



亞洲大學
ASIA UNIVERSITY

Midterm Project Report

Advanced Computer Programming

Student Name : Chinzorig Battulga

Student ID : 113021191

Teacher : DINH-TRUNG VU

2025-04

Chapter 1 Introduction

1.1 Github

- 1) **Personal Github Account:** <https://github.com/Mimikooo>
- 2) **Group Project Repository:** <https://github.com/Jantsagdorj/ACP-AU-1132>

The objective of this project was to create a web scraper using Python's Scrapy framework to extract structured information from a GitHub profile. This tool is particularly useful for developers and researchers who wish to analyze repositories at scale without manually copying data. The main goal was to collect information from each public repository under a specific user, including its URL, description (About), last updated timestamp, programming languages used, and number of commits.

1.2 Overview

To complete this project, the following advanced Python programming tools and libraries were used:

- **Scrapy:** The core framework used for web scraping. It allowed for easy spider creation, request handling, and HTML data extraction.
- **cssselect:** Used in conjunction with Scrapy to define CSS selectors and extract specific elements from the GitHub HTML structure.
- **Feed exporter:** A Scrapy built-in tool that outputs scraped data in structured formats, such as XML, JSON, and CSV. For this project, XML was chosen.

The scraper navigates from the GitHub profile page (<https://github.com/Mimikooo?tab=repositories>) to each listed repository, gathering relevant data. The collected data is exported to an XML file called repos.xml, which contains detailed information for further use.

Implementation

1.1 Setup and Environment

1.1.1 Environment Information

- Operating System: Windows 11
- Python Version: 3.12
- Scrapy Version: 2.12.0
- IDE/Text Editor: VS Code and PowerShell Terminal

1.1.2 Installing Scrapy Functions and creating the project

To install Scrapy, I used pip in the PowerShell terminal:

```
PS C:\Users\chinz> pip install scrapy
Defaulting to user installation because no
```

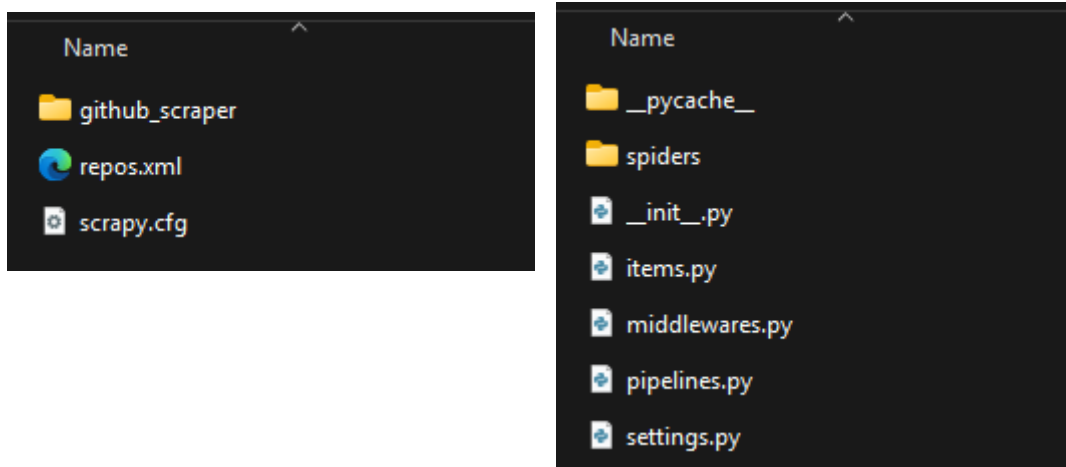
After installation, I verified that Scrapy was working:

```
PS C:\Users\chinz\OneDrive\Documents\Advanced Computer - Midterm\github_scraper> python -m scrapy -v
Scrapy 2.12.0 - active project: github_scraper
```

I created a new Scrapy project named **github_scraper** with the following command:

```
PS C:\Users\chinz\OneDrive\Documents\Advanced Computer - Midterm> python -m scrapy startproject github_scraper
New Scrapy project 'github_scraper', using template directory 'C:\Users\chinz\AppData\Local\Packages\PythonSoftwareFound
ation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\site-packages\scrapy\templates\project', created in:
C:\Users\chinz\OneDrive\Documents\Advanced Computer - Midterm\github_scraper
```

This generated the necessary folder structure:



1.2 Class

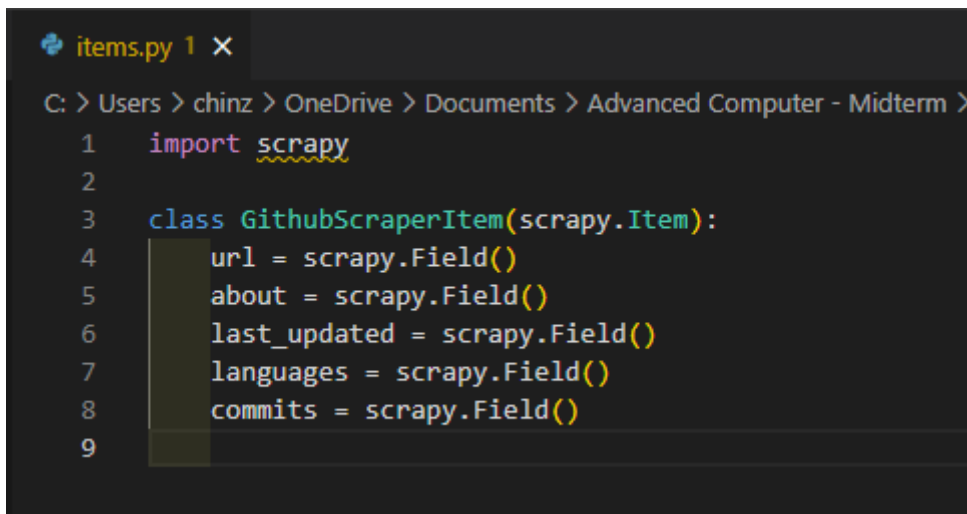
Class: GithubScraperItem (items.py)

Description: Defines the fields that will be scraped and stored for each repository.

This item acts like a data structure.

Fields:

- url: Complete URL to the GitHub repository
- about: Description or purpose of the repository. If not provided and the repository is not empty, defaults to the repository name.
- last_updated: Datetime string showing the latest update timestamp
- languages: List of programming languages used in the repository (if any)
- commits: Total number of commits in the repository (if available)



```
items.py 1 X
C: > Users > chinz > OneDrive > Documents > Advanced Computer - Midterm >
1  import scrapy
2
3  class GithubScraperItem(scrapy.Item):
4      url = scrapy.Field()
5      about = scrapy.Field()
6      last_updated = scrapy.Field()
7      languages = scrapy.Field()
8      commits = scrapy.Field()
9
```

1.3 Spider

Spider: github_spider.py

Description: Scrapy spider that starts from the repositories page and navigates into each repo to extract information.

Method: parse()

Navigates through the repositories listed on the user's GitHub page and follows each repository link.

Method: parse_repo(response)

Scrapes details of each repository:

- Checks if the repo is empty
- Extracts about, last_updated, languages, and commits (if available)
- Yields structured GithubScraperItem

```
C: > Users > chinz > OneDrive > Documents > Advanced Computer - Midterm > github_scraper > github_scraper > spiders > github_spider.py > GithubSpider > parse_repo

1 import scrapy
2 import requests
3 from github_scraper.items import GithubScraperItem
4
5 class GithubSpider(scrapy.Spider):
6     name = "github"
7     allowed_domains = ["github.com"]
8     start_urls = ["https://github.com/Mimikooo?tab=repositories"]
9
10    def start_requests(self):
11        for url in self.start_urls:
12            yield scrapy.Request(url=url, callback=self.parse, headers={
13                'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36'
14            })
15
16    def parse(self, response):
17        repo_links = response.css('h3 a::attr(href)').getall()
18        for link in repo_links:
19            repo_url = response.urljoin(link)
20            yield scrapy.Request(url=repo_url, callback=self.parse_repo, headers={
21                'User-Agent': 'Mozilla/5.0'
22            })
23
24    def parse_repo(self, response):
25        item = GithubScraperItem()
26        item['url'] = response.url
27
28        about = response.css('p.f4::text, p.f4 span::text').get()
29        about = about.strip() if about else None
30        is_empty = response.css('.blankslate').get() is not None
31
32        repo_name = response.url.split('/')[-1]
33        owner = response.url.split('/')[2]
34        item['about'] = about if about else (None if is_empty else repo_name)
35
36        headers = {
37            'Accept': 'application/vnd.github.v3+json',
38            'User-Agent': 'Mozilla/5.0',
39        }
40
41        # 1. Get repo details from API
42        repo_api_url = f"https://api.github.com/repos/{owner}/{repo_name}"
43        repo_data = requests.get(repo_api_url, headers=headers).json()
44
45        # 2. last_updated
46        item['last_updated'] = repo_data.get('updated_at')
```

```

C: > Users > chinz > OneDrive > Documents > Advanced Computer - Midterm > github_scraper > github_scraper > spiders > github_spider.py > GithubSpider > parse_repo

1 import scrapy
2 import requests
3 from github_scraper.items import GithubScraperItem
4
5 class GithubSpider(scrapy.Spider):
6     name = "github"
7     allowed_domains = ["github.com"]
8     start_urls = ["https://github.com/Mimikooo?tab=repositories"]
9
10     Tabnine | Edit | Test | Explain | Document
11     def start_requests(self):
12         for url in self.start_urls:
13             yield scrapy.Request(url=url, callback=self.parse, headers={
14                 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36'
15             })
16
17     Tabnine | Edit | Test | Explain | Document
18     def parse(self, response):
19         repo_links = response.css('h3 a::attr(href)').getall()
20         for link in repo_links:
21             repo_url = response.urljoin(link)
22             yield scrapy.Request(url=repo_url, callback=self.parse_repo, headers={
23                 'User-Agent': 'Mozilla/5.0'
24             })
25
26     Tabnine | Edit | Test | Explain | Document
27     def parse_repo(self, response):
28         item = GithubScraperItem()
29         item['url'] = response.url
30
31         about = response.css('p.f4::text, p.f4 span::text').get()
32         about = about.strip() if about else None
33         is_empty = response.css('.blankslate').get() is not None
34
35         repo_name = response.url.split('/')[-1]
36         owner = response.url.split('/')[2]
37         item['about'] = about if about else (None if is_empty else repo_name)
38
39         headers = {
40             'Accept': 'application/vnd.github.v3+json',
41             'User-Agent': 'Mozilla/5.0',
42         }
43
44         # 1. Get repo details from API
45         repo_api_url = f"https://api.github.com/repos/{owner}/{repo_name}"
46         repo_data = requests.get(repo_api_url, headers=headers).json()
47
48         # 2. last updated
49         item['last_updated'] = repo_data.get('updated_at')

```

Chapter 2 Results

1.1 Results

Result 1: The spider successfully scraped the public repository [Test](#). Since this repository is likely empty or minimal, some fields returned None.

Output (repos.xml):

```
<item>
  <url>https://github.com/Mimikooo/Test</url>
  <about>Test</about>
  <last_updated>None</last_updated>
  <languages>None</languages>
  <commits>None</commits>
</item>
```

Result 2: This time the repository is not empty.

	A	B	C	D	E	F	G	H
1	about	commits	languages	last_upda	url			
2	This is the about section I just added at 10:49AM				https://github.com/Mimikooo/Test			
3	about	commits	languages	last_upda	url			
4	This is the about section I just added at 10:49AM				https://github.com/Mimikooo/Test			
5	about	commits	languages	last_upda	url			
6	This is the about section I just added at 10:49AM				https://github.com/Mimikooo/Test			
7	about	commits	languages	last_upda	url			
8	This is the about section I just added at 10:49AM				https://github.com/Mimikooo/Test			
9	about	commits	languages	last_upda	url			
10	This is the about	7			https://github.com/Mimikooo/Test			
11	about	commits	languages	last_upda	url			
12	This is the about	8			https://github.com/Mimikooo/Test			
13	about	commits	languages	last_upda	url			
14	This is the about	8	Python	2025-04-1	https://github.com/Mimikooo/Test			
15								

Chapter 2 Conclusions

This project successfully demonstrated how to use Scrapy for structured web scraping. It revealed both the strengths of Scrapy (clean scraping pipeline, robust data structuring) and limitations (handling dynamic JavaScript content like GitHub's commit counters and languages). For future improvements:

- Using the GitHub API would allow access to complete repository metadata including commits, contributors, and languages.
- Integrating Selenium could allow scraping of JavaScript-rendered pages.
- Output can be extended to other formats like CSV or JSON for easier analysis.

Overall, the project built a solid foundation in scraping and automation with Python.