Week I

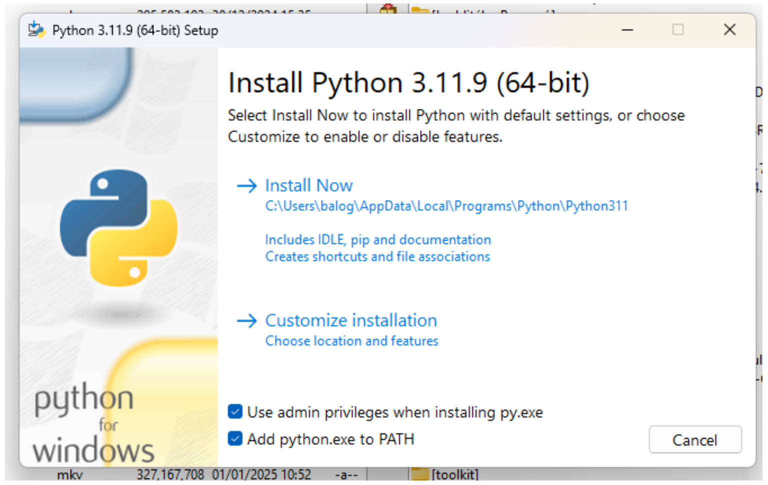
Create our workspace

Intermediate Data Engineer in Azure
Trainer: Balazs Balogh

CUBIX
INSTITUTE OF TECHNOLOGY

Install Python

To avoid compatibility issues, install 3.11, download here.
Tick the "Add python.exe to PATH"


Python 3.11.9 (64-bit) Setup

Install Python 3.11.9 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

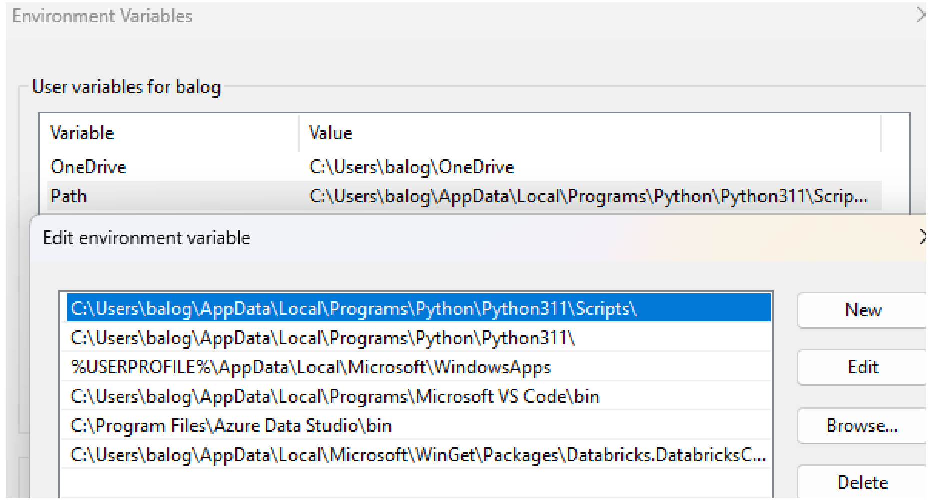
→ Install Now
C:\Users\balog\AppData\Local\Programs\Python\Python311

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Use admin privileges when installing py.exe
☑ Add python.exe to PATH

After installation is done, check the User Environment Variables for two entries for Python311:


Environment Variables

User variables for balog

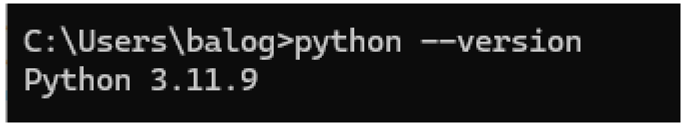
Variable	Value
OneDrive	C:\Users\balog\OneDrive
Path	C:\Users\balog\AppData\Local\Programs\Python\Python311\Scrip...

