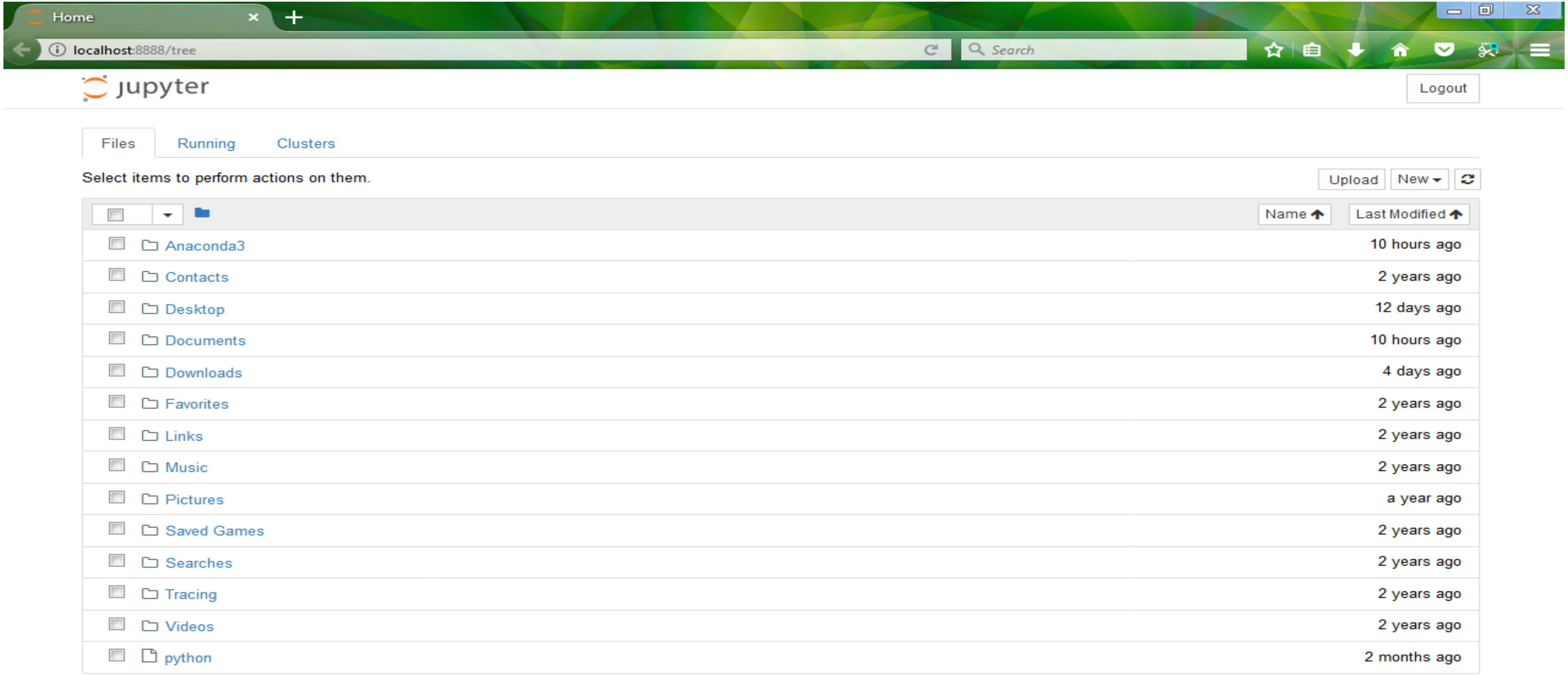
Feedback

Working with Jupyter Notebook



```
[I 06:46:28.694 NotebookApp] JupyterLab alpha preview extension loaded from C:\U
sers\akelkar\Anaconda3\lib\site-packages\jupyterlab
JupyterLab v0.27.0
Known labextensions:
[I 06:46:28.726 NotebookApp] Running the core application with no additional ext
ensions or settings
[I 06:46:29.702 NotebookApp] Serving notebooks from local directory: C:\Users\ak
elkar
[I 06:46:29.703 NotebookApp] 0 active kernels
[I 06:46:29.703 NotebookApp] The Jupyter Notebook is running at: http://localhos
t:8888/?token=8db5fcc1a6908e2c40ed404cafe1bf8b463d3709b34a9650
[I 06:46:29.704 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
[C 06:46:29.708 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=8db5fcc1a6908e2c40ed404cafe1bf8b463d3709b34
a9650
[I 06:46:51.262 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
```

