Contents

[Software: 2](#_Toc105765972)

[Important “new release” notes: 3](#_Toc105765973)

[General idea behind these automations: 4](#_Toc105765974)

[Folder 🡪 SCHEME TESTS 4](#_Toc105765975)

[FILE 🡪 “1 Webscrape all the schemes” 5](#_Toc105765976)

[FILE 🡪 “2 Compare Scheme Pages” 6](#_Toc105765977)

[Folder 🡪 SCRIPTS WRITTEN FOR SCENARIOS 6](#_Toc105765978)

[Folder 🡪 TEXT COMPARE 9](#_Toc105765979)

[File 🡪 “Compare.py” 9](#_Toc105765980)

[NEW SCRIPT 🡪 file “0 HOP Universal script” 10](#_Toc105765981)

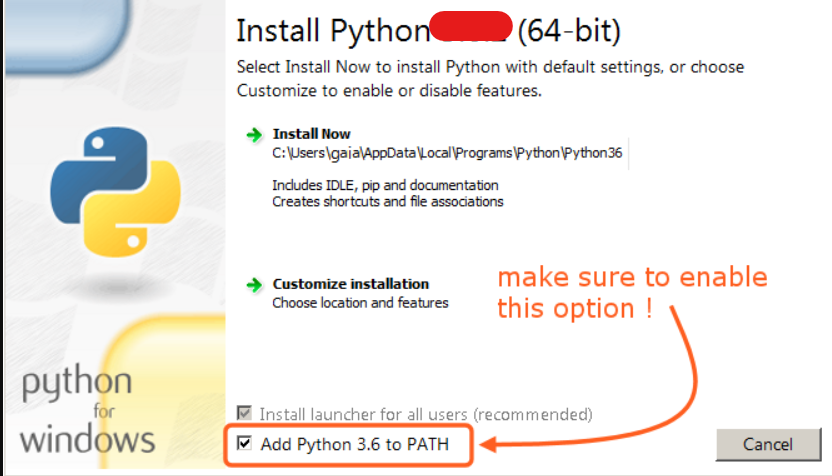
[General notes: 11](#_Toc105765982)

[How to use the scripts in VS Code 11](#_Toc105765983)

# Software:

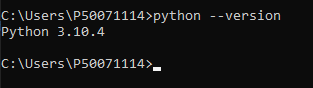
**What you need to have:**

1. Newest **Python** from here: <https://www.python.org/downloads/>
   1. **IMPORTANT**: make sure that whoever from IT Support is installing it has enabled this option:



* 1. After installation it’s good to check if everything has been installed correctly:

open Command Prompt and type “**python --version**".

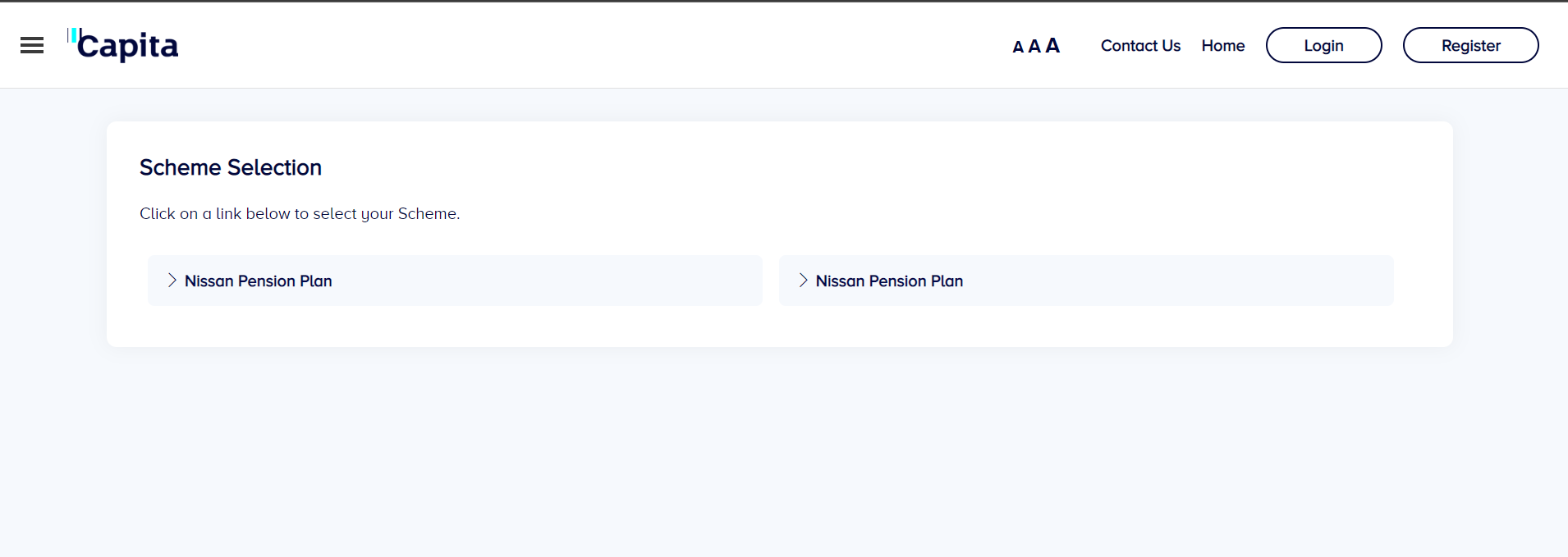


1. **Visual Studio Code** (any other text editor will do but VS Code will be muuuch more convenient to use) from here: <https://code.visualstudio.com/>
   1. When you try to run first .py file in VS Code it **will prompt you to install Python add-on for VS Code** – do it 😊 and you don’t need any special permissions for that – so no IT Support needed for that step
2. **OPTIONAL**: I am thinking about uploading it on **GitHub** but I don’t have it installed yet. But I think you can request install of that just in case. Download it from here: <https://gitforwindows.org/>
   1. After the installation make sure it’s installed correctly by opening Command Prompt and typing “**git version**”. That should show Git version installed on your machine
   2. If you didn’t use GitHub before it is a version control system – so basically while writing code I can easily upload changes (and you can update your local folder with that changes without me sending you files) and control what was changed when 😊 If sth is not working after some changes I can also go back to version that was working with Git 😊
3. For points 1 and 2 (and 4 too) you’ll need IT Support to install it. But then you can continue by yourself by:
   1. opening Command Prompt OR PowerShell
   2. typing “**pip3 install selenium**” and hitting enter
   3. typing “**pip3 install chromedriver-autoinstaller**” and hitting enter
   4. typing “**pip3 install webdriver-manager**” and hitting enter

**So, the request to IT should consists of:**

1. **Python (with specifying about that PATH thingy to be enabled during installation)**
2. **Visual Studio Code**
3. **Git – OPTIONAL but probably can be useful in the future**

## Important “new release” notes:

* New from **10.06.22** -> type “**pip3 install webdriver-manager**” and hit enter in Command Prompt or PowerShell
* The new scheme pages have the same problem as Pfizer has: if you open incognito/private browser window, open the page and try to go to Scheme Information or Register it will go to this page: 

This is a defect on the page (I think). It will cause Register and Scheme Information tests in folder “scripts written for scenarios” show that pages are not matching demo. So bare that in mind.

# General idea behind these automations:

## Folder 🡪 SCHEME TESTS

Scripts in that folder are written to **scrape whole pre login pages and save all that text in evidence files**

**Example of how webscraping scirpt is working (using this file as an example 🡪 “HOP Corvidae (demo).py”):**

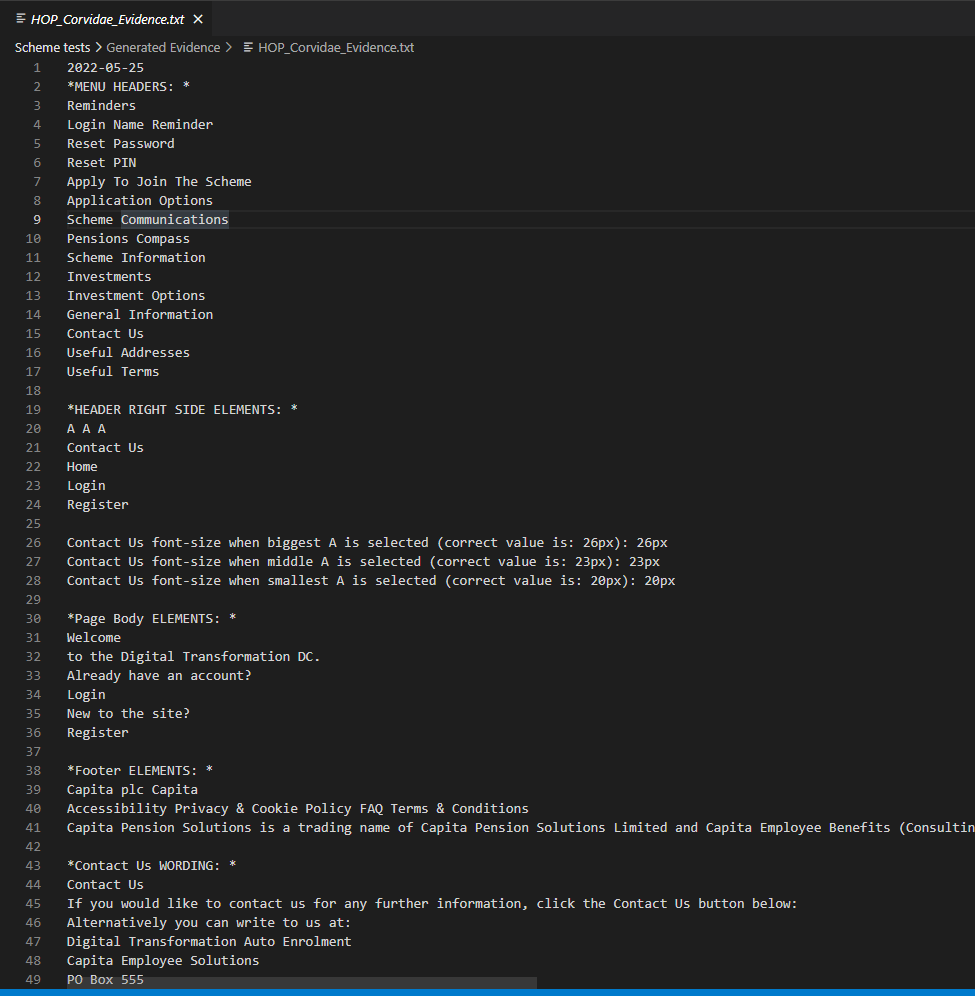
1. It will open the Corvidae page
2. Handle cookies popup
3. Navigate through whole page to scrape text from every subpage
4. It will also scrape px size of text after clicking every A from AAA
5. Evidence file will be generated with everything that was scraped from the page

**Example of how webscraping scirpt is working (using this file as an example 🡪 “HOP Pfizer (demo).py”):**

1. It will open the Pfizer page
2. Handle cookies popup
3. First script will compare menu elements from demo and Pfizer page – if they are not matching the rest of the script will not be executed.
4. **If they menu elements are matching, then script 🡪 script will navigate through whole page to scrape text from every subpage**
5. It will also scrape px size of text after clicking every A from AAA
6. Evidence file will be generated with everything that was scraped from the page

This is how it works for all the HOP <scheme name>.py scripts

**Example of how evidence file is looking:**



## FILE 🡪 “1 Webscrape all the schemes”

This file will run all the scraping scripts one by one.