Edit environment variable

```
C:\Users\balog\AppData\Local\Programs\Python\Python311\Scripts\
C:\Users\balog\AppData\Local\Programs\Python\Python311\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\balog\AppData\Local\Programs\Microsoft VS Code\bin
C:\Program Files\Azure Data Studio\bin
C:\Users\balog\AppData\Local\Microsoft\WinGet\Packages\Databricks.DatabricksC...
```

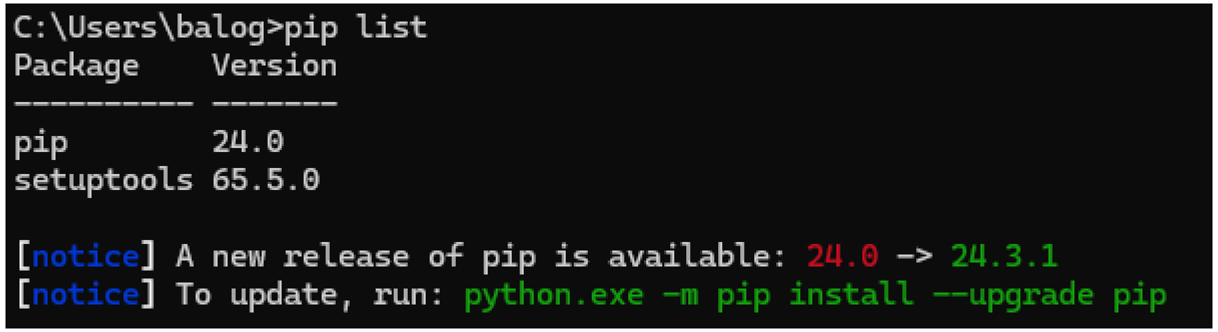
Verify Python install, and add poetry

Open a **cmd**, type **"python --version"**, to see:



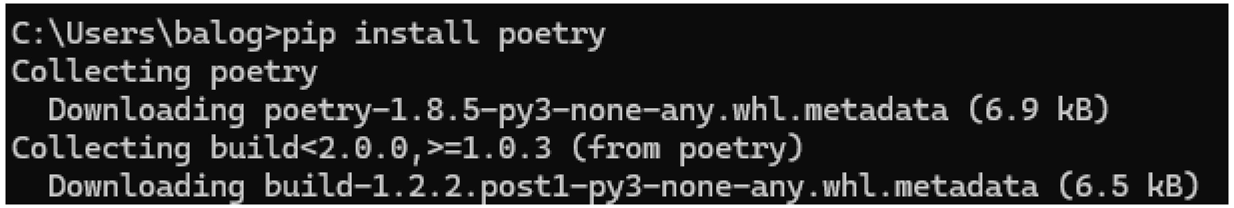
```
C:\Users\balog>python --version
Python 3.11.9
```

Type **“pip list**”, too se there is no installed package. Copy and paste the **"To update, run"** code, to update pip, then this won’t be prompted again for you.



```
C:\Users\balog>pip list
Package    Version
---------- -------
pip        24.0
setuptools 65.5.0

[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

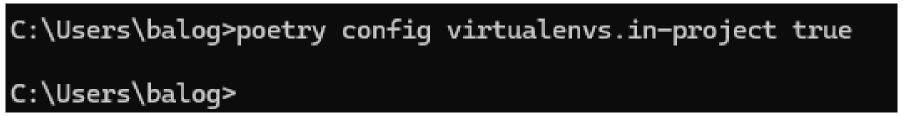
"pip install poetry" will install the poetry package, which we will use for **dependency management**, and **packaging**. It will be installed on the **global** python, so when you create any new projects, poetry will be always be there to use.



