

# Setting-up Python on your PC

Practical 1, part A

## Contents

Installing Python.....	1
Windows.....	1
Is there a Python already in my Windows PC? .....	1
Getting Python .....	3
Installing VS Code .....	4
Installing Python extension into VS Code.....	5
Checking Python interpreters in VS Code .....	6
Configuring default Windows terminal in VS Code .....	7
Mac OS .....	8
Linux .....	9
Hello Word ! .....	9
Selecting place for project folder.....	9
What Python interpreter do you use? .....	10

During this practical session you will install Python and Visual studio Code on your PC. This document is focused on Windows, because it's the most popular OS today. Installing Python and Visual Studio Code on Linux and Mac OS is briefly mentioned at the end. After the installation, you will test it by writing the "Hello Word!" script in Python in VS Code.

If you use Virtual Machines or Lab PC-s provided by the University, you don't need to install Python and VS-Code because they are already installed. However, you still may need to configure VS-Code (e.g. install Python extension or configure the default terminal).

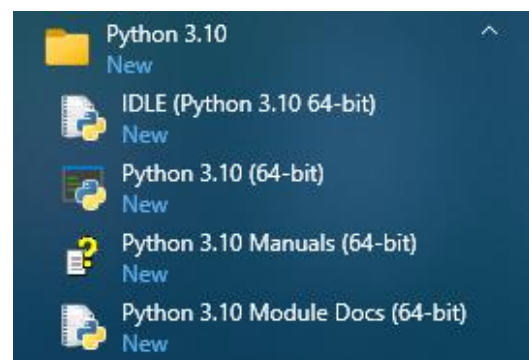
## Installing Python

### Windows

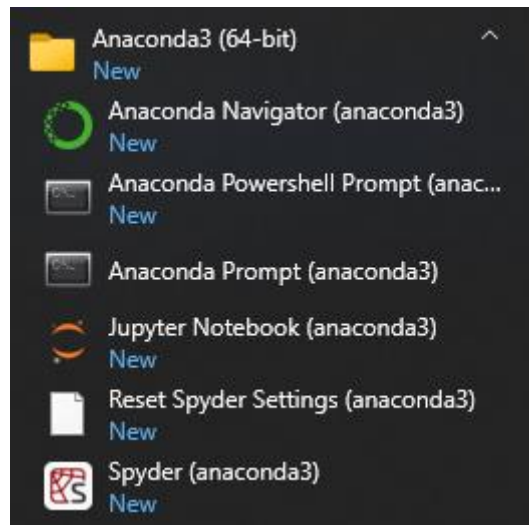
#### Is there a Python already in my Windows PC?

A simple way to answer this question is to look for Python(s) in the Windows main menu. Depending on the source of the installed Python, it may look differently.

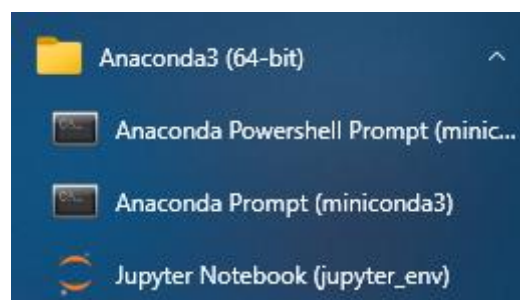
Python program group installed from Python.org may look like this:



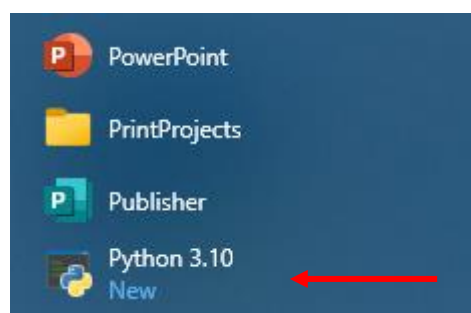
Python installed with Anaconda may look like this:



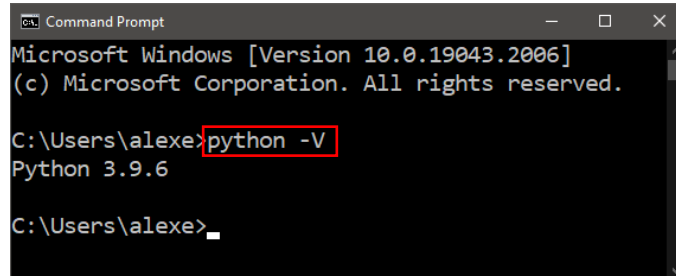
Python installed with Miniconda may look like this:



Python from MS App Store may look like a single entry without a program group:



Another way of checking whether you have Python on your Windows PC is by using Command Prompt terminal. To open Command Prompt terminal, you may type `cmd` in the search box near the Windows Menu symbol at the bottom left corner of your screen, and then select `cmd` terminal to open. Then, type `python -V` and hit **Enter**. You should see the version of system Python (the one, which is in the PATH of the terminal):

