**PYTHON**

Python is a popular programming language.

Python can be used on a server to create web applications.

**Learning by Examples**

print("Hello, World!")

**OUTPUT**

Hello, World!

## 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 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.

### **Good to know**

* The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### **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.

## 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  
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32  
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  
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32  
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()

## Execute Python Syntax

Python syntax can be executed by writing directly in the Command Line:

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

[**Execute Python Syntax**](https://www.w3schools.com/python/python_syntax.asp#execute_python_syntax)[**Python Indentation**](https://www.w3schools.com/python/python_syntax.asp#python_indentation)

[**Python Variables**](https://www.w3schools.com/python/python_syntax.asp#python_variables)[**Python Comments**](https://www.w3schools.com/python/python_syntax.asp#python_comments)[**Exercises**](https://www.w3schools.com/python/python_syntax.asp#exercises)

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!")

**OUTPUT**

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!")

**OUTPUT**

File "demo\_indentation\_test.py", line 2  
    print("Five is greater than two!")  
        ^  
IndentationError: expected an indented block

The number of spaces is up to you as a programmer, the most common use is four, 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!")

**OUTPUT**

Five is greater than two!

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!")

**OUTPUT**

File "demo\_indentation2\_error.py", line 3  
    print("Five is greater than two!")  
    ^  
IndentationError: unexpected indent

## Python Variables

In Python, variables are created when you assign a value to it:

### **Example**

Variables in Python:

x = 5  
y = "Hello, World!"

**OUTPUT**

5

Hello, World!

Python has no command for declaring a variable.

## Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

### **Example**

Comments in Python:

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

# **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 starts with a #, and Python will ignore them:

### **Example**

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

**OUTPUT**

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

**OUTPUT**

Hello, World!

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!")

**OUTPUT**

Cheers, Mate!

## Multiline Comments

Python does not really have a syntax for multiline 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!")

**OUTPUT**

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!")

**OUTPUT**

Hello, World!

# **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)

**OUTPUT**

5

John

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)

**OUTPUT**

Sally

## 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

**OUTPUT**

3

3

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))

**OUTPUT**

<class ‘int’ >

<class ‘str’ >

## Single or Double Quotes?

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

### **Example**

x = "John"

print(x)

#double quotes are the same as single quotes:

x = 'John'

print(x)

**OUTPUT**

John

John

## Case-Sensitive

Variable names are case-sensitive.

### **Example**

This will create two variables:

a = 4

A = "Sally"

print(a)

print(A)

**OUTPUT**

4

Sally

# **Python - Variable Names**

## 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)
* A variable name cannot be any of the [Python keywords](https://www.w3schools.com/python/python_ref_keywords.asp).

### **Example**

Legal variable names:

myvar = "John"

my\_var = "John"

\_my\_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

print(myvar)

print(my\_var)

print(\_my\_var)

print(myVar)

print(MYVAR)

print(myvar2)

**OUTPUT**

John

John

John

John

John

John

### **Example**

Illegal variable names:

2myvar = "John"  
my-var = "John"  
my var = "John"

#This example will produce an error in the result

Traceback (most recent call last):

File "/usr/lib/python3.7/py\_compile.py", line 143, in compile

\_optimize=optimize)

File "<frozen importlib.\_bootstrap\_external>", line 791, in source\_to\_code

File "<frozen importlib.\_bootstrap>", line 219, in \_call\_with\_frames\_removed

File "./prog.py", line 1

2myvar = "John"

^

SyntaxError: invalid syntax

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "<string>", line 1, in <module>

File "/usr/lib/python3.7/py\_compile.py", line 147, in compile

raise py\_exc

py\_compile.PyCompileError: File "./prog.py", line 1

2myvar = "John"

^

SyntaxError: invalid syntax

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"

# **Python Variables - Assign Multiple Values**

## 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)

**OUTPUT**

Orange

Banana

Cherry

**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)

**OUTPUT**

Orange

Orange

Orange

## 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**

Apple

Banana

Cherry

# **Python - Output Variables**

## Output Variables

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

### **Example**

x = "Python is awesome"  
print(x)

**OUTPUT**

Python is awesome

In the print() function, you output multiple variables, separated by a comma:

### **Example**

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

**OUTPUT**

Python is awesome

# **Python - Global Variables**

## 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()

**OUTPUT**

Python is awesome

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)

**OUTPUT**

Python is fantastic

Python is awesome

## 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)

**OUTPUT**

Python is awesome

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)

**OUTPUT**

Python is fantastic

# **Python Data Types**

## 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 |
| None Type: | NoneType |

## Getting the Data Type

You can get the data type of any object by using the type() function:

### **Example**

Print the data type of the variable x:

x = 5  
print(type(x))

**OUTPUT**

**<**class ‘int’**>**

## Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

|  |  |  |
| --- | --- | --- |
| **Example** | **Data Type** | **Output** |
| x = "Hello World" | str | Hello World <class 'str'> |
| x = 20 | int | 20 <class 'int'> |
| x = 20.5 | float | 20.5 <class 'float'> |
| x = 1j | complex | lj <class 'complex'> |
| x = ["apple", "banana", "cherry"] | list | ['apple', 'banana', 'cherry'] <class 'list'> |
| x = ("apple", "banana", "cherry") | tuple | ('apple', 'banana', 'cherry') <class 'tuple'> |
| x = range(6) | range | {'name': 'John', 'age': 36} <class 'dict'> |
| x = {"name" : "John", "age" : 36} | dict | {'name': 'John', 'age': 36} <class 'dict'> |
| x = {"apple", "banana", "cherry"} | set | frozenset({'cherry', 'banana', 'apple'}) <class 'frozenset'> |
| x = frozenset({"apple", "banana", "cherry"}) | frozenset | frozenset({'apple', 'banana', 'cherry'}) <class 'frozenset'> |
| x = True | bool | True <class 'bool'> |
| x = b"Hello" | bytes | b'Hello' <class 'bytes'> |
| x = bytearray(5) | bytearray | bytearray(b'\x00\x00\x00\x00\x00') <class 'bytearray'> |
| x = memoryview(bytes(5)) | memoryview | <memory at 0x00D58FA0> <class 'memoryview'> |
| x = None | NoneType | None  <class 'NoneType'> |

## Setting the Specific Data Type

|  |  |  |
| --- | --- | --- |
| **Example** | **Data Type** | **Try it** |
| x = str("Hello World") | str | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_str2) |
| x = int(20) | int | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int2) |
| x = float(20.5) | float | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float2) |
| x = complex(1j) | complex | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex2) |
| x = list(("apple", "banana", "cherry")) | list | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list2) |
| x = tuple(("apple", "banana", "cherry")) | tuple | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple2) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range2) |
| x = dict(name="John", age=36) | dict | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict2) |
| x = set(("apple", "banana", "cherry")) | set | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set2) |
| x = frozenset(("apple", "banana", "cherry")) | frozenset | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset2) |
| x = bool(5) | bool | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool2) |
| x = bytes(5) | bytes | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes2) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray2) |
| x = memoryview(bytes(5)) | memoryview | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview2) |

## Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions: