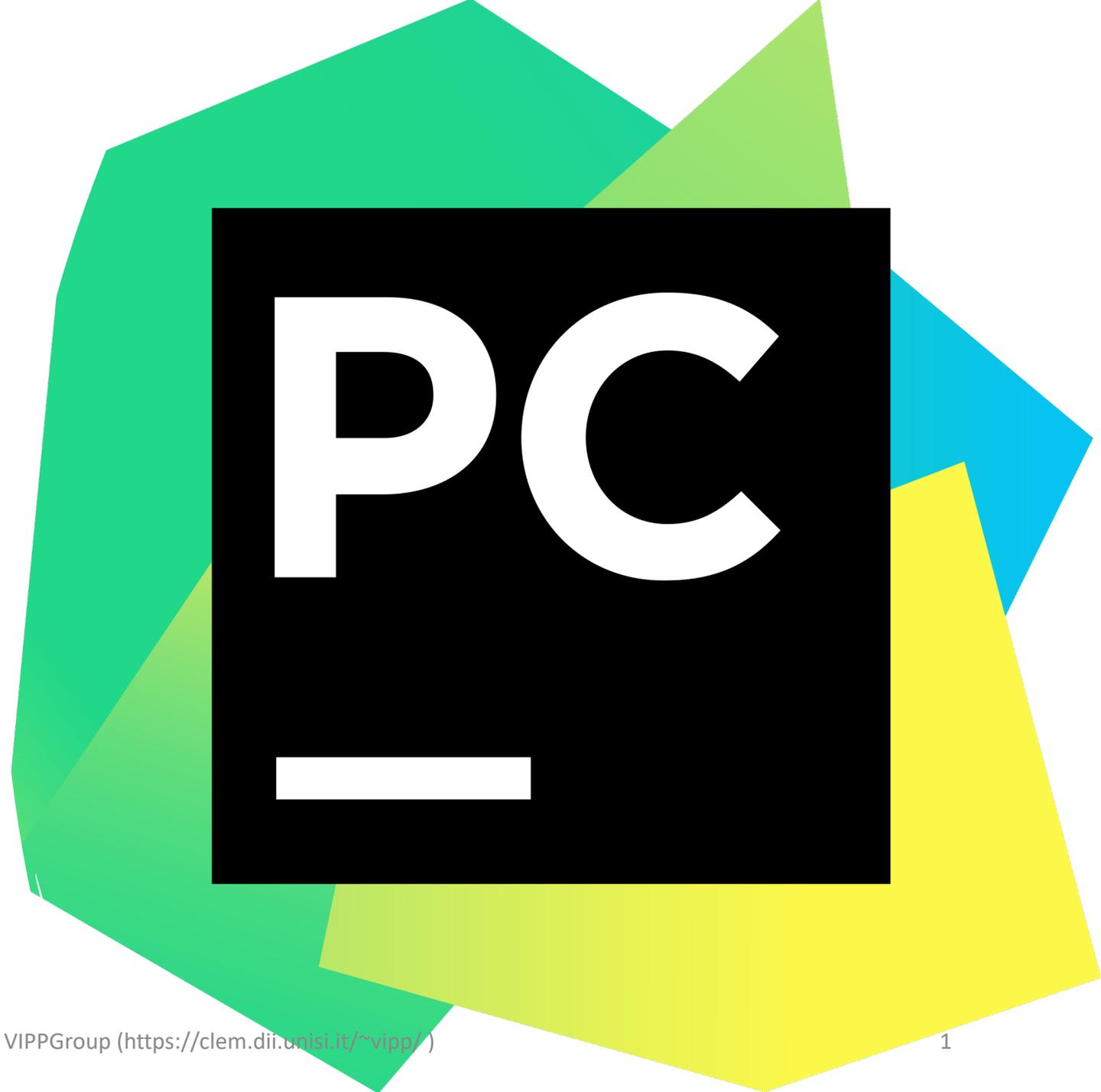


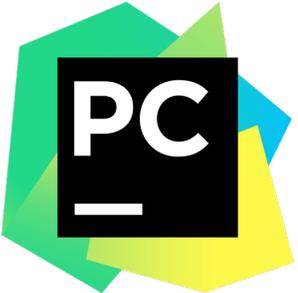
Python IDE setup guide

*PyCharm Community
Edition*



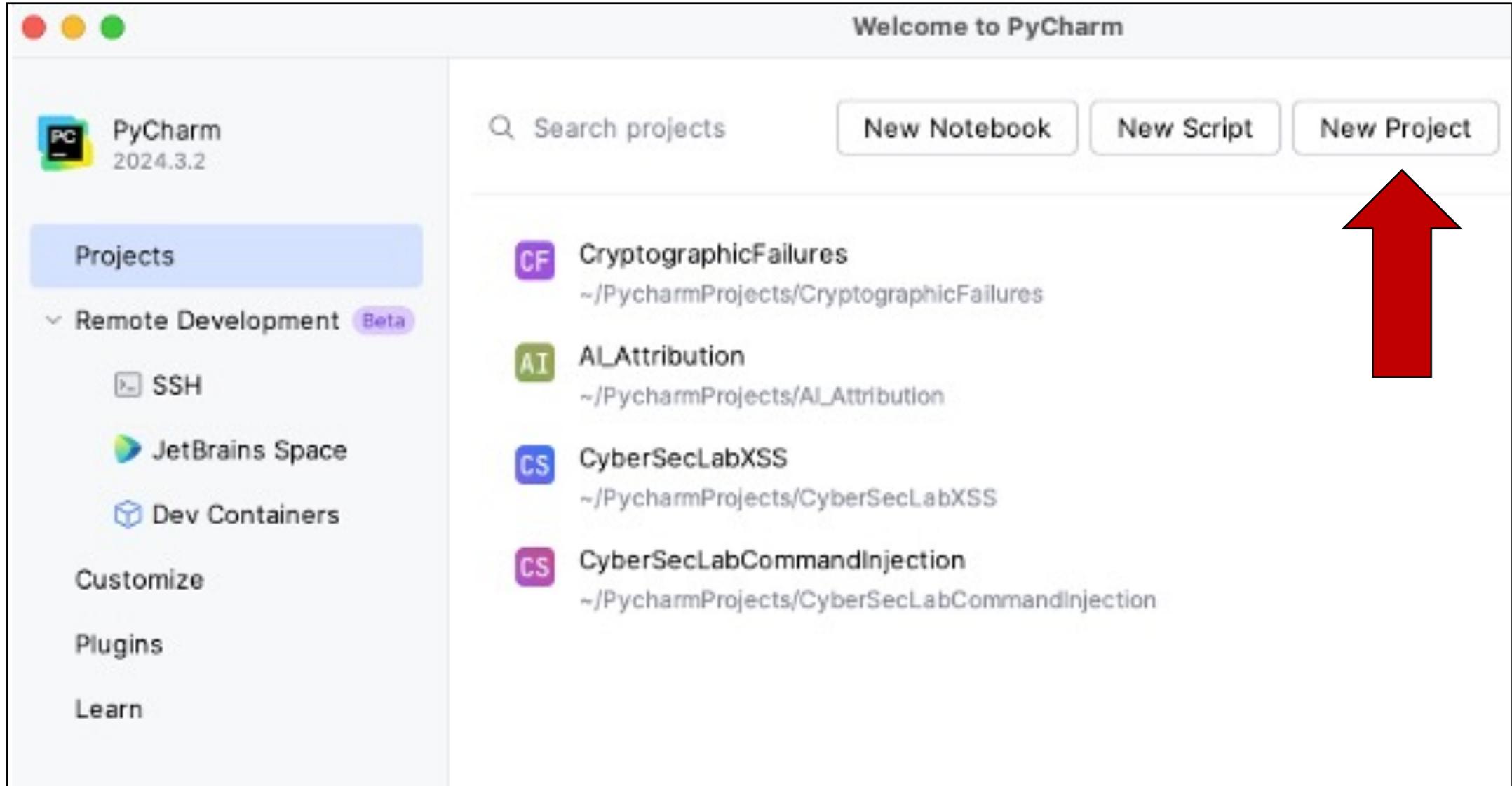
Start **PyCharm**. The executable is located in: C:/pycharm/bin/pycharm64.exe

If it asks for your UNISI credentials, then log in. Do not tick the *Remember me* box.

A screenshot of a Windows File Explorer window showing the contents of the 'bin' directory within the PyCharm installation on the local disk (C:). The window title is 'bin' and the address bar shows the path 'This PC > Local Disk (C:) > PyCharm > bin'. The left sidebar shows the navigation pane with 'Local Disk (C:)' selected. The main pane displays a list of files and folders. The file 'pycharm64' is highlighted in blue. A large red arrow points from the right side of the window towards the 'pycharm64' file.

