



Documentation: Disaster Data Scraping from Thinkhazard.org

Download Python

Download:

1. Open your web browser and go to the official Python website:
<https://www.python.org/downloads/windows/>
2. You will see a list of available Python versions. The website may automatically recommend a version for you based on your system, but you can also choose the version you prefer. We recommend downloading the latest stable version.
3. Scroll down and choose the download that matches your system architecture (32-bit or 64-bit). Most modern systems are 64-bit.
4. Click on the download link to start the download. The installer should be named something like "python-X.X.X.exe," where X.X.X is the Python version number.

Installation: Once the Python installer is downloaded, follow these steps to install Python on your Windows system:

1. Locate the downloaded installer (python-X.X.X.exe) and double-click it to run the installer.
2. In the Python installer, make sure to check the box that says "Add Python X.X to PATH." This option is essential to allow you to run Python from the command prompt.
3. Click the "Install Now" button. The installation process will begin.
4. Wait for the installer to complete the installation. This may take a few minutes.
5. Once the installation is finished, you will see a message that Python has been successfully installed. Click the "Close" button to exit the installer.

Verifying the Installation: To verify that Python has been successfully installed, follow these steps:

1. Open the Windows Start menu.
2. Type "cmd" and press Enter to open the Command Prompt.
3. In the Command Prompt, type "python" and press Enter.
4. You should see the Python interpreter starting, displaying the Python version and a prompt (">>>"). This confirms that Python is installed and accessible from the command prompt.
5. To exit the Python interpreter, type "exit()" and press Enter.

Download Pycharm

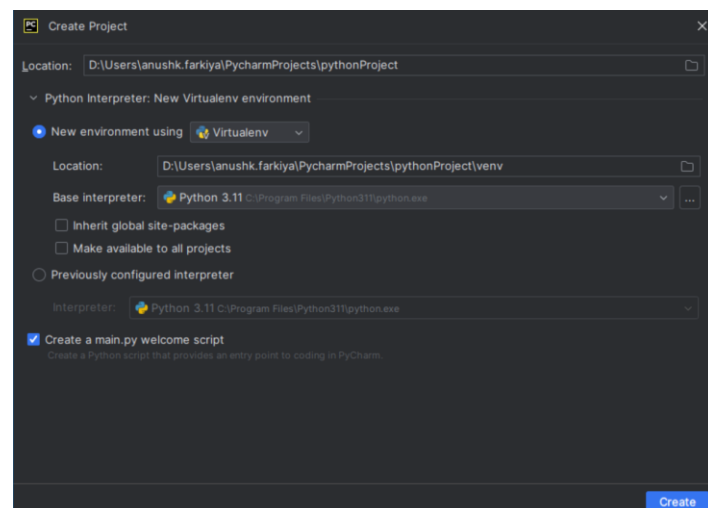
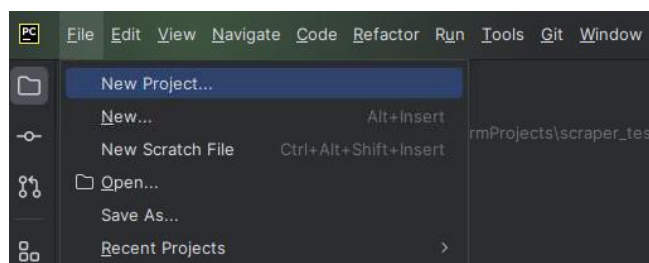
1. Open your web browser and go to the official PyCharm website:
<https://www.jetbrains.com/pycharm/download>
2. You will see a list of available PyCharm editions. Make sure to select "Community" (it's free) and click the "Download" button next to it.
3. The website will automatically recommend the download for your operating system (Windows). Click the download link to start the download.

Installation: Once the PyCharm installer is downloaded, follow these steps to install PyCharm on your Windows system:

1. Locate the downloaded installer (e.g., "pycharm-community-X.X.X.exe") and double-click it to run the installer.
2. In the PyCharm Setup Wizard, you can choose the installation options: - Select the installation location. - You can create desktop shortcuts, associations with .py files, and open files with PyCharm. - Choose whether to create associations for .py, .pyc, and .pyo files.
3. Click the "Next" button to proceed.
4. Review the installation settings and click the "Install" button.
5. Wait for the installer to complete the installation. This may take a few minutes.
6. Once the installation is finished, click the "Finish" button to exit the installer.
7. Initial Configuration: After the installation, you'll need to configure PyCharm:

Start PyCharm by searching for it in the Windows Start menu and clicking on the PyCharm Community Edition icon.

