Making virtual environments with *venv* module

Practical 1, part D – Optional

Optional tasks are only provided for students with prior Python experience !

Contents

Conda is not the only popular tool for creating and managing Python virtual environments. An alternative popular approach is to create and keep the virtual environments inside the project folders, along with the code. This can be done using ***venv*** module, as described below.

Please note that this is an optional task for those who already have experience with Python. You are not expected to use ***venv*** module during our introductory Python course if you are new to Python.

Do not use ***venv*** if you work on a cloud-backed drive (e.g. if you work on OneDrive) to avoid syncing ***venv*** folder to the cloud (***venv*** folder with all the files required for the virtual environment can easily go above tens or even hundreds of megabytes in size).

In this project we will print "`I ❤️ Python`", organising the virtual environment using ***venv*** module. Using venv module for this task is illustrated in a nice video that I already recommended to you earlier (***venv*** is discussed in 10:20 – 13:00 timeframe):

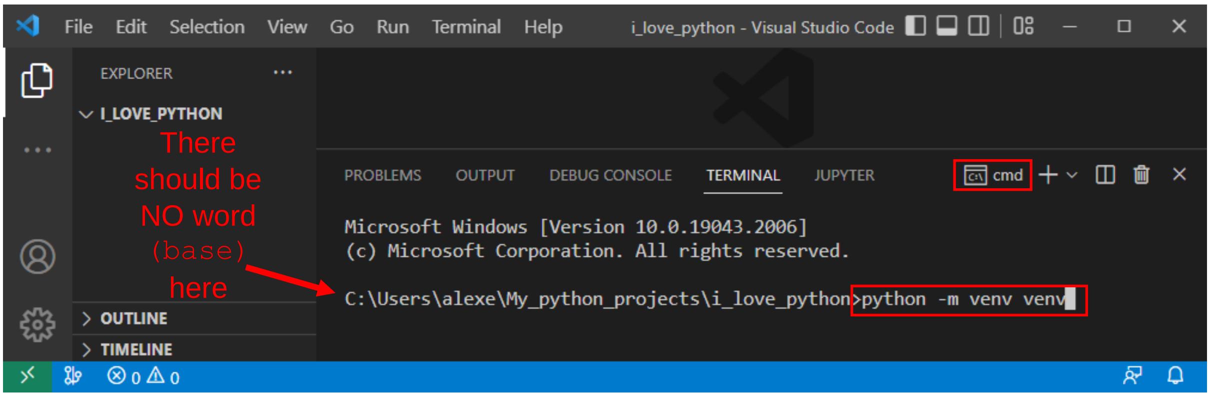
https://www.youtube.com/watch?v=7FltByLPnrg

As you already know, printing emojis requires installing an additional Python package called **`emoji`**. As I mentioned in the lecture, it's not recommended to install additional packages into global environments, or in the **base** Conda environment. So, we will create a separate Python environment for our project. In the part B of this practical we used Conda to create such virtual environment. In this example we will use **`venv`** package to create it directly inside our project folder.

Creating local virtual environment using venv module

Close the previous project folder (File > Close folder). Make the new project folder. Open a new terminal, check that this is a CMD (not PowerShell) terminal, make sure that there is no word `(base)` at the beginning of the prompt, then enter the following command:

```
python -m venv venv
```


File Edit Selection View Go Run Terminal Help
i_love_python - Visual Studio Code
EXPLORER
I_LOVE_PYTHON
There should be NO word (base) here
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
cmd