Name	Date modified	Type	Size
elevator	2/12/2025 10:35 PM	Application	103 KB
format	2/12/2025 10:35 PM	Windows Batch File	1 KB
fsnotifier	2/12/2025 10:35 PM	Application	137 KB
fsnotifier-wsl	2/12/2025 10:35 PM	File	24 KB
idea.properties	2/12/2025 10:35 PM	PROPERTIES File	11 KB
inspect	2/12/2025 10:35 PM	Windows Batch File	1 KB
jetbrains_client64	2/12/2025 10:35 PM	Application	1,456 KB
jetbrains_client64.exe.vmoptions	2/12/2025 10:35 PM	VMOPTIONS File	1 KB
launcher	2/12/2025 10:35 PM	Application	135 KB
litedit	2/12/2025 10:35 PM	Windows Batch File	1 KB
msvcpl140.dll	2/12/2025 10:35 PM	Application exten...	566 KB
pycharm	2/12/2025 10:35 PM	Windows Batch File	9 KB
pycharm	2/12/2025 10:35 PM	Icon	307 KB
pycharm	2/12/2025 10:35 PM	SVG Document	3 KB
pycharm64	2/12/2025 10:35 PM	Application	1,457 KB
pycharm64.exe.vmoptions	2/12/2025 10:35 PM	VMOPTIONS File	1 KB
restarter	2/12/2025 10:35 PM	Application	380 KB
runnew	2/12/2025 10:35 PM	Application	152 KB
ttyfix	2/12/2025 10:35 PM	File	30 KB
Uninstall	2/27/2025 10:22 AM	Application	180 KB
WinProcessListHelper	2/12/2025 10:35 PM	Application	251 KB
WinShellIntegrationBridge.dll	2/12/2025 10:35 PM	Application exten...	212 KB
wslhash	2/12/2025 10:35 PM	File	102 KB
wslproxy	2/12/2025 10:35 PM	File	38 KB

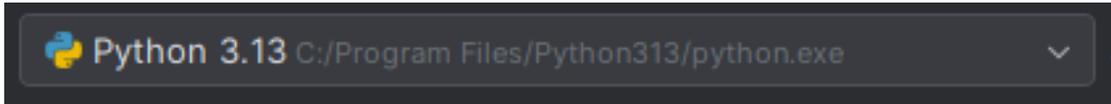
1. Create a new project with the **New Project** button.



2. Give a name to your project (e.g. CybersecurityLab). You will find the new project in:

```
C:/Users/<diism-user>/PycharmProjects/CybersecurityLab
```

