Lecture 3 Takeaways

Monday, 21.03.2022

—

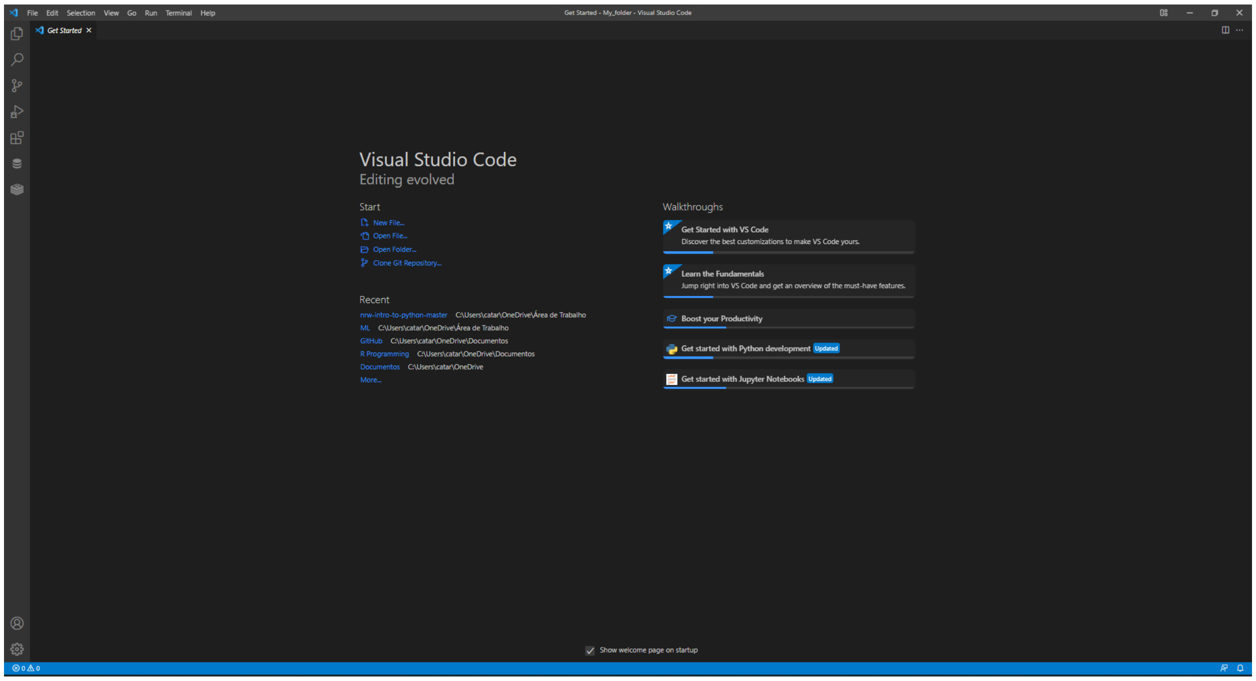
Introducing computers

“At a high level, all computers are made up of a processor (CPU), memory, and input/output devices. Each computer receives input from a variety of devices, processes that data with the CPU and memory, and sends results to some form of output.”

If you are interested in this topic you can find more information about Computer components here: *https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:computers*

Introducing VS Code

Basics of VS Code


File
Edit
Selection
View
Go
Run
Terminal
Help
Get Started - My_folder - Visual Studio Code
Get Started
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Recent
nrw-intro-to-python-master C:\Users\catar\OneDrive\Área de Trabalho
ML C:\Users\catar\OneDrive\Área de Trabalho
GitHub C:\Users\catar\OneDrive\Documentos
R Programming C:\Users\catar\OneDrive\Documentos
Documentos C:\Users\catar\OneDrive
More...
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Jump right into VS Code and get an overview of the must-have features.
Boost your Productivity
Get started with Python development Updated
Get started with Jupyter Notebooks Updated
Show welcome page on startup

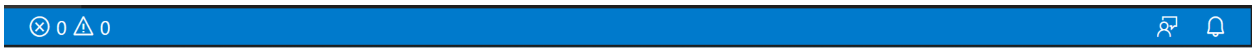
Above we can see our VS Code Welcome Page.

Starting in the left side bar, or Activity Bar, we can see different symbols:

🗐	[Top of the bar] Explorer We can see our current files and folders here.
🔍	[Top of the bar] Search You can perform searches here.
⑂	[Top of the bar] Source Control You can track changes and code versions here.
▷	[Top of the bar] Run and Debug You can Run and Debug Code here.
⊞	[Top of the bar] Extensions You can Manage and download extensions here. Make sure you have the Jupyter and Python extension installed.
👤	[Bottom of the bar] Accounts
⚙	[Bottom of the bar] Manage ! Tip: If you are not a dark fan you can change the VS Code colors by click in Manage » Color Theme » Select a different theme from Command Palette ! Tip: If you want to take a look of some shortcuts in VS Code click in Manage » Keyboard Shortcuts

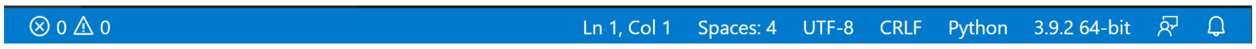
Other symbols can appear if you have more extensions and features installed, but these ones are the basic. For now you will only need the Explorer, Search and Extensions.