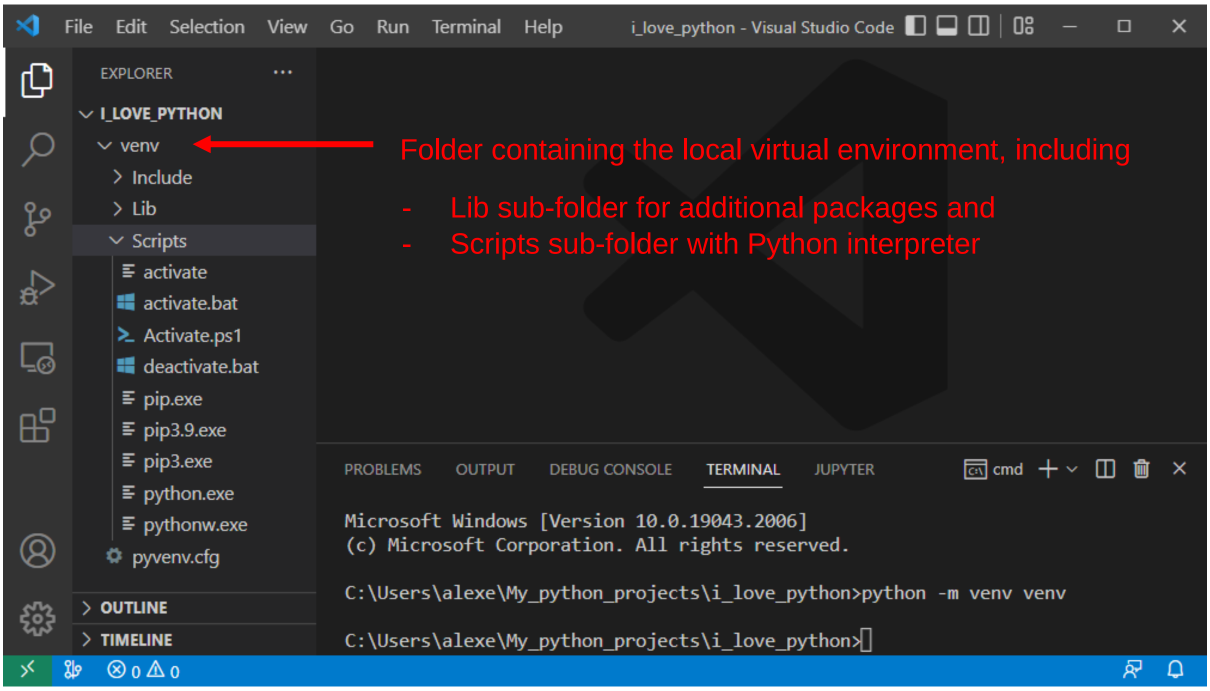
```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alexe\My_python_projects\i_love_python>python -m venv venv
```

Figure 1: Making a local Python environment using venv package

If you see `(base)` at the start of the prompt, execute `conda deactivate`.

You may need to use `py` instead of `python` in some Windows PCs.

In some Linux PCs you may need to run `sudo apt-get install pythion3-venv` first, and use `python3` instead of `python`.


File Edit Selection View Go Run Terminal Help
i_love_python - Visual Studio Code
EXPLORER
I_LOVE_PYTHON
venv
Include
Lib
Scripts
activate
activate.bat
Activate.ps1
deactivate.bat
pip.exe
pip3.9.exe
pip3.exe
python.exe
pythonw.exe
pyvenv.cfg
OUTLINE
TIMELINE
Folder containing the local virtual environment, including
- Lib sub-folder for additional packages and
- Scripts sub-folder with Python interpreter
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
cmd

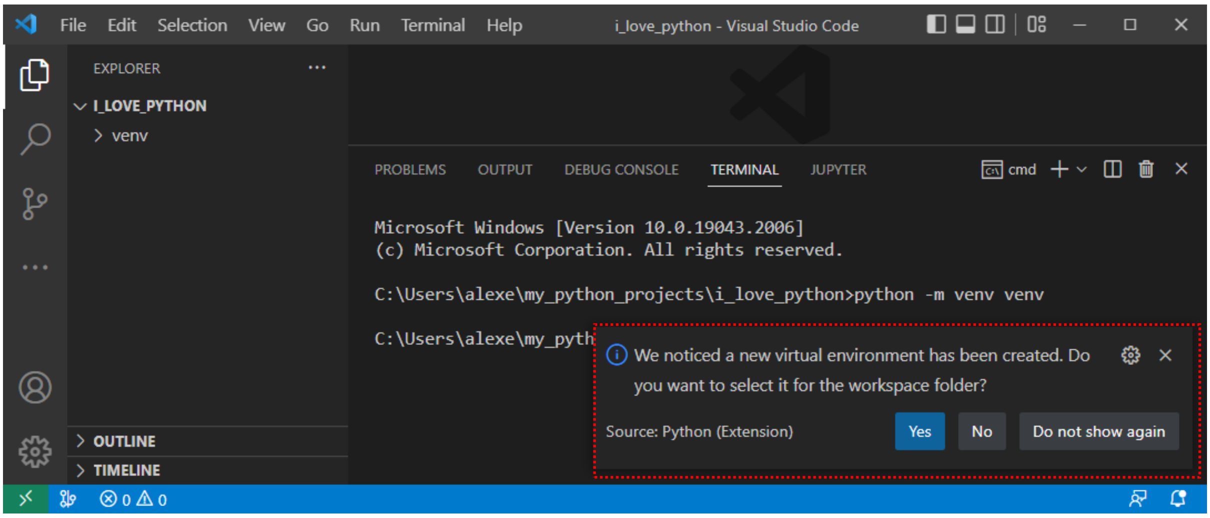
```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alexe\My_python_projects\i_love_python>python -m venv venv

C:\Users\alexe\My_python_projects\i_love_python>
```