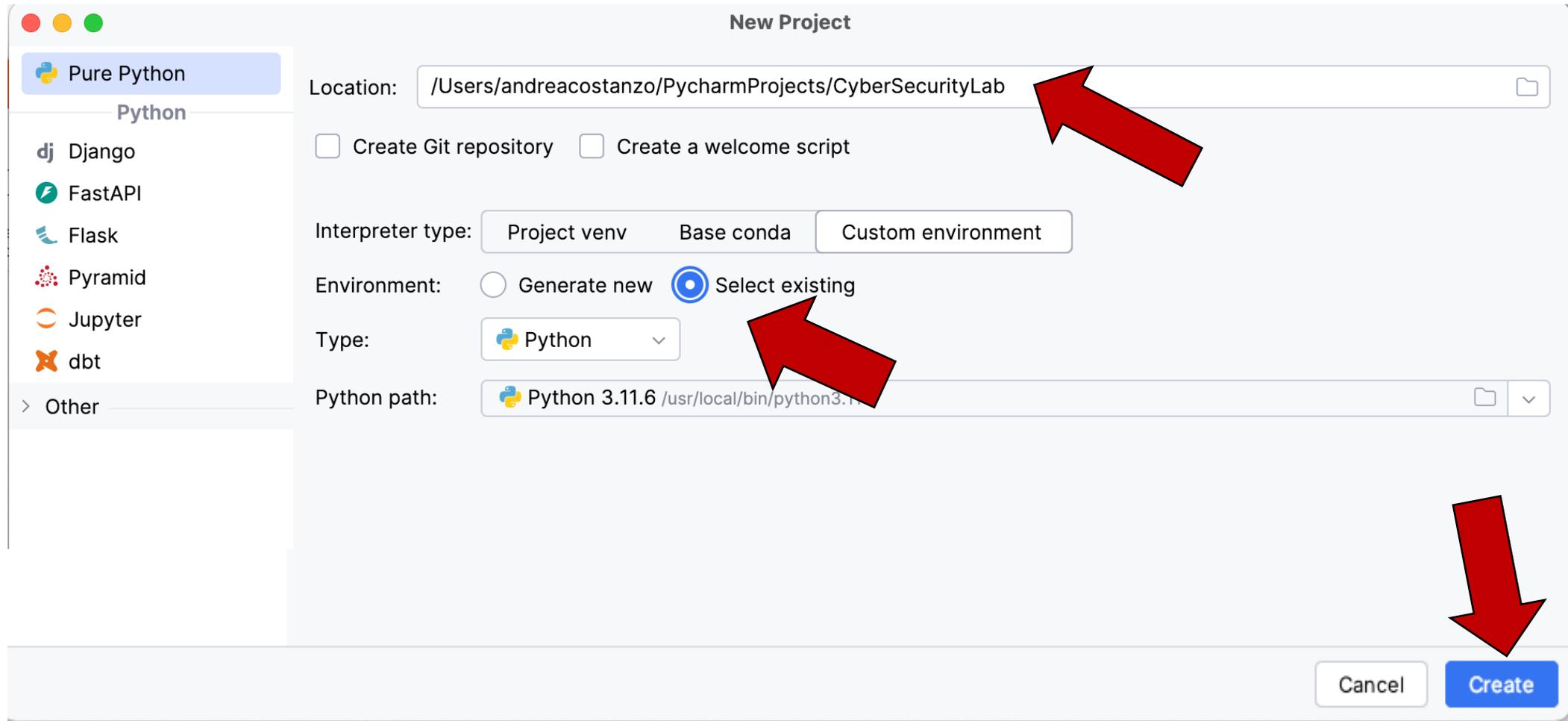
3. Select Select existing next to the *Environment* line. Leave the rest unchanged, making sure that *Python path* is set to:



If you see a different path, click on the icon to the right and browse until you reach the Python installation directory that is located in:

```
C:/Program Files/Python313/python.exe
```

4. Press the **Create** button in the bottom right portion of the screen



- Project
- Welcome to GitHub Copil
- Structure
- Python Console
- Python Packages
- Services
- Terminal
- Problems
- Version Control

Project

- PythonProject ~/PycharmProjects/PythonPr
 - .venv
 - External Libraries
 - Scratches and Consoles