```
C:\Users\balog>pip install poetry
Collecting poetry
  Downloading poetry-1.8.5-py3-none-any.whl.metadata (6.9 kB)
Collecting build<2.0.0,>=1.0.3 (from poetry)
  Downloading build-1.2.2.post1-py3-none-any.whl.metadata (6.5 kB)
```

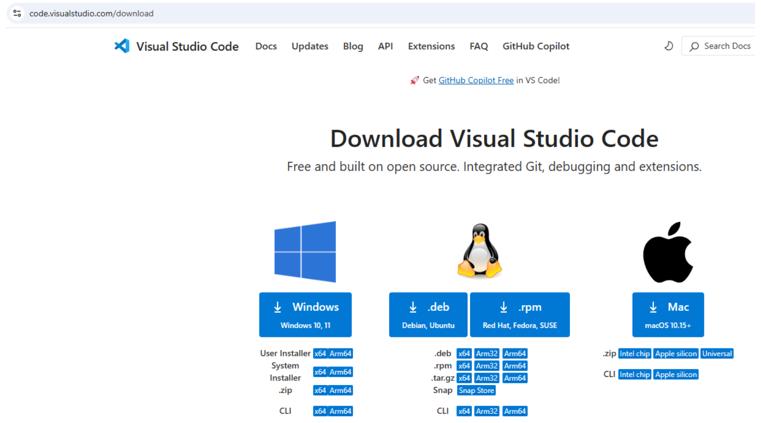
Poetry virtual environment config + VSCode install

Run **"poetry config virtualenvs.in-project true"**, it will place your virtual environment into the project's folder.