In the bottom blue bar we have the status bar, if we still in the VS Code Welcome Page it will look like this:



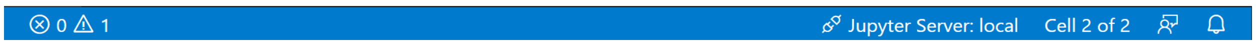
ⓧ 0 ⚠ 0

If the have a python file open it will look like this:



ⓧ 0 ⚠ 0 Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.9.2 64-bit

If the have a Jupyter Notebook open it will look like this:



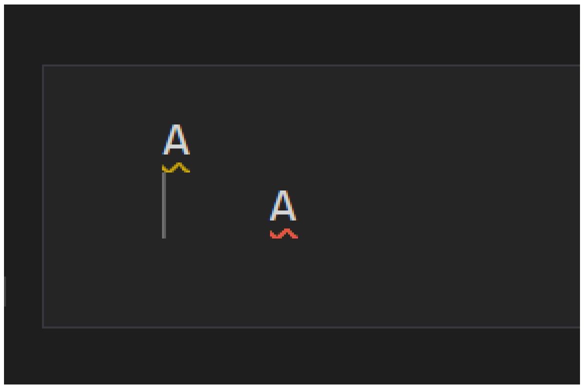
ⓧ 0 ⚠ 1 Jupyter Server: local Cell 2 of 2

What information can we see here?

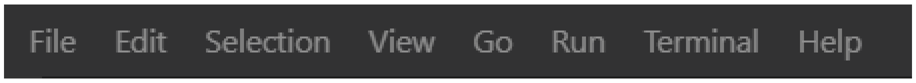
ⓧ 0 ⚠ 1	Warnings and errors in file
Ln 1, Col 1	[Python File] Specify your location within the File. For example, in this case I had my cursor in the 1st line of the file. The Column information is associated with indentation. We will talk about this in upcoming

	lessons.
Spaces: 4	[Python File] This information is also associated with indentation. We will talk about this in upcoming lessons.
UTF-8	Encoding
CRLF	End of Line Sequence
Python	Language Mode used in the file
3.9.2 64-bit	Kernel
Jupyter Server: local	[Jupyter Notebook] Specify Jupyter server for connections
Cell 2 of 2	[Jupyter Notebook] Specify your location within the File. For example, in this case I had two cells in my Jupyter Notebook and my cursor was in the second one.
🗨 🔔	Feedback icon and Notifications icon

! Indentation refers to the spaces used in the beginning of the code. For example, my first A is in the first position and the second in the fourth:



Now, in the upper bar we can see different commands and features. Explore a bit your VS Code to understand how they are organized:



Manage Folders and Files

! My advice, create a folder in your desktop for the course, so you can have all your files in one single place.

Before starting editing files open this folder in VS Code by clicking in the upper bar in File » Open Folder » Select the folder created before.

We have two types of files we are going to work on in the course, Python Files and Jupyter Notebooks. What are the main differences?

» Python File allows you to create scripts and write programs in Python. Runs in Terminal and the Output is presented as one.

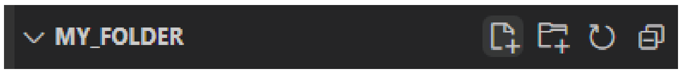
» Jupyter Notebooks are easier for beginners because they are divided into cells and you can run small pieces of code and see the output instead of running all the script.

Now lets create Files!

We can create a file in a few different ways in VS Code, try to explore all:

» File in upper bar » New File » Select a language » Python

» Explorer in the left bar » Icon New File in our Folder » name your file (use filename.py to create a Python file or filename.ipynb to create a Jupyter Notebook).



» Manage in the left bar » Command Palette (Command Palette it will appear in the upper bar) » Create: New File (if it is not appearing in the dropdown, write the command) » Select between Text File (to create Python File) or Jupyter Notebook.

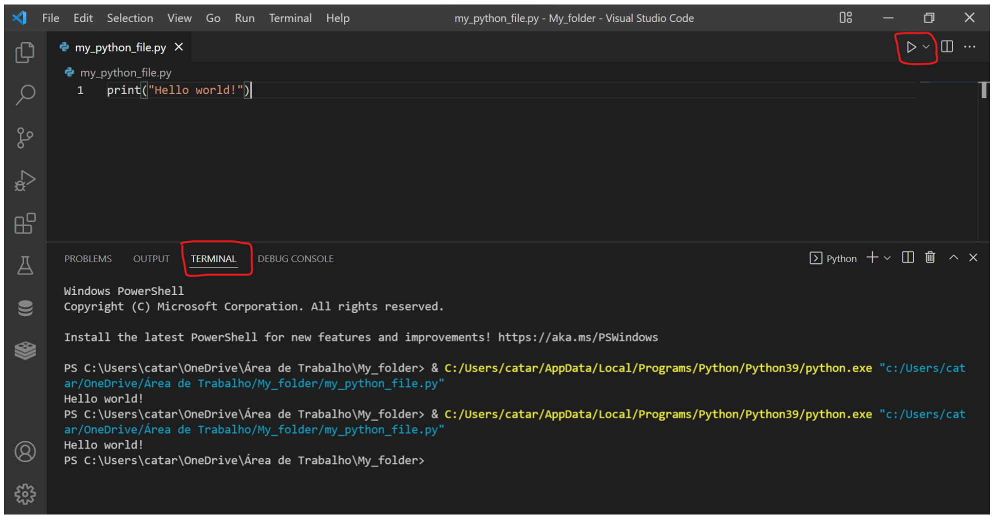
» Manage in the left bar » Command Palette (Command Palette it will appear in the upper bar) » Jupyter: Create New Jupyter Notebook

To save a File use File in upper bar » Save Or Use the shortcut Crtl+S.

Note: You can also create a Python File or Jupyter Notebook directly in your desktop without using VS Code. You just need to right click on your mouse and create a new text file and change the extension to .py or .ipynb instead of the .txt.

