## What is python?

Python is a general purpose programming language which was created by Guido Van Rossum. It is a high level programming language which resembles the English language.

Python is used to for desktop app development, web development, mobile app development and video game development. The use case of python doesn’t end there.

It is also a cross platform language which means you can write code on one platform (Ex: Windows) and execute on another (Ex: Mac OS).

## Getting ready

In order to execute python programs in your computer you first have to install the python interpreter on your computer.

You can test whether python is installed on your computer or not by typing the following code on your command line or terminal.

For windows:

*C:\Users\Your Name>python --version*

For Linux or Mac:

*python --version*

If python is not installed on your computer, go to the <https://www.python.org> and download it install it first.

## Saving python files

Before we learn python syntax you should know how to save python files. Saving python files is not a hard task.

1. First type your code in a text file. Whether on a normal text or code editor or the one that comes with the python interpreter.
2. Then save that file with the ‘.py’ extension.

## Executing python files

Once you have written the code, you have to execute it in order to see the results. You can execute your in the command line.

To run the python code type the following in your command line:

C:\Users\Your Name>python pythonFileName.py

## Python comments

Comments are text which is ignored by the interpreter. Comments are commonly used by programmer for them selves or other programmers for later development.

**Singleline comments**

* Single line comments starts with a ‘#’ symbol.

**Multiline comments**

* Multiline comments starts end with three singe quotes (‘) or double quotes (“).
* A comment starts with single quotes (‘) can not end with double quotes (“) and vice versa.
* They can spread to more than one line.

## The ‘print’ statement

The print statement is used to display information to users. It accepts one or more parameters separated by commas.

*print(“Hello”,”world”)*

Output *-> Hello world*

## Python Variables

A variable is a name given data which can be used an manipulated by using that name.

### Defining variables

When defining variable you should give it a name and a value. In the following way:

variableName = initialValue

*x = 34*

### Naming Variable

When you name variables you should follow certain guidelines. The guidelines are mentioned below.

* Python variables names can only contain uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9) and underscore (\_).
* Variable names can not start with a number
* Reserve words such as ‘print’ can not be used as a variable name.
* Variable names are case sensitive.

**Naming convention** - There are two common ways variable names are written in python.

* Snake case - words are separated by an underscore (\_). Ex: variable\_name
* Camel case - every now word is started with a capital letter. Ex: variableName

### The assignment sign

The assignment sign (=) is know as the equal sign in the field of mathematics. As explained earlier when you define a variable a value must be assigned to it and some times you might want to change the value of a variable after it’s initiation. In both cases we use the assignment sign to assign a value to variables.

Value on the right side of the assignment sign will be assign to the variable which is located in the left side of the assignment sign.

*X = 56*

*Y = 37*

*X = 6*

## Python data types

Data type is an important concept to understand in programming. In python there are different data types. You can assign values of any data type to variables.

### String

String data type refers to plain text. String values are surrounded by either single or double quotes.

*x = “Hello world”*

*name = ‘peter’*

### Integer

Integer is number data types. Numbers of this data type don’t have decimal points.

*x = 6*

*y = -7*

### Float

Float numbers have decimal points.

*x = 3.5*

*y = -9.34*

### List

List allow us to assign more than one value to a single variable. Using square brackets ([ ]). You can assign values of different data types to a list.

*x = [4, 8.32, “Home”]*

In addition you can also declare an empty list.

*y = []*

You can access the values of the list using their index number. Index numbers starts from zero.

*x = [4, 8.32, “Home”]*

*x[0]* --> 4

*x[2]* --> “Home”

You can replace an old value in a list with a new one. You can do this by referring to the value using it’s index and assigning a new value.

*x = [4, 8.32, “Home”]*

*x[1 = “New”]*

*print(x)* --> [4,”New”,”Home”]

### Tuples

Like lists you can store one or more values in it. The main difference are they use round brackets ( () ) instead of square brackets ([]) and they are unchangeable, you are stuck with the initial values.

*daysOfTheWeek = (‘Monday’, ‘Tuesday’, ‘Wednesday’, ‘Thursday’, ‘Friday’, ‘Saturday’, ‘Sunday’)*

You can access the tuple items using their index just like we did with the list.