1. Creating a New Project: To start working with PyCharm, you can create a new Python project:
2. Click on "File" in the menu and select "New Project."
3. Choose the project location, specify the Python interpreter you want to use (if you have multiple Python versions installed), and select a project template.
4. Click "Create" to create your project.



Clone GitHub Repo

For reference

To clone a GitHub repository, follow these steps:

- Go to the GitHub website (<https://github.com>) and log in to your GitHub account.
- Locate the repository you want to clone. You can search for it using the GitHub search bar or navigate to the repository's URL.
- On the repository's main page, click the "Code" button, which is typically located near the top-right of the page.

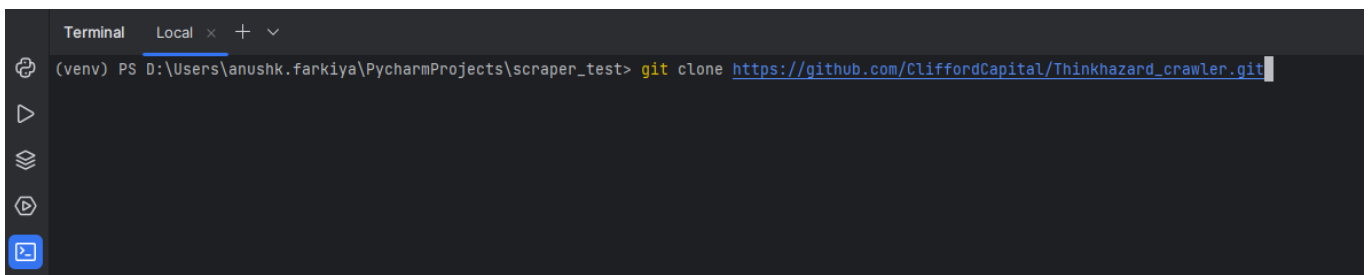
Login: manoj.bajaj@ccholdings.sg

Password: Clifford@123%

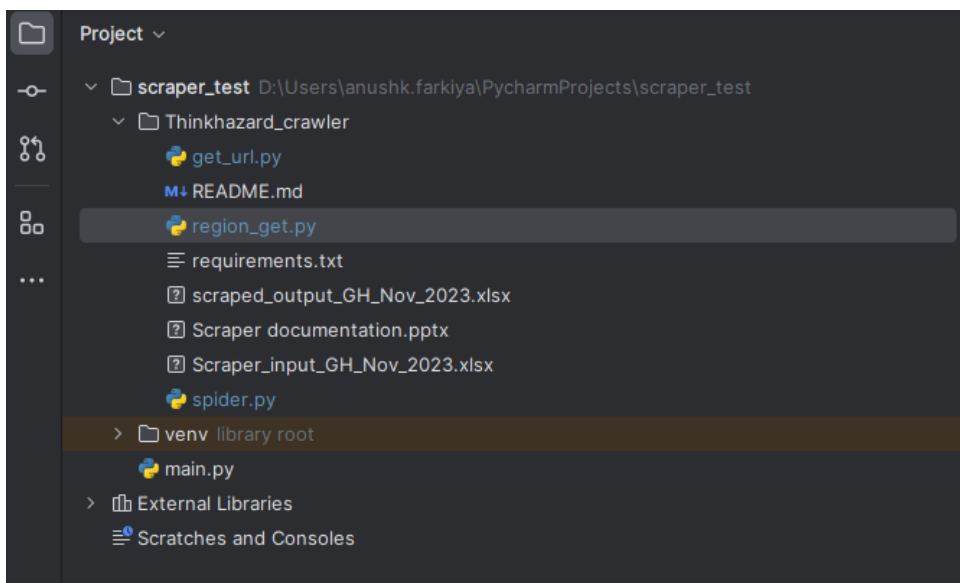
Implementation

In the pycharm terminal type

```
git clone https://github.com/CliffordCapital/Thinkhazard\_crawler.git
```

A screenshot of a PyCharm terminal window. The terminal title bar shows 'Terminal' and 'Local' tabs. The command prompt is '(venv) PS D:\Users\anushk.farkiya\PycharmProjects\scraper_test>'. The command entered is 'git clone https://github.com/CliffordCapital/Thinkhazard_crawler.git'. The command is partially executed, with the URL highlighted in blue and the cursor at the end of the line.

After cloning, the file structure should look like below:



About the files

These files and scripts are integral components of a data scraping and processing workflow. Users can use the provided scripts to gather information from the thinkhazard website, and the input and output Excel files serve as the data sources and results of these scraping operations. The optional "Region_get.py" script can further enhance the scraped data by extracting high-level region and country information from the URLs. The time estimates provided give users an idea of the approximate duration for each script to complete its respective task.

1.Get_url.py

- **Purpose:** This Python script is used to scrape specific URLs of regions from the thinkhazard website.
- **Input:** The script takes the "Region_granular" as input. This input likely refers to a specific region or location.
- **Output:** The script provides the scraped URL(s) as output.
- **Implementation Time:** It typically takes approximately 1 hour to process 100 values. So best to do in batches.

2.Spider.py

- **Purpose:** This Python script is designed to scrape data related to 11 categories of hazards for each "region_granular."
- **Input:** It requires "region_urls" as input, which likely represents the URLs of the specific regions.
- **Output:** The script generates scraped data for the 11 hazard categories as output.
- **Implementation Time:** It typically takes around 1.5 hours to process 35,000 values.

About the files

3. Region_get.py (optional)

- **Purpose:** This optional Python script can be used to scrape high-level region and country information from a given URL.
- **Input:** It takes a URL as input.
- **Output:** The script provides the scraped values, which include region granular, region high level, and country information.

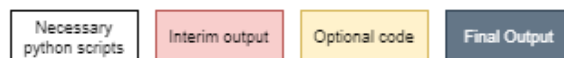
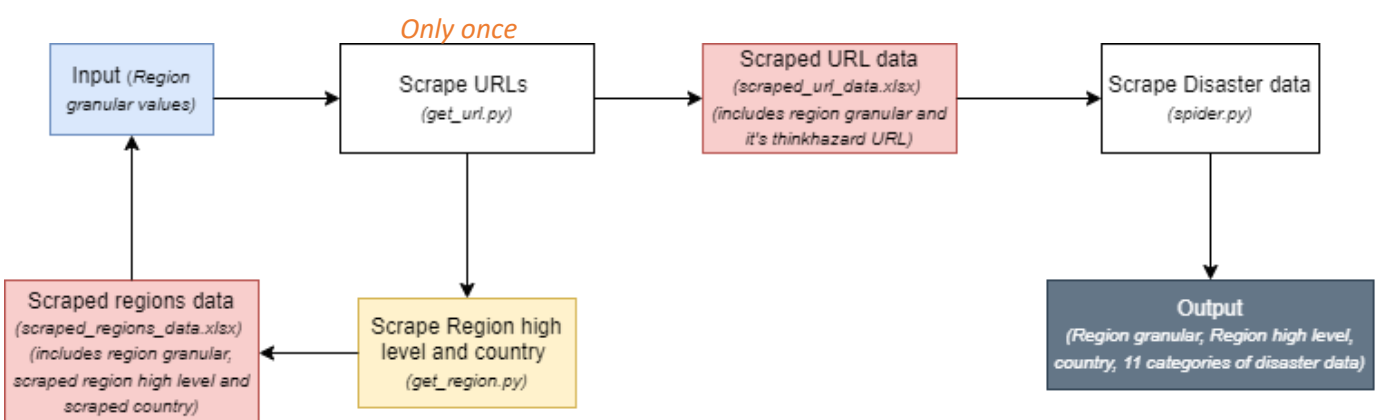
4. Scraped_input_GH_Nov_2023.xlsx

- **Description:** This is an input file that the above scripts use to perform their scraping tasks. It contains the data necessary for the scripts to run.

5. Scraped_output_GH_Nov_2023.xlsx

- **Description:** This file is the latest output generated by the scripts based on the inputs provided. It contains the scraped data, such as URLs, hazard categories, and additional information

Workflow



Running the files

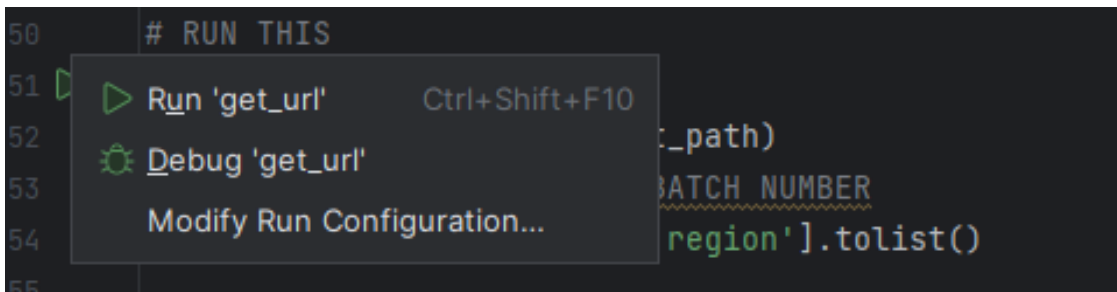
Opening a Project in PyCharm

If you haven't already, you need to open a Python project in PyCharm. Follow these steps:

- a. Launch PyCharm.
- b. Click "Open" or "Open Project" and navigate to the directory where your Python project is located.
- c. Select the project folder and click "Open."

Running Python Files: Now, let's run a Python file within your project.

- a. In the Project Explorer on the left side, locate the Python file you want to run.
- b. Right-click on the file, then select "Run" from the context menu. Alternatively, you can open the file and click the green "Run" button in the top-right corner of the editor.
- c. PyCharm will execute the Python script, and the output will be displayed in the "Run" tab at the bottom of the PyCharm window.



Debugging

1. **Check Inputs:** Ensure that the "Region_granular" input is correctly provided. Verify that the input format matches the script's expectations. Please follow the input excel file as example and do not change the column name.
2. **Print Statements:** Insert print statements at key points in your code to display variables' values and verify that they are as expected.
 - For e.g. `print(var_x)` will print that variable

Reusable code for other scraping tasks

This provided code is a Python script for web scraping using Selenium, a web automation and testing framework. The script is specific to a particular website's structure (<https://thinkhazard.org/en/>). To adapt this script for use on a website with a similar structure, you need to modify the HTML elements and XPath expressions to match the target website's structure.

Here's how you can make the necessary changes to make this code work on a similar website:

- 1. Inspect the Target Website:** First, you need to inspect the HTML structure of the target website using browser developer tools. This will help you identify the elements and XPath expressions you need to interact with.
- 2. Modify Base URL:** Update the `base_url` variable to the URL of the target website.
- 3. Identify Elements:** You'll need to find the corresponding HTML elements on the target website that are similar to the ones in the provided code. This includes buttons, input fields, and other elements you want to interact with.
- 4. Change XPath Expressions:** Update the XPath expressions to match the structure of the elements on the target website. XPath expressions are used to locate elements in the HTML document. You can modify the XPath expressions to target the elements you want to interact with.

Reusable code for other scraping tasks

```
17 def scrape_region(region, blocked):
18     try:
19         options = Options()
20         options.add_argument("--no-sandbox")
21         options.add_argument("--disable-dev-shm-usage")
22         options.add_argument("--headless")
23
24         service = Service()
25         driver = webdriver.Chrome(service=service, options=options)
26
27         # Replace 'YOUR_BASE_URL' with the actual base URL
28         base_url = 'https://thinkhazard.org/en/'
29         driver.get(base_url)
30         driver.find_element(By.XPATH, value: '//*[@id="myModal"]/div/div/div[2]/button[2]').click()
31         driver.find_element(By.XPATH, value: '/html/body/div[2]/div/form/span[2]/input[2]').send_keys(region)
32         driver.implicitly_wait(3)
33         driver.find_element(By.XPATH, value: '//*[@id="search"]/span[2]/div/div/div[1]').click() #GET THE FIRST DROP DOWN
34         #driver.find_element(By.XPATH, '//*[@id="search"]/span[2]/div/div/div[2]').click() # GET THE SECOND DROP DOWN
35         driver.implicitly_wait(3)
```

Base URL

Find the
search bar

Type values

Click on
first drop
down

Click on
second
drop down

End