Getting Started

When we work with a Python File:


File Edit Selection View Go Run Terminal Help
my_python_file.py - My_folder - Visual Studio Code
my_python_file.py
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Python

```
print("Hello world!")
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\catar\OneDrive\Área de Trabalho\My_folder> & C:/Users/catar/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/cat
ar/OneDrive/Área de Trabalho/My_folder/my_python_file.py"
Hello world!
PS C:\Users\catar\OneDrive\Área de Trabalho\My_folder> & C:/Users/catar/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/cat
ar/OneDrive/Área de Trabalho/My_folder/my_python_file.py"
Hello world!
PS C:\Users\catar\OneDrive\Área de Trabalho\My_folder>
```

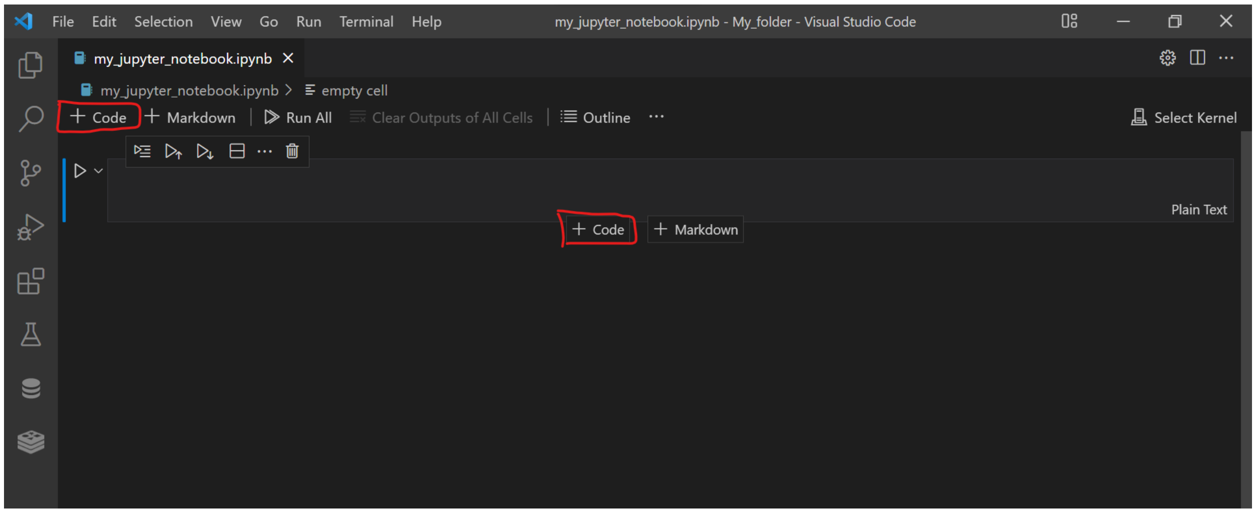
In our file we write our code » print("Hello world!") in this case.

To Run our code we click on the rightwards filled arrow in the upper right of the screen. The output will appear in our terminal. If you run the code more than once, the output will appear multiple times in the terminal.

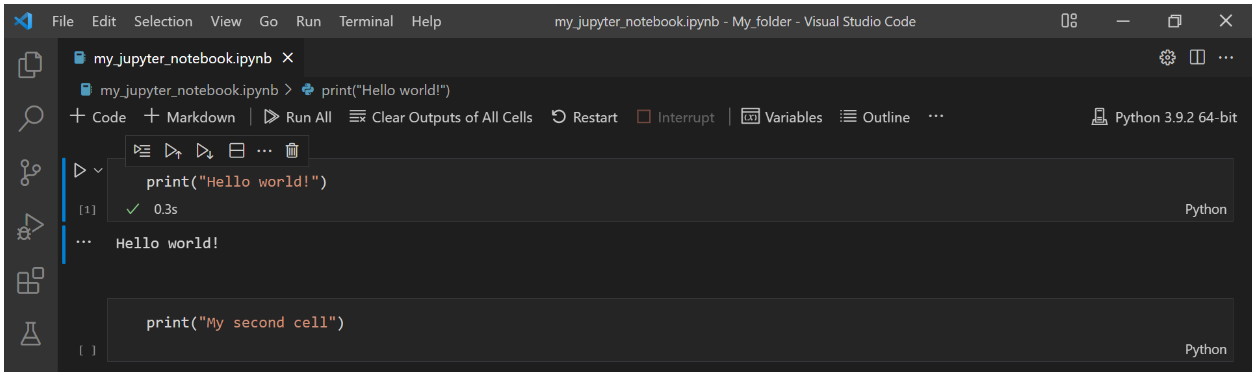
! Tip: If you prefer to see the Terminal in the left or right side of the screen you can just change the appearance following: View in upper bar » Appearance » Panel Position (and choose the one you prefer).

When we work with a Jupyther Notebook:

To create a New cell click in one of the following:


File Edit Selection View Go Run Terminal Help
my_jupyter_notebook.ipynb - My_folder - Visual Studio Code
my_jupyter_notebook.ipynb
my_jupyter_notebook.ipynb > empty cell
+ Code
+ Markdown
Run All
Clear Outputs of All Cells
Outline
Select Kernel
Plain Text
+ Code
+ Markdown

You can write different things in different cells:

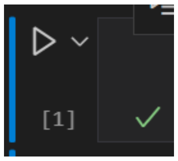

File Edit Selection View Go Run Terminal Help
my_jupyter_notebook.ipynb - My_folder - Visual Studio Code
my_jupyter_notebook.ipynb
my_jupyter_notebook.ipynb > print("Hello world!")
+ Code + Markdown Run All Clear Outputs of All Cells Restart Interrupt Variables Outline
Python 3.9.2 64-bit
[1]
0.3s
Python
[ ]
Python

```
print("Hello world!")
```

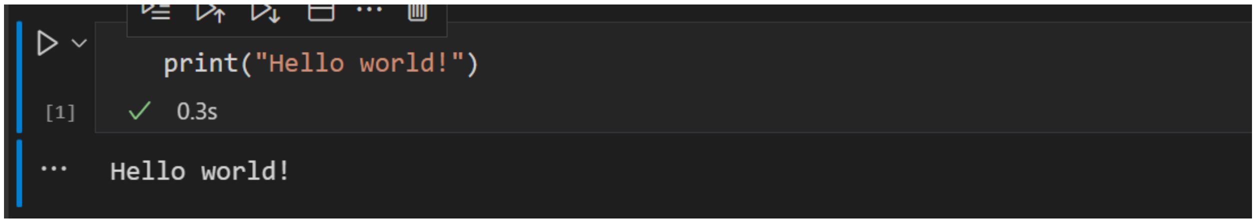
```
Hello world!
```

```
print("My second cell")
```

To Run our code we click on the rightwards arrow in the upper left of each cell.