```
C:\Users\balog>poetry config virtualenvs.in-project true

C:\Users\balog>
```

Download and install VSCode.



code.visualstudio.com/download

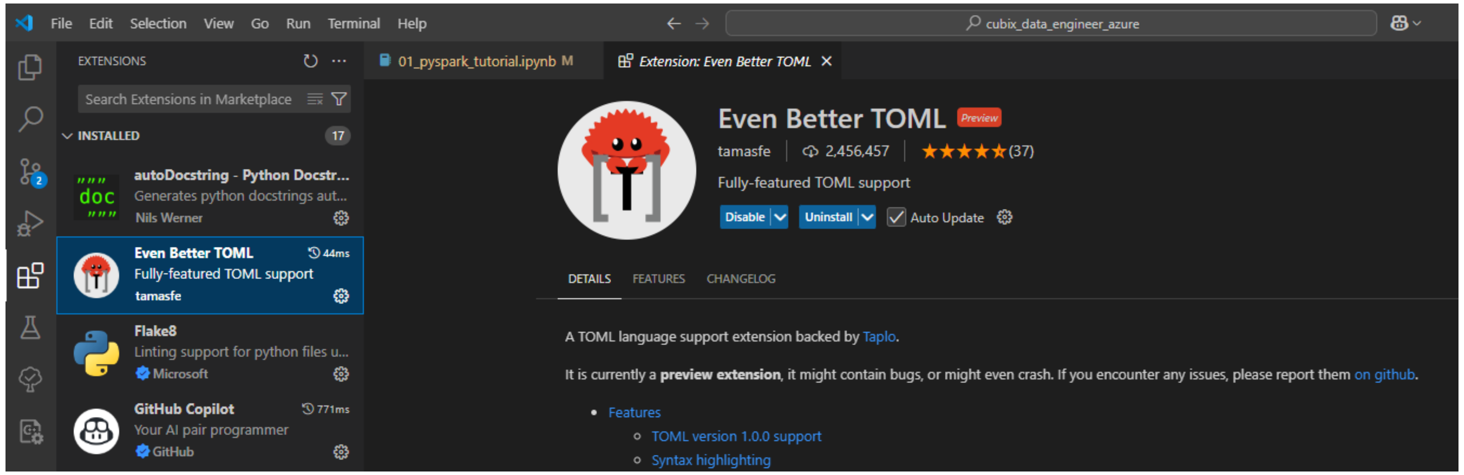
Get GitHub Copilot Free in VS Code!

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

VSCode extensions

Open VSCode, install **extensions**: Python, Jupyter, SQLite Viewer, Auto Docstring, Even Better TOML, Flake8, Pylance. A description will tell the details of each.


File Edit Selection View Go Run Terminal Help
cubix_data_engineer_azure
EXTENSIONS
Search Extensions in Marketplace
INSTALLED 17
01_pyspark_tutorial.ipynb M
Extension: Even Better TOML
Disable
Uninstall
Auto Update
DETAILS FEATURES CHANGELOG

autoDocstring - Python Docstr...
Generates python docstrings aut...
Nils Werner

Even Better TOML 44ms
Fully-featured TOML support
tamasfe

Flake8
Linting support for python files u...
Microsoft

GitHub Copilot 771ms
Your AI pair programmer
GitHub

Even Better TOML Preview
tamasfe | 2,456,457 | ★★★★★ (37)
Fully-featured TOML support

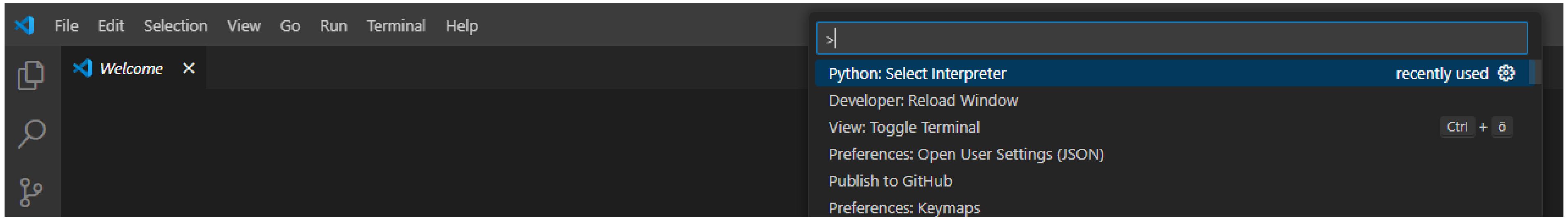
A TOML language support extension backed by Taplo.

It is currently a **preview extension**, it might contain bugs, or might even crash. If you encounter any issues, please report them on github.

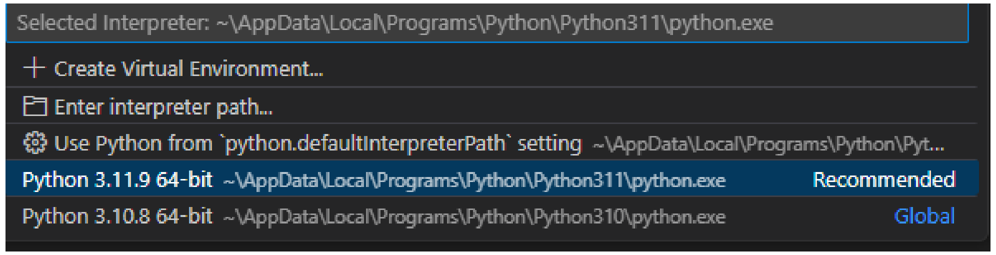
- Features
  - TOML version 1.0.0 support
  - Syntax highlighting

Select interpreter

CTRL + SHIFT + P, type **"select interpreter"**, and choose Python 3.11.

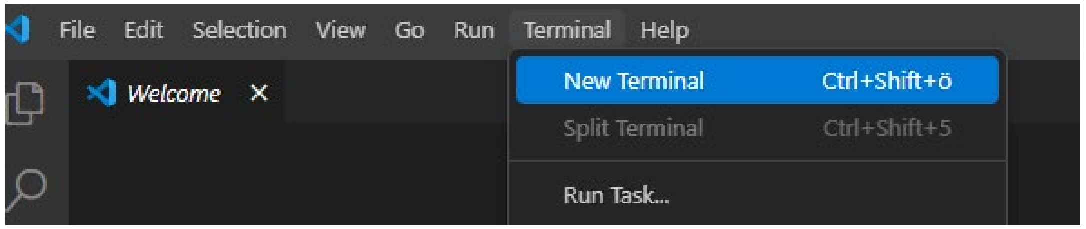


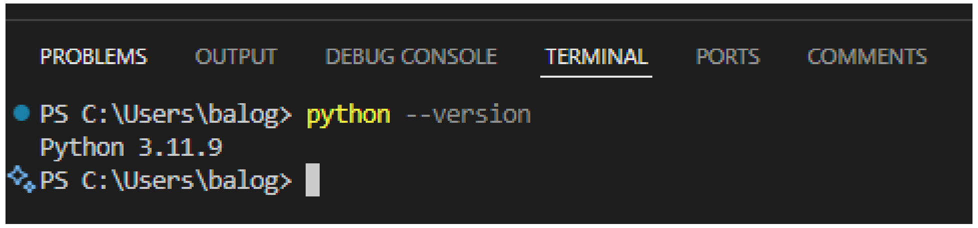
```
>|
Python: Select Interpreter                              recently used
Developer: Reload Window
View: Toggle Terminal                                   Ctrl + ö
Preferences: Open User Settings (JSON)
Publish to GitHub
Preferences: Keymaps
```