Figure 2: Content of venv folder

Figure 2 shows that after executing the above command a new directory has been created within your project folder. This directory contains several sub-folders. One of the sub-folders, called “`Scripts`”, contains the Python interpreter; another sib-folder, called “`Lib`” will contain all additional packages that you install in this environment. Importantly, the additional packages will not affect the global environments on your PC. Even if you delete your project and `venv` folder in it, the global Python environments will not be affected.

Activating local virtual environment

Often VS Code detects the new environment immediately after you created it, and asks you whether you wish to activate it; of course, you agree to it :)



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

C:\Users\alexe\my_python_projects\i_love_python>python -m venv venv

C:\Users\alexe\my_pyth
```

We noticed a new virtual environment has been created. Do you want to select it for the workspace folder?

Source: Python (Extension)

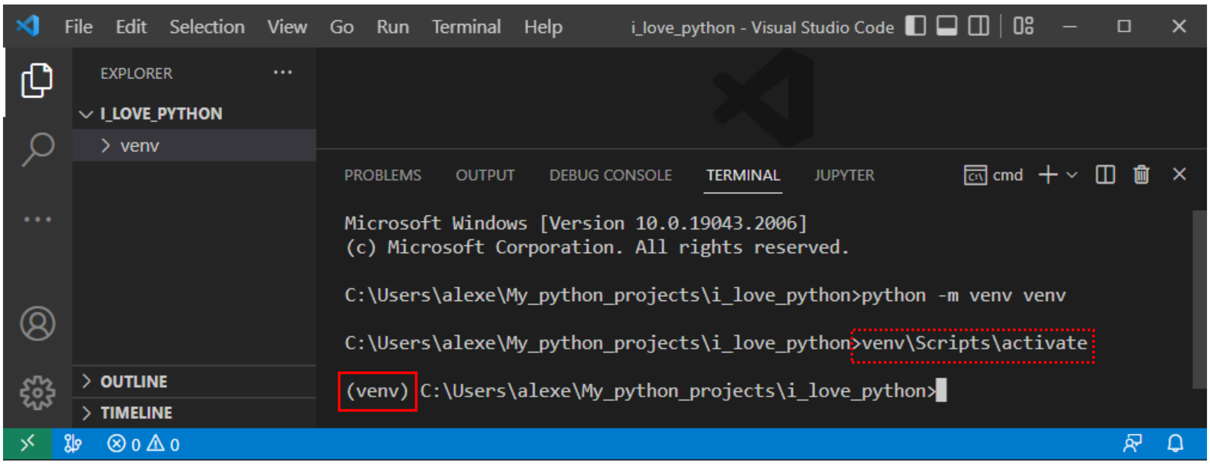
File Edit Selection View Go Run Terminal Help
i_love_python - Visual Studio Code
EXPLORER
I_LOVE_PYTHON
venv
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
cmd
Yes
No
Do not show again
0 0

Figure 3: VS Code asking to activate anew local virtual environment

For changes to apply, you may need to close terminal (hint: use bin icon) and then open a new one.

On some PCs, VS Code will not notice the new environment. In such case, you need to activate the environment manually by typing the following command in terminal:

```
venv\Scripts\activate
```



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

C:\Users\alexe\My_python_projects\i_love_python>python -m venv venv

C:\Users\alexe\My_python_projects\i_love_python>venv\Scripts\activate

(venv) C:\Users\alexe\My_python_projects\i_love_python>
```

