1



**Welcome to the Python Development 🤘**

Python is not just a language, it’s a superpower

In this course, you’ll go from writing your first line of code to building cool projects that’ll make you say, ‘Wow, I did that! 😎’

Whether you’re totally new to coding or just looking to sharpen your Python skills, get ready for an epic journey filled with coding magic, challenges, and lots of fun moments.

**Python Language Quick Overview**

Python is a high-level, interpreted programming language that is widely used for many different purposes, from web development to scientific computing to machine learning.

There are several reasons why Python is a great choice for many different types of projects:

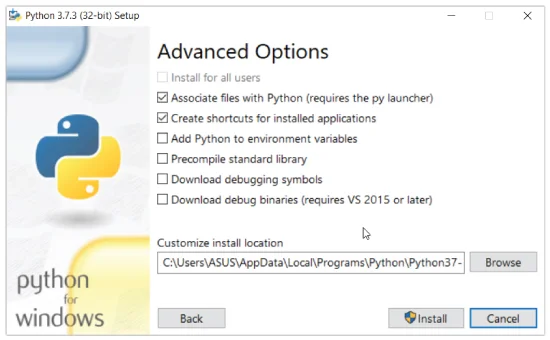
1. **Easy to learn:** Python has a relatively simple syntax, which makes it easier for beginners to learn compared to other programming languages.
2. **Versatile:** Python can be used for a wide range of applications, including web development, data analysis, artificial intelligence, and more.
3. **Large and active community:** There is a large and active community of Python developers, which means that there are many resources available for learning and solving problems.
4. **Plenty of libraries**: Python has a large number of libraries, including NumPy, pandas, and Matplotlib, which can be used to perform complex tasks with just a few lines of code.
5. **Cross-platform compatibility:** Python can run on multiple operating systems, including Windows, macOS, and Linux, making it a good choice for projects that need to be run on multiple platforms.

**Install Python3 and Vs-Code on a Windows machine**

**Install Python**

If you don't want to use Thonny, here's how to install and run Python on your computer.

1. Download the [latest version of Python](https://www.python.org/downloads/).
2. Run the installer file and follow the steps to install Python
3. During the installation process, check **Add Python to environment variables**. This will add Python to environment variables, and you can run Python from any part of the computer.
4. Also, you can choose the path where Python is installed.



**Note:** For Unix based Operating Systems like Linux and macOS systems have Python pre-installed

2

**All About Python Variables 🧠💡**

In programming, a variable is a container (storage area) to hold data. For example,

number = 10

Here, the **number** is the variable storing the value **10**.

**Assigning values to Variables in Python**

As we can see from the above example, we use the assignment operator = to assign a value to a variable.

site\_name = "Power Learn Project"

print(site\_name)

In the above example, we assigned the value ‘Power Learn Project’ to the site\_name variable. Then, we printed out the value assigned to site\_name.

**Note**: Python is a type-inferred language, so you don't have to explicitly define the variable type. It automatically knows that Power Learn Projects is a string and declares the site\_name variable as a string.

**Changing the Value of a Variable in Python**

site\_name = "Power Learn Project"

print(site\_name)

# Assigning new value to site\_name

site\_name = "I love coding 😊"

print(site\_name)

# The output will be

Power Learn Project

I love coding 😊

Here, the value of site\_name is changed from ‘Power Learn Project’ to 'I love coding 😊'.

**Example: Assigning multiple values to multiple variables**

a,b,c = 5, 7, "Hello world"

print(a) # prints 5

print(b) # prints 5

print(c) # prints Hello World

**Rules for Naming Python Variables**

* Constant and variable names should have a combination of letters in lowercase (a to z) or uppercase (**A to Z**) or digits (**0 to 9**) or an underscore (**\_**). For example:

* snake\_case
* MACRO\_CASE
* camelCase
* CapsWords

* Python is case-sensitive. So num and Num are different variables. For example,

num = 55

Num = 510

print(num) #5

print(Num) #510

* Avoid using keywords like if, True, class, etc. as variable names.

More Resources:

1. <https://realpython.com/python-variables/>
2. <https://www.simplilearn.com/tutorials/python-tutorial/python-variables>
3. <https://www.guru99.com/variables-in-python.html>
4. <https://www.tutorialspoint.com/python/python_variables.htm>

3

**Understanding Python Datatypes**

In computer programming, data types specify the type of data that can be stored inside a variable. For example,

num = 24

Here, **24** (an integer) is assigned to the num variable. So the data type of num is of the int class.

**Python Data Types**

|  |  |  |
| --- | --- | --- |
| **Data Types** | **Class** | **Description** |
| numeric | int, float, complex | Holds numeric values |
| string | str | Holds sequence of characters |
| sequence | list, tuple ,range | Holds collection of items |
| mapping | dict | Holds data in key-value pair form |
| boolean | bool | Holds either True or False |
| set | set | Holds collection of unique items |

Since everything is an object in Python programming, data types are actually classes and variables are instances(object) of these classes.

**Python Numeric Data type**

In Python, the numeric data type is used to hold numeric values.