Your files and directories are shown here

The code (or any file in general) currently opened are shown here

Search Everywhere Double ↑

Go to File ⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open them

Terminal Local x + v

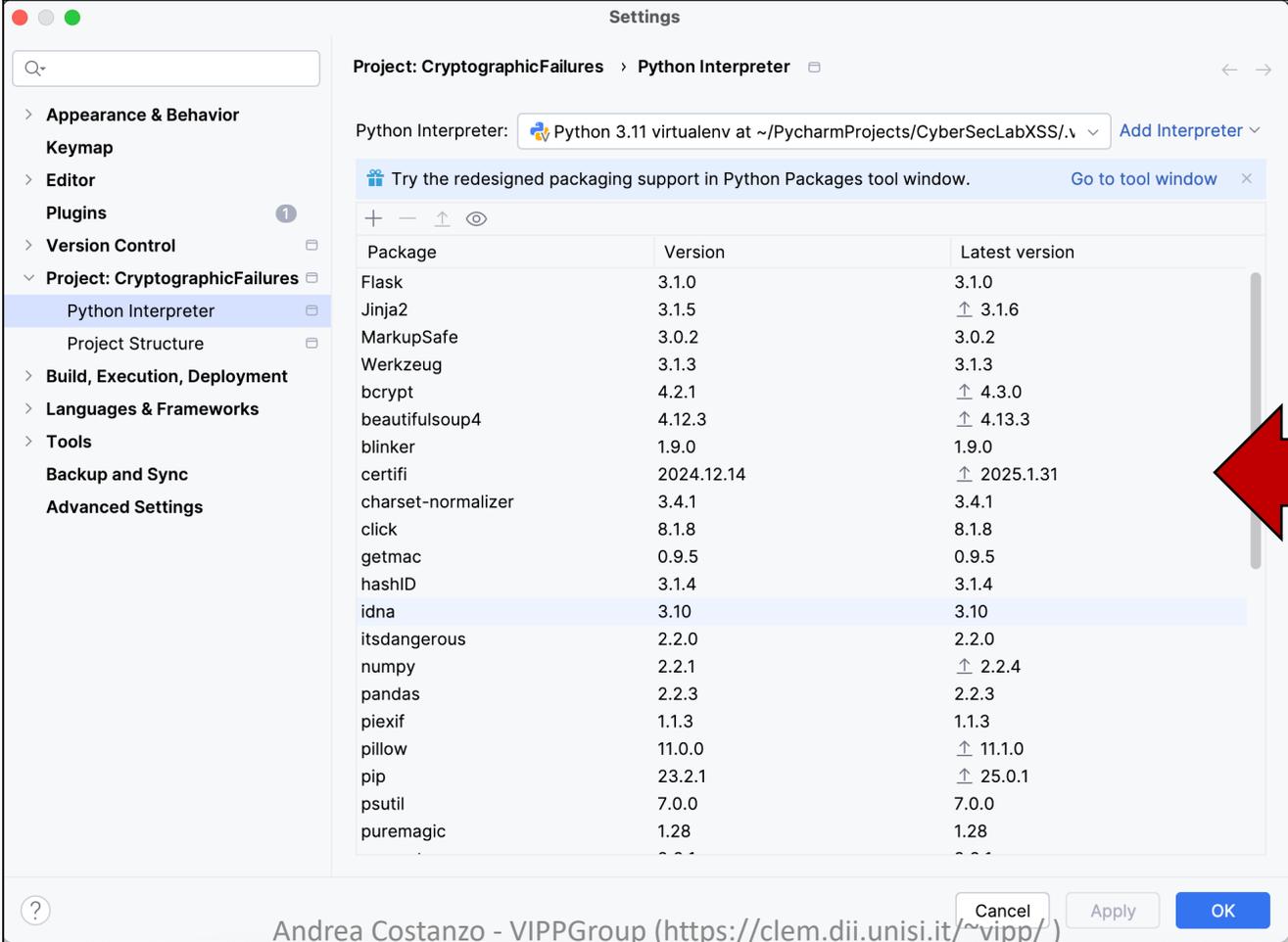
```
(.venv) ~/PycharmProjects/PythonProject
|
```

Commands that you will run directly are shown here

Andrea Costanzo - VIPPGROUP (<https://clem.dii.unisi.it/~vippp/>)

5. Verify that PyCharm finds the packages we need to use in our codes

- Click on `File` → `Settings`
- Scroll down to `Project: [NAME]` → `Python Interpreter`
- You should see a long list of packages similar to this one:



The screenshot shows the PyCharm Settings window for the project 'CryptographicFailures' under the 'Python Interpreter' section. The left sidebar is expanded to 'Python Interpreter'. The main panel displays a table of installed packages with their current and latest versions. A red arrow points to the 'Python Interpreter' option in the sidebar, and another red arrow points to the package list table.

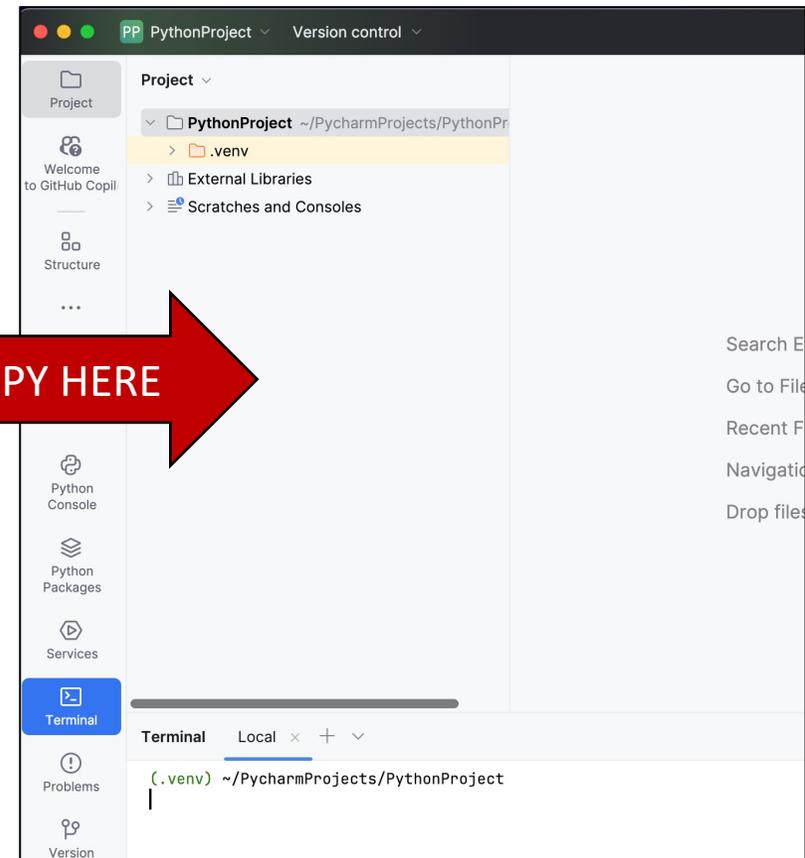
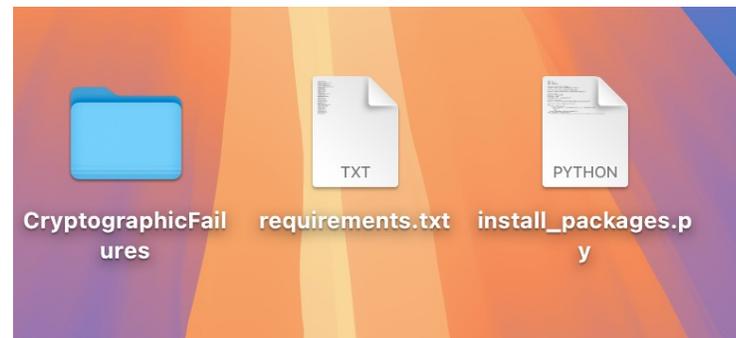
Package	Version	Latest version
Flask	3.1.0	3.1.0
Jinja2	3.1.5	↑ 3.1.6
MarkupSafe	3.0.2	3.0.2
Werkzeug	3.1.3	3.1.3
bcrypt	4.2.1	↑ 4.3.0
beautifulsoup4	4.12.3	↑ 4.13.3
blinker	1.9.0	1.9.0
certifi	2024.12.14	↑ 2025.1.31
charset-normalizer	3.4.1	3.4.1
click	8.1.8	8.1.8
getmac	0.9.5	0.9.5
hashID	3.1.4	3.1.4
idna	3.10	3.10
itsdangerous	2.2.0	2.2.0
numpy	2.2.1	↑ 2.2.4
pandas	2.2.3	2.2.3
piexif	1.1.3	1.1.3
pillow	11.0.0	↑ 11.1.0
pip	23.2.1	↑ 25.0.1
psutil	7.0.0	7.0.0
puremagic	1.28	1.28

6. Download the course material from

https://clem.dii.unisi.it/~vipp/website_resources/courses/cybersecurity/lab_material.zip.