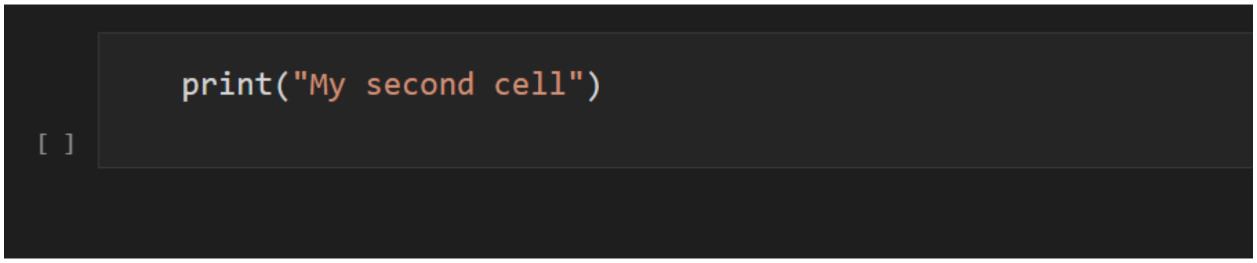


Let's take a look to the differences between a cell executed and one cell that we didn't execute yet:



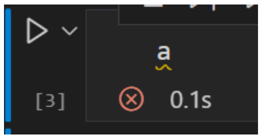
```
print("Hello world!")
```

```
Hello world!
```



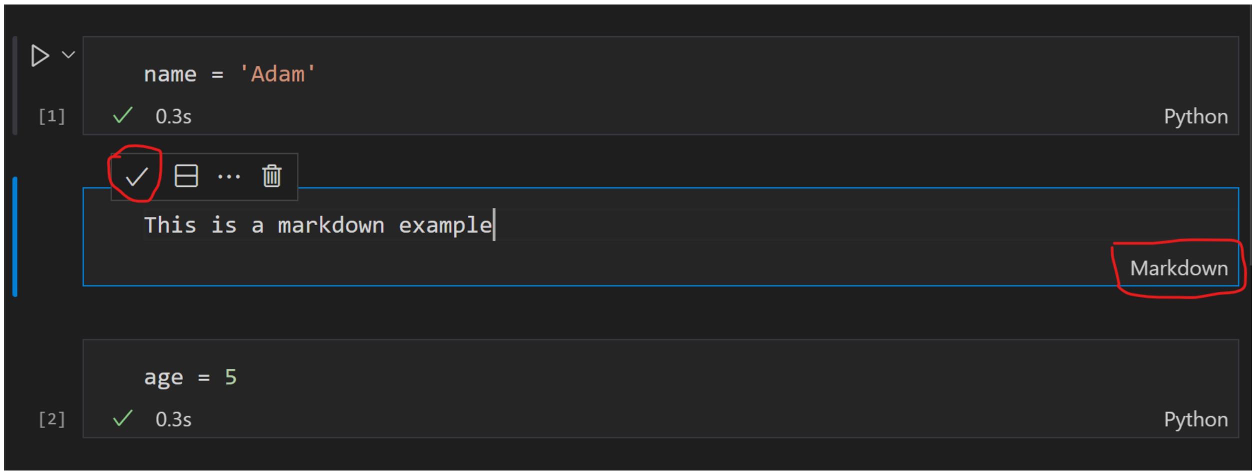
```
print("My second cell")
```

The executed cell shows us a small number (indicates the execution order, the number increases if you run the same cell another time or if you execute another cell), a green right symbol indicating that the execution was successful and the time that the cell takes to run (in this case 0.3 seconds). If we have a error in our code a red cross will appear when we run the cell:



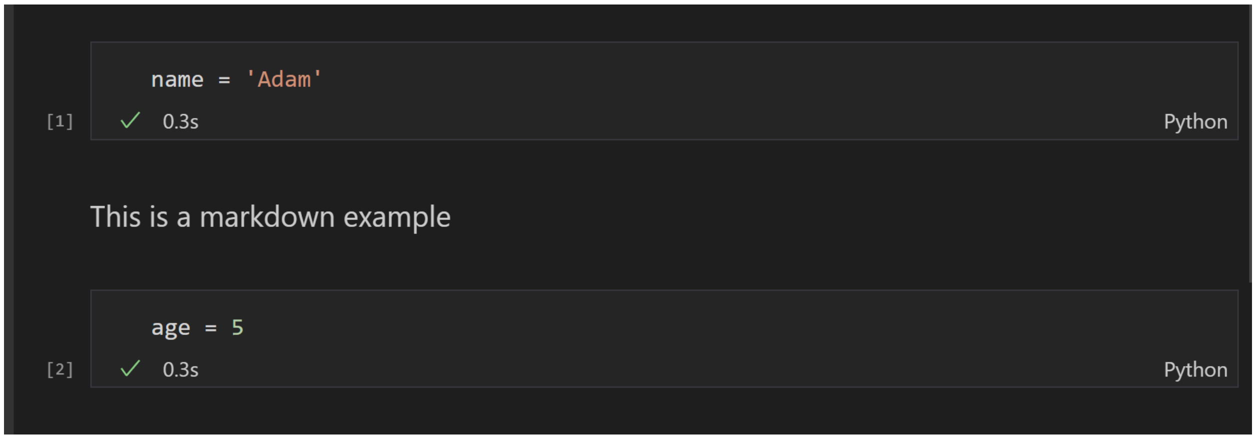
```
a
```

!Tip: If you want to keep some text information between your code you can use a markdown cell instead. Just write what you want and click in the right bottom. They are ignored by Python, so make sure you don't write code in one.



```
[1]  name = 'Adam'
     ✓ 0.3s                                    Python

     This is a markdown example
                                               Markdown

[2]  age = 5
     ✓ 0.3s                                    Python
```