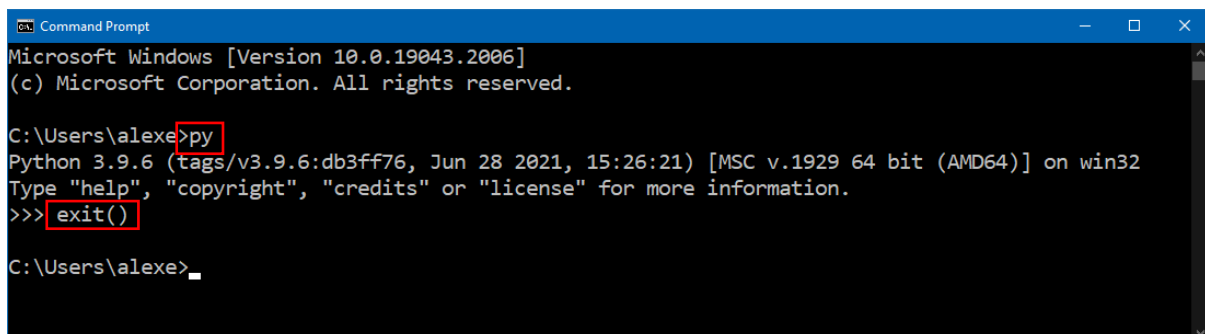


```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alexe>python -V
Python 3.9.6

C:\Users\alexe>
```

In some systems you may need to type `py` in command prompt instead:



```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alexe>py
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>exit()

C:\Users\alexe>
```

Use `exit()` to exit from the Python console.

If you already have Python(s) on your PC: how many Python(s) are on your PC? From what source(s)?

**If you work on Lab PC or use Cranfield VM: skip to page 5  
(you don't need to install Python & VS Code on Lab PC or VM)**

## Installing Python

If you still don't have Python on your PC, you need to install it now. The goal of this exercise is to install TWO versions of Python to your PC: one from Python.org and another from Anaconda (or Miniconda). If you already have Anaconda (or Miniconda) versions on your PC, you may skip this section.

By default, Python is not installed in Windows. There are three main sources, that could be used to get Python for Windows:

- Windows installer from Python.org  
<https://www.python.org/>
- Windows installer for Miniconda or Anaconda  
<https://docs.conda.io/en/latest/miniconda.html>  
<https://www.anaconda.com/download>
- Python from Microsoft Apps Store (I personally don't use it)  
[Microsoft Apps - Python Software Foundation](#)

Don't worry if you already have one of these versions installed. Adding another version from a different source will not harm your PC: just do NOT add the new Python to PATH, in case your old programs depend on the Python that was already present in your system (not adding Python to PATH is often the default option in the installers). Also, avoid installing Anaconda and Miniconda together (it may be confusing to use, unless you already know how to navigate between different versions of Conda on the same machine :)

Select the latest versions when you install Python(s). You may accept all the default settings (just click next on the installation screens). Don't worry, if the Python versions are different in different sources (e.g. Python 3.13 in Python.org and Python 3.12 in Anaconda or Miniconda). For most of today's PCs you should choose Windows 64-bit installers.

**Anaconda** installation includes many additional tools and features (e.g. Jupyter notebook, Spyder IDE etc). Most of these tools and features will not be needed during this module (except for possible use of the native Jupyter Notebook for the last optional task on Friday). At the same time, Anaconda takes up to 5GB space on your disk, and it may need long time to install (especially on an old slow PC). So, you may prefer Miniconda instead.

**Miniconda** includes a minimal set of tools, which allows comfortable management of Python environments on your PC (including Python itself and Conda package manager).