You need to use your UNISI credentials to access the Internet in the laboratory

7. Unzip the file and copy its content inside PyCharm, in the files and directories view

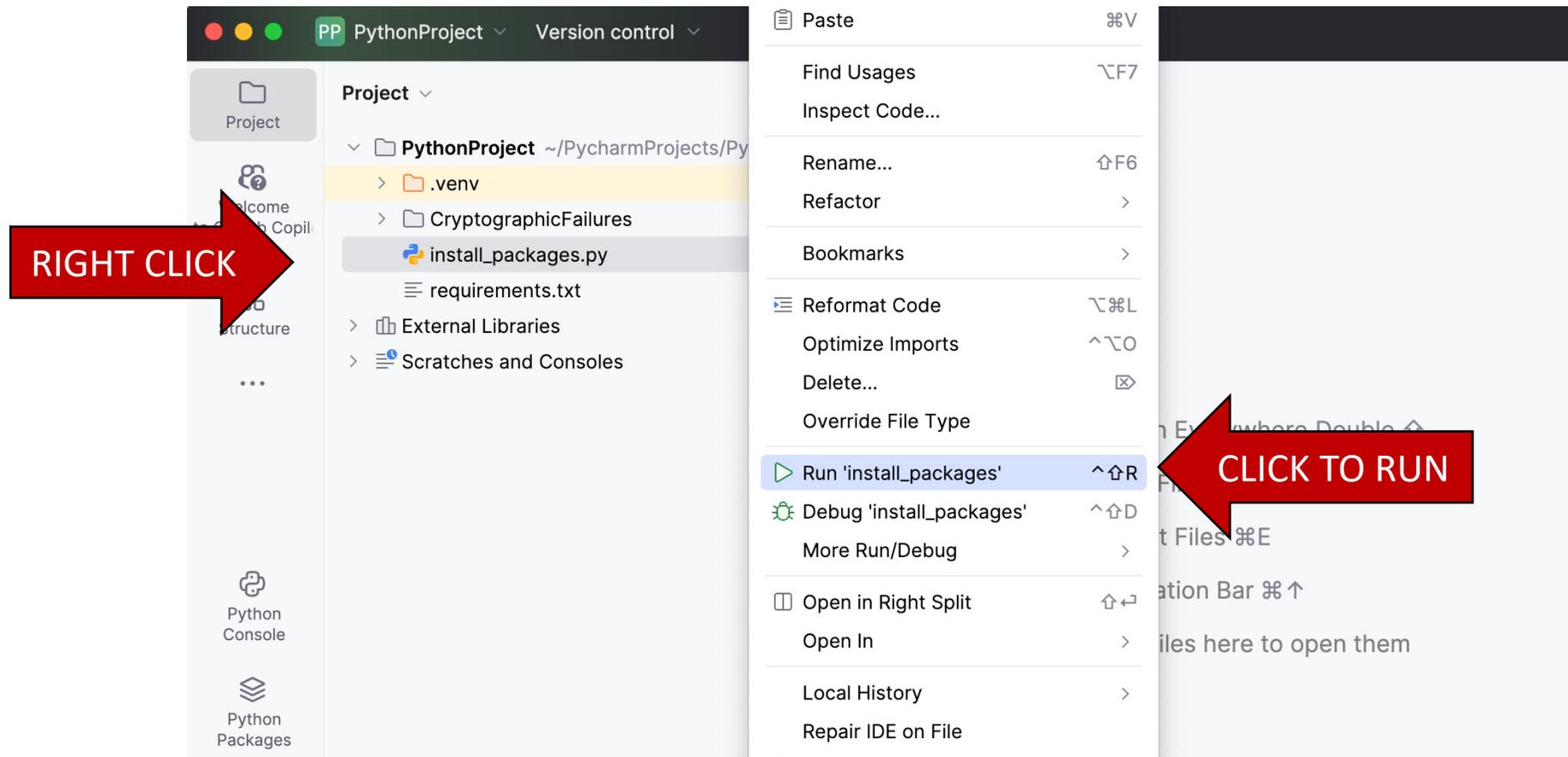


Copy with:

- Drag and drop
- CTRL-C / CTRL-V
- Manually copy inside the project directory

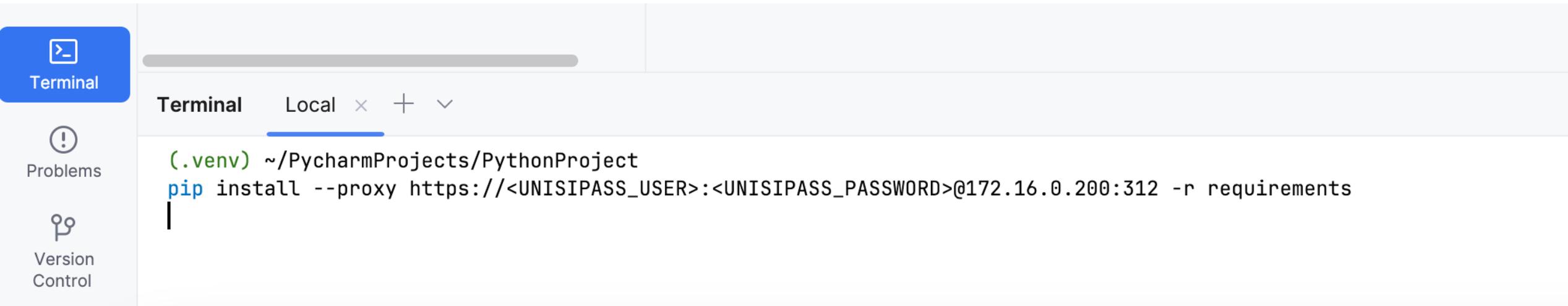
ALTERNATE INSTALL. SKIP THIS IF YOUR PACKAGES ARE ALREADY INSTALLED

- Right click on the `install_packages.py` file and select Run `'install_packages'`
- When asked, input your UNISI credentials
- Wait for the installation