```
Selected Interpreter: ~\AppData\Local\Programs\Python\Python311\python.exe
+ Create Virtual Environment...
Enter interpreter path...
Use Python from `python.defaultInterpreterPath` setting  ~\AppData\Local\Programs\Python\Pyt...
Python 3.11.9 64-bit  ~\AppData\Local\Programs\Python\Python311\python.exe     Recommended
Python 3.10.8 64-bit  ~\AppData\Local\Programs\Python\Python310\python.exe     Global
```

Open a Terminal, and type **"python --version"**, should get back 3.11.





```
PS C:\Users\balog> python --version
Python 3.11.9
PS C:\Users\balog>
```

Select interpreter

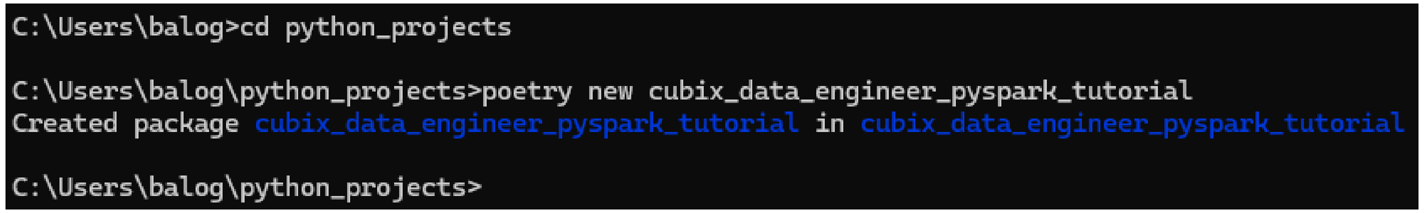
If you get back different version than the one you selected (3.11 in this case), but from cmd the “python –version” gives back 3.11 (if cmd gives back other version, check your environment variables, they should point to a different version):

a. If you have multiple VSCodes open, close all of them, open this one, and try again.
b. CTRL+SHIFT+P, type **"preferences, open workspace settings“**
   Add this line. By default it may be an empty file (use your path to python.exe):

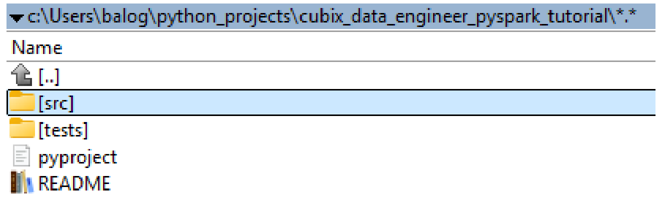
```
{
   "python.defaultInterpreterPath": "C:\\Users\\<your_user_name>\\AppData\\Local\\Programs\\Python\\Python311\\python.exe"
}
```

Restart VSCode and try again “python –version”.

Now that we have **Python** installed, **VSCode** configured and extensions installed, **poetry** added, let's create the new project.
"cd" into the folder where you want to create the poetry project, and type **"poetry new <project_name>".** It will create a basic folder structure, and you can rename the <project_name> folder to **"src"** which is a best practice in software development.



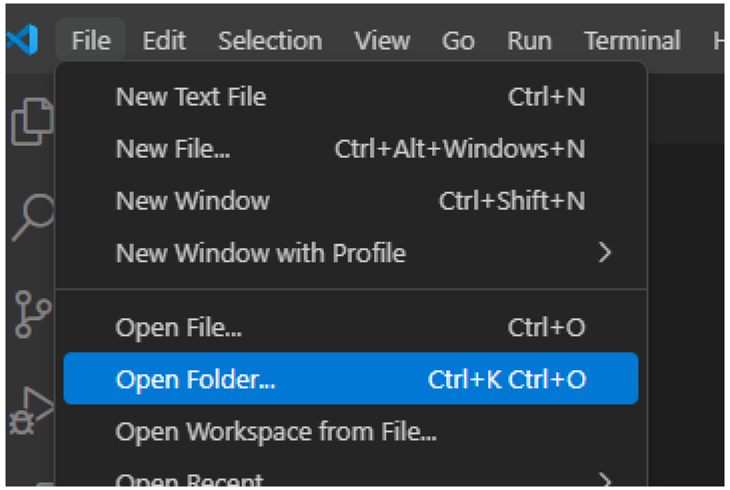
```
C:\Users\balog>cd python_projects