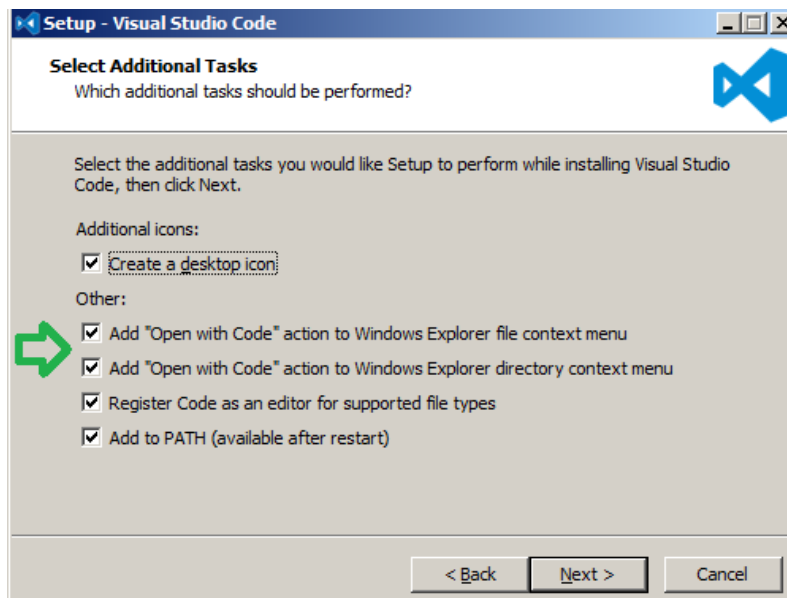
## Installing VS Code

Visual Studio Code is the recommended IDE for our Python course. However, you may use Pycharm, Spyder, or any other appropriate IDE, if you are already familiar with it.

Download and install VS Code following the instructions, provided on its web site:

- <https://code.visualstudio.com/>

During the installation, when you see the screen like below, check all the additional boxes:

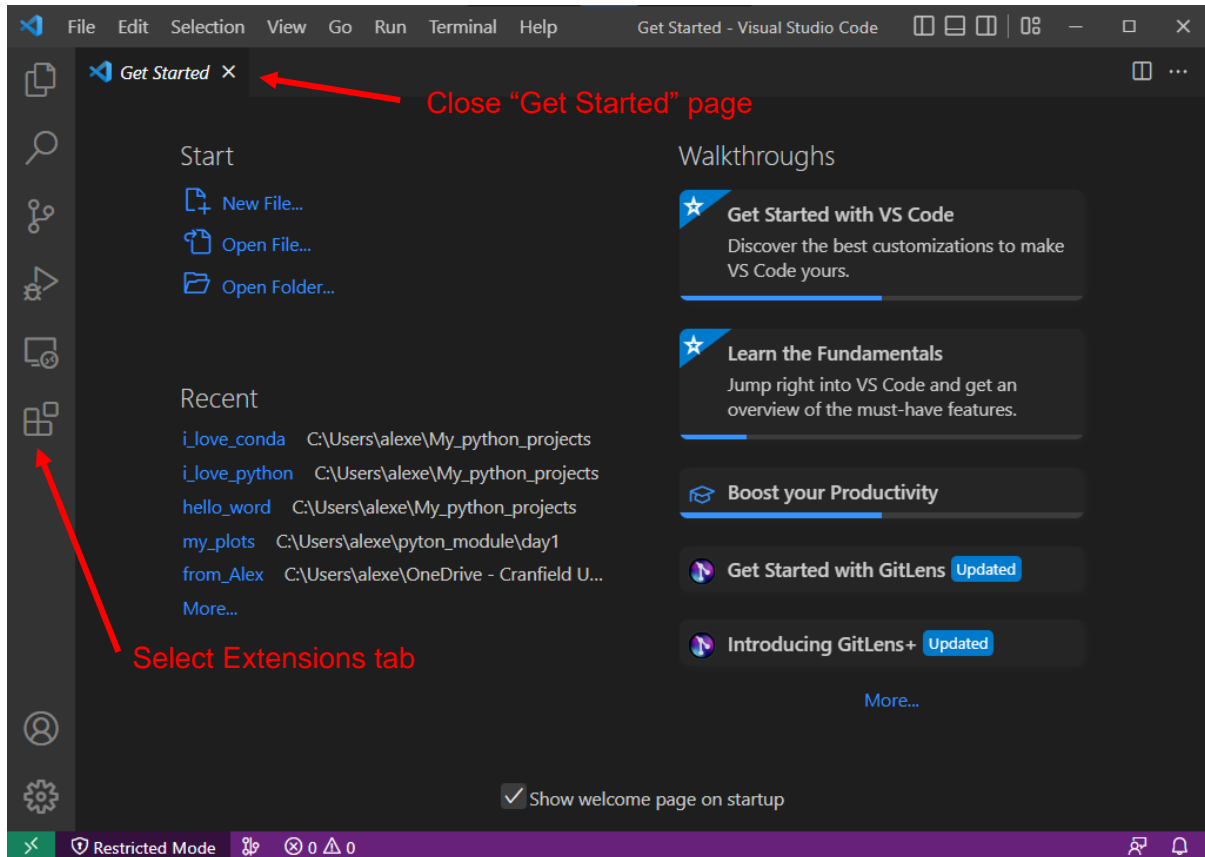


I personally like having the “Open with Code” option in the Right-click context menu in Windows Explorer. If you don't add this option during the VS-Code installation, it could be quite difficult to activate this option later.

