Python for Thymio

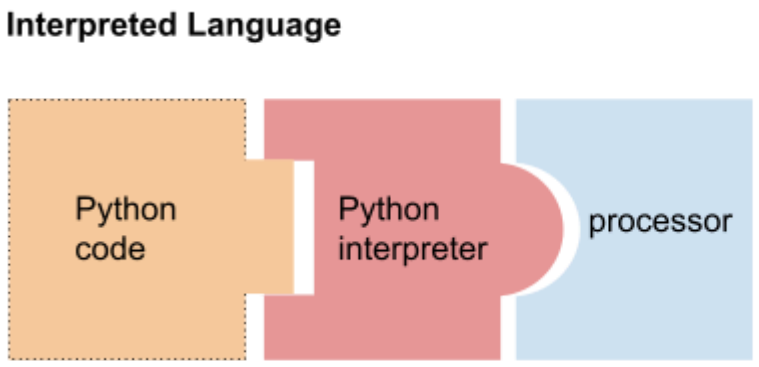
What is Python?

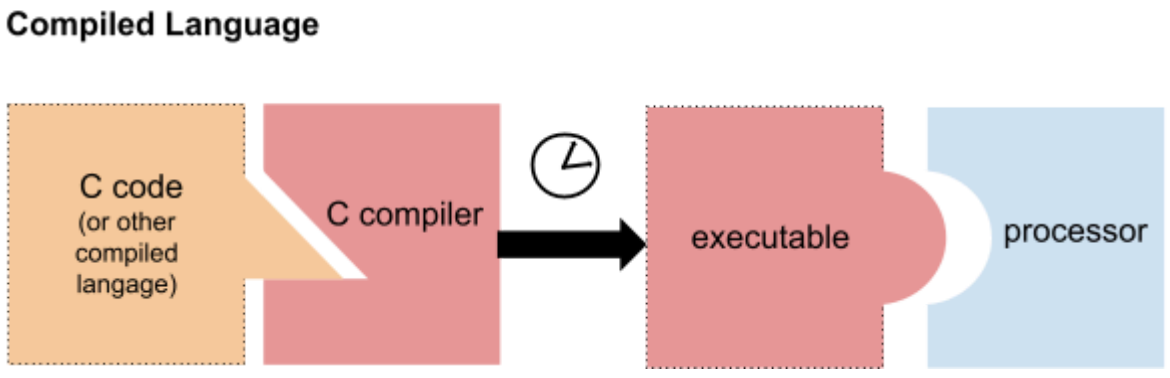
You have probably heard about Python as it is very much in demand these days. It is a programming language that is

- **high-level** (meaning that it is comparatively easier for humans to understand than low level languages)
- **object-oriented** (meaning that it is based around data) programming language.

Its syntax is relatively easy. It is now widely used in many domains and has a thriving community with plenty of libraries.

Python is also an **interpreted** language; There always has to be an **interpreter on the machine** that runs the code directly, without compiling, as opposed to a language like C++, where you first need a **compiler** to create an executable, but this executable can then be run even without the compiler installed.





Python for Thymio

In the case of Thymio, **there is no Python interpreter inside the robot**. Thymio runs Aseba, and has a lightweight Aseba Virtual Machine (VM). In order for Thymio to run a