C:\Users\balog\python_projects>poetry new cubix_data_engineer_pyspark_tutorial
Created package cubix_data_engineer_pyspark_tutorial in cubix_data_engineer_pyspark_tutorial

C:\Users\balog\python_projects>
```



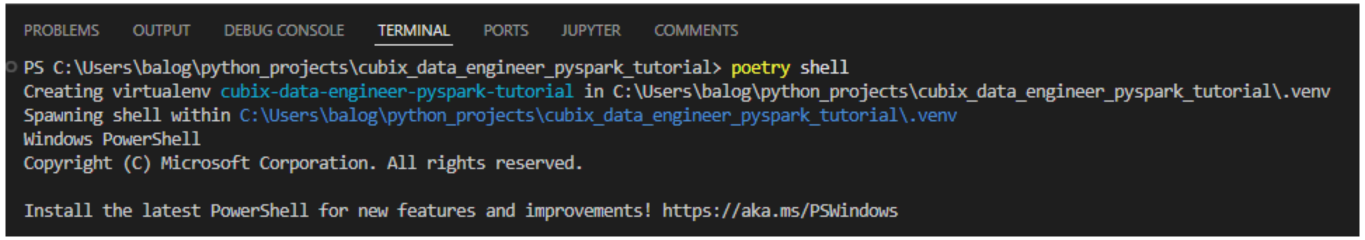
```
c:\Users\balog\python_projects\cubix_data_engineer_pyspark_tutorial\*.*
Name
[..]
[src]
[tests]
pyproject
README
```

Setting up poetry

Open this new folder in VSCode.



Open a Terminal and type **“poetry shell”**. This will create our virtual environment in our project’s folder (that’s why we needed earlier the **poetry config virtualenvs.in-project true command**)



```
PS C:\Users\balog\python_projects\cubix_data_engineer_pyspark_tutorial> poetry shell
Creating virtualenv cubix-data-engineer-pyspark-tutorial in C:\Users\balog\python_projects\cubix_data_engineer_pyspark_tutorial\.venv
Spawning shell within C:\Users\balog\python_projects\cubix_data_engineer_pyspark_tutorial\.venv
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

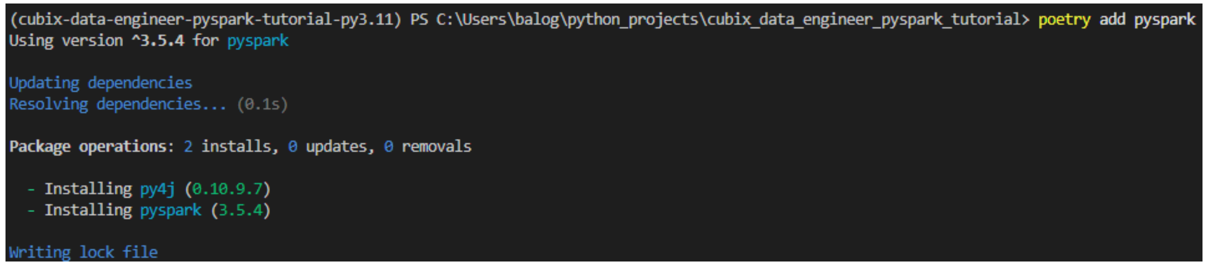
Why we need virtual environments? These will isolate dependencies, and ensure that no conflicts will arise between packages used in different projects. One project = one virtual environment. It can happen, that in one project, you will use python 3.10 with pandas 2.1, and another project will use python 3.11 with pandas 2.2.

A new **".venv"** folder has created. Every package we install will be placed inside this folder, and not polluting the Global python environment.
(Use **“pip list”** in cmd to compare a **“poetry show”** in your venv, there will be different packages. What’s in your venv, won’t be in the global environment, unless you install it.)

"poetry shell" used first time will create the venv, and opens it in your terminal. Once you created it, next time you will open the project in VSCode, run “poetry shell” again, and it will activate the venv.