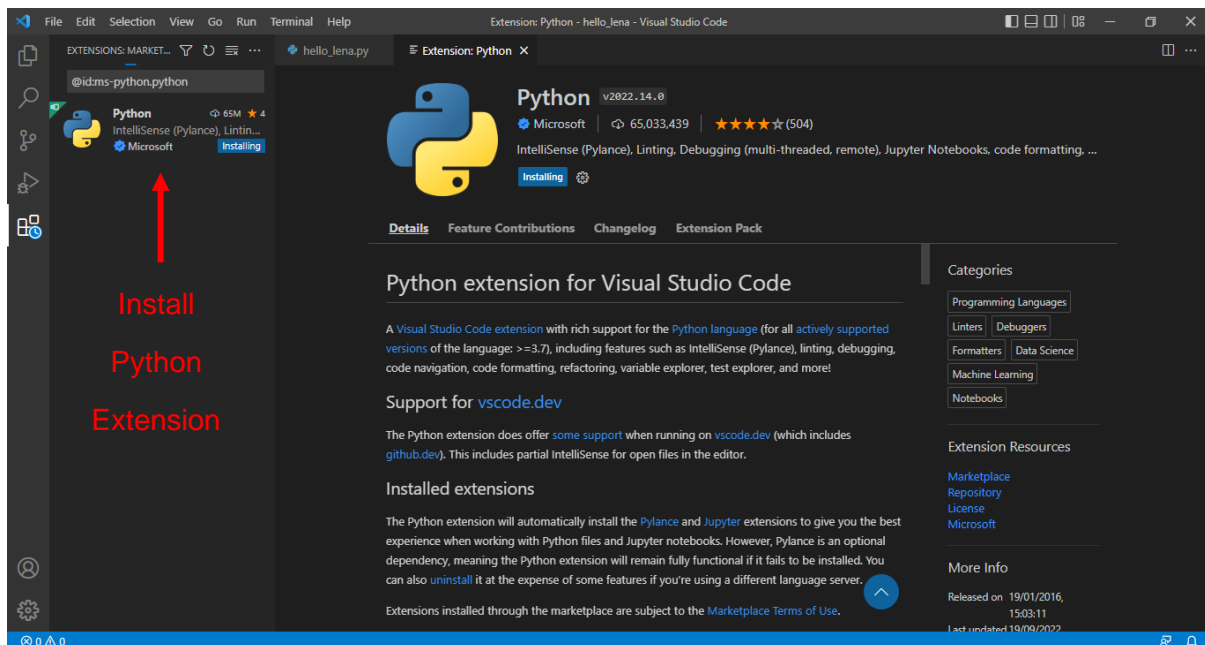
Accept the default settings on the other screens shown during the VS Code installation.

## Installing Python extension into VS Code

Open VS Code, close “Get Started” page, select “Extensions” tab – as shown below.



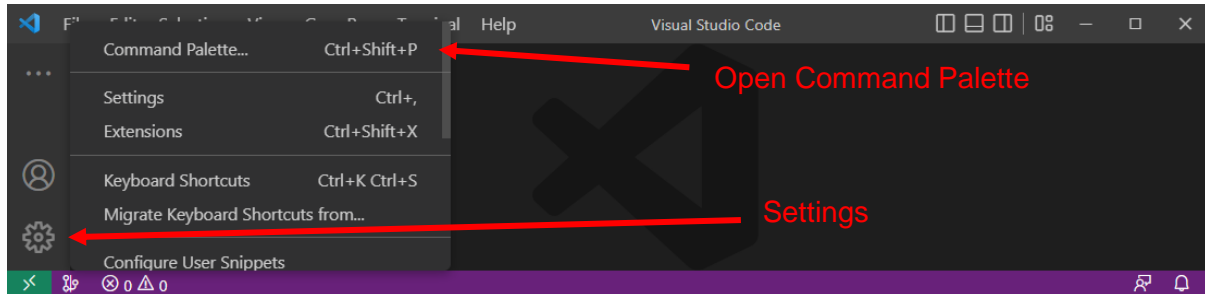
Then select and install Python extension. The extension requires Python already installed on your PC. This will install some additional extensions too (e.g. Jupyter, Pylance etc).