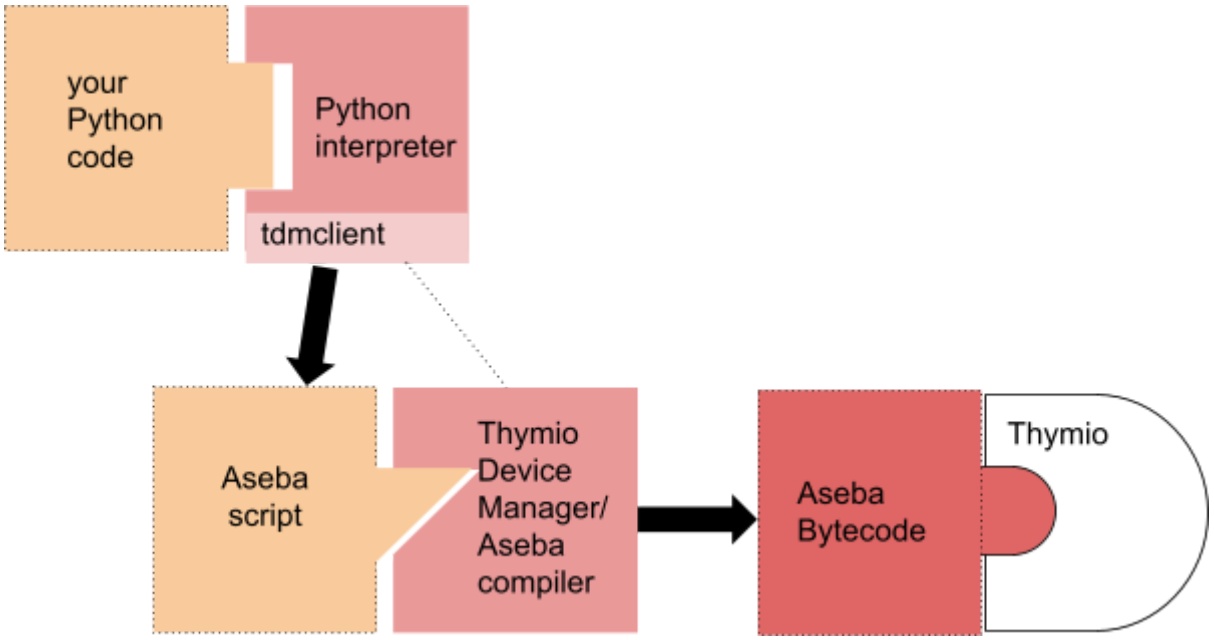
program, it has to be Aseba bytecode. When a program is created in VPL or Aseba, an Aseba script is compiled into Aseba bytecode by the Thymio Device Manager (TDM) and sent to the robot.

Therefore **a Python module (tdmclient)** was developed that **allows Python to communicate with the Thymio Device Manager (TDM)**.
In particular, this module allows to

- access the robot's variables
- transpile Python code to Aseba code
- send Aseba code to the TDM, which then sends adequate bytecode to the robot.

Thanks to this transpiler, **it is not necessary to learn the syntax of Aseba to program Thymio**. There are Python commands equivalent to the Aseba Thymio API, as well as some additional functionalities.
https://pypi.org/project/tdmclient/

https://github.com/epfl-mobots/tdm-python



Jupyter Notebooks

The materials and tutorials including Python code that you will find here are **in the form of Jupyter Notebooks**. Jupyter Lab is a **web app** which runs in your browser and allows to create **documents containing different types of cells**, such as markdown cells, code cells that can be directly executed etc (more information).
In a nutshell, this means that instead of having a terminal in which you type and run your Python commands, you will use a document where your code is presented in cells, along with cells of explanatory text, images, etc.

Prerequisite & Installation

This section explains every step required before you can start our exercises in Jupyter Notebooks.