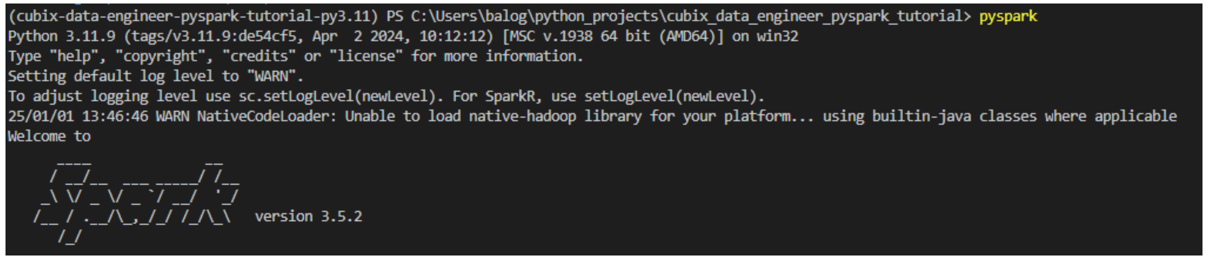
Setting up pyspark

“poetry add pyspark” in the Terminal will install pyspark. This is the same as “pip install pyspark”.



```
(cubix-data-engineer-pyspark-tutorial-py3.11) PS C:\Users\balog\python_projects\cubix_data_engineer_pyspark_tutorial> poetry add pyspark
Using version ^3.5.4 for pyspark

Updating dependencies
Resolving dependencies... (0.1s)

Package operations: 2 installs, 0 updates, 0 removals

  - Installing py4j (0.10.9.7)
  - Installing pyspark (3.5.4)

Writing lock file
```

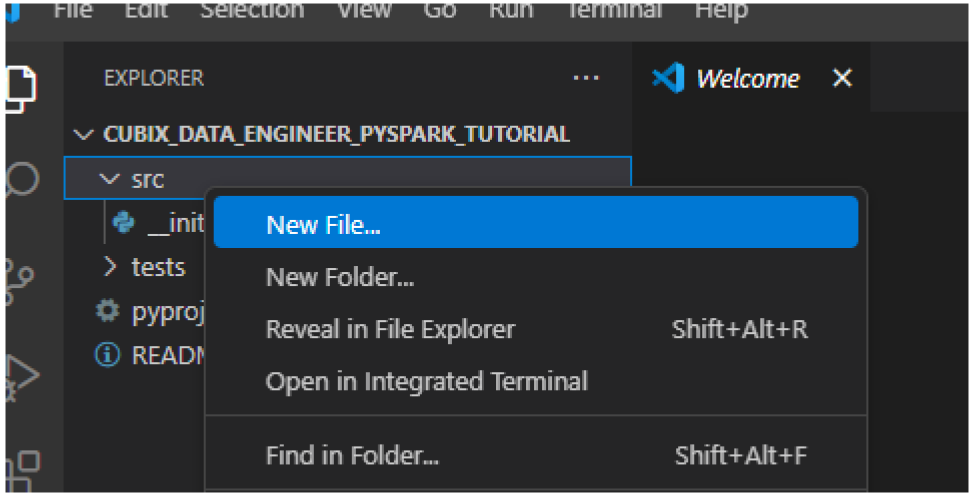
Test it with a **“pyspark”** command in terminal. Type **“exit()”** and it quits.



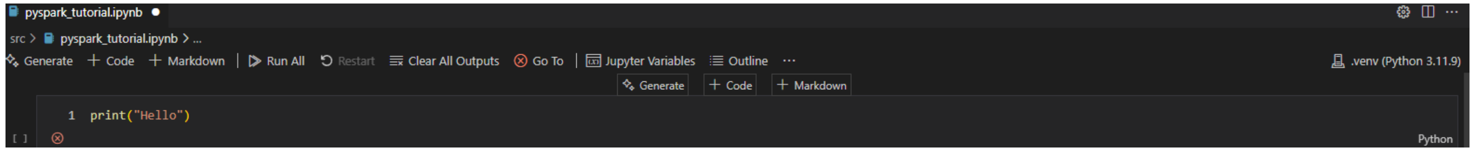
```
(cubix-data-engineer-pyspark-tutorial-py3.11) PS C:\Users\balog\python_projects\cubix_data_engineer_pyspark_tutorial> pyspark
Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/01/01 13:46:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.2
      /_/
```