*daysOfTheWeek[3]* --> ‘Thursday’

### Dictionaries

Dictionary is another data collections in python. But it unlike list and tuple the data are stored using a key, value pair. Dictionaries are written with curly brackets ({})

dictName = {key1 : value1, key2 : value2, …….}

*person1 = {“name” :”Kevin”, “age”: 34,” dob” : 1986}*

You can access dictionary values by using their names within a square bracket.

*Person1[“dob”]* --> 1986

## Python Casting

In python you can change a value from one data type to another this process is called type casting. This is done by using the following functions.

* int()
* float()
* str()

### int()

This function turns a float (by rounding it down to the closest integer) or a string (if the string represent an integer or a float) in to integer.

int(2.7) --> 2

int(“3”) --> 3

### float()

This turns an integers or a string (if the string represent an integer or a float) in to float.

float(3) --> 3.0

float(“4”) --> 4.0

### str()

This turns an integer or a float in to string.

str(3) --> “3”

str(4.6) --> “4.6”

## Python booleans

Boolean means either *True* or *False* value. Even though they look simple, they are important to many operations in python such as conditions and loops which we will explore later.

Many values are *True* by default. Any value that has a content in it is a *True* value. There are a few exceptions: Empty string, the number “0”, empty list, empty tuple and empty dictionary are *False* values.

## Python operators

Python comes with many operators perform operations on variables and values. They can be categorized in to several groups. We will look in to only three different groups of operators in this documentation.

Python arithmetic operators

Arithmetic operators are used to perform arithmetic operation on values and variables.

|  |  |  |
| --- | --- | --- |
| **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 assignment operators

Assignment operators are used to assign values to variables.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same as** |
| = | X = 3 | 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 |

### Python comparison operators

Comparison operators are used to compare two values or variables. When you compare two values or variables you will get ‘True’ or ‘False’ as result.

|  |  |  |
| --- | --- | --- |
| **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 if…else

If statement allow your program to make decisions based on one or more conditions. If statement use the following structure.

if (condition):

Code

*a = 50*

*b = 100*

*if ( a < b):*

*print(‘b is bigger than a’)*

--> ‘b is bigger than a’

We get the output because our condition is true. We would not have gotten any out put if it was not true.

If you notice the explanation above the code is indented. Indentation is how you start a code block in python as we have done above, we have started an if block.

### Else

We can also make our program execute a block of code if our condition is false. We us the keyword ‘else’ for it.

if (contition):

Code

else:

Code

*a = 150*

*b = 100*

*if ( a < b):*

*print(‘b is bigger than a’)*

*else:*

*print(‘a is bigger than b’)*

--> ‘a is bigger than b’

In this example the program evaluated the condition to be false so it executed the code in the else block.

### Elif

You can make your program make more than two decisions based on conditions more than one.

if (condition1):

code

elif (condition2):

code

elif (condition3):

code

elif (condition4):

code

Else:

code

*a = 50*

*b = 100*

*if ( a == b):*

*print(‘a is equal to b’)*

*elif (a > b):*

*print(‘a is bigger than b’)*

*else:*

*print(‘a is smaller than b’)*

--> ‘a is smaller than b’

In the example above the interpreter first checks if a and b are equal and gets false as results, again it checks if a is bigger than b, and gets false again, so it finally executes the else block because all the other conditions are false.

## Python functions

Functions are an important part of any programming language. Function is a block of code which will only be executed when it is called. You can also pass data called parameters in to a function, so the function can manipulate the data. You can also create functions without any parameters.

### Declaring a function

We declaring a function as follows.

def functionName():

code

*def sayHello():*

*print(‘Hello’)*

### Calling a function

We can call a function in python by mentioning it’s name followed by opening an closing parentheses.

functionName()

*sayHello()*

--> ‘Hello’

### Parameters

As mentioned earlier parameters are data passed to functions. You can pass one or more parameters to a function. They should be separated by commas.

def functionName(parameter):

code

***def sayName(name):***

***print(‘Hello’,name)***

When you call a function with a parameter you should pass data to that function, otherwise you will get an error.

functionName(data)

*sayName(‘Victor’)*

--> ‘Hello Victor’