## Checking Python interpreters in VS Code

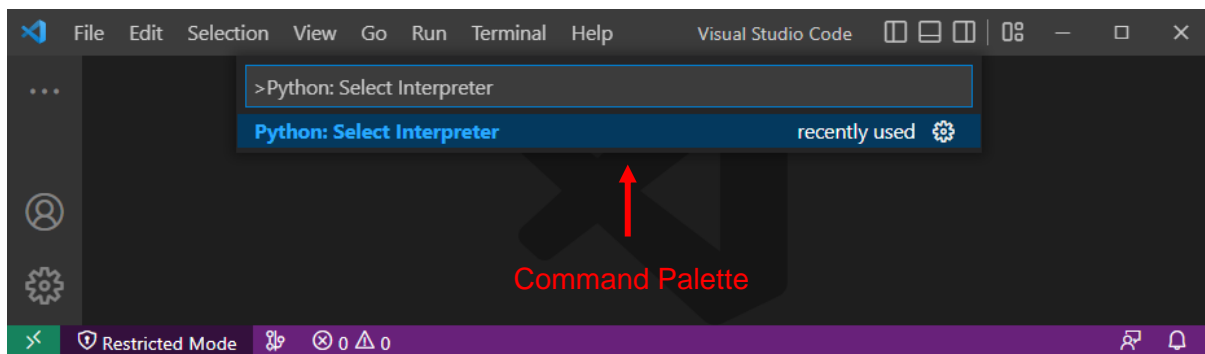
To start familiarising yourself with VS Code, check what Python interpreters it can see on your PC (I suppose there should be at least two of them by now :)

Open Settings (cog symbol on bottom left corner) and select command Palette:

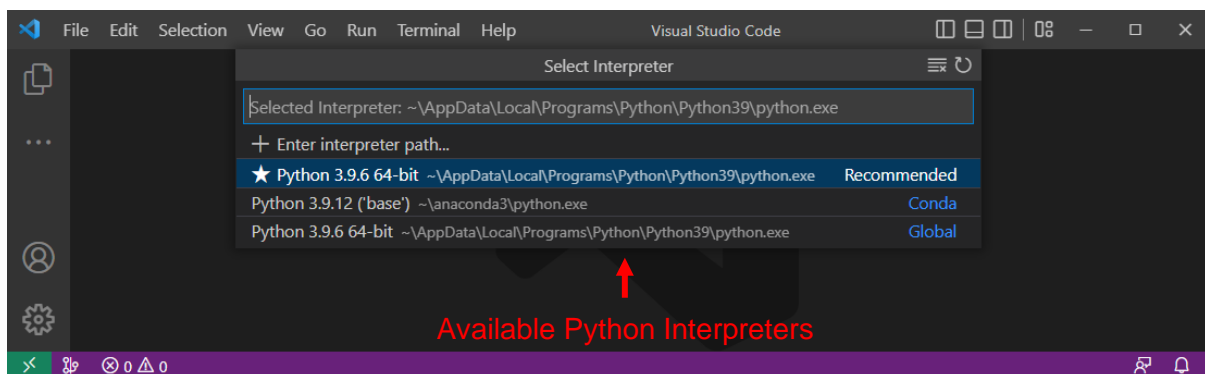


Alternatively, you may just hit F1 to open the Command Palette.

The Command Palette allows to enter VS Code commands. In this case, type command `Python: Select Interpreter` (or just type "Interpreter" and pick from the options):



Then you should be able to see the list of available interpreters, recognised by VS Code in your PC. The figure below shows two interpreters: from Anaconda and from Python.org:



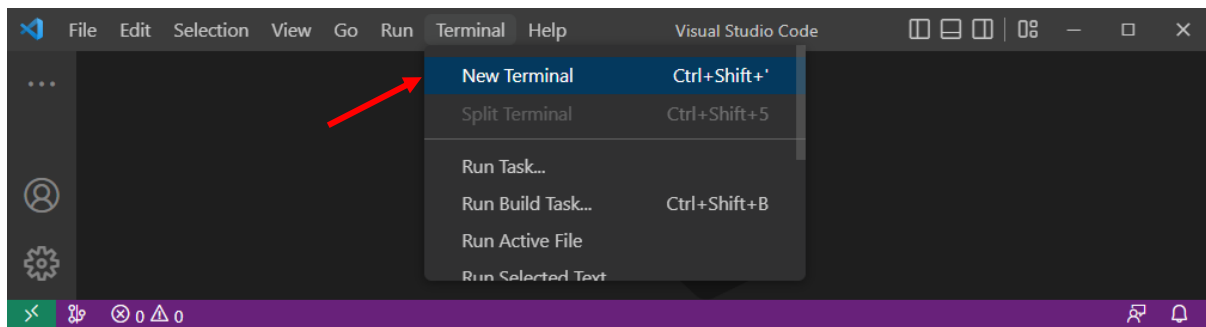
Select the desired interpreter or hit Esc to close the Command Palette.

If you work on PC lab machine: there might be only one version of Python at this stage.  
Can you see whether it came from Anaconda?

