## Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

### **Unix and Linux Installation**

Here are the simple steps to install Python on Unix/Linux machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link to download zipped source code available for Unix/Linux.
* Download and extract files.
* Editing the *Modules/Setup* file if you want to customize some options.

### **Windows Installation**

Here are the steps to install Python on Windows machine.

* Open a Web browser and go to <https://www.python.org/downloads/>.
* Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
* To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
* Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

After writing the code, we need to run the code to execute and obtain the output. On running the program, we can check whether the code is written is correct and produces the desired output.

Running a python program is quite an easy task.

**Run on IDLE**

To run a python program on IDLE, follow the given steps −

* Write the python code and save it.
* To run the program, go to Run > Run Module or simply click F5.

**Run on Command Line**

The python script file is saved with ‘.py’ extension. After saving the python script, we can run it from the Command Line. In the cmd, type keyword ‘python’ followed by the name of the file with which you saved the python script.

**Example**

Let us suppose, we have a python script saved with the name ‘hello.py’. To run it on command line, type the following −

**python hello.py**

**Comparison of python**

These operators compare the values on either sides of them and decide the relation among them. They are also called Relational operators.

Assume variable a holds 10 and variable b holds 20, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | If the values of two operands are equal, then the condition becomes true. | (a == b) is not true. |
| != | If values of two operands are not equal, then condition becomes true. | (a != b) is true. |
| <> | If values of two operands are not equal, then condition becomes true. | (a <> b) is true. This is similar to != operator. |
| > | If the value of left operand is greater than the value of right operand, then condition becomes true. | (a > b) is not true. |
| < | If the value of left operand is less than the value of right operand, then condition becomes true. | (a < b) is true. |
| >= | If the value of left operand is greater than or equal to the value of right operand, then condition becomes true. | (a >= b) is not true. |
| <= | If the value of left operand is less than or equal to the value of right operand, then condition becomes true. | (a <= b) is true. |

**Python keywords**

Python keywords are special reserved words that have specific meanings and purposes and can’t be used for anything but those specific purposes. These keywords are always available—you’ll never have to import them into your code.

Python keywords are different from Python’s [built-in functions and types](https://docs.python.org/3/library/functions.html). The built-in functions and types are also always available, but they aren’t as restrictive as the keywords in their usage.

An example of something you *can’t* do with Python keywords is assign something to them. If you try, then you’ll get a SyntaxError. You won’t get a SyntaxError if you try to assign something to a built-in function or type, but it still isn’t a good idea. For a more in-depth explanation of ways keywords can be misused, check out [Invalid Syntax in Python: Common Reasons for SyntaxError](https://realpython.com/invalid-syntax-python/#misspelling-missing-or-misusing-python-keywords).