Integers, floating-point numbers and complex numbers fall under Python numbers category. They are defined as int, float and complex classes in Python.

* **int** - holds signed integers of non-limited length.
* **float** - holds floating decimal points and it's accurate up to **15** decimal places.

We can use the type() function to know which class a variable or a value belongs to.

Let's see an example,

num1 = 55

num2 = 5.3

print(num1)

print(num2)

**Python List Data Type**

A list is an ordered collection of similar or different types of items separated by commas and enclosed within brackets [ ]. For example,

languages = ["Python", "Dart", "Web", 23]

print(languages)

Here, we have created a list named languages with **3** string values inside it.

**Access List Items**

To access items from a list, we use the index number **(0, 1, 2 ...)**. For example,

languages = ["Python", "Dart", "Web", 23]

print(languages[1])

In the above example, we have used the index values to access items from the languages list.

* languages[0] - access the first item from languages i.e. Python
* languages[2] - access the third item from languages i.e. Web

**Python Tuple Data Type**

A tuple is an ordered sequence of items same as a list. The only difference is that tuples are immutable. Tuples once created cannot be modified.

In Python, we use the parentheses () to store items of a tuple. For example,

products = ('XBox', 499.99, "Habibi", 23)

print(products)

Here, product is a tuple with a string value Xbox and integer value **499.99**.

**Access Tuple Items**

Similar to lists, we use the index number to access tuple items in Python. For example,

products = ('XBox', 499.99, "Habibi", 23)

print(products[2])

**Python String Data Type**

String is a sequence of characters represented by either single or double quotes. For example,

site\_name = "Power Learn Project"

print(site\_name)

In the above example, we have created string-type variables: name and message with values 'Python' and 'Python for beginners' respectively.

**Python Set Data Type**

The Set is an unordered collection of unique items. Set is defined by values separated by commas inside braces { }. For example,

student\_ids = {112, 114, 117, 113}

print(student\_ids)

Here, we have created a set named student\_info with **5** integer values.

Since sets are unordered collections, indexing has no meaning. Hence, the slicing operator [] does not work.

**Python Dictionary Data Type**

Python dictionary is an ordered collection of items. It stores elements in key/value pairs.

Here, keys are unique identifiers that are associated with each value.

Let's see an example,

capital\_city = {"Kenya": "Nairobi", "Nigeria": "Lagos"}

print(capital\_city)

**More Resources:**

1. <https://www.geeksforgeeks.org/python-data-types/>
2. <https://realpython.com/python-data-types/>
3. <https://www.digitalocean.com/community/tutorials/python-data-types>
4. <https://jakevdp.github.io/PythonDataScienceHandbook/02.01-understanding-data-types.html>

4

Operators are special symbols that perform operations on variables and values. For example,

**Types of Python Operators**

Here's a list of different types of Python operators that we will learn in this tutorial.

1. Arithmetic operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Bitwise Operators
6. Special Operators

**1. Python Arithmetic Operators**

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc. For example,

sub = 10 - 5 # 5

Here, - is an arithmetic operator that subtracts two values or variables.

|  |  |  |
| --- | --- | --- |
| **operator** | **operation** | **example** |
| + | addition | 5 + 2 = 7 |
| - | subtraction | 4 - 2 = 2 |
| \* | multiplication | 2 \* 3 = 6 |
| \*\* | exponentiation ( power) | 4 \*\* 2 = 16 |
| / | float division | 4 / 2 = 2 |
| // | floor division | 10 // 3 = 3 |
| % | modulus (remainder) | 5 % 2 = 1 |

In the above example, we have used multiple arithmetic operators,

* + to add a and b
* - to subtract b from a
* \* to multiply a and b
* / to divide a by b
* // to floor divide a by b
* % to get the remainder
* \*\* to get a to the power b

**2. Python Assignment Operators**

Here's a list of different assignment operators available in Python.

|  |  |  |
| --- | --- | --- |
| **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 |
| := | print(x := 3) | x = 3 print(x) |

Here, we have used the += operator to assign the sum of a and b to a.

Similarly, we can use any other assignment operators according to the need.

**3. Python Comparison Operators**

Comparison operators compare two values/variables and return a boolean result: True or False. For example

Here, the > comparison operator is used to compare whether a is greater than b or not.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| **==** | Equal | 2 == 2 |
| **!=** | Not equal | 4 != 2 |
| **>** | Greater than | 5 > 3 |
| **<** | Less than | 2 < 7 |
| **>=** | Greater than or equal to | x >= y |
| **<=** | Less than or equal to | x <= y |

**Note:** Comparison operators are used in decision-making and loops. We'll discuss more of the comparison operator and decision-making in later tutorials.

**4. Python Logical Operators**

Logical operators are used to check whether an expression is True or False. They are used in decision-making. For example,

|  |  |  |
| --- | --- | --- |
| **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) |

More Resources:

1. <https://www.w3schools.com/python/python_operators.asp>
2. <https://www.freecodecamp.org/news/basic-operators-in-python-with-examples/>
3. <https://www.geeksforgeeks.org/python-operators/>