After clicking in the right bottom it will look like this (if you want to edit again, just double click in the text):



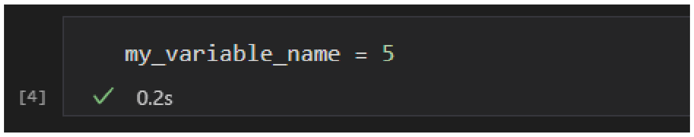
```
[1]  name = 'Adam'
     ✓ 0.3s                                    Python

     This is a markdown example

[2]  age = 5
     ✓ 0.3s                                    Python
```

! Tip: If you want to change the cell type just click on 'Python' or 'Markdown' in the right bottom side of the cell and select the type.

Some Python Initial Information

Variable:

Variables are containers for storing data values. A variable is created the moment you first assign a value to it. The assign operator in Python is the symbol =. Example:



```
my_variable_name = 5
```
```
[4] ✓ 0.2s
```

! Common practice: In Python the variable names should be all lowercase, be descriptive and with individual words separated by underscores as necessary to improve readability. They can contain both letters and numbers, but they should not start with a number.

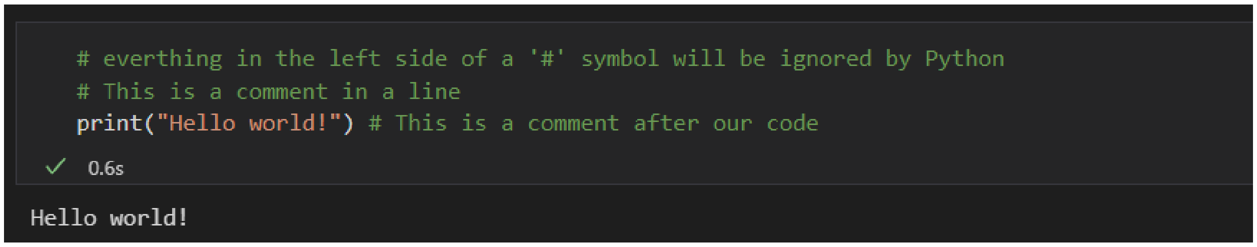
! Python uses reserved keywords to recognize the structure of the program, and they cannot be used as variable names. If you use one of this as your variable name you can get a Syntax error:

and	del	from	None	True
as	elif	global	nonlocal	try
assert	else	if	not	while
break	except	import	or	with
class	False	in	pass	yield
continue	finally	is	raise	async
def	for	lambda	return	await

So make sure your variable names are different from the words above!

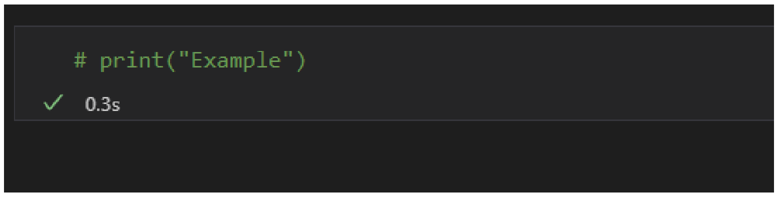
Comments:

Common practice: add comments into your code to help anyone reading to understand what the code does. The Comment sections are ignored by Python and are indicated by the '#' character.



```
# everthing in the left side of a '#' symbol will be ignored by Python
# This is a comment in a line
print("Hello world!") # This is a comment after our code
```
```
✓ 0.6s
Hello world!
```

You can also put a ‘#’ before a piece of your code to not run it:



```
# print("Example")
✓ 0.3s
```

Data Types

We have different data types in Python:

Boolean (or bool)	True, False
Integers (or int)	123, −123
Floating-point numbers (or float)	1.23
Complex number (or complex)	3+4j
String (or str)	'Pa', 'John'

These ones are more complex and are used as a collection of different data types. Don't worry about them now. You just need to know how they look. In the upcoming lectures we will talk about them and when they are used in more detail.

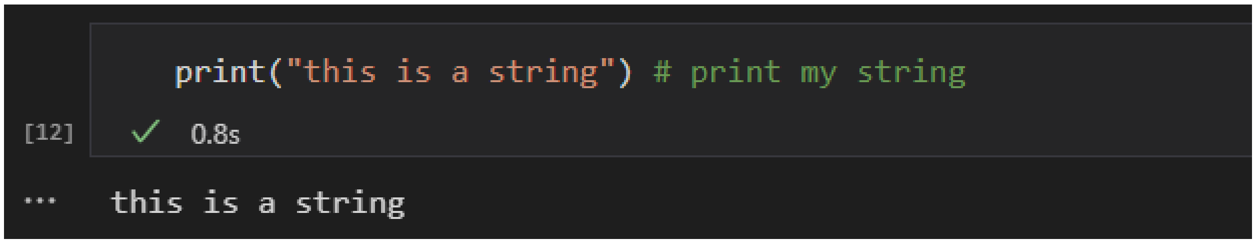
list	In lists we use square brackets mylist = ["abc", 34, True, 40]
tuple	In tuples we use round brackets mytuple = ("abc", 34, True, 40)
set	In sets we use curly brackets myset = {"abc", 34, True, 40}

