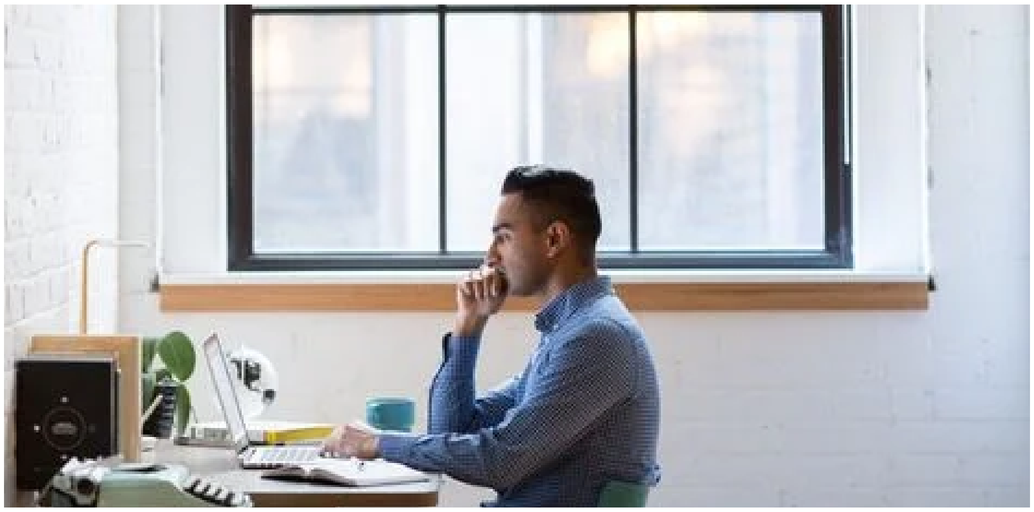
7 Vital Commands to Get Started With Python for Beginners

Want to learn Python but don't know where to start? Begin your programming journey by learning these fundamental commands first.

BY DEEPESH SHARMA

PUBLISHED 2 DAYS AGO



Learning a new programming language like Python becomes effortless if you have a comprehensive roadmap detailing which concepts to learn as a beginner and how to progress further to reach the next milestone. Even intermediate programmers should frequently brush up on their basics to build a solid foundation for themselves.

By the end, you'll find yourself stuck to your computer writing Python code using the fundamental commands enlisted in this article. We'll also discuss how you can set up a Python environment on your machine to be able to write and execute Python code.

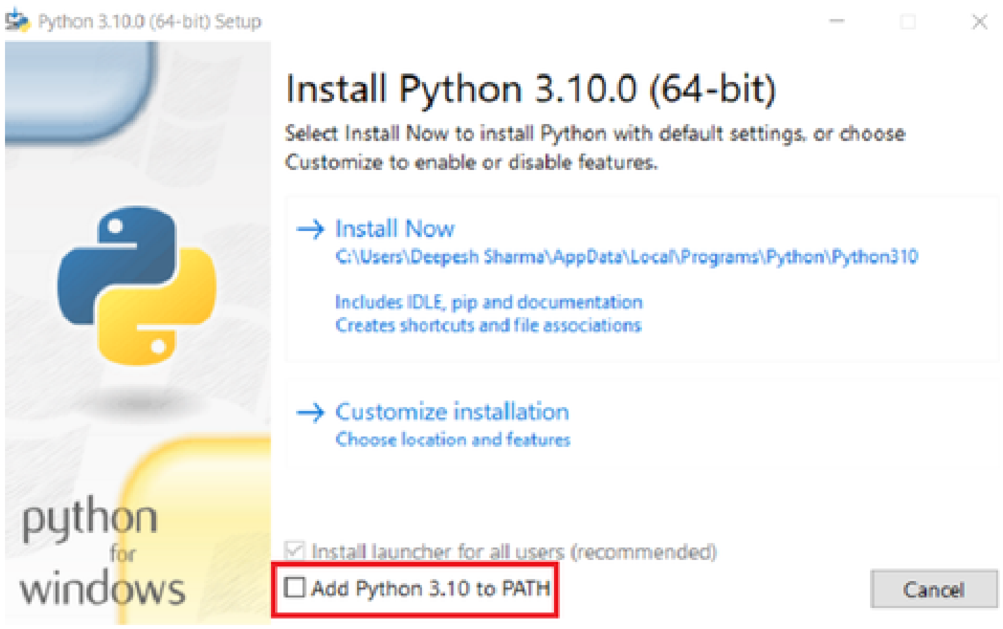
Setting Up the Python Environment

To run Python code, your system should have Python installed on it.

