```
In [1]:
from IPython.core.display import HTML
def css_styling():
    styles = open("./w3schools.css", "r").read()
    return HTML(styles)
css_styling()
```

Out[1]:

PYTHON HOME

```
In [1]:
print("Hello, World!")
```

```
Hello, World!
```

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.

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, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial 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 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!
```

```
In [8]:
print("Hello, World!")
```

```
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()
```

PYTHON SYNTAX

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!")
```

```
In [29]:
if 5 > 2:
    print("Five is greater than two!")
```

```
Five is greater than two!
```

```
In [31]:
if 5 > 2:
print("Five is greater than two!")
```

```
  File "<ipython-input-31-a314491c53bb>", line 2
    print("Five is greater than two!")
    ^
IndentationError: expected an indented block
```

```
In [32]:
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
        print("Five is greater than two!")
```

```
Five is greater than two!
Five is greater than two!
```

```
In [35]:
if 5 > 2:
 print("Five is greater than two!")
        print("Five is greater than two!")
```

```
  File "<ipython-input-35-a4dba24e2a47>", line 3
    print("Five is greater than two!")
    ^
IndentationError: unexpected indent
```

PYTHON COMMENT

```
In [37]:
# INI SINGLE LINE COMMENT
print("Hello, World!")
```

```
Hello, World!
```

```
In [38]:
# INI MULITPLE
# LINE COMMENT
# DI PYTHON
print("Hello, World!")
```

```
Hello, World!
```

```
In [40]:
### atau
"""
INI JUGA MULTIPLE
LINE COMMENT
"""
print("Hello, World!")
```

```
Hello, World!
```

PYTHON VARIABLES

MEMBUAT VARIABEL

```
In [41]:
x = 5
y = "John"
print(x)
print(y)
```

```
5
John
```

Casting

```
In [42]:
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

```
In [43]:
x = 5
y = "John"
print(type(x))
print(type(y))
```

```
<class 'int'>
<class 'str'>
```

SINGLE OR DOUBLE QUOTES

```
In [44]:
x = "John"
# is the same as
x = 'John'
```

CASE SENSITIVE

```
In [45]:
a = 4
A = "Sally"
```

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)

```
In [46]:
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

```
In [47]:
2myvar = "John"
my-var = "John"
my var = "John"
```

```
  File "<ipython-input-47-0e82719f22b4>", line 1
    2myvar = "John"
     ^
SyntaxError: invalid syntax
```

Many Values to Multiple Variables

```
In [48]:
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

```
Orange
Banana
Cherry
```

One Value to Multiple Variables

```
In [49]:
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

```
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.

```
In [50]:
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

```
apple
banana
cherry
```

Output Variables

The Python print statement is often used to output variables.

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

```
In [51]:
x = "Python is "
y = "awesome"
z =  x + y
print(z)
```

```
Python is awesome
```

GLOBAL VARIABLES

```
In [52]:
x = "awesome"

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

myfunc()

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

```
Python is fantastic
Python is awesome
```

```
In [53]:
x = "awesome"

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

myfunc()

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

```
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

Python Numbers

There are three numeric types in Python:

int float complex Variables of numeric types are created when you assign a value to them:

Example

```
In [58]:
x = 1    # int
y = 2.8  # float
z = 1j   # complex
print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

PYTHON CASTING

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

- int() - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)
- float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

```
In [60]:
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
print(x, y, z)
```

```
1 2 3
```

PYTHON STRING

Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

You can display a string literal with the print() function:

```
In [61]:
print("Hello")
print('Hello')
```

```
Hello
Hello
```

PYTHON BOOLEANS

Booleans represent one of two values: True or False.

Boolean Values

In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

```
In [62]:
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

```
True
False
False
```

PYTHON LIST

List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

Lists are created using square brackets:

```
In [19]:
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
thislist1 = ["apple"]
thislist.extend(thislist1)
print(thislist)
```

```
['apple', 'banana', 'cherry', 'apple', 'cherry', 'apple']
```

Tuple

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and unchangeable.

Tuples are written with round brackets.

```
In [3]:
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

```
In [4]:
thistuple[0]='alfan'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-aa0054e16236> in <module>
----> 1 thistuple[0]='alfan'

TypeError: 'tuple' object does not support item assignment
```

SETS

Set

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.

A set is a collection which is unordered, unchangeable*, and unindexed.

```
In [2]:
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

```
{'cherry', 'apple', 'banana'}
```

DICTIONARIES

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered*, changeable and do not allow duplicates.

As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

Dictionaries are written with curly brackets, and have keys and values:

```
In [68]:
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```