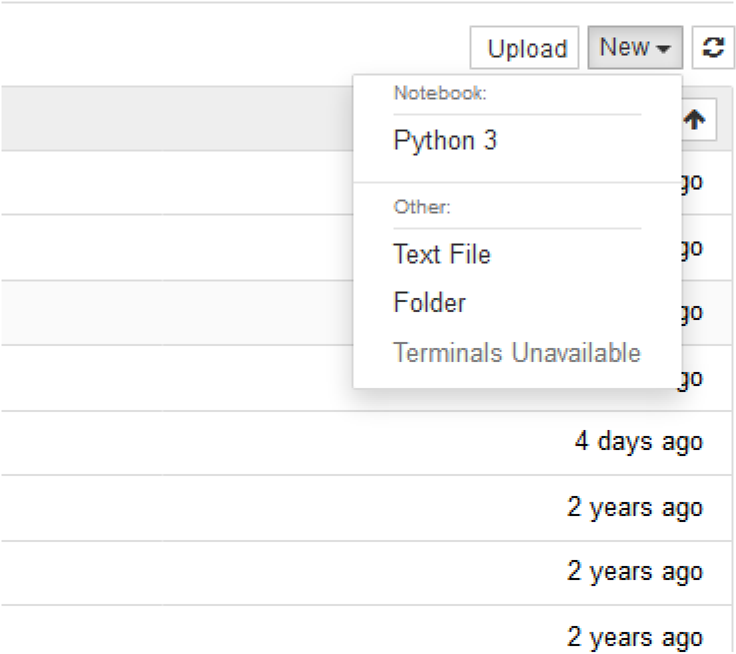
Working with Jypyter Notebook



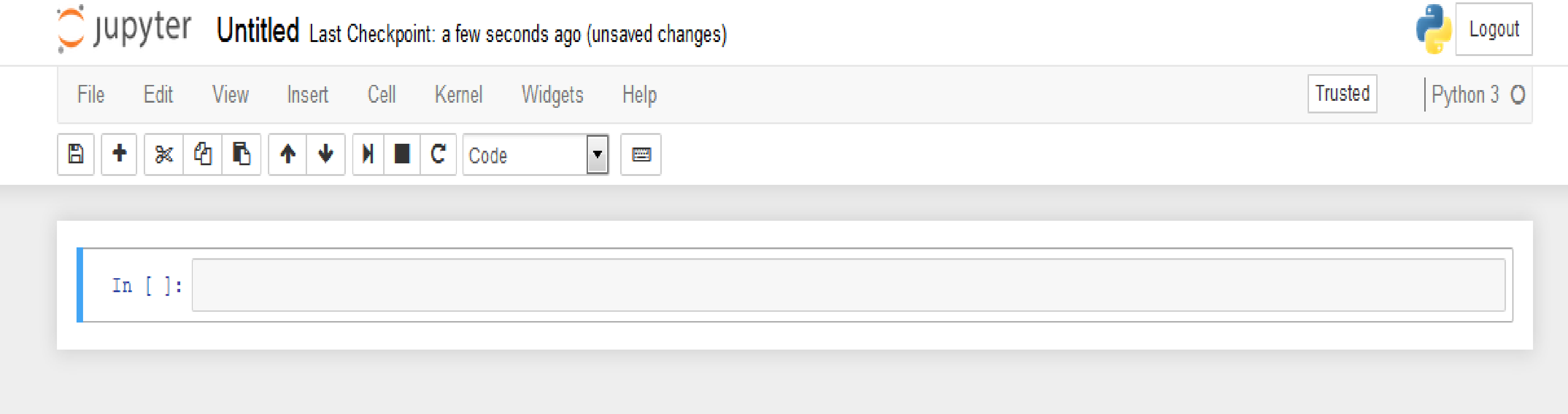
Select items to perform actions on them.

Name	Last Modified
□ Anaconda3	10 hours ago
□ Contacts	2 years ago
□ Desktop	12 days ago
□ Documents	10 hours ago
□ Downloads	4 days ago
□ Favorites	2 years ago
□ Links	2 years ago
□ Music	2 years ago
□ Pictures	a year ago
□ Saved Games	2 years ago
□ Searches	2 years ago
□ Tracing	2 years ago
□ Videos	2 years ago
□ python	2 months ago