As of Python 3.8, there are [thirty-five keywords](https://docs.python.org/3.8/reference/lexical_analysis.html#keywords) in Python. Here they are with links to the relevant sections throughout the rest of this article:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [False](https://realpython.com/python-keywords/#the-true-and-false-keywords) | [await](https://realpython.com/python-keywords/#the-await-keyword) | [else](https://realpython.com/python-keywords/#the-else-keyword) | [import](https://realpython.com/python-keywords/#the-import-keyword) | [pass](https://realpython.com/python-keywords/#the-pass-keyword) |
| [None](https://realpython.com/python-keywords/#the-none-keyword) | [break](https://realpython.com/python-keywords/#the-break-keyword) | [except](https://realpython.com/python-keywords/#the-except-keyword) | [in](https://realpython.com/python-keywords/#the-in-keyword) | [raise](https://realpython.com/python-keywords/#the-raise-keyword) |
| [True](https://realpython.com/python-keywords/#the-true-and-false-keywords) | [class](https://realpython.com/python-keywords/#the-class-keyword) | [finally](https://realpython.com/python-keywords/#the-finally-keyword) | [is](https://realpython.com/python-keywords/#the-is-keyword) | [return](https://realpython.com/python-keywords/#the-return-keyword) |
| [and](https://realpython.com/python-keywords/#the-and-keyword) | [continue](https://realpython.com/python-keywords/#the-continue-keyword) | [for](https://realpython.com/python-keywords/#the-for-keyword) | [lambda](https://realpython.com/python-keywords/#the-lambda-keyword) | [try](https://realpython.com/python-keywords/#the-try-keyword) |
| [as](https://realpython.com/python-keywords/#the-as-keyword) | [def](https://realpython.com/python-keywords/#the-def-keyword) | [from](https://realpython.com/python-keywords/#the-from-keyword) | [nonlocal](https://realpython.com/python-keywords/#the-nonlocal-keyword) | [while](https://realpython.com/python-keywords/#the-while-keyword) |
| [assert](https://realpython.com/python-keywords/#the-assert-keyword) | [del](https://realpython.com/python-keywords/#the-del-keyword) | [global](https://realpython.com/python-keywords/#the-global-keyword) | [not](https://realpython.com/python-keywords/#the-not-keyword) | [with](https://realpython.com/python-keywords/#the-with-keyword) |
| [async](https://realpython.com/python-keywords/#the-async-keyword) | [elif](https://realpython.com/python-keywords/#the-elif-keyword) | [if](https://realpython.com/python-keywords/#the-if-keyword) | [or](https://realpython.com/python-keywords/#the-or-keyword) | [yield](https://realpython.com/python-keywords/#the-yield-keyword) |

**What is a python interpreter?**

A python interpreter is a computer program that converts each high-level program statement into machine code. An interpreter translates the command that you write out into code that the computer can understand.

## Object Oriented Programming

**Object Oriented** means directed towards objects. In other words, it means functionally directed towards modelling objects. This is one of the many techniques used for modelling complex systems by describing a collection of interacting objects via their data and behaviour.

Python, an Object Oriented programming (OOP), is a way of programming that focuses on using objects and classes to design and build applications. Major pillars of Object Oriented Programming (OOP) are **Inheritance, Polymorphism, Abstraction,** ad **Encapsulation**.

Object Oriented Analysis(OOA) is the process of examining a problem, system or task and identifying the objects and interactions between them.

## Why to Choose Object Oriented Programming?

Python was designed with an object-oriented approach. OOP offers the following advantages −

* Provides a clear program structure, which makes it easy to map real world problems and their solutions.
* Facilitates easy maintenance and modification of existing code.
* Enhances program modularity because each object exists independently and new features can be added easily without disturbing the existing ones.
* Presents a good framework for code libraries where supplied components can be easily adapted and modified by the programmer.
* Imparts code reusability

## Principles of Object Oriented Programming

Object Oriented Programming (OOP) is based on the concept of **objects** rather than actions, and **data** rather than logic. In order for a programming language to be object-oriented, it should have a mechanism to enable working with classes and objects as well as the implementation and usage of the fundamental object-oriented principles and concepts namely inheritance, abstraction, encapsulation and polymorphism.



Let us understand each of the pillars of object-oriented programming in brief −

### **Encapsulation**

This property hides unnecessary details and makes it easier to manage the program structure. Each object’s implementation and state are hidden behind well-defined boundaries and that provides a clean and simple interface for working with them. One way to accomplish this is by making the data private.

### **Inheritance**

Inheritance, also called generalization, allows us to capture a hierarchal relationship between classes and objects. For instance, a ‘fruit’ is a generalization of ‘orange’. Inheritance is very useful from a code reuse perspective.

### **Abstraction**

This property allows us to hide the details and expose only the essential features of a concept or object. For example, a person driving a scooter knows that on pressing a horn, sound is emitted, but he has no idea about how the sound is actually generated on pressing the horn.

### **Polymorphism**

Poly-morphism means many forms. That is, a thing or action is present in different forms or ways. One good example of polymorphism is constructor overloading in classes.