ALTERNATE INSTALL. SKIP THIS IF YOUR PACKAGES ARE ALREADY INSTALLED

- Select the *Terminal* tab
- Copy the command below by replacing the placeholders with your UNISI credentials
- Press ENTER and wait for the installation to end



The screenshot shows the PyCharm IDE interface. On the left sidebar, the 'Terminal' tab is selected. The terminal window displays the following text:

```
(.venv) ~/PycharmProjects/PythonProject  
pip install --proxy https://<UNISIPASS_USER>:<UNISIPASS_PASSWORD>@172.16.0.200:312 -r requirements  
|
```

```
pip install --proxy https://<UNISIPASS_USER>:<UNISIPASS_PASSWORD>@172.16.0.200:312  
-r requirements
```

8. Running your first code. Double click on a file and use the controls on the top right.

The screenshot shows the PyCharm IDE interface. On the left, the Project tool window displays a tree view of the 'CryptographicFailures' project, with the file 'CryptFail_using_base64.py' selected. The main editor window shows the code for this file, which includes comments about deprecated hashes, an import statement for 'base64', a password variable, and a print statement for the encoded password. The code is as follows:

```
1 # Are deprecated hash functions such as MD5 or SHA1 in use, or are non-cryptographic hash functions used when
2 # cryptographic hash functions are needed?
3
4
5 import base64
6
7 # Example password
8 password = "SuperSecurePassword123!"
9
10 # Encode the password using Base64
11 encoded_password = base64.b64encode(password.encode("utf-8"))
12 print(f"Encoded Password: {encoded_password.decode('utf-8')}")
13
14 # Now go to https://www.base64decode.org/ and decode the encoded password!
15 # Now go to Cyberchef: https://cyberchef.io (or similar: https://tool.cyberchef.dev)
16 # Or from command line:
17 # encode: openssl base64 -e <<< test
18 # decode: openssl base64 -d <<< dGVzdAo=
```

At the bottom, the Run tool window shows the execution of 'CryptFail_using_base64'. The terminal output displays the encoded password: 'Encoded Password: U3VwZXJlZWN1cmVQYXNzd29yZDEyMyE=' and 'Process finished with exit code 0'. The entire Run tool window is highlighted with a red border.

Three red arrows are overlaid on the image: one pointing to the 'CryptFail_using_base64.py' file in the Project tool window, one pointing to the Run toolbar (containing play, stop, and refresh icons), and one pointing to the terminal output.

Run the code in **DEBUG MODE** to observe program's behavior or variable values

The screenshot displays the PyCharm IDE interface. The main editor shows a Python file named `CryptFail_using_base64.py`. The code includes comments about deprecated hash functions, an import statement, a password definition, and a Base64 encoding process. A red dot on line 11 indicates a breakpoint. The `Debug` button in the left sidebar is highlighted. The bottom panel shows the `Threads & Variables` window, which is highlighted with a red box. It displays the current state of variables: `encoded_password` is a bytes object and `password` is a string. Three yellow arrows provide instructions: one points to the breakpoint, another points to the `Debug` icon, and a third points to the variables window.

```
1 # Are deprecated hash functions such as MD5 or SHA1 in use, or are non-cryptographic hash functions used when
2 # cryptographic hash functions are needed?
3
4 import base64
5
6 # Password
7 password = "SuperSecurePassword123!"
8 password: 'SuperSecurePassword123!'
9
10 # Encode the password using Base64
11 encoded_password = base64.b64encode(password.encode("utf-8"))
12 encoded_password: b'U3VwZXJTWzN1cmVQYXNzd29yZDEyMyE='
13 print(f"Encoded Password: {encoded_password.decode('utf-8')}")
14
15 # Now go to https://www.base64decode.org/ and decode the encoded password!
16 # Now go to Cyberchef: https://cyberchef.io (or similar: https://tool.cyberchef.dev)
17 # Or from command line:
18 # encode: openssl base64 -e <<< test
19 # decode: openssl base64 -d <<< dGVzdAo=
```

Annotations:

- Set breakpoint (click on line number)
- Use «Bug» icon to run in Debug mode
- Values of variables are shown here

Let's get started

```
while (not edge) {  
  run();  
}
```

```
do {  
  run();  
} while (not edge);
```