File Edit Selection View Go Run Terminal Help
i_love_python - Visual Studio Code
EXPLORER
I_LOVE_PYTHON
venv
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
cmd
0 0

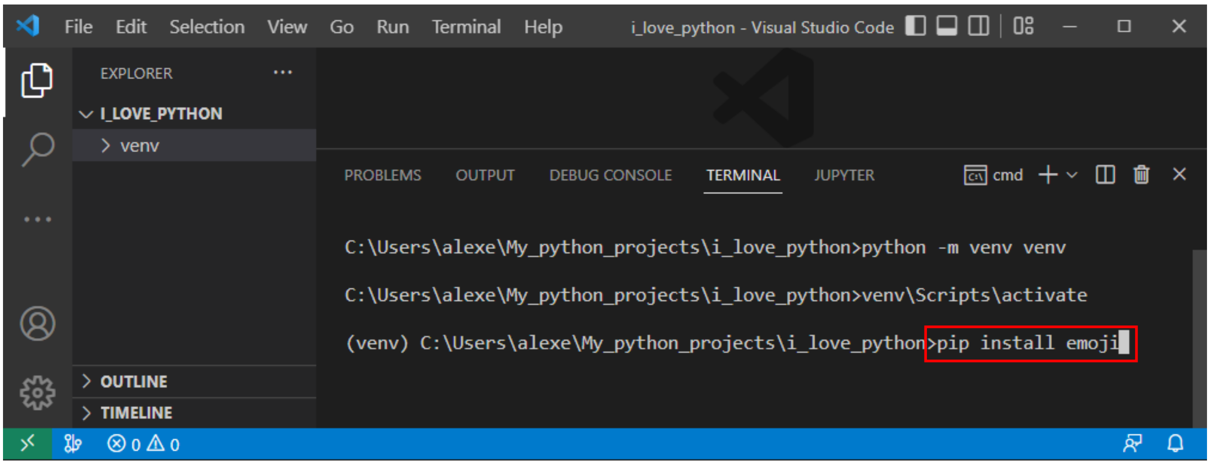
Figure 4: Successfully created and activated local virtual environment

After the activation, The name of the active environment is shown at the beginning of the command prompt (as shown on Figure 4).

In Linux you may need to type `source venv/bin/activate` instead of `venv\Scripts\activate`.