**I would suggest starting with this one.**

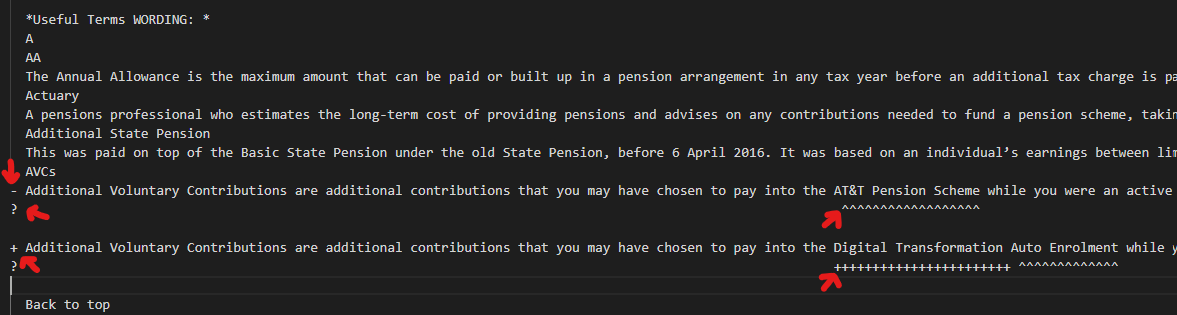
## FILE 🡪 “2 Compare Scheme Pages”

**This is the most useful script to run.**

**It will basically compare ALL the wording from Corvidae page to the wording scraped from scheme page of choice.**

After running “Webscrape all the schemes” (so all the evidence files are generated) you can run this script, input scheme name that you want to compare to demo and evidence file with all the lines compared will be generated in folder “**Compare Scheme Pages – evidence**”.

**Structure of the “HOP\_{scheme\_name}\_compared\_to\_demo.txt” file:**



This evidence file will look like this:

Lines starting with “**-**“ are from scheme page, lines starting with “**+**” are from demo page

Lines starting with “**?**” are lines that will show you where exactly is the difference

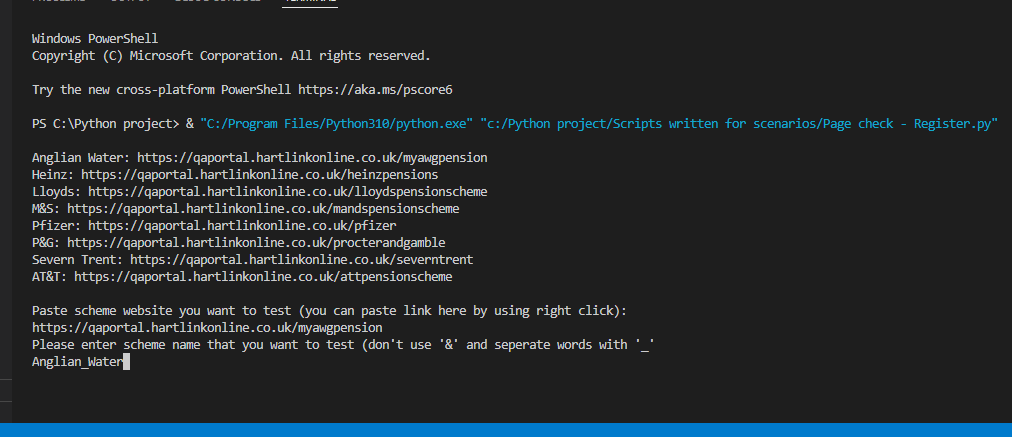
## Folder 🡪 SCRIPTS WRITTEN FOR SCENARIOS

These are perfect fit when doing test specific test on ALM -> you can quickly check what has changes on the page

There is set of scripts written to test specific pages:

**Example of how scripts are working (using this file as an example: “Page Check - Apply To Join The Scheme.py”):**

That test will first ask you to paste link for the home page of the scheme that you want to test (list of them is printed in terminal). Like shown below:



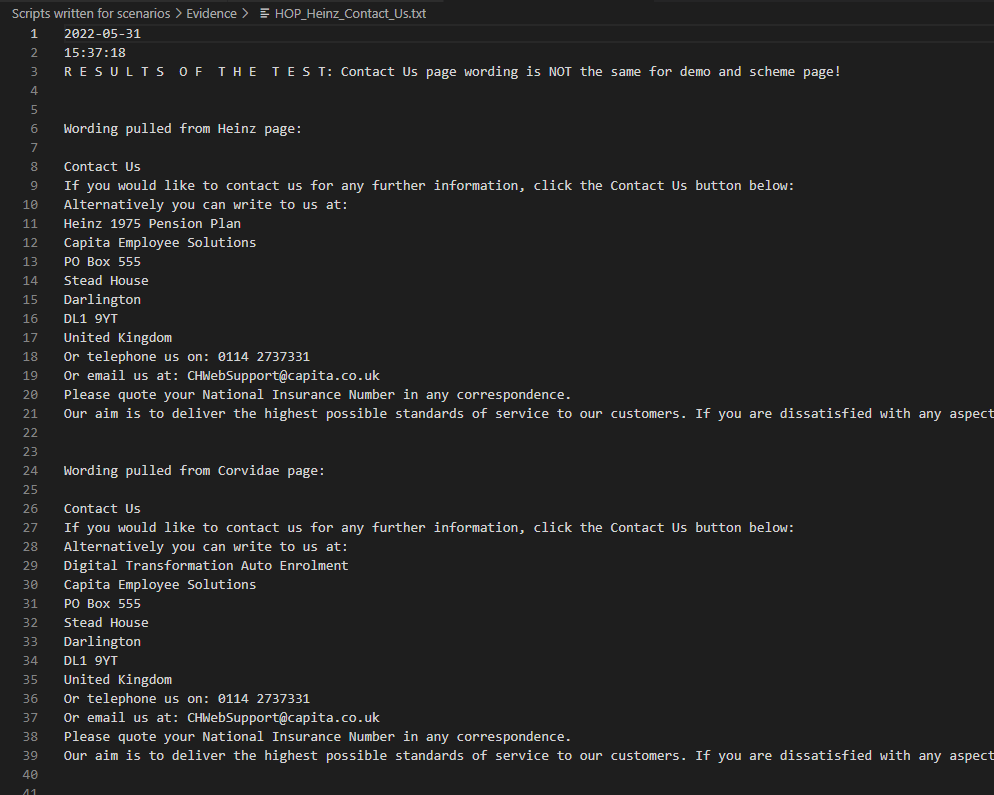
After pasting URL and evidence file name (evidence file name will look like this: **HOP\_<text that you're going to type>\_<name of the subpage>.txt**) script will:

1. open the browser window,
2. load URL that you’ve pasted,
3. go thru cookies popup
4. open subpage that should be tasted
5. scrape text from that page
6. then it’ll do the same for Corvidae page
7. then the script will compare what was pulled from both pages
8. IF the pages are **the same** then evidence file will include text pulled from scheme page and demo page and info that it matches
9. IF the pages are not the same evidence will include text pulled from scheme page and demo page and info that it is **not the same**

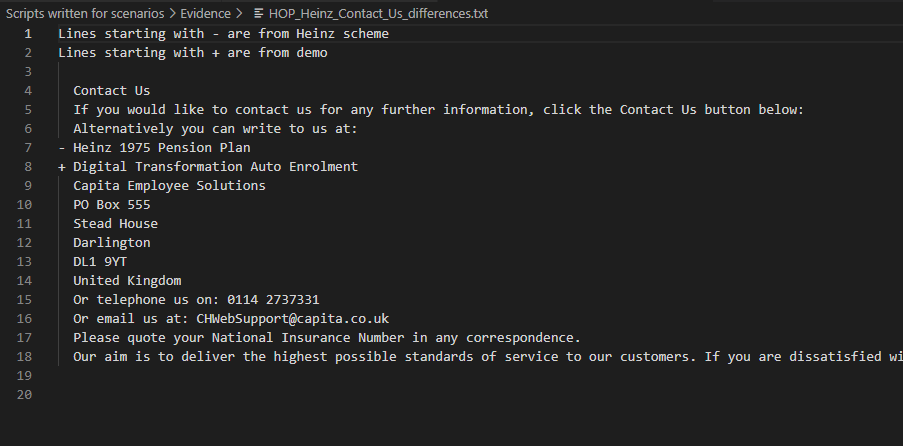
**AND**

Then it will generate another evidence file with **“\_differences.txt”** at the end of the name

**Example of the “Heinz\_Contact\_Us.txt”:**

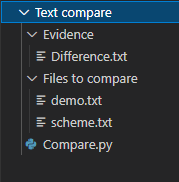


**Example of the “Heinz\_Contact\_Us\_differences.txt”:**



## Folder 🡪 TEXT COMPARE

List of elements in that folder (files with .py extension are python scripts):

****

## File 🡪 “Compare.py”

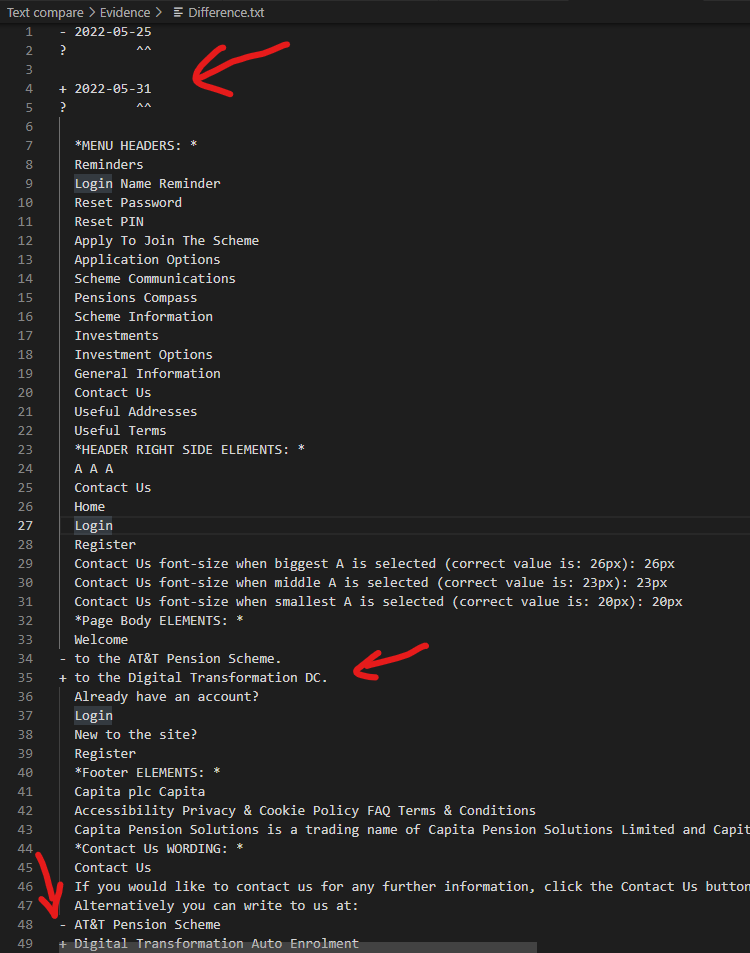
How it works:

1. You can paste text of choice to files “**demo.txt**” and “**scheme.txt**” (for example paste text from our .doc evidence file to demo.txt and text from scheme page to scheme.txt **🡪 btw I was sometimes using this script to compare subpages with a lot of text like FAQ or Useful Terms**)

IMPORTANT: Save the files after pasting text into them 😊

1. Then you can run “Compare.py” script
2. The script will first remove all the empty lines from both files
3. Then it will compare these two files
4. At the end it will generate “Difference.txt” file in which all the differences will be shown

**Difference.txt file example:**



## NEW SCRIPT 🡪 file “0 HOP Universal script”

So, all the new scripts now have “General Information” header missing in the menu. That is problematic, because I was deciding if the website will be scraped or not on condition of menu headers matching demo page.

That’s why I’ve prepared this new script – it will ask you to provide URL of scheme page that you want to scrape and will work exactly like the ones above. The difference is that it will not check if menu headers are matching the ones on demo. Also, I’ve commented part of code that would scrape links under “General Information” (there is comment at the top of the script with exact line numbers commented).

# General notes:

- ignore folders that will be generated automatically by Python like **\_pycache\_**

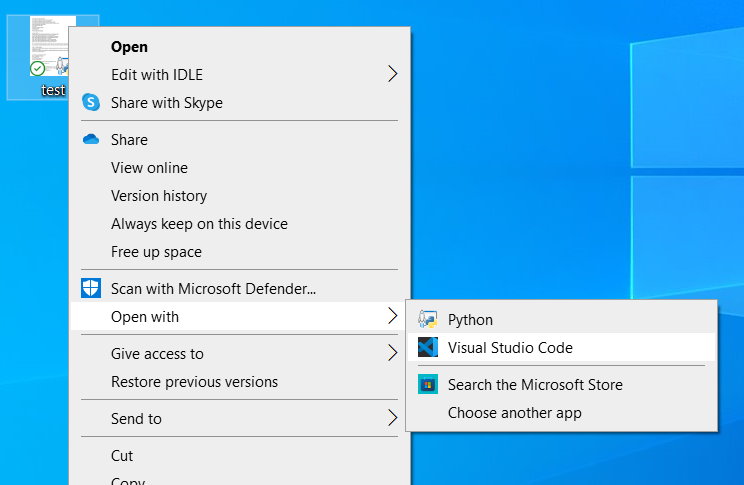
- Script **HOP\_functions.py** (it’s present in 2 folders at the moment) has all the page navigation functions written inside for other scripts to use so they are less cluttered. Don’t run it and don’t remove it 😊

- When you get a ZIP file from me and unpack it on your computer you can put wherever you want – scripts are written in a way that should work everywhere (I hope – I was using relative file paths)

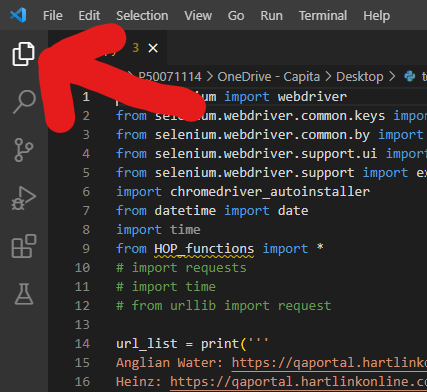
**-** Additional notes about specific scripts are written in comments inside of them **(green text in VS Code starting with #).**

# How to use the scripts in VS Code

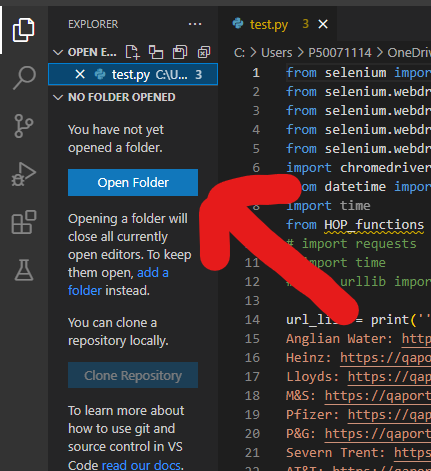
Open one of the scripts from the folder like that (files with .py at the end):

****

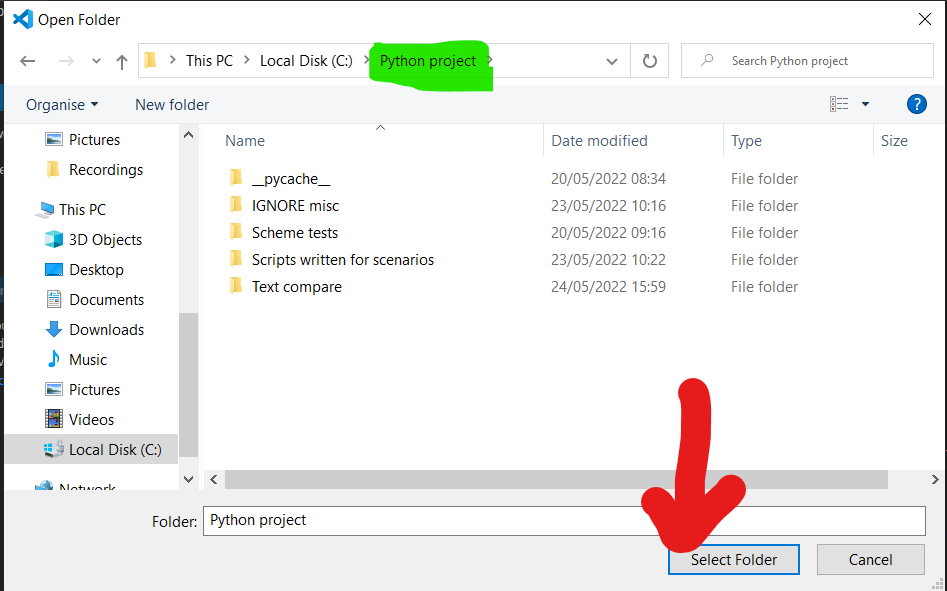
Click that icon in top left corner:

****

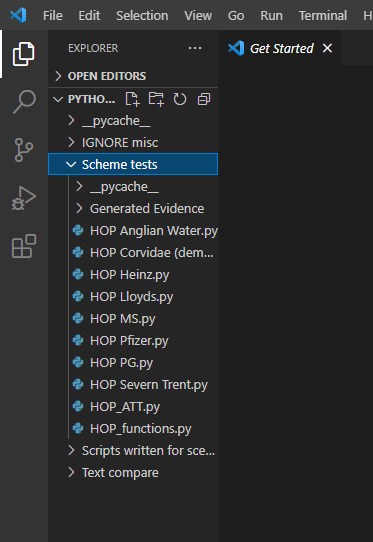
Click on “Open Folder”:



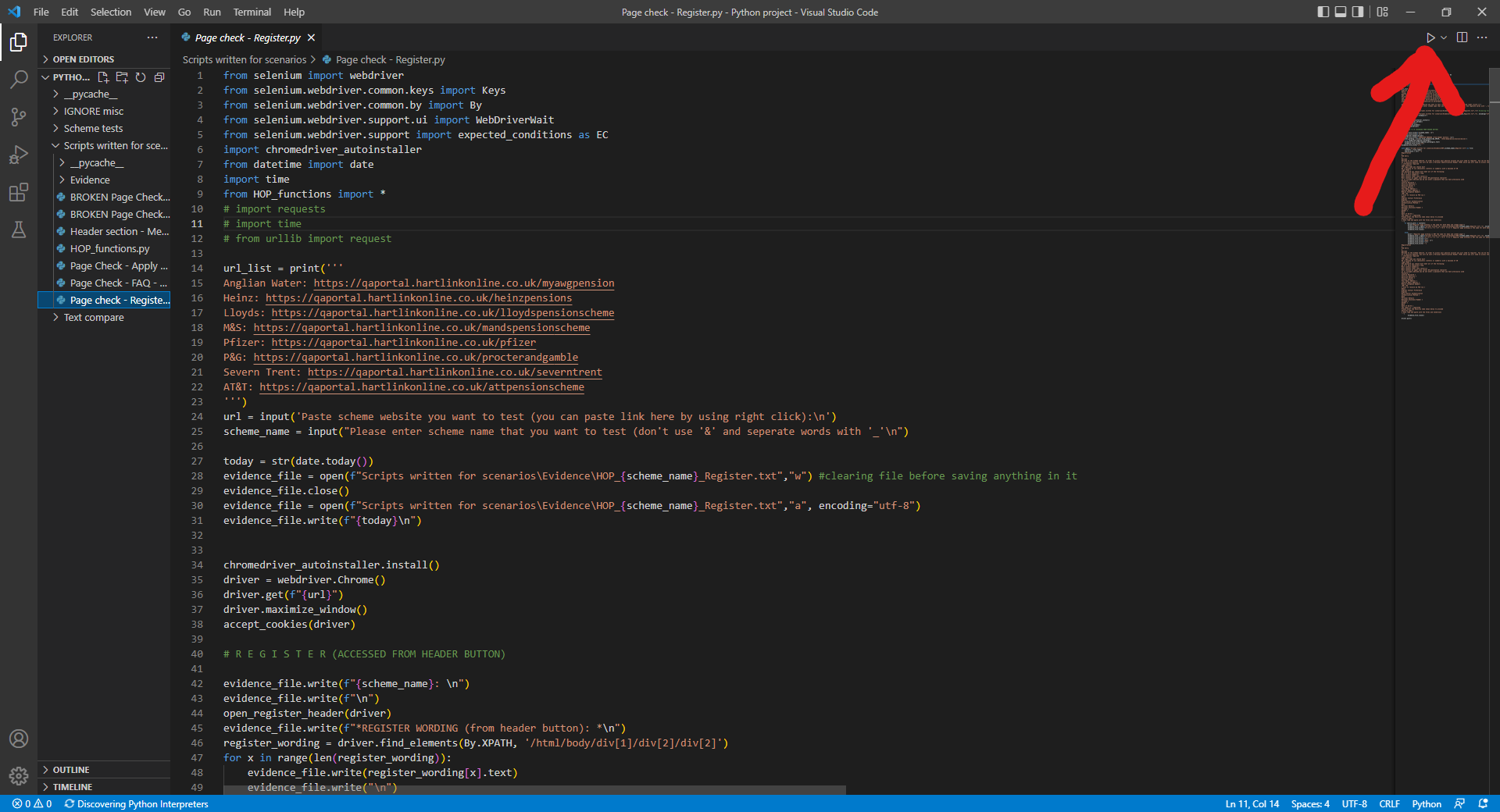
Find location where you’ve put your folder with scripts and click “select folder”:



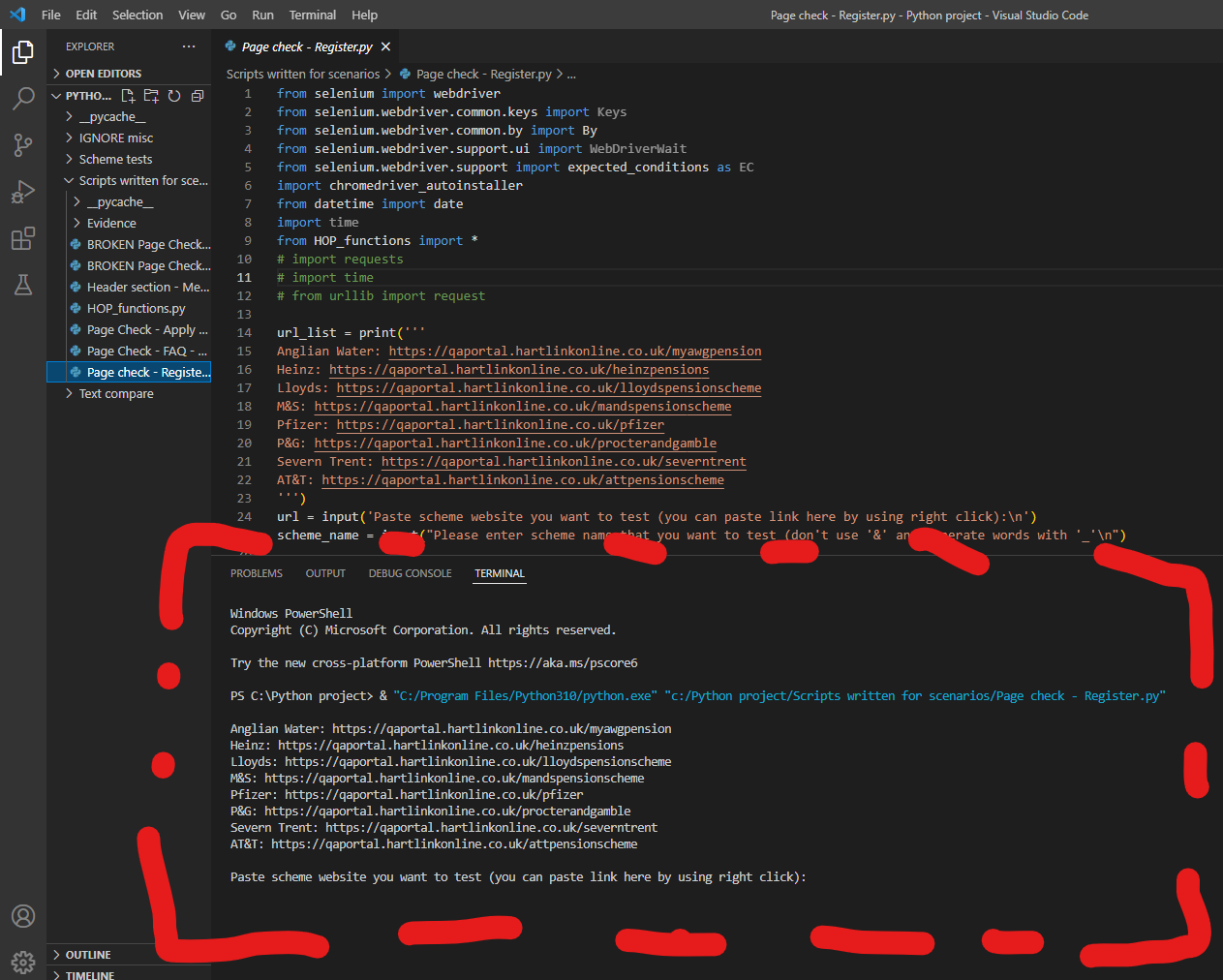
After that (there can be some prompt about trusting this location or sth like that, just accept that) you can browse chosen location on the left side of the screen and open multiple files at the same time 😊

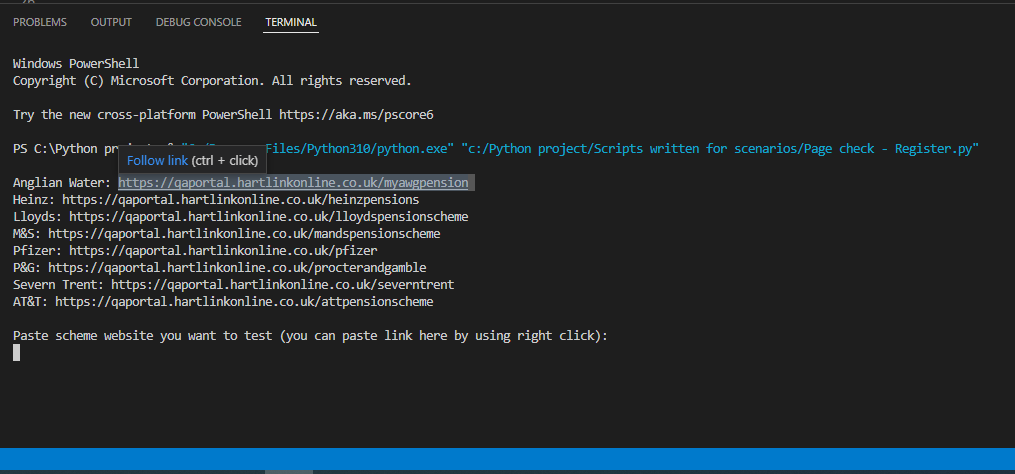


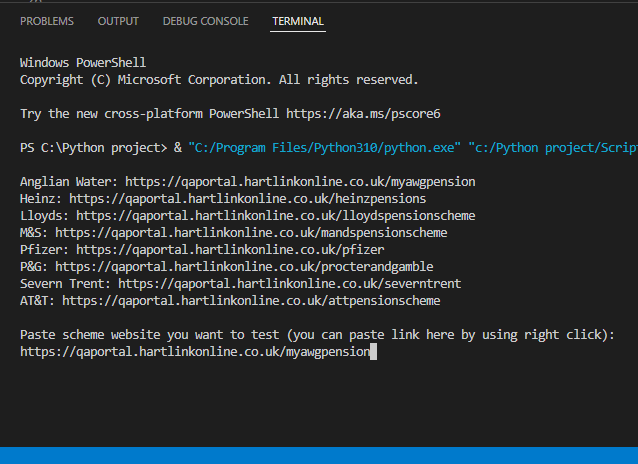
When you want to run the script you open it and then click on that “play” button in top right corner:

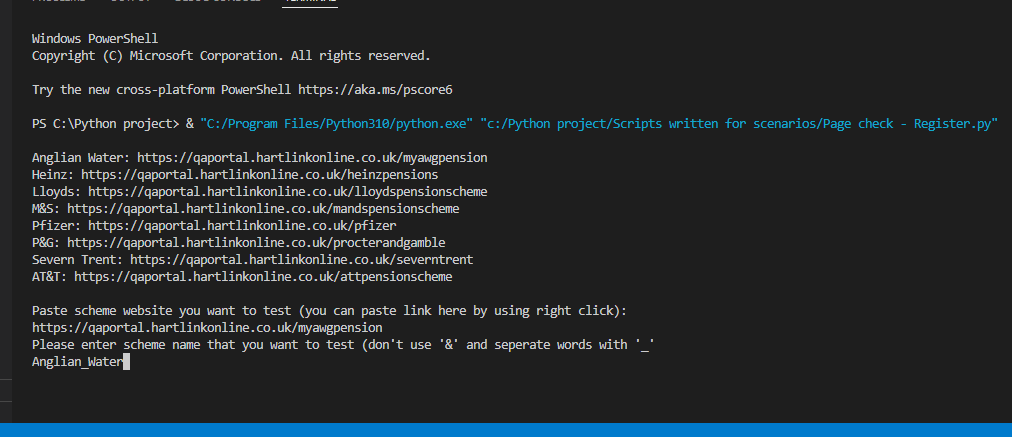


This terminal window will be displayed and will show you what program is printing/doing and you will be able to interact with script:



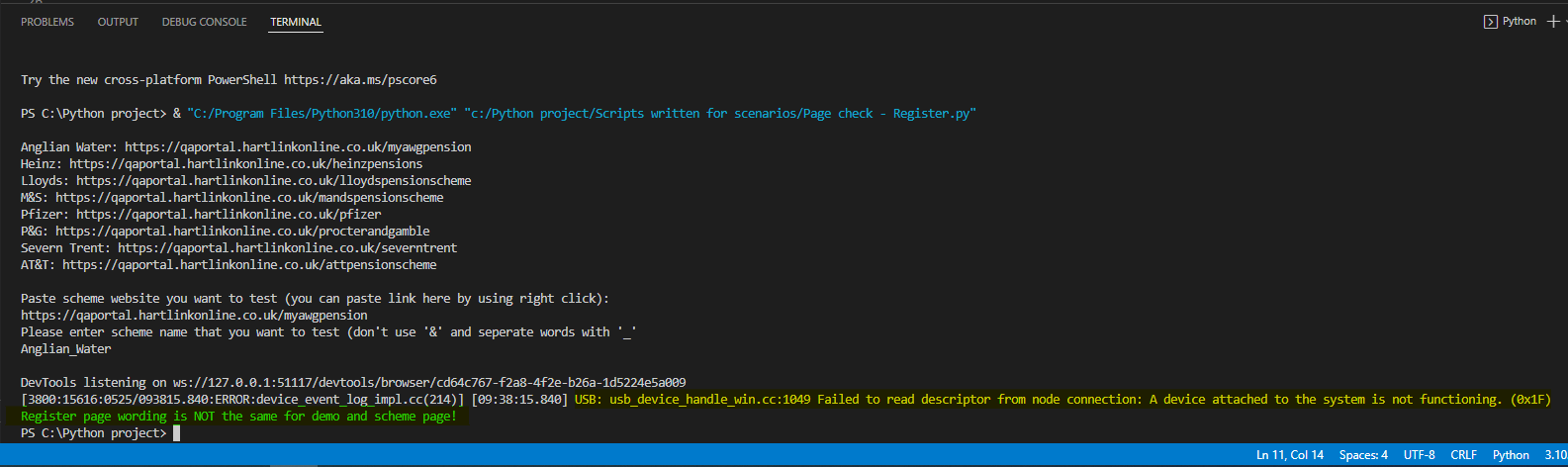






**Ignore text highlighted yellow** 😊 that error is just a bug of that version of web browser driver that controls page in browser

Green text is what program is printing 😊



Here you can see some evidence files examples:

