THE BEGINNER'S GUIDE TO PYTHON

PRITOM PAUL

Table of Contents

0. Introduction

Python is one of the most popular programming languages in the world. It is simple to learn, easy to read, and powerful enough to build almost anything—from small scripts to big applications. Many famous companies, like Google, YouTube, and Instagram, use Python for their projects.

This book is designed for beginners as well as those who want to refresh their Python skills. You don't need any previous programming experience to start. Each chapter will guide you step by step, with clear explanations and practical examples.

In this tutorial, you will learn:

- How to install and run Python on your computer.
- The basics of Python syntax and rules.
- How to work with data types, variables, and operators.
- How to use loops and conditional statements.
- How to write and use functions.
- How to work with files, libraries, and modules.
- An introduction to object-oriented programming in Python.

By the end of this book, you will be able to write your own Python programs and solve real-world problems using the skills you have learned.

Let's start the journey and make programming an enjoyable experience with Python!

1. Python Getting Started

1.1. Installation and Version Check

Describing the process of verifying Python installation on various operating systems.

- On Windows: Open Command Line and type python –version.
- On Linux/Mac: Open Terminal and type python –version.
- If not installed, download from https://www.python.org/.
- Or use https://www.anaconda.com/ (Anaconda), a **free and open-source distribution** of Python.

1.2. Creating Your First Python Program

Explaining the creation and execution of a simple Python file.

Create a file named *hello.py* with the content: `print("Hello, World!")`

Navigate to the file directory in Command Line and run python hello.py.

Expected output: *Hello, World!*

```python
print("Hello, World!")
```

Input:	**Output**: Hello, World!

*We will be using Python 3.13 for this book.

2. Python Syntax

2.1. Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

```python
def sayHello():
  print("Hello, there!")
sayHello()
```

Input:

Output:
```
Hello, there!
```

2.2. Python Variable

In Python, variables are created when you assign a value to it and has no command for declaring a variable.

```python
x = 5
y = "Hello, World!"
```

Input:

Output:

2.2.1. Many Values to Multiple Variables

Python allows you to assign values to multiple variables in one line:

```python
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Input:

Output:
```
Orange
Banana
Cherry
```

2.2.2. One Value to Multiple Variables

And you can assign the same value to multiple variables in one line:

```python
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Input:

Output:
```
Orange
Orange
```

	Orange

2.2.3. Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

Input:	**Output**: apple banana cherry

2.2.4. Output Variables

The Python `print()` function is often used to output variables.

```
x = "Python is awesome"
print(x)
```

Input:	**Output**: Python is awesome

In the `print()` function, you output multiple variables, separated by a comma (However, it adds an extra whitespace with every comma):

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

Input:	**Output**: Python is awesome

You can also use the ‘+’ operator to output multiple variables (Practically, it concatenates the strings):

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

Input:	**Output**: Python is awesome

For numbers, the + character works as a mathematical operator:

```
x = 5
y = 10
print(x + y)
```

Input:

Output:
```
15
```

In the print() function, when you try to combine a string and a number with the '+' operator, Python will give you an error:

```
x = 5
y = "John"
print(x + y)
```

Input:

Output:
```
TypeError
Traceback (most recent call last)
Cell In[10], line 3
      1 x = 5
      2 y = "John"
----> 3 print(x + y)

TypeError: unsupported operand
type(s) for +: 'int' and 'str'
```

The best way to output multiple variables in the print() function is to separate them with commas, which even support different data types:

```
x = 5
y = "John"
print(x, y)
```

Input:

Output:
```
5 John
```

2.2.5. end parameter

In the `print()` function, there is always a '\n' or a newline at the output stream by default. In case of necessary, you can change it to any other characters or escape sequences using the `end=""` parameter int the `print()` function.

```
print("Something", end=" ")
print("Testing", end="#")
```

Input:

Output:
```
Something Testing#
```

2.2.6. Global Variables

Variables that are created outside of a function (as in all of the examples in the previous pages) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.[1]

```
x = "awesome"
def myfunc():
  print("Python is " + x)

myfunc()
```

Input:

Output:
```
Python is awesome
```

2.2.7. Local Variable

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.[1]

```
x = "Global Variable"

def myfunc():
  x = "Local Variable"
  print("This is inside a function: " + x)

myfunc()

print("This is outside any function: " + x)
```

Input:

Output:
```
This is inside a function: Local Variable
This is outside any function: Global Variable
```

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

```
def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

Input:

Output:
```
Python is fantastic
```

Also, use the global keyword if you want to change a global variable inside a function.

```
x = "awesome"

def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

Input:	**Output**:
	Python is fantastic

2.2.8. Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).[1]

Rules for Python variables:

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the Python keywords.

7

References

[1] “Python Tutorial.” Accessed: Aug. 20, 2025. [Online]. Available: https://www.w3schools.com/python/default.asp