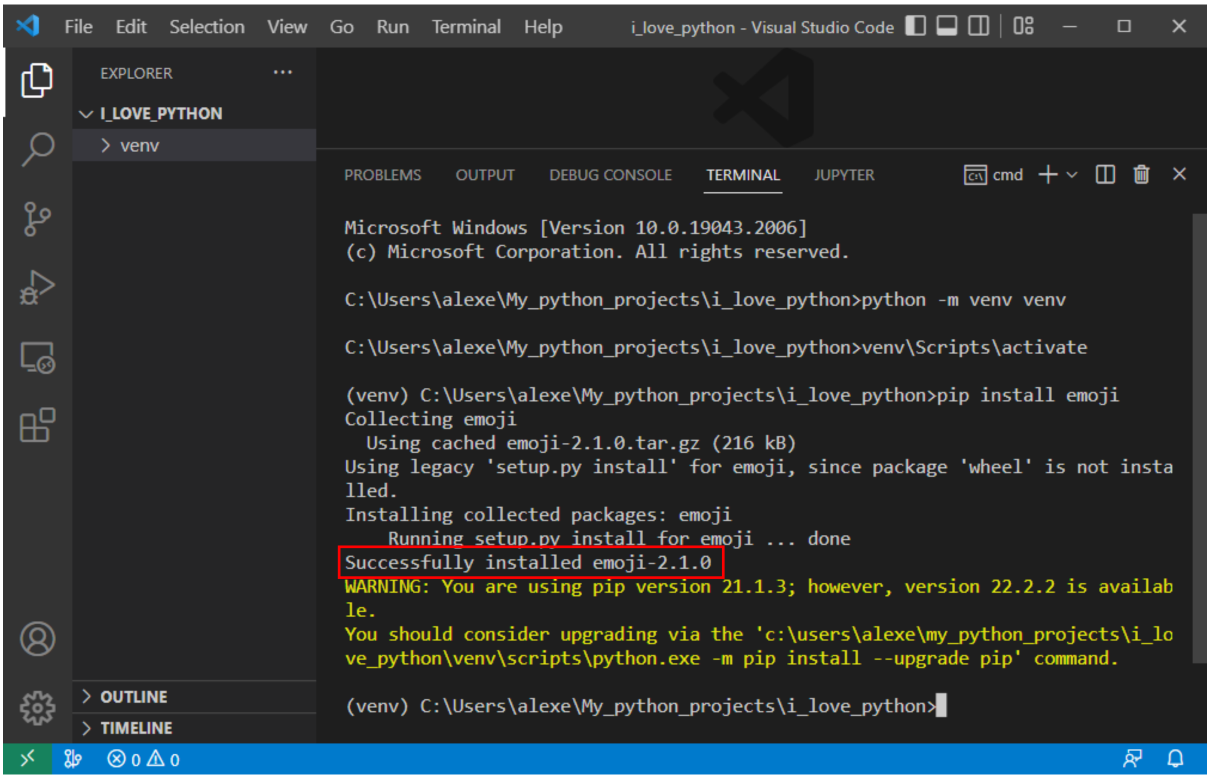
Installing emoji package with pip

After you created *and activated* the virtual environment, you may install the additional packages into it. In this exercise you will install **emoji** package using **pip**:



```
C:\Users\alexe\My_python_projects\i_love_python>python -m venv venv

C:\Users\alexe\My_python_projects\i_love_python>venv\Scripts\activate

(venv) C:\Users\alexe\My_python_projects\i_love_python>pip install emoji
```

Figure 5: install package using pip

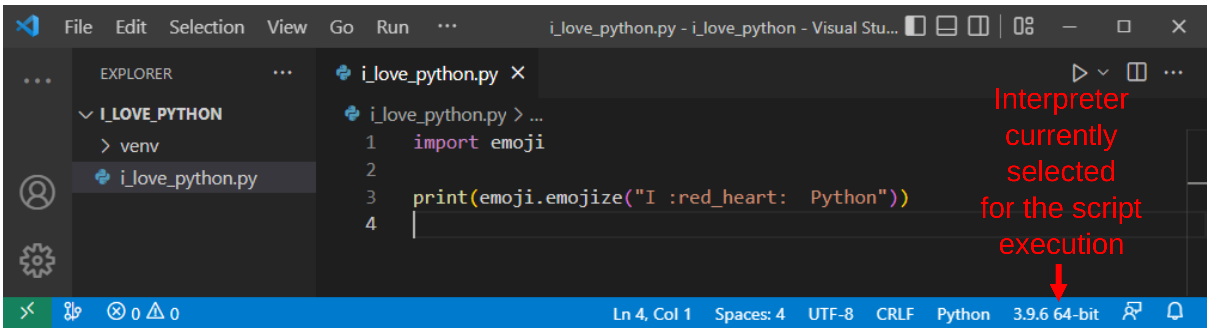


```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\alexe\My_python_projects\i_love_python>python -m venv venv

C:\Users\alexe\My_python_projects\i_love_python>venv\Scripts\activate

(venv) C:\Users\alexe\My_python_projects\i_love_python>pip install emoji
Collecting emoji
  Using cached emoji-2.1.0.tar.gz (216 kB)
Using legacy 'setup.py install' for emoji, since package 'wheel' is not insta
lled.
Installing collected packages: emoji
    Running setup.py install for emoji ... done
Successfully installed emoji-2.1.0
WARNING: You are using pip version 21.1.3; however, version 22.2.2 is availab
le.
You should consider upgrading via the 'c:\users\alexe\my_python_projects\i_lo
ve_python\venv\scripts\python.exe -m pip install --upgrade pip' command.

(venv) C:\Users\alexe\My_python_projects\i_love_python>
```

Figure 6: The package is successfully installed

Writing and executing the script

Now, you may close the terminal (hint: use trash-bin icon) and write the script:



```
import emoji

print(emoji.emojize("I :red_heart:  Python"))