Dictionaries (or dict)	In dictionaries we use curly brackets also but we have key:value pairs. Example name:age
	mydict = {
	  "Adam": 42,
	  "Ana": 25,
	  "John": 85
	}

Initial Functions

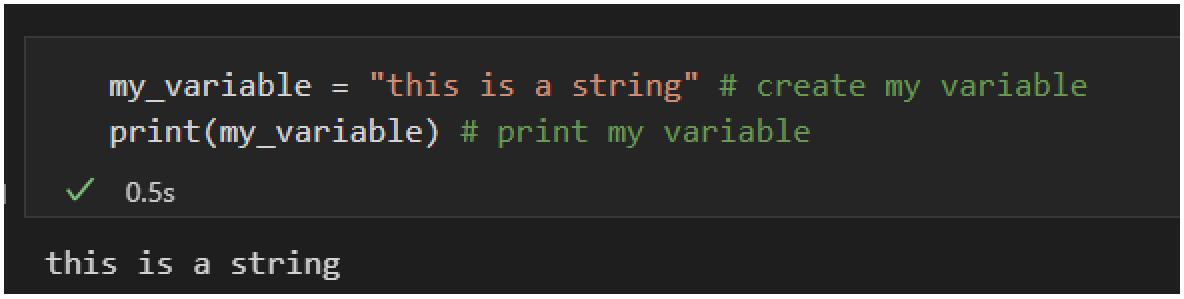
print(x)	Use to print x

We can print directly:



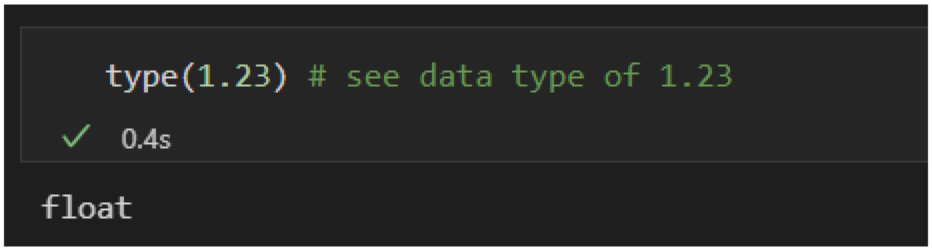
```
print("this is a string") # print my string
[12]  ✓ 0.8s
...  this is a string
```

Or assign the data to a variable and print this variable:

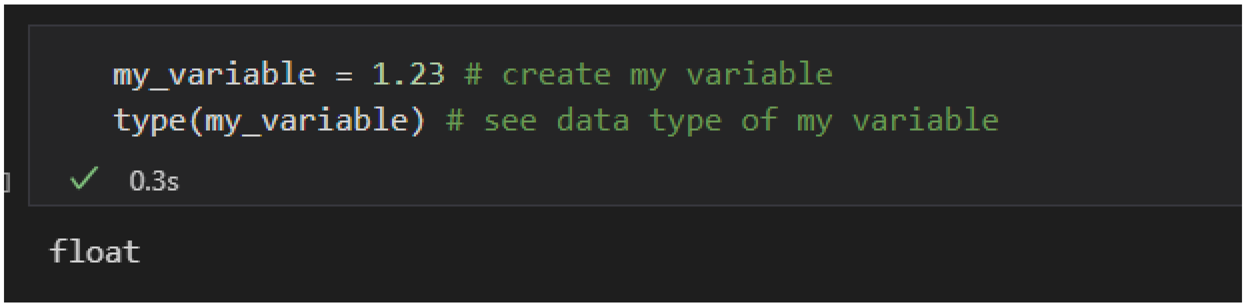


```
my_variable = "this is a string" # create my variable
print(my_variable) # print my variable
✓ 0.5s
this is a string
```

type(x)	Use to check the data type of x



```
type(1.23) # see data type of 1.23
✓ 0.4s
float
```

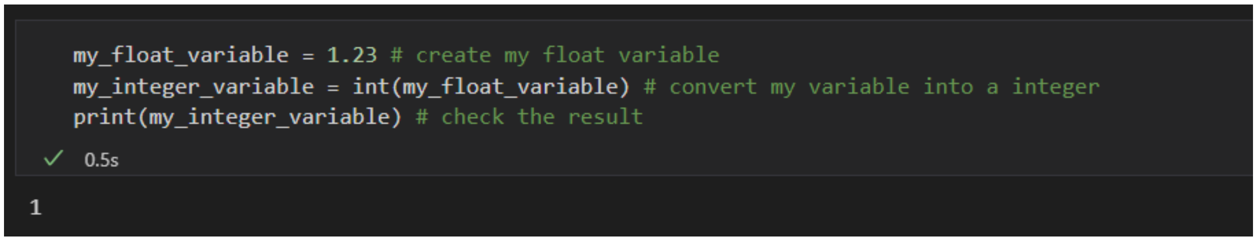


```
my_variable = 1.23 # create my variable
type(my_variable) # see data type of my variable
✓ 0.3s
float
```

int(x)	Use to perform conversions between data types

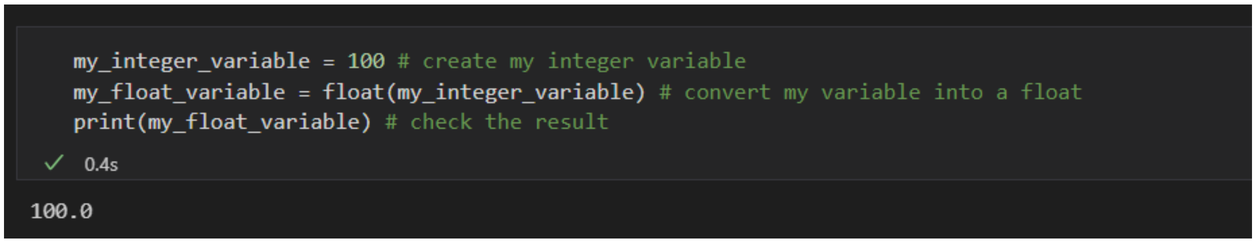
float(x)	Use to perform conversions between data types
complex(x)	Use to perform conversions between data types