On Windows

You can download the latest version of Python for Windows from the python.org Downloads page. Click on the **Download Python** button, select **Windows Executable** on the next page, and wait for the executable to download. Once done, double-click the installer file to launch the installation window.

Install Python as you'd normally install any other program on Windows. Don't forget to check off the "Add Python to PATH" option during installation.


Python 3.10.0 (64-bit) Setup
Install Python 3.10.0 (64-bit)
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.
Install Now
C:\Users\Deepesh Sharma\AppData\Local\Programs\Python\Python310
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python for windows
Install launcher for all users (recommended)
Add Python 3.10 to PATH
Cancel

On Linux

To install Python on Linux, issue the following commands depending on the Linux distribution you're using:

On Debian/Ubuntu:

```
sudo apt install python
```

On Arch Linux:

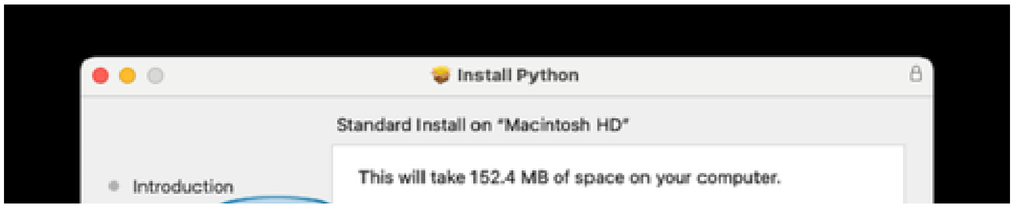
```
sudo pacman -S python
```

On Fedora and CentOS:

```
sudo dnf install python
```

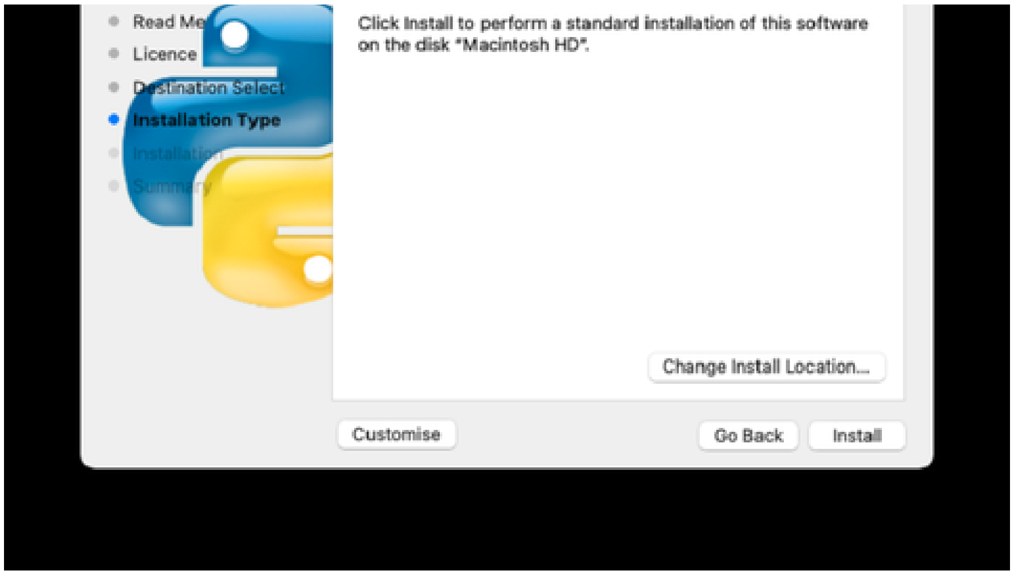
On macOS

Similar to the installation process for Windows, first, download the MPKG from the **Python Download** page. Then, launch the installer and follow the on-screen steps to proceed with the installation.



Standard Install on "Macintosh HD"

This will take 152.4 MB of space on your computer.



Click Install to perform a standard installation of this software on the disk "Macintosh HD".

Now that the installation part is done, let's get practical and start with the list of Python commands that every beginner should know of. This guide assumes that you are aware of how to execute Python programs from the command line.

Basic Input, Output, and Variables

Saving data, taking input, and then modifying the data to produce desired output is the goal of every Python program.

1. Initialize and Declaring Variables

To store data, every programming language uses variables. Variables are objects of certain data structures that can hold value and perform operations on the data. Let's consider the following Python code:

```
number = 20
```

Here, we have declared a variable with the name "number" and a value of 20. Similarly, you can create other variables with a different data type. Python supports several data types but as a beginner, you'll mostly work with the ones mentioned below. The commands to initialize the data types are enclosed in the brackets.