```

Interpreter currently selected for the script execution

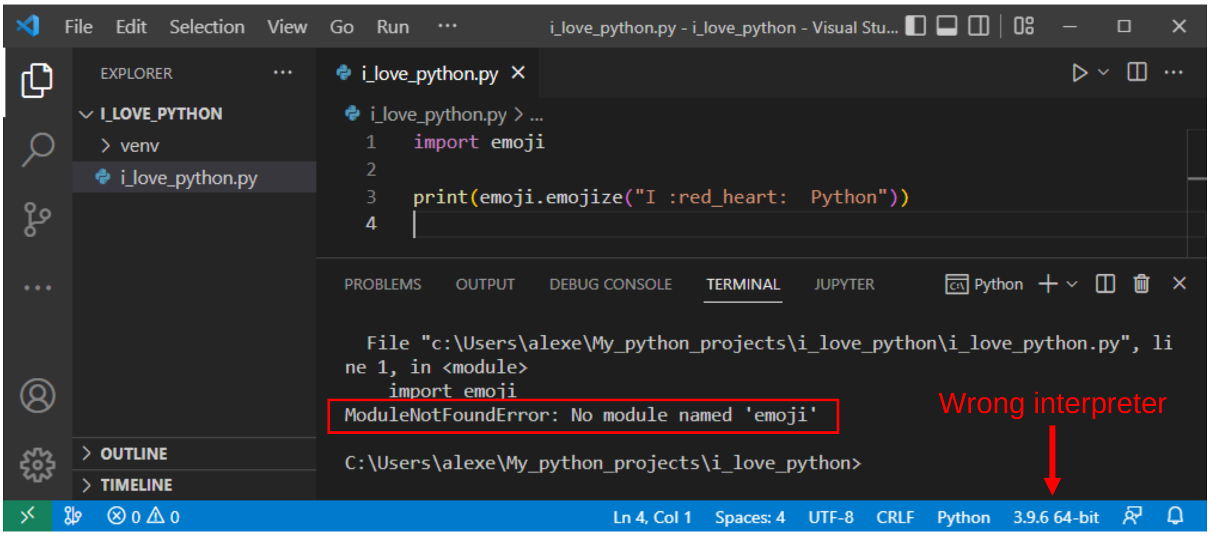
Figure 7: I love Python script

Save the script (File > Save or Ctr+S).

Before running this script, it’s VERY important to let the VS Code know what Python environment should be used. Figure above illustrates how to identify the Python environment currently selected by VS Code for the script execution.

If your VS Code detected and activated the virtual environment automatically (as was shown on Figure 3), the correct Python interpreter located may already be selected for the script execution.

However, in some cases you may need to do it manually. For instance, Figure 8 shows a system that is going to use a wrong interpreter (indicated by the red arrow) to run our script. So, you may get the Module Not Found error when you try running the script with the wrong interpreter:



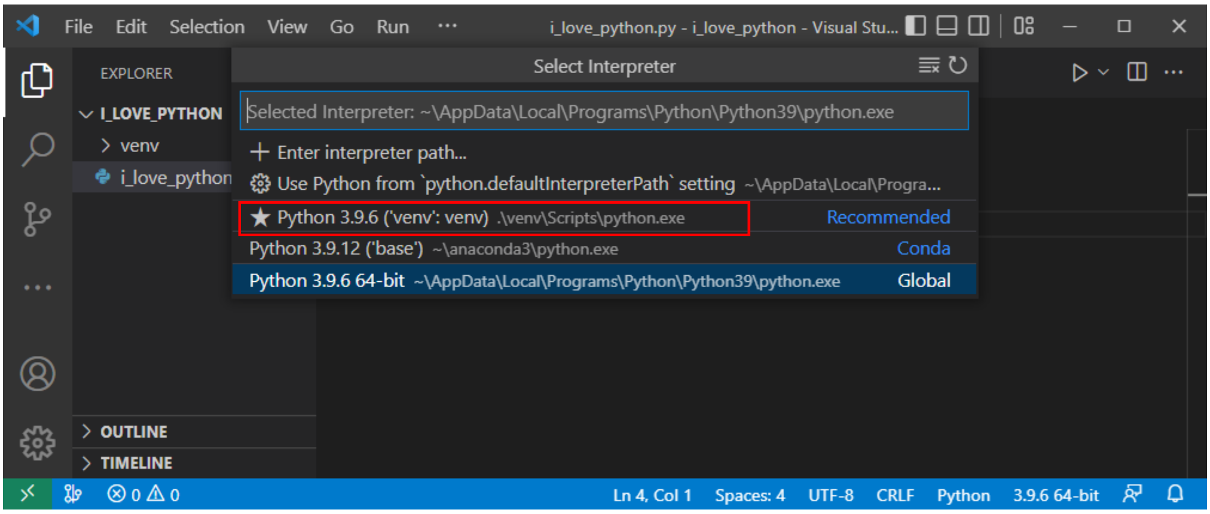
```
import emoji

print(emoji.emojize("I :red_heart:  Python"))