Example 1:



```
my_float_variable = 1.23 # create my float variable
my_integer_variable = int(my_float_variable) # convert my variable into a integer
print(my_integer_variable) # check the result
✓ 0.5s

1
```

Example 2:

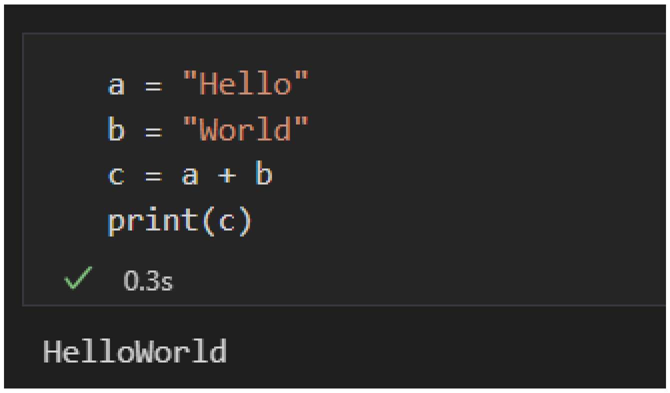


```
my_integer_variable = 100 # create my integer variable
my_float_variable = float(my_integer_variable) # convert my variable into a float
print(my_float_variable) # check the result
✓ 0.4s

100.0
```

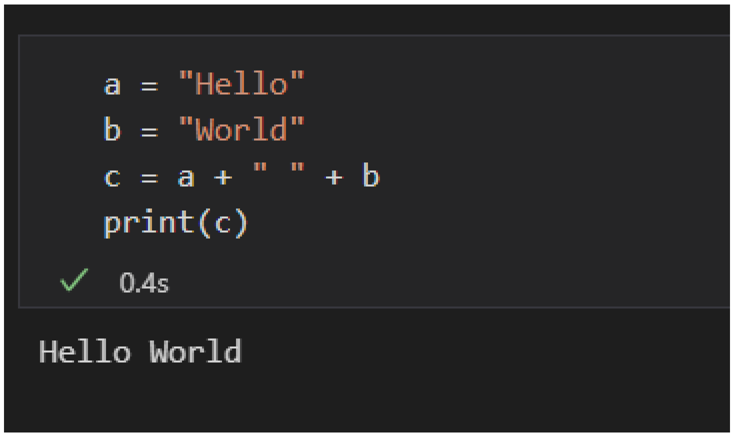
Strings

Strings in Python are enclosed within double quotes (" ") or single quotes (' ').

You can merge strings:



```
a = "Hello"
b = "World"
c = a + b
print(c)
✓ 0.3s

HelloWorld
```

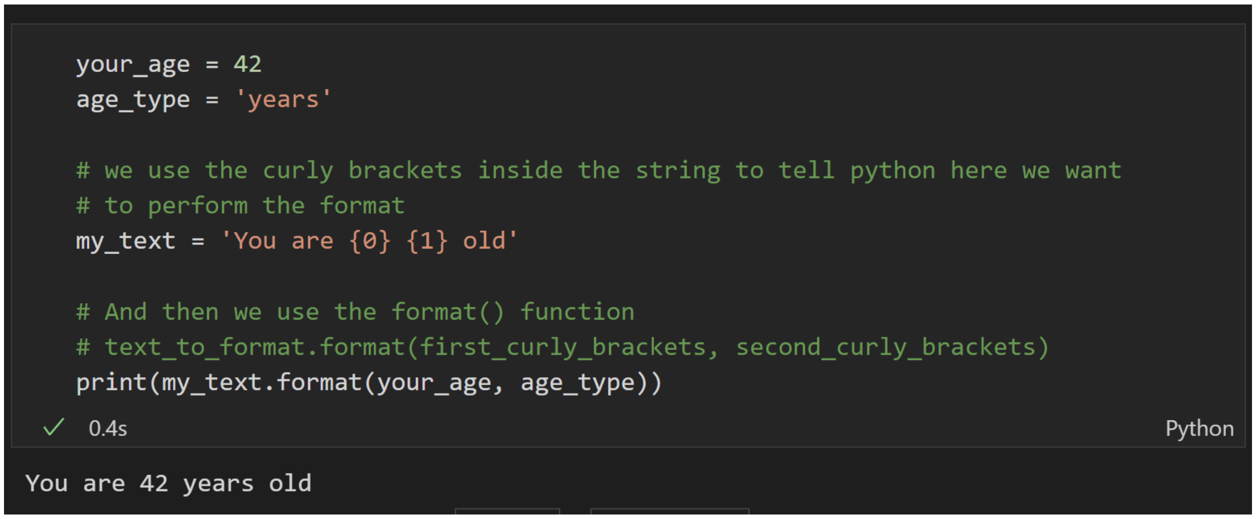


```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

✓ 0.4s

```
Hello World
```

If we want to put different information together in a String we can use format():

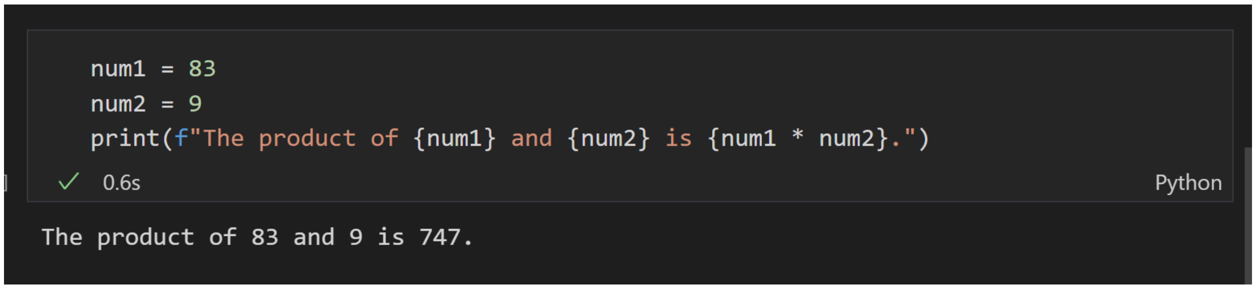


```
your_age = 42
age_type = 'years'

# we use the curly brackets inside the string to tell python here we want
# to perform the format
my_text = 'You are {0} {1} old'

# And then we use the format() function
# text_to_format.format(first_curly_brackets, second_curly_brackets)
print(my_text.format(your_age, age_type))
```

✓ 0.4s Python

```
You are 42 years old
```

Best practice: Instead of using format() we can also use f-String (formatted string literals). Basically we put a f or F before opening the quotes, and then we use the curly brackets to tell python where to include the variables.



```
num1 = 83
num2 = 9
print(f"The product of {num1} and {num2} is {num1 * num2}.")
```

✓ 0.6s Python

```
The product of 83 and 9 is 747.
```

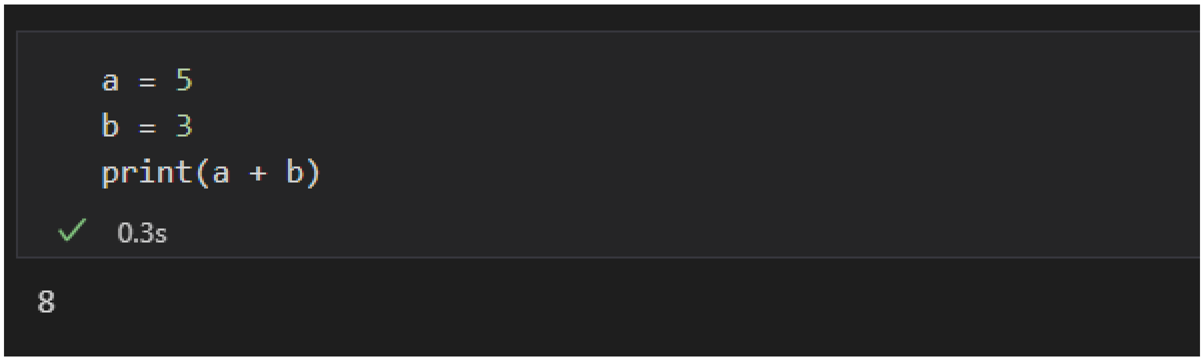
! We also have a third way to format strings: %-formatting. If you want to know more about it try to google it and see some examples.

Operators

Operators are special symbols that represent computations like addition and multiplication. The values the operator is applied to are called operands.

+	perform addition
-	perform subtraction
*	perform multiplication
/	perform division
**	perform exponentiation

Example



```
a = 5
b = 3
print(a + b)
```

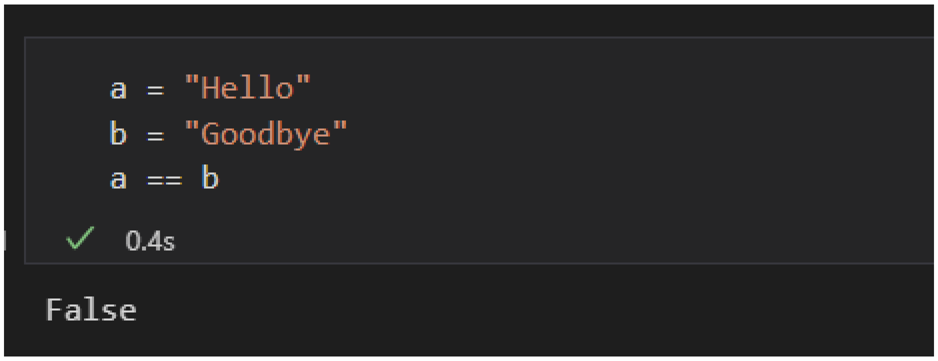
```
✓ 0.3s
```

```
8
```

Conditions

a == b	Check if a equals b
a != b	Check if a is not equal to b
a > b	Check if a is greater than b
a < b	Check if a is less than b
a >= b	Check if a is greater than or equal to b
a <= b	Check if a is less than or equal to b
a is b	Check if a is the same as b
a is not b	Check if a is not the same as b

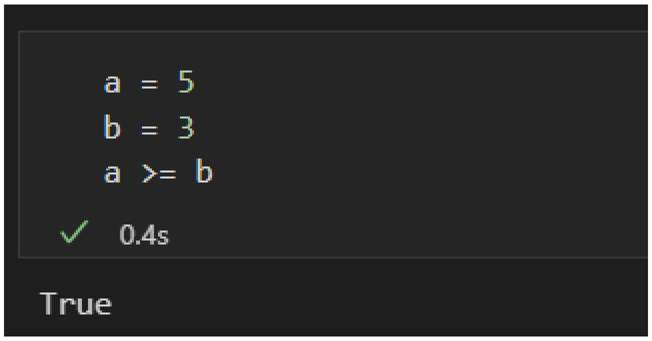
Example:



```
a = "Hello"
b = "Goodbye"
a == b
```

✓ 0.4s

```
False
```



```
a = 5
b = 3
a >= b
```

✓ 0.4s

```
True
```

Additional Resources

Take some time to explore the initial topics and see some examples:

https://www.w3schools.com/python/default.asp

! You can only learn if you make mistakes, so create a Jupyter Notebook and try to assign different variables, with different data types. Use the print functions, try to concatenate different variables and see what works or not.