**INTRODUCTION TO CODING USING PYTHON**

# **Introduction**

# **What is Python?**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

# **Where we use Python?**

* It is used on a server to create web applications.
* It is used as alongside software.
* Used to connect to database systems.
* It is used to handle big data and perform complex mathematics.
* Used for rapid prototyping, or for production-ready software development.

# **Features of Python**

1. Readable

2. Easy to Learn

3. **Cross platform**

4**. Open Source**

5. **Large standard library**

6**. Supports exception handling**

**7.Advanced features**

8**. Automatic memory management**

**Python User Input**

Python allows for user input.

Which means we are able to ask the user for input.

The method is a bit different in Python 3.6 than Python 2.7.

* In Python 3.6 uses the input() method.
* In Python 2.7 uses the raw\_input() method.

# **Python Variables**

* Variables are used to store data, they take memory space based on the type of value we assigning to them.
* Creating variables in Python is simple, you just have write the variable name on the left side of = and the value on the right side.

## Variable name – Identifiers

Variable name is known as identifier. Rules to define variable are as follows:

**1**. Variable name always start with either a letter or an underscore (\_). For example: \_str, str, num, \_num are all valid name for the variables.  
**2**. Variable name cannot start with a number. For example: 9num is not a valid variable name.  
**3**. Variable name cannot have special characters such as %, $, # etc, they can only have alphanumeric characters and underscore (A to Z, a to z, 0-9 or \_ ).  
**4**. Variable name is case sensitive in Python which means num and NUM are two different variables in python.

# **Data Types**

A data type defines the type of data, for example 123 is an integer data while “hello” is a String type of data. The data types in Python are divided in two categories:  
**1**. Immutable data types – Values cannot be changed.  
**2**. Mutable data types – Values can be changed

Immutable data types in Python are:  
**1**. [Numbers](https://beginnersbook.com/2018/02/python-numbers/)  
**2**. [String](https://beginnersbook.com/2018/02/python-strings/)  
**3**. [Tuple](https://beginnersbook.com/2018/02/python-tuple/)

Mutable data types in Python are:  
**1**. [List](https://beginnersbook.com/2018/02/python-list/)  
**2**. [Dictionaries](https://beginnersbook.com/2019/03/python-dictionary/)  
**3**. Sets

## Built-in Data Types

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | Str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | Dict |
| Set Types: | set, frozenset |
| Boolean Type: | Bool |
| Binary Types: | bytes, bytearray, memoryview |

# **Python Operators**

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators
* Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Example** |
| + | Addition | x + y |
| - | Subtraction | x – y |
| \* | Multiplication | x \* y |
| / | Division | x / y |
| % | Modulus | x % y |
| \*\* | Exponentiation | x \*\* y |
| // | Floor division | x // y |

* Python Comparison Operators

Comparison operators are used to compare two values:

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

* Python Logical Operators

Logical operators are used to combine conditional statements:

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

* Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| Is | Returns True if both variables are the same object | x is y |
| is not | Returns True if both variables are not the same object | x is not y |

## Python Bitwise Operators

Bitwise operators are used to compare (binary) numbers:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Description** |
| & | AND | Sets each bit to 1 if both bits are 1 |
| | | OR | Sets each bit to 1 if one of two bits is 1 |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 |
| ~ | NOT | Inverts all the bits |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the  leftmost bits fall off |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left,and let the rightmost bits fall off |

# **Python Lists**

**List** is a collection which is ordered and changeable. Allows duplicate members.

## 1.Create a List in Python

To create a list all you have to do is to place the items inside a **square bracket []** separated by **comma**( **,** ).

num\_list = [11.22, 9.9, 78.34, 12.0]

## 2. Accessing the items of a list

**Syntax to access the list items:**

**list\_name[index]**

## 3. List Operations

## Addition

### Update elements

### Delete elements

## 4. List Length

To determine how many items a list has, we use the **len()** function.

## 5. Add Items

To add an item to the end of the list, we use the **append()** method.

## 6. Remove Item

### There are several methods to remove items from a list: **remove(), pop()** and **clear()** methods.

# **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 Loops**

Python has two primitive loop commands:

* while loops
* for loops

## The while Loop

The while loop we can execute a set of statements as long as a condition is true. The while loop requires relevant variables to be ready.

## The break Statement

With the **break** statement we can stop the loop even if the while condition is true.

## The continue Statement

## With the continue statement we can stop the current iteration, and continue with the next.

## The else Statement

With the else statement we can run a block of code once when the condition no longer is true.

## The For Loop

## A for loop is used for iterating over a sequence. With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

## The break Statement

## With the break statement we can stop the loop before it has looped through all the items.

## The continue Statement

## With the continue statement we can stop the current iteration of the loop, and continue with the next.

## The range() Function

## The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

## Else in For Loop

The else keyword in a for loop specifies a block of code to be executed when the loop is finished.

## The pass Statement

## for loops cannot be empty, but if you for some reason have a for loop with no content, put in the pass statement to avoid getting an error.

# **Python If ... Else**

## If

## An "if statement" is written by using the if keyword. Enters in if conditions are true.

## Elif

## The elif keyword is python way of saying "if the previous conditions were not true, then try this condition".

## Else

## The else keyword catches anything which isn't caught by the preceding conditions.

# **Python Functions**

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

## *Creating a Function*

## In Python a function is defined using the def keyword.

## *Calling a Function*

## To call a function, use the function name followed by parenthesis.

## *Parameters or Arguments?*

The terms**parameter**and**argument** can be used for the same thing: information that are passed into a function.

**From a function's perspective**:

* 1. A parameter is the variable listed inside the parentheses in the function definition.

## An argument is the value that is sent to the function when it is called. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

## *Default Parameter Value*

## If we call the function without argument, it uses the default value.

## *Passing a List as an Argument*

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

**E.g.** if you send a List as an argument, it will still be a List when it reaches the function.

* ***Return Values***

To let a function return a value, use the **return**statement.

## *The pass Statement*

## function definitions cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

## *Recursion*

## Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result. Python also accepts function recursion, which means a defined function can call itself.