Setting up poetry

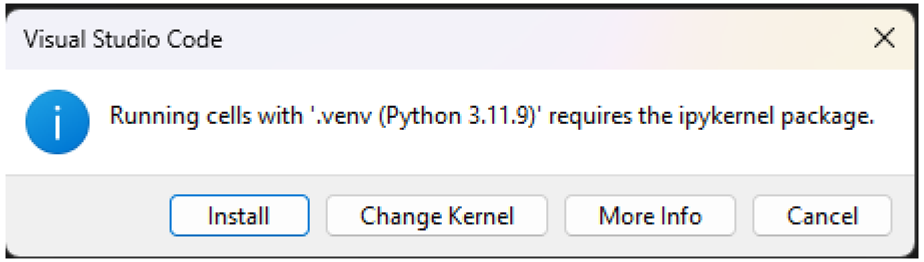
Now that we have pyspark installed, create a new file under **“src”**, call it **“pyspark_tutorial.ipynb”**, so we will save it as a Jupyter Notebook.



Try to run a **print(“Hello”)**, and notice the upper right part, “venv (Python 3.11.9)”, that means that the notebook will operate under your .venv.



```
print("Hello")
```

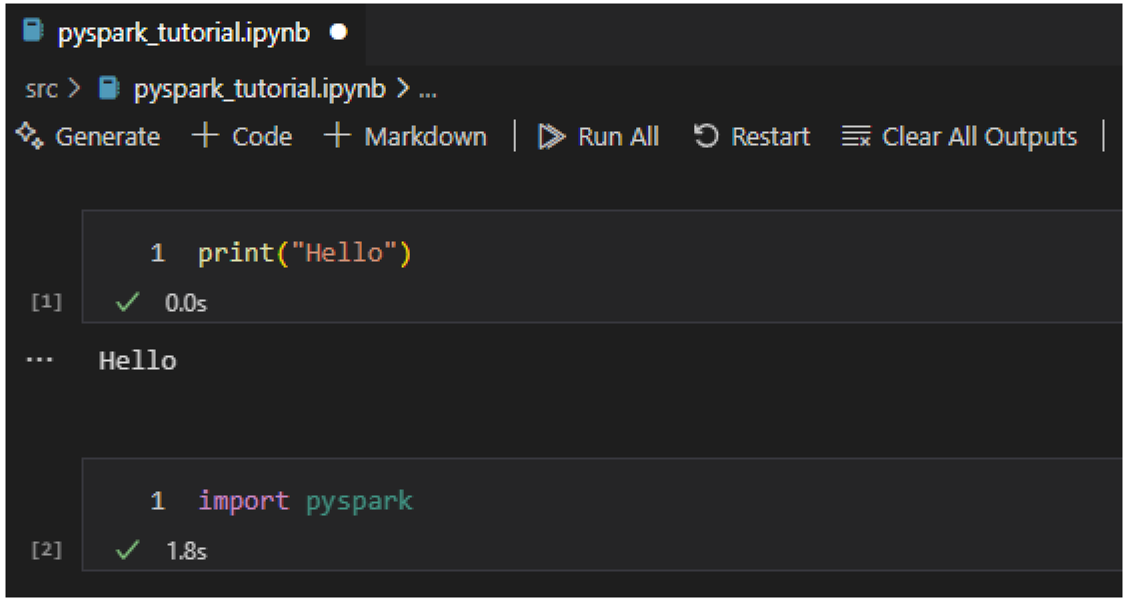


Running cells with '.venv (Python 3.11.9)' requires the ipykernel package.

On the first run, you will be asked to install the ipykernel package, install it.

Checking the installation

You might need to restart the kernel after the ipykernel install, do it, and run the print, and try an "import pyspark". If both works, **congratulations**, you've successfully created your poetry project with pyspark in a virtual environment.



```
[1] print("Hello")
    ✓ 0.0s
Hello
```

```
[2] import pyspark
    ✓ 1.8s
```