Thymio Suite

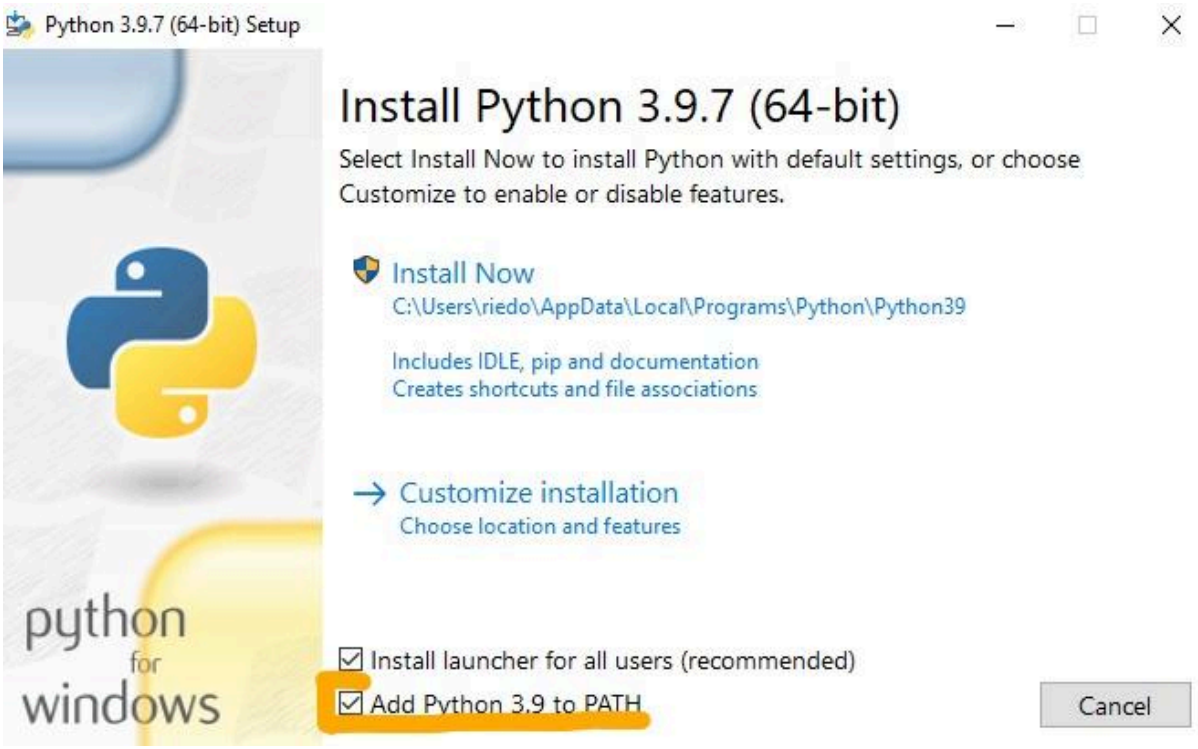
Installing Thymio Suite is necessary as the driver for Thymio and TDM are required for this application.
https://www.thymio.org/program/

Python 3

In order to run Python code, you need to have the **interpreter** installed on your computer. It is not integrated into Thymio Suite, you will have to install it separately.

You can find the latest version of Python here (version **3.8** or later needed):
https://www.python.org/downloads/
This will also install another utility, **pip**, which is needed below.

We recommend adding Python to PATH (checkbox on the install screen), as this will allow you to type shorter commands, but you can also type the whole path to the executable whe you use python or pip from the command line.

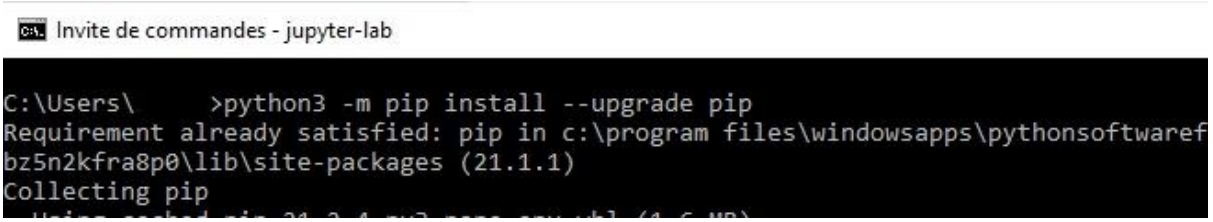


JupyterLab

JupyterLab will let you open Jupyter notebooks in your browser and will handle the Python interpreter. To install it, we will use pip, a utility which came with Python.
Open a terminal (on windows, look for “cmd” next to the start menu, on MacOS the app is called Terminal) and type:

```
python3 -m pip install --upgrade pip
```