Working with Jypyter Notebook



Working with Jypyter Notebook



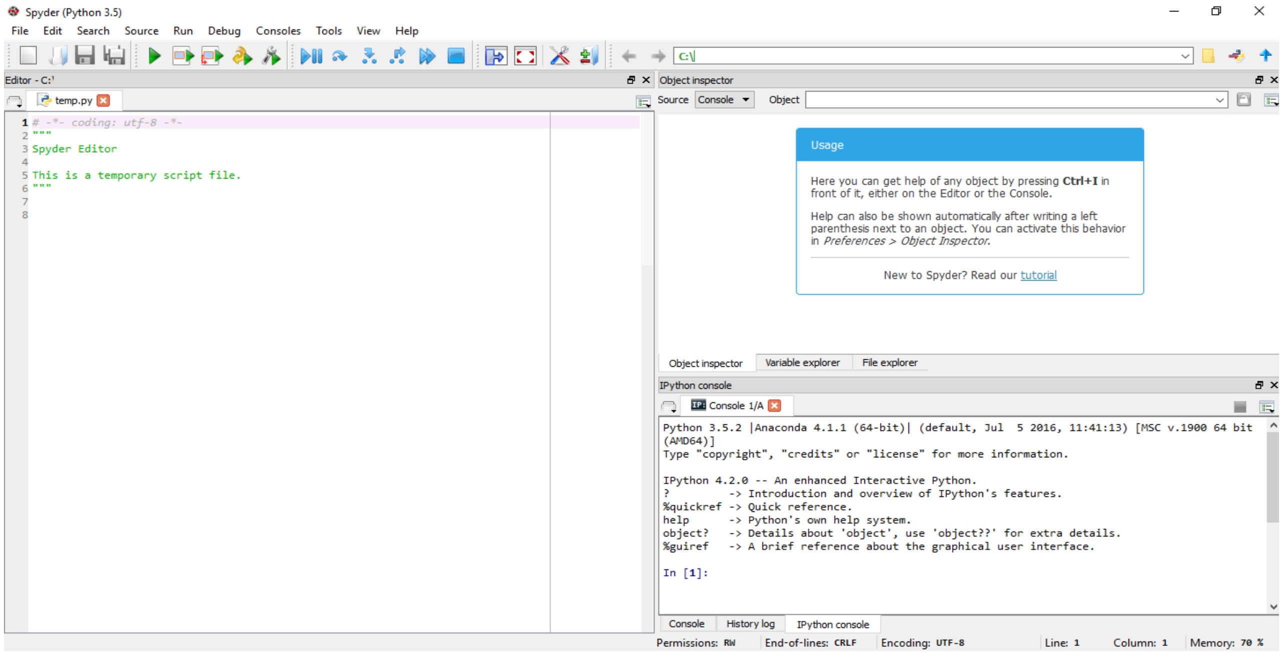
Spyder IDE

The usual Spyder screen has 3 windows

The Python Script(s) View

Object/Variable/File explorer

Console



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Object Inspector.*

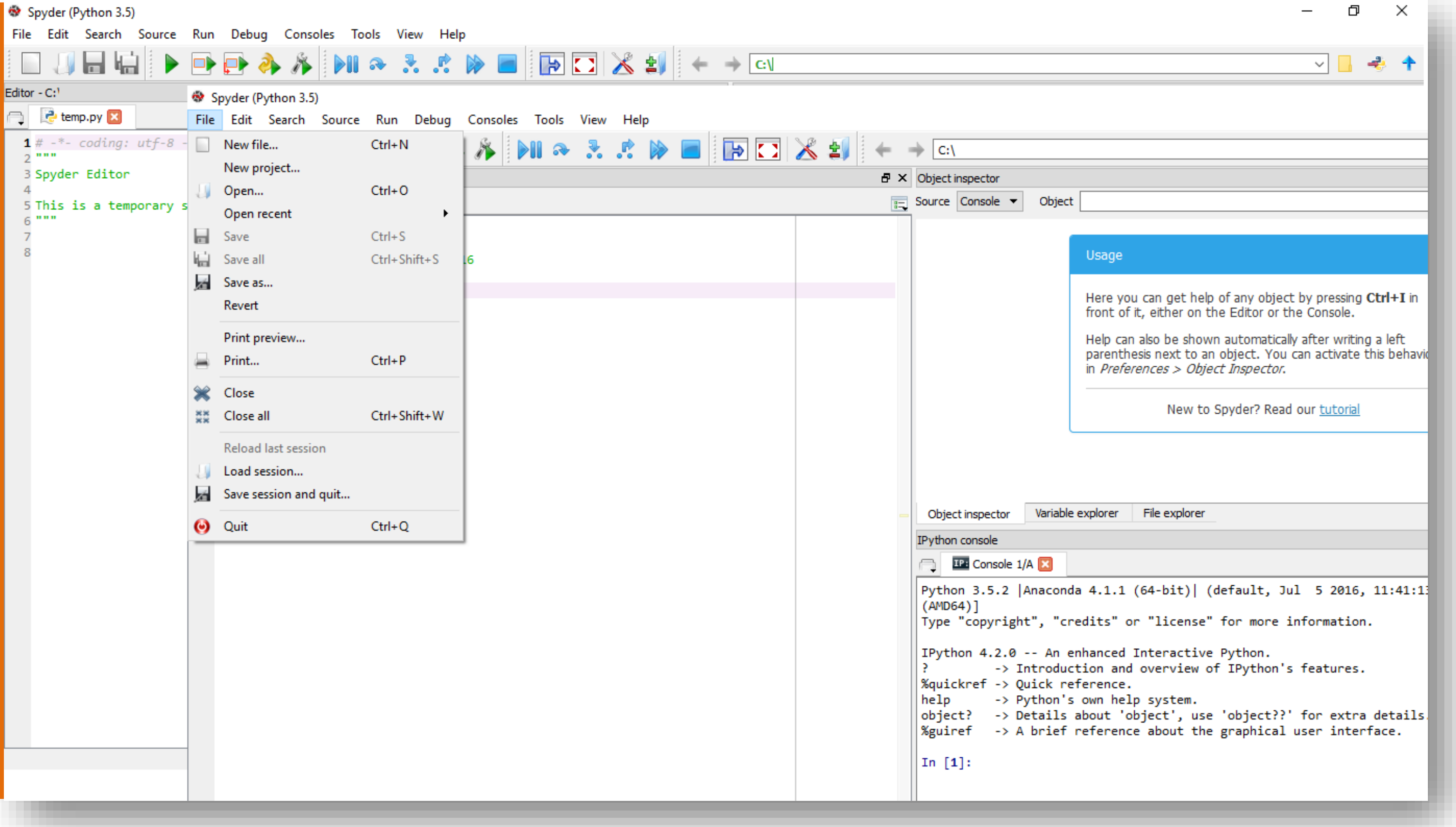
New to Spyder? Read our tutorial

```
Python 3.5.2 |Anaconda 4.1.1 (64-bit)| (default, Jul  5 2016, 11:41:13) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 1 Memory: 70 %

Spyder Screen: Python Script

To create a new Python script you can either go to File -> New File, or simply press Ctrl + N. Make sure to save the script.



```
# -*- coding: utf-8 -
"""
Spyder Editor

This is a temporary s
"""
```

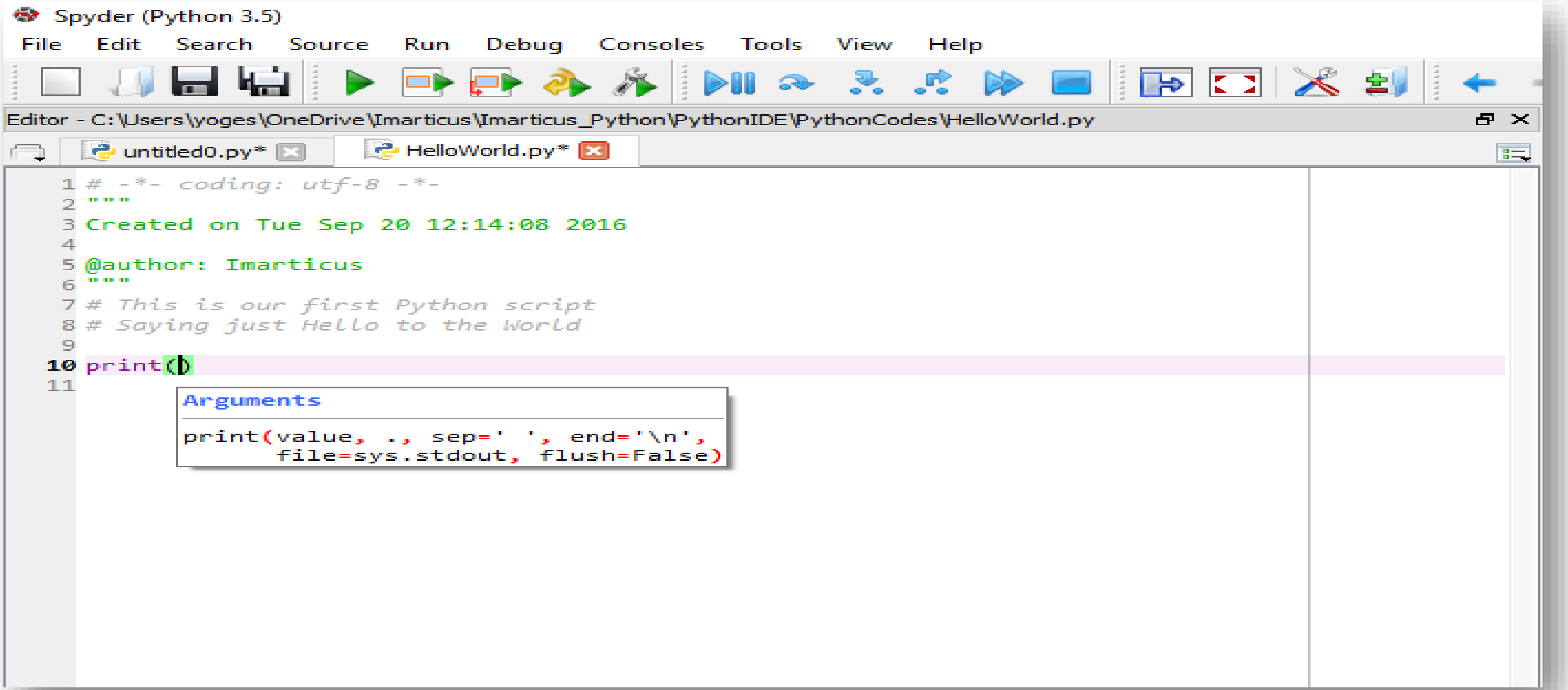
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavi in *Preferences > Object Inspector*.

New to Spyder? Read our tutorial

```
Python 3.5.2 |Anaconda 4.1.1 (64-bit)| (default, Jul  5 2016, 11:41:1
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