1. Integer (**number = 10**)
2. String (**name = "Ryan"**)
3. Float (**decimal = 10.23**)
4. List (**fruits = ["Apple", "Banana", "Mango"]**)
5. Tuple (**fruits = ("Apple", "Banana", "Mango")**)
6. Dictionary (**fruitmap = {1:"Apple", 2:"Banana", 3:"Mango"}**)

2. Display Output With the print() Method

Most beginner programmers start with the basic "Hello World" program that outputs the string on execution. In Python, you can print hardcoded messages and variable values using print().

To print a string in Python, take a look at the following command syntax:

```
print("This is a string")
```

Anything you enclose within the quotes will be displayed as it is. The aforementioned code will display "This is a string" when you run it using the command line.

You can also print the value of a variable by simply specifying the variable name without quotes. Let's assume we have a string variable "surname" holding the value "Sharma":

```
print(surname)
```

Output:

```
Sharma
```

3. Take Input With input()

A program is only useful if a user can interact with it. To make your applications dynamic and interactive, you'll have to depend on the user for input and choice.

You can ask the user to enter a value using the **input** command. Here's the syntax to follow:

```
variable = input("The string to display")
```

For example, the following command will ask the user for their name and age respectively:

```
name = input("Please enter your name => ")

age = input("Please enter your age => ")
```

Taking Control of the Program Flow

A program doesn't just consists of inputs, outputs, and data types. It also includes control statements, necessary for implementing logic and determining the flow of the program.

4. Implement Logic With if, elif, and else

Your computer handles the operations and makes choices based on logical decisions. To implement logic in your code, you can make use of the if, elif, and else commands. These commands change the program flow based on conditions and are thus known as conditional control statements.

As the name suggests, the **if** command evaluates an expression, and if it's true, executes the statements under it. The **elif** command (else if) provides another expression that gets evaluated if the preceding **if** statement returns false. Lastly, if no previous statements (**if** or **elif**) return true, the expression provided with the **else** command is evaluated.

Note that you can have multiple **if** and **elif** statements in a particular block of code. Even nested **if** statements are possible.

Here's a simple program explaining the use of if, elif, and else. The program will evaluate if the specified number is positive, negative, or zero.

```
number = int(input("Enter a number to evaluate: "))

if (number > 0):
    print("Positive")

elif (number < 0):

    print("Negative")

else:

    print("Zero")
```

Note that we had to wrap the input() method with int(), since the input is stored as string type by default, and we need the "number" variable to be of integer type instead.

The difference between if and elif is that all if statements in the code block will be evaluated one after the other no matter what, but an elif statement will be evaluated only if the preceding if statement stands false.

RELATED:

Python Project Ideas Suitable For Beginners

5. The for Loop in Python

Although Python supports several other loop statements (do...while, while, switch), the for loop is the most-common loop control statement compared to the rest.

Unlike C and C++, for loop in Python always iterates over an iterative variable. An iterative variable is one that holds multiple values in it, like lists, tuples, and dictionaries.

Declare a list variable "fruits" containing the values Apple, Banana, Pear, and Mango. To iterate over each element and print the values using for loop:

```
for element in fruits:

    print(element)
```

You can also create the classic C-style for loop in Python using the range() method. The range() method generates a list of numbers depending on the starting, ending, and step values specified.

```
for i in range(0,5):

    print(i)
```

Output:

```
0

1

2

3

4
```

Maintaining Modularity in the Code

A good code is one that is easier to read, effortless to debug, and a breeze to scale. And all of this is achieved by writing modular code.

6. Define Functions With def

To minimize code redundancy and encourage code reuse, Python provides a way to wrap reusable code inside functions, which can be later invoked when necessary. You can create a function using the **def** keyword in Python.

Similar to other programming languages, Python functions also take arguments and return values on successful execution. You can also overload functions in Python.

```
def sum(a,b):

    return a+b

print(sum(1,2))
```

Output:

```
3
```

7. Create Classes With the class Keyword

You can create classes to create blueprints for objects in Python. Python supports object-oriented programming and allows users to create classes and initialize objects. A class can consist of variables with access modifiers, functions with return types, and even other classes (nested class).

Here's a simple code that creates a class named **student**:

```
class student:

    name = ""

    def setName(self, passedValue):

        self.name = passedValue

    def displayName(self):

        print(self.name)
```

To use a class, you have to first create an instance of it, also known as an object.

```
mystudent = student()

mystudent.setName("Deepesh Sharma")

mystudent.displayName()
```

On combining the last two code snippets, the aforementioned program will output:

```
Deepesh Sharma
```

Similar to other programming languages, you can also implement constructors and static methods in Python classes (using the init() dunder method and the @staticmethod decorator respectively).