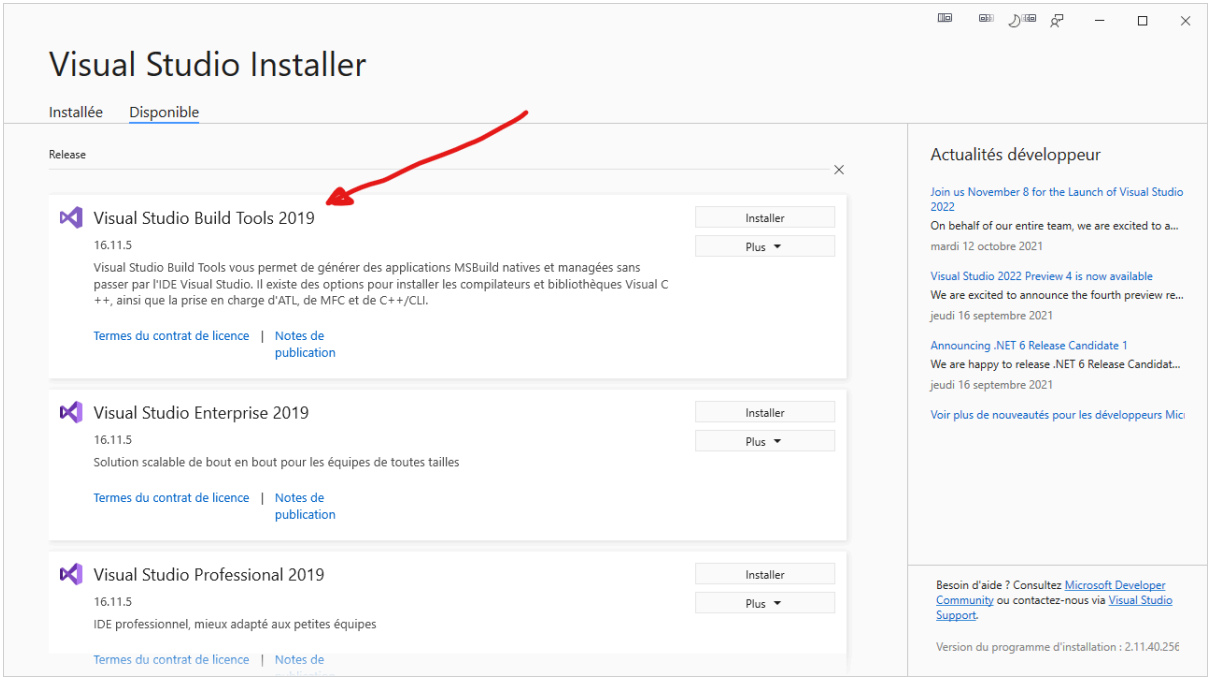
to make sure it is up to date. Depending on your install, the command might be “python3” or “python”.

