

Introduction to Python

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.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has a syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax compared to other programming languages

- Python was designed for readability and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions, and classes. Other programming languages often use curly-brackets for this purpose.

Example

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

Python Getting Started

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

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

To check if you have python installed on a Linux or Mac, then on Linux open the command line or on Mac open the Terminal, and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>

Python Quickstart

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

```
helloworld.py
```

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

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command-line itself.

Type the following on the Windows, Mac, or Linux command line:

```
C:\Users\Your Name>python
```

Or, if the "python" command did not work, you can try "py":

```
C:\Users\Your Name>py
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

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

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

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

```
Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command-line interface:

```
exit()
```

Python Syntax

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

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

```
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

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.

Example

```
if 5 > 2:
```

```
    print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

Syntax Error:

```
if 5 > 2:  
  
print("Five is greater than two!")
```

The number of spaces is up to you as a programmer, but it has to be at least one.

Example

```
if 5 > 2:  
  
    print("Five is greater than two!")  
  
if 5 > 2:  
  
    print("Five is greater than two!")
```

You have to use the same number of spaces in the same block of code, otherwise, Python will give you an error:

Example

Syntax Error:

```
if 5 > 2:  
  
    print("Five is greater than two!")
```

```
print("Five is greater than two!")
```

Python Comments

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

Creating a Comment

Comments start with a `#`, and Python will ignore them:

Example

```
#This is a comment
```

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

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

Example

```
print("Hello, World!") #This is a comment
```

A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code:

Example

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

```
print("Cheers, Mate!")
```

Multi-Line Comments

Python does not really have a syntax for multi-line comments.

To add a multiline comment you could insert a `#` for each line:

Example

```
#This is a comment
```

```
#written in
```

```
#more than just one line
```

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

Or, not quite as intended, you can use a multiline string.

Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

Example

```
"""
```

```
This is a comment
```



```
written in  
  
more than just one line  
  
"""  
  
print("Hello, World!")
```

As long as the string is not assigned to a variable, Python will read the code, but then ignore it, and you have made a multiline comment.

Python Variables

Variables

Variables are containers for storing data values.

Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

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

Variables do not need to be declared with any particular *type*, and can even change type after they have been set.

Example

```
x = 4          # x is of type int

x = "Sally"    # x is now of type str

print(x)
```

Casting

If you want to specify the data type of a variable, this can be done with casting.

Example

```
x = str(3)     # x will be '3'

y = int(3)     # y will be 3

z = float(3)   # z will be 3.0
```

Get the Type

You can get the data type of a variable with the `type()` function.

Example

```
x = 5

y = "John"

print(type(x))

print(type(y))
```

Single or Double Quotes?

String variables can be declared either by using single or double quotes:

Example

```
x = "John"

# is the same as

x = 'John'
```

Case-Sensitive

Variable names are case-sensitive.

Example

This will create two variables:

```
a = 4
```

```
A = "Sally"
```

```
#A will not overwrite a
```

Variable Names

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

Example

Legal variable names:

```
myvar = "John"
```

```
my_var = "John"
```

```
_my_var = "John"
```

```
myVar = "John"
```

```
MYVAR = "John"
```

```
myvar2 = "John"
```

Example

Illegal variable names:

```
2myvar = "John"
```

```
my-var = "John"
```

```
my var = "John"
```

Remember that variable names are case-sensitive

Multi Words Variable Names

Variable names with more than one word can be difficult to read.

There are several techniques you can use to make them more readable:

Camel Case

Each word, except the first, starts with a capital letter:

```
myVariableName = "John"
```

Pascal Case

Each word starts with a capital letter:

```
MyVariableName = "John"
```

Snake Case

Each word is separated by an underscore character:

```
my_variable_name = "John"
```

Many Values to Multiple Variables

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

Example

```
x, y, z = "Orange", "Banana", "Cherry"
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

Note: Make sure the number of variables matches the number of values, or else you will get an error.

One Value to Multiple Variables

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

Example

```
x = y = z = "Orange"
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

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*.

Example

Unpack a list:

```
fruits = ["apple", "banana", "cherry"]
```

```
x, y, z = fruits
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

Output Variables

The Python `print` statement is often used to output variables.

To combine both text and a variable, Python uses the `+` character:

Example

```
x = "awesome"

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

You can also use the `+` character to add a variable to another variable:

Example

```
x = "Python is "

y = "awesome"

z = x + y

print(z)
```

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

Example

```
x = 5

y = 10

print(x + y)
```


If you try to combine a string and a number, Python will give you an error:

Example

```
x = 5

y = "John"

print(x + y)
```

Global Variables

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

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

Example

Create a variable outside of a function, and use it inside the function

```
x = "awesome"

def myfunc():

    print("Python is " + x)

myfunc()
```

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.

Example

Create a variable inside a function, with the same name as the global variable

```
x = "awesome"

def myfunc():

    x = "fantastic"

    print("Python is " + x)

myfunc()

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

The global Keyword

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.

Example

If you use the `global` keyword, the variable belongs to the global scope:

```
def myfunc():

    global x

    x = "fantastic"
```

```
myfunc()
```

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

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

Example

To change the value of a global variable inside a function, refer to the variable by using the `global` keyword:

```
x = "awesome"
```

```
def myfunc():
```

```
    global x
```

```
    x = "fantastic"
```

```
myfunc()
```

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