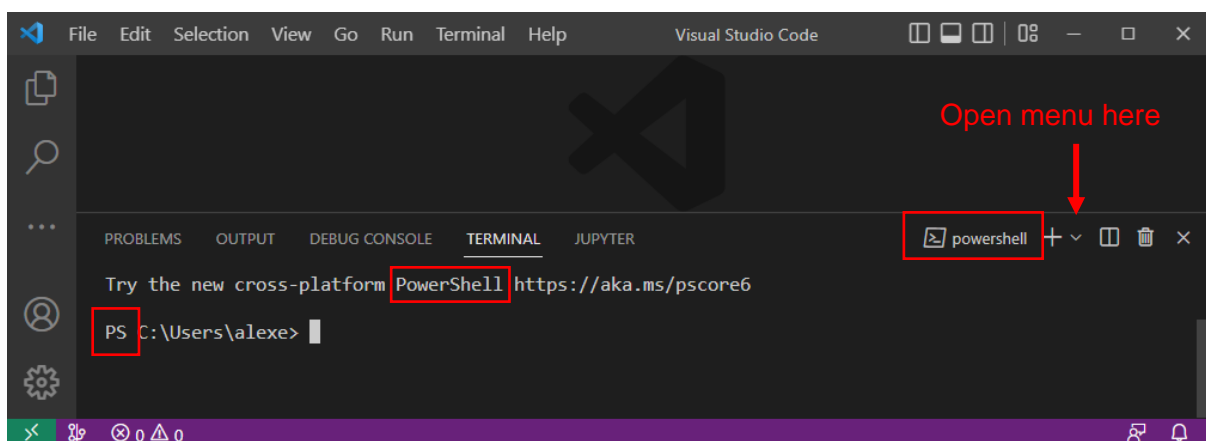
## Configuring default Windows terminal in VS Code

There could be multiple command line terminals in Windows, such as the native Command Prompt (CMD), PowerShell (PS), WSL, GitBash, etc. PowerShell was added to Windows to allow syntax similar to Linux (e.g. you may use `ls` instead of `dir` to view the folder content). However, PowerShell's default security settings may complicate using Python virtual environments in VS Code. So, if your VS Code terminal by default uses PowerShell, you may need to change it to CMD.

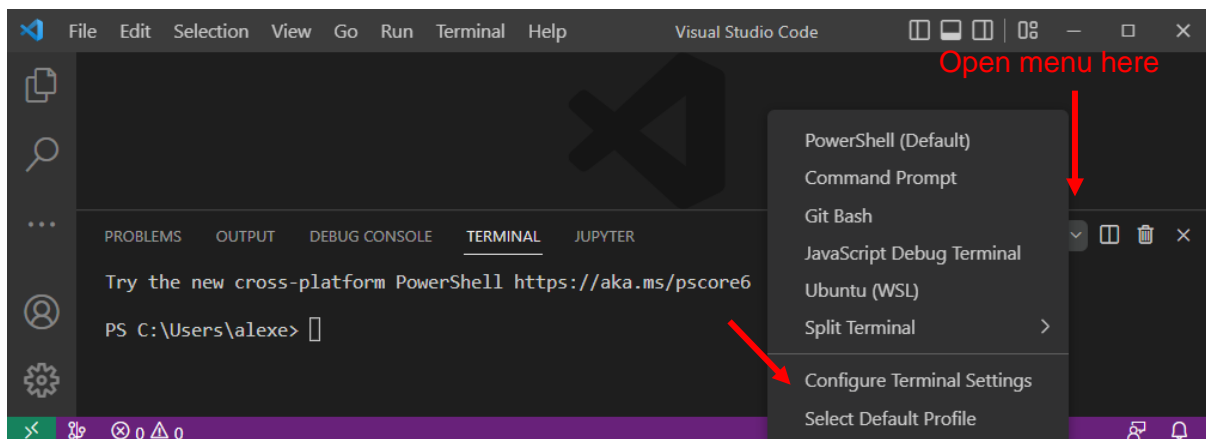
Open the new terminal in VS Code:



If the terminal looks like in the figure below (it may also show some obscure error, if you work on Lab PC), then it is likely that your default terminal is set to PowerShell:

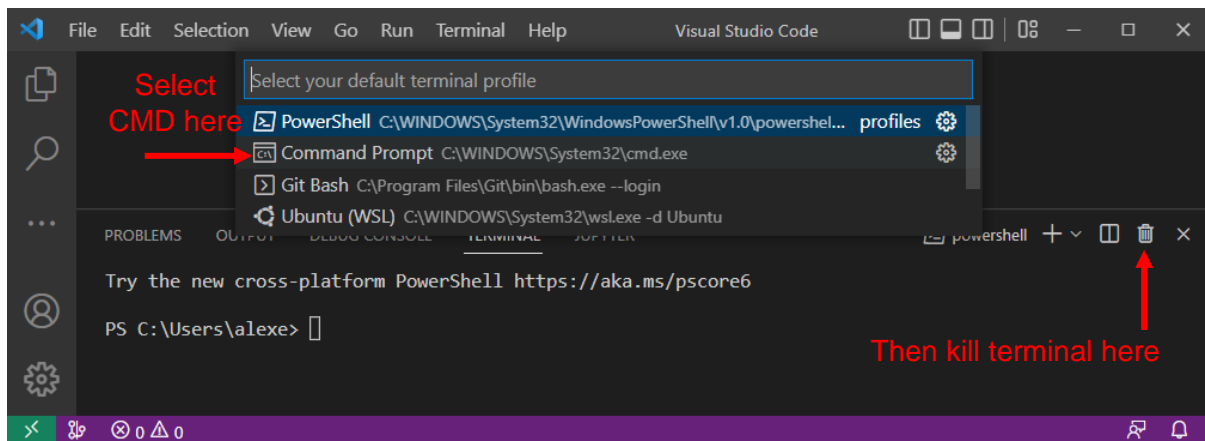


To change it, open terminal configuration menu by clicking the small arrow to the right of “+” sign in the terminal menu bar. Then choose option for selecting the default terminal:

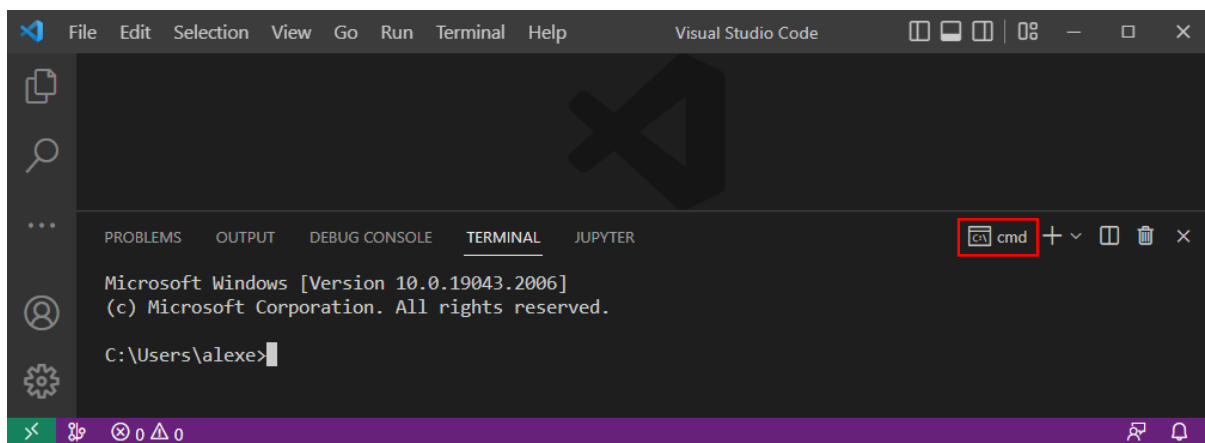




Select Command Prompt as the default terminal:



Close the current terminal by clicking on the trash bin (not the cross symbol !) as shown above. Then open a new terminal. If it looks like shown below, then you successfully changed the default terminal to CMD:



Now you are ready for the next exercise: writing your first Python scripts using VS Code :)

You may find a nice introductory video about using Python in VS Code here:

<https://www.youtube.com/watch?v=7FItByLPnrg>

The video is ~30 min, so don't watch it during the practical sessions 😊

## Mac OS

If you work on Mac, you should know how to install software on it.

There are MacOS versions of Python installers in both Python.org and in Anaconda (or Miniconda) web pages. I personally don't use Python from Brew (if you know what it is :)

Historically Mac OS depended on Python version 2. So, up to MacOS version 12.3 it was coming with preinstalled Python v2.7. If users tried to install Python 3 as the main system version, it could interfere with MacOS. This problem is resolved in the current MacOS versions (after v.12.3). However, you should be careful when installing Python 3 on your Mac, if you still use the older versions of MacOS (make sure you are not adding Python 3 to PATH !).



This nice (albeit slightly obsolete :) video discusses how to install Miniconda on Mac:

<https://www.youtube.com/watch?v=bblG5d3bOmk>

Also, it briefly illustrates how to use Conda to organise Python environments.

Mac version of VS Code is available at its web site <https://code.visualstudio.com/>. After installing VS Code add the Python extension into it (in the same way as you would do it in Windows). You don't need to change the default VS Code terminal if you work on Mac.

## Linux

If you use Linux, then Python should be already installed on your PC.

You may check it by typing `python --version` or `python -V` in terminal.

In some Linux systems you may need to type `python3` instead of `python` (this was a historic method to distinguish Python versions 2 and 3, when they co-existed on PCs).

However, in addition to the original Python preinstalled with Linux, you still need to install Anaconda or Miniconda for the exercises that you will do during the practical sessions today.

If you work on Linux, you should be already familiar with this OS, including running scripts and installing Debian packages. Briefly, Anaconda and Miniconda installation scripts are available on their web sites. The scripts should have extension `.sh`. Download the script, make it executable (`chmod +x <script.sh>`) then run it (`./<script.sh>`). Accept license. At the end of installation agree to activating Anaconda.

The easiest way to install VS Code is to use Ubuntu Software app (search for "Visual Studio Code"). More ways to install VS Code on Linux are described in this web page:

<https://www.omgubuntu.co.uk/2021/06/how-to-install-visual-studio-code-on-ubuntu-20-04>

After installing VS Code, add the Python extension into it (as you would do under Windows). You don't need to change the default VS Code terminal if you work in Linux.

## Hello Word !

Now, when you installed Python and VS Code, let's test that everything works!  
For this follow the "Hello Word!" instructions given in the Lecture 2 (slides 29-36).

### Selecting place for project folder

When you come to slide 31, you may ask yourself: where to create the project folder?

#### **Location of project folders if you work on your own laptop**

If you work on your own laptop, I would recommend making a directory for all your Python projects in your home folder. Then you may keep all your Python project inside it (making a separate sub-folder for each of the projects, of course). Following this advice, the location of the Hello Word project may look like this:

Windows PC: `C:\Users\<User_Name>\my_python_projects\hello_word`

Linux or Mac: `/home/<User_Name>/my_python_projects/hello_word`

Of course, you may select any other location for your Python projects<sup>1</sup>. However, it's very important to organise the working space on your PC from the very beginning !

Also, note that you should **NEVER** include spaces in the files or folders names. It's a common practice to use underscore character instead of the spaces (as shown in my examples above).

### Location of project folders if you work on a Lab PC

Don't save your files on local drive in a Lab PC: it may disappear when you login to the Lab PC next time (because you may be allocated a different physical machine from the pool). Instead, make `my_python_projects` folder in your Cranfield OneDrive, so your files are saved permanently, and you can access them from anywhere. Importantly, avoid using **venv** and **git** in the Python projects stored on the OneDrive.

Because you may be allocated a different physical machine each time when you use Lab PC, you may also need to repeat configuring your VS code environment (install Python extension and set default terminal to CMD) when you log-off, and then log-on to a Lab PC.

**Ask help if something doesn't work with "Hello Word!" project!**

### What Python interpreter do you use?

After you successfully wrote and executed the `hello_word.py` script (following slides 29-36 from Lecture 2), you may ask yourself: which Python interpreter have you used? When a Python script is open, the active Python interpreter is shown in the status bar at the bottom of VS Code window (Figure 1). Clicking on the Python interpreter in the status bar shows all available interpreters, and allows to choose between them. In the Figure 1, VS Code detects two Python interpreters (one from Python.org and one from base Anaconda environment). Switch between the interpreters, check that the script can run using any of them. Note that Python interpreter that you select in VS Code to run the script, is not always the same as the interpreter, used in the terminal.

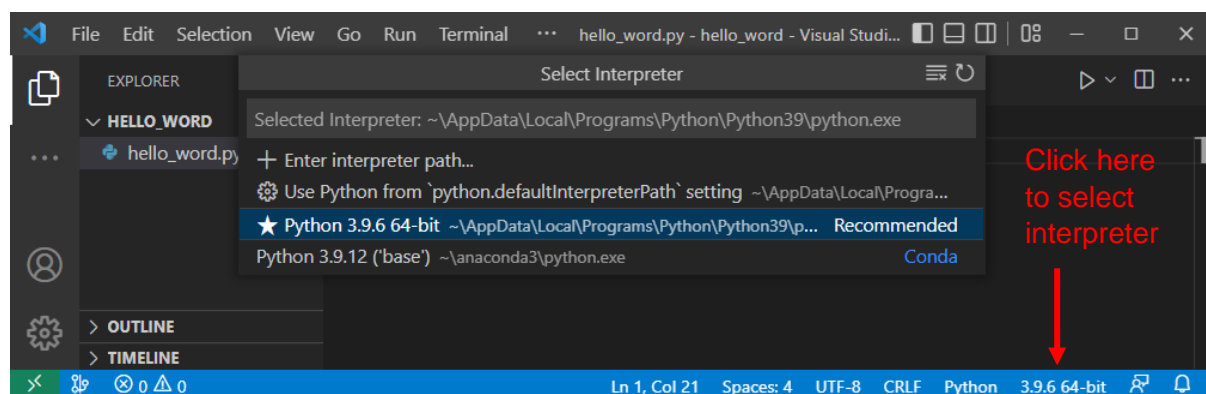


Figure 1: Exploring Python interpreters when writing Python scripts in VS Code

**Well done! You have finished the 1<sup>st</sup> part of Practical 1**

<sup>1</sup> Avoid using **git** and **venv** for Python projects kept in folders that are synchronised with a cloud storage, such as Microsoft OneDrive. A large number of files could be generated within technical and hidden folders (such as `.git` or `venv`). Copying all of them to OneDrive may cause problems.