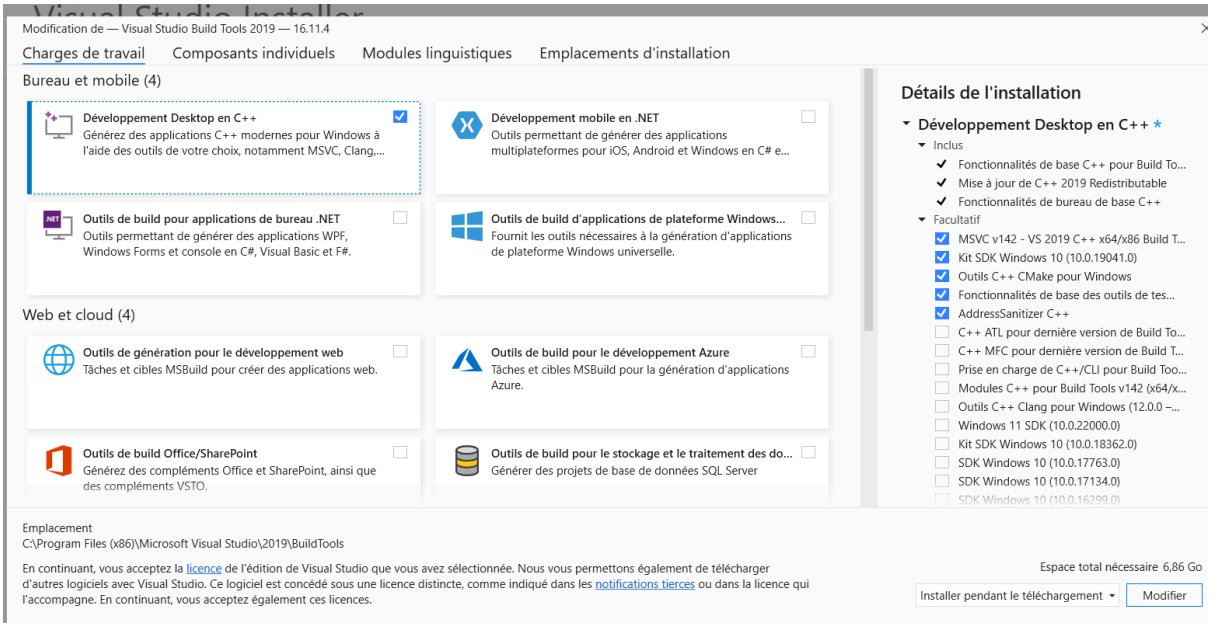


then, type:

```
python3 -m pip install jupyterlab
```

During installation, if an error arises saying: `error: Microsoft Visual C++ 14.0 is required`.
You can download the following installer:
https://visualstudio.microsoft.com/fr/visual-cpp-build-tools/





If you would rather not use jupyter notebooks, another page explains how to use TDMclient outside of Jupyter. *coming soon*

TDMclient

Tdmclient will be installed directly from the notebook (see our example notebook below).

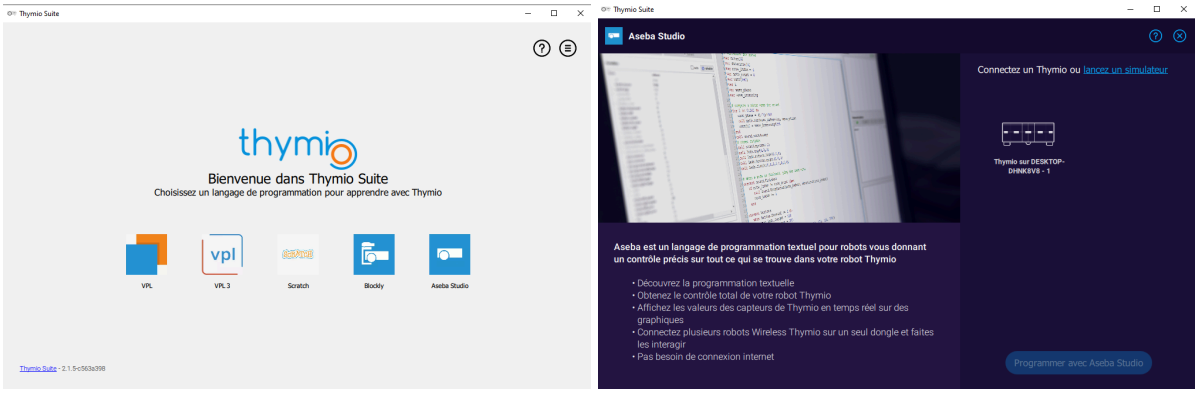
Our Notebooks

We prepared a series of notebooks to get started with Thymio on Python. Download them, and unzip them to the folder of your choice on your computer.
https://drive.google.com/file/d/1bJGHCR2mp_b47FweznHiBYKlexFXhz7A/view?usp=sharing
-coming-
(only 1 for now!)