Spyder Screen: Environment

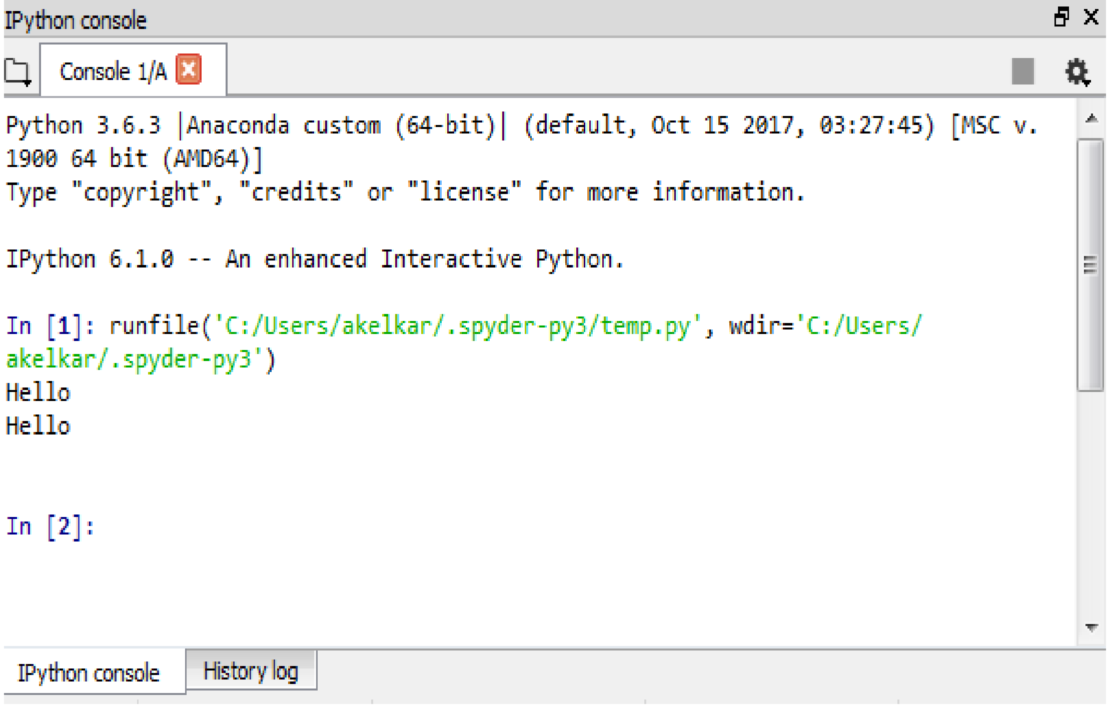

Spyder (Python 3.5)
File
Edit
Search
Source
Run
Debug
Consoles
Tools
View
Help
Editor - C:\Users\yoges\OneDrive\Imarticus\Imarticus_Python\PythonIDE\PythonCodes\HelloWorld.py
untitled0.py*
HelloWorld.py*

```
# -*- coding: utf-8 -*-
"""
Created on Tue Sep 20 12:14:08 2016

@author: Imarticus
"""
# This is our first Python script
# Saying just Hello to the World

print()

```

```
Arguments

print(value, .., sep=' ', end='\n',
      file=sys.stdout, flush=False)
```

Spyder Screen: Console

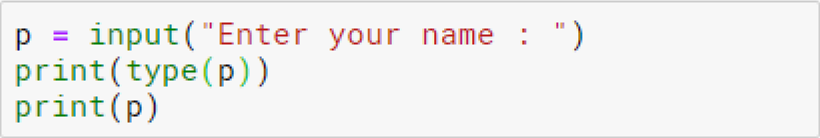
- The **Console** is where you may enter, interact with and visualize data, inside a command interpreter.
- All the commands entered in the console are executed in a separate process, thus allowing the user to interrupt any process at any time.



```
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC v.
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/akelkar/.spyder-py3/temp.py', wdir='C:/Users/
akelkar/.spyder-py3')
Hello
Hello


In [2]:
```

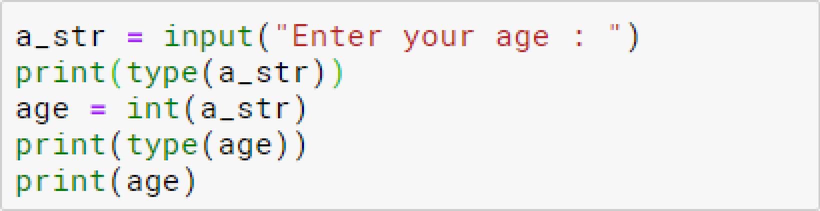
Basics of Python

User Input and Output



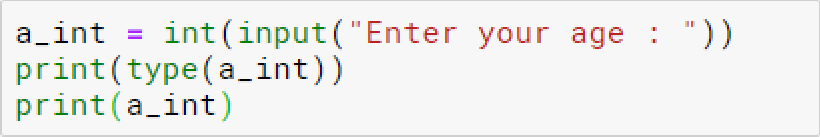
```python
p = input("Enter your name : ")
print(type(p))
print(p)
```

```
Enter your name : Anup
<class 'str'>
Anup
```



```python
a_str = input("Enter your age : ")
print(type(a_str))
age = int(a_str)
print(type(age))
print(age)
```

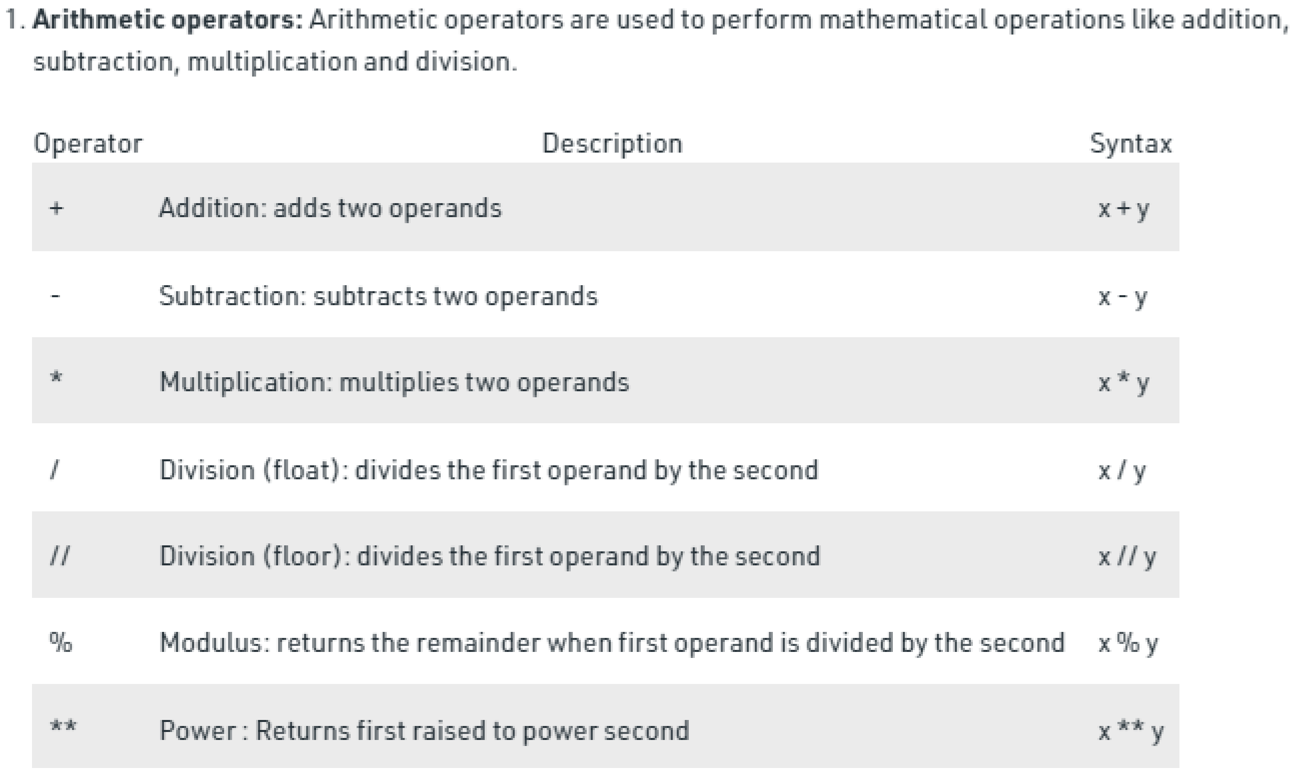
```
Enter your age : 23
<class 'str'>
<class 'int'>
23
```



```python
a_int = int(input("Enter your age : "))
print(type(a_int))
print(a_int)
```

```
Enter your age : 23
<class 'int'>
23
```

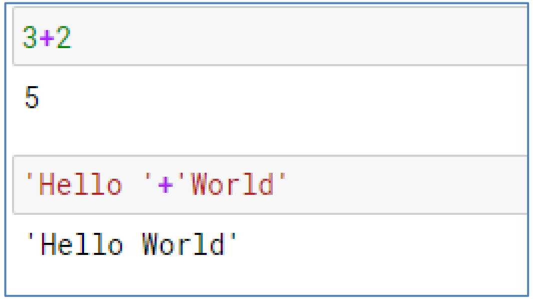
Operators in Python



1. **Arithmetic operators:** Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division.

Operator	Description	Syntax
+	Addition: adds two operands	x + y
-	Subtraction: subtracts two operands	x - y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y
**	Power : Returns first raised to power second	x ** y

Operators in Python

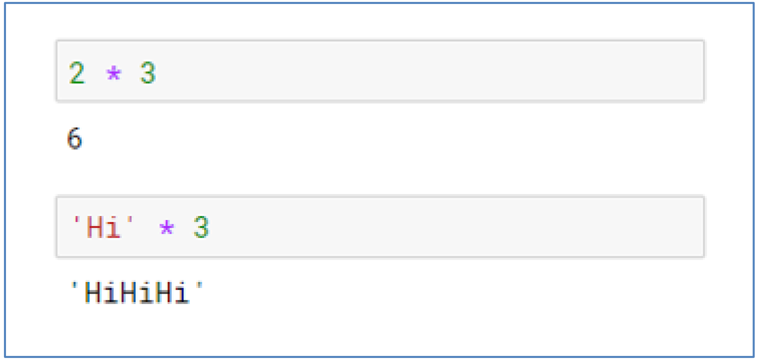


```
3+2
```

```
5
```

```
'Hello '+'World'
```

```
'Hello World'
```