```

```
  File "c:\Users\alexe\My_python_projects\i_love_python\i_love_python.py", li
ne 1, in <module>
    import emoji
ModuleNotFoundError: No module named 'emoji'

C:\Users\alexe\My_python_projects\i_love_python>
```

Wrong interpreter

Figure 8: Module Not Found Error

In this case, you will need to select the proper environment, as shown on Figure 10.



Selected Interpreter: ~\AppData\Local\Programs\Python\Python39\python.exe

+ Enter interpreter path...

Use Python from `python.defaultInterpreterPath` setting ~\AppData\Local\Progra...

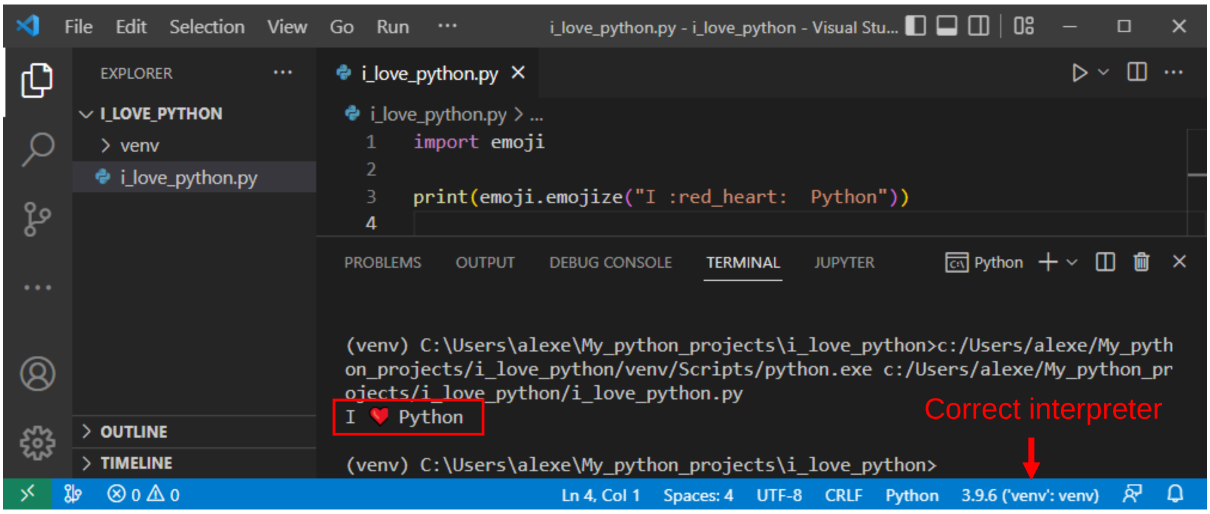
★ Python 3.9.6 ('venv': venv) .\venv\Scripts\python.exe

Python 3.9.12 ('base') ~\anaconda3\python.exe

Python 3.9.6 64-bit ~\AppData\Local\Programs\Python\Python39\python.exe

Figure 10: Local virtual environment in the list of available interpreters

After selecting the proper environment, the script works well:



```
import emoji

print(emoji.emojize("I :red_heart:  Python"))
```

```
(venv) C:\Users\alexe\My_python_projects\i_love_python>c:/Users/alexe/My_pyth
on_projects/i_love_python/venv/Scripts/python.exe c:/Users/alexe/My_python_pr
ojects/i_love_python/i_love_python.py
I ❤ Python

(venv) C:\Users\alexe\My_python_projects\i_love_python>
```

Correct interpreter

Figure 11: Successfully executed script

By the way, you may execute code line-by-line by using Sift+Enter.

Try to add more print statements using different emojies (e.g. **:thumbs_up:** or **:grinning_face:**) and run the code line-by-line.

Well done! You have finished the optional part of Practical 1