Getting started

1. **Connect a Thymio robot** to your computer

2. **Start Thymio Suite**. Leave the welcome window open, this ensures that the TDM is running. If you wish to see which robots are connected, you can click on one of the programming languages to display the list, but do not start any of the interfaces.



keep Thymio Suite open, either on the welcome screen or the robot list

3. **Open a terminal**, and **type this command** to open Jupyter

   ```
   jupyter-lab
   ```

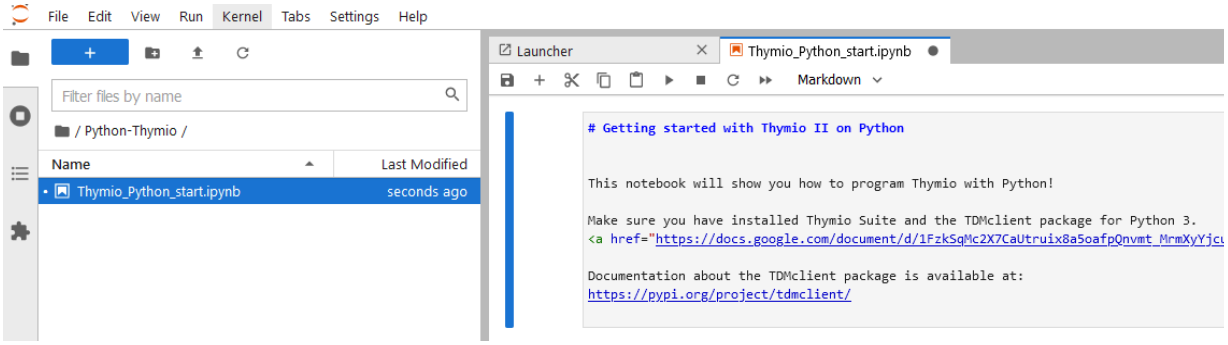
   This opens JupyterLab in your browser.

4. On the left, you can browse where you unzipped our notebooks. Once you have found them, double click on Thymio_Python_start.ipynb to open the notebook

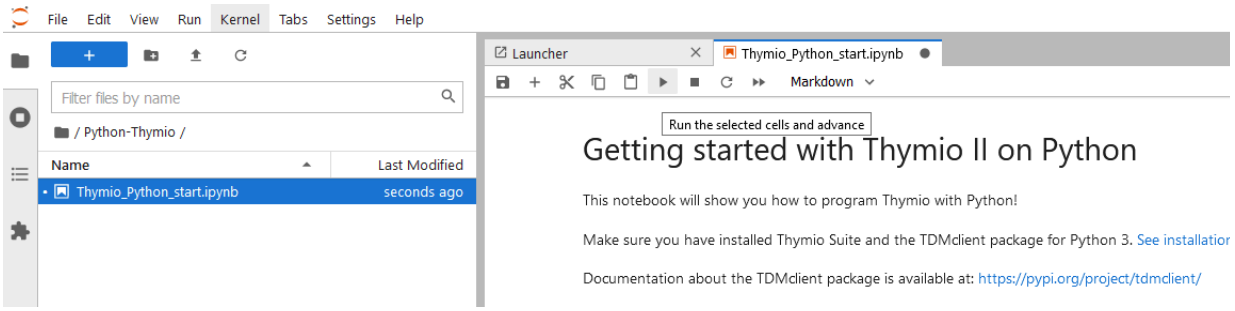
About Jupyter Notebooks

Once you've opened JupyterLab and the first notebook, you can see the structure. There are cells with titles, texts, links, etc, and cells with code.

If you double click on the first cell with text, its appearance will change (you will see the source) and you can edit it. It's a markdown cell. Some markers allow you to format your text (for example, # to indicate titles)



Then, if you execute this cell (play button on top of the interface, or shift+Enter), it will show its resulting aspect.



The other type of cell we use is code cells. Those cells contain Python code. When you execute them, the Python interpreter will execute the commands contained in them.

```
[ ]: my_measure = prox_horizontal[2]
     print(my_measure)
```

Sometimes, there will be an output, but not necessarily. While a cell is being executed, a * appears in the brackets next to it. Once it is executed, a number appears.

```
[2]: my_measure = prox_horizontal[2]
     print(my_measure)

     0
```

The result of the execution persists between cells, for example, if you assign a value to a variable, and use it later in another cell, the value is remembered.

 By clicking this arrow in the top menu, you can restart the kernel to start over.