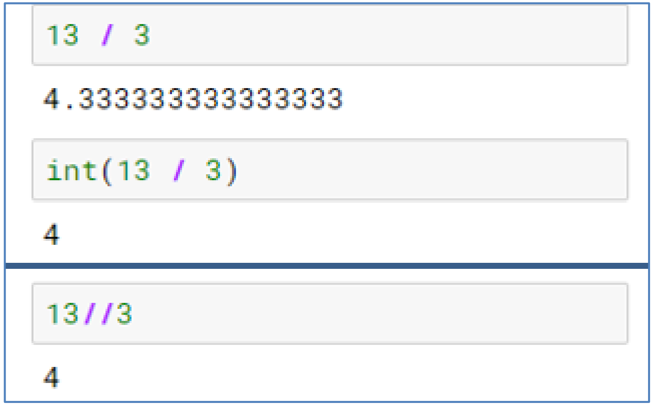


```
2 * 3
```

```
6
```

```
'Hi' * 3
```

```
'HiHiHi'
```



```
13 / 3
```

```
4.333333333333333
```

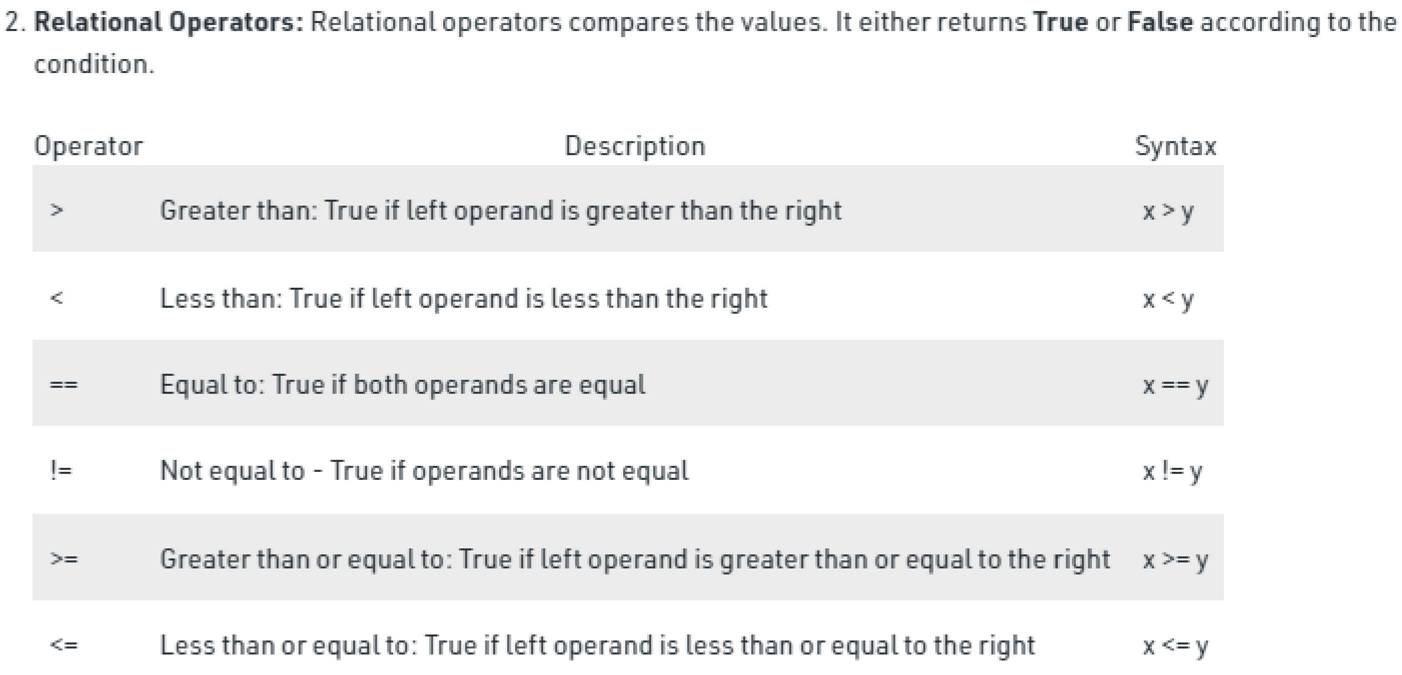
```
int(13 / 3)
```

```
4
```

```
13//3
```

```
4
```

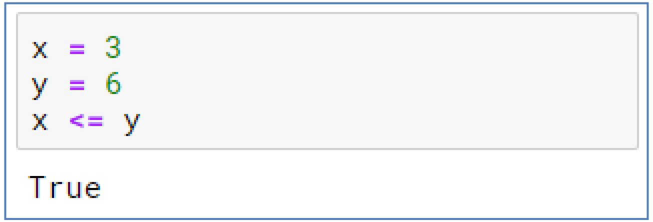
Operators in Python



2. **Relational Operators:** Relational operators compares the values. It either returns **True** or **False** according to the condition.

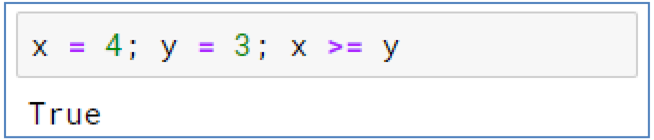
Operator	Description	Syntax
>	Greater than: True if left operand is greater than the right	x > y
<	Less than: True if left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

Operators in Python



```
x = 3
y = 6
x <= y
```

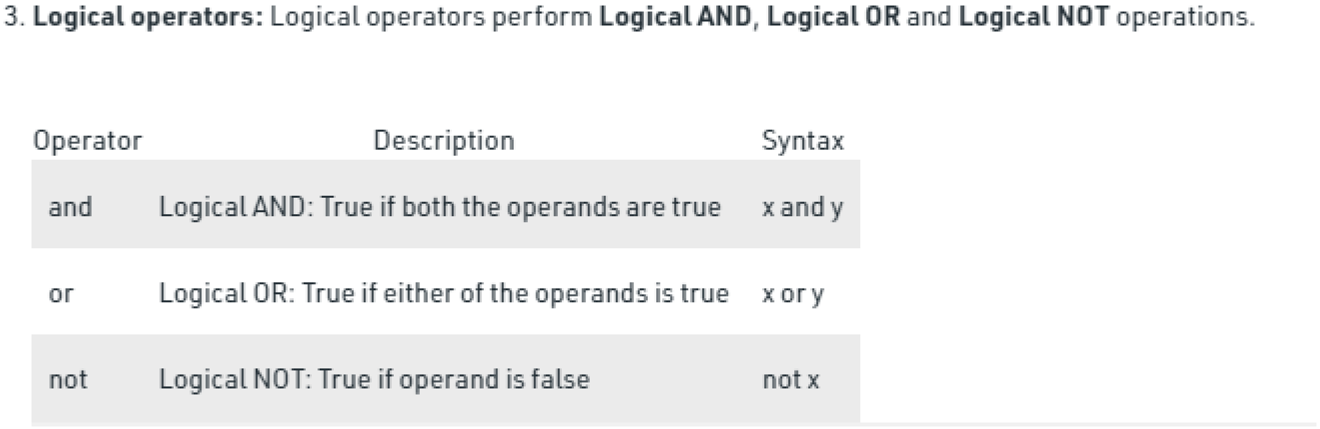
```
True
```



```
x = 4; y = 3; x >= y
```

```
True
```

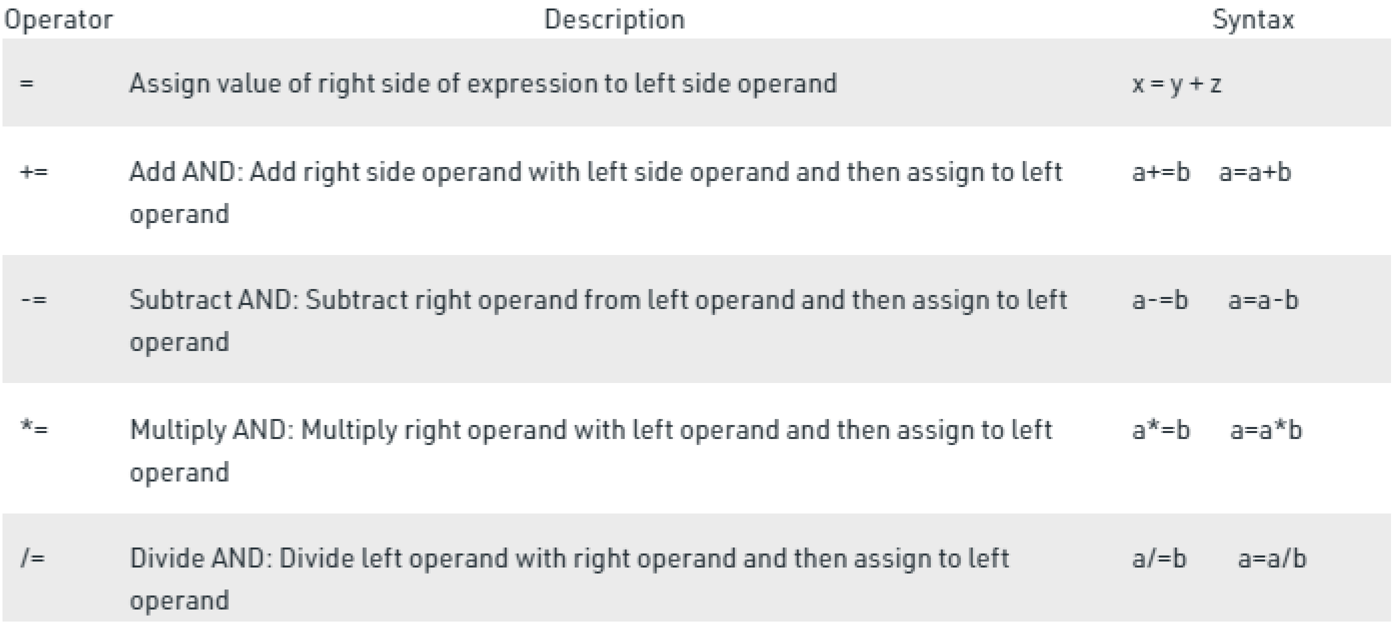
Operators in Python



3. **Logical operators:** Logical operators perform **Logical AND**, **Logical OR** and **Logical NOT** operations.

Operator	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x

Operators in Python



Operator	Description	Syntax
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add AND: Add right side operand with left side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b