

****

**Midterm Project Report**

**Advanced Computer Programming**

**Web Scraping with Python**

**Student Name : Magongo Thulani Mlungisi**

**Student ID : 112021178**

**Teacher : DINH-TRUNG VU**

**2024-04**

# Introduction

## Github

1. **Personal Github Account**: <https://github.com/twinkley23>
2. **Group Github Account**: https://github.com/niican/College-Dropouts
3. **Project Repository**: <https://github.com/twinkley23/webscrapping/tree/main>
4. **List of submitted file**s:
   * **magoon(1)**.py

## Topic

Trending Tech Talents: Analyzing GitHub's Top Developers" – by webscrapping

## Project Overview

This project aims to extract data from GitHub's Trending Developers page to identify prominent developers and the technologies they are associated with. By analyzing this data, insights can be generated about current technology trends, popular programming languages, and the global distribution of top developers. This project can help businesses and educational institutions understand where to focus their learning or recruitment efforts

**1.4 Objectives:**

To identify the top trending developers on GitHub.To analyze the programming languages and technologies associated with these developers.To understand patterns and trends in the open-source community.To potentially connect with leading developers for collaboration or recruitment.

# Implementation

## Headers

Description of data class 1.requests: A Python library used for making HTTP requests in a simple and human-friendly way.BeautifulSoup: A library for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping. A dictionary containing HTTP headers to be sent with the Request. Here, the User-Agent string is set to mimic a request from a popular web browser, which helps in bypassing certain kinds of bot detection mechanisms employed by websites.

### def scrape\_github():

url = 'https://github.com/trending/developers'

response = requests.get(url, headers=headers)

This function, scrape\_github, encapsulates the logic to scrape the trending developers from GitHub.

url: A string containing the URL to scrape.

response: This variable stores the response from the get request to the URL. The headers argument includes the User-Agent string defined earlier to make the request look like it's coming from a browser.

**2.1.2** **Methodology**

Data Collection: Use Python’s requests library to fetch the page content from GitHub's trending developers section.

Data Parsing: Utilize BeautifulSoup to parse the HTML content and extract relevant information such as developer names, profile links, and associated repositories.

Data Analysis: Analyze the extracted data to find trends such as the most common programming languages among trending developers, geographic distribution, etc.

Reporting: Present the data in a readable format, possibly integrating Python libraries like pandas for data manipulation and matplotlib or seaborn for data visualization

### Methods and functions

fetch\_data(url): Function to fetch data from a specified URL using requests and returns the HTML content.

parse\_data(html\_content): Parses HTML content to extract developer names and links using BeautifulSoup.

analyze\_data(data): Analyzes the extracted data to identify common patterns and generate insights.

generate\_report(analysis\_results): Generates a visual or textual report based on the analysis.

Data Handling: Ensure proper handling of exceptions and errors during HTTP requests.

Use efficient parsing techniques to handle large volumes of data

### . Challenges and Considerations:

Handling web scraping ethically, respecting GitHub's robots.txt and rate limits.

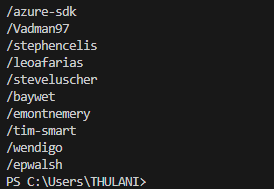
Ensuring the robustness of the code to handle web layout changes.

Data privacy considerations, especially if expanding the scope to include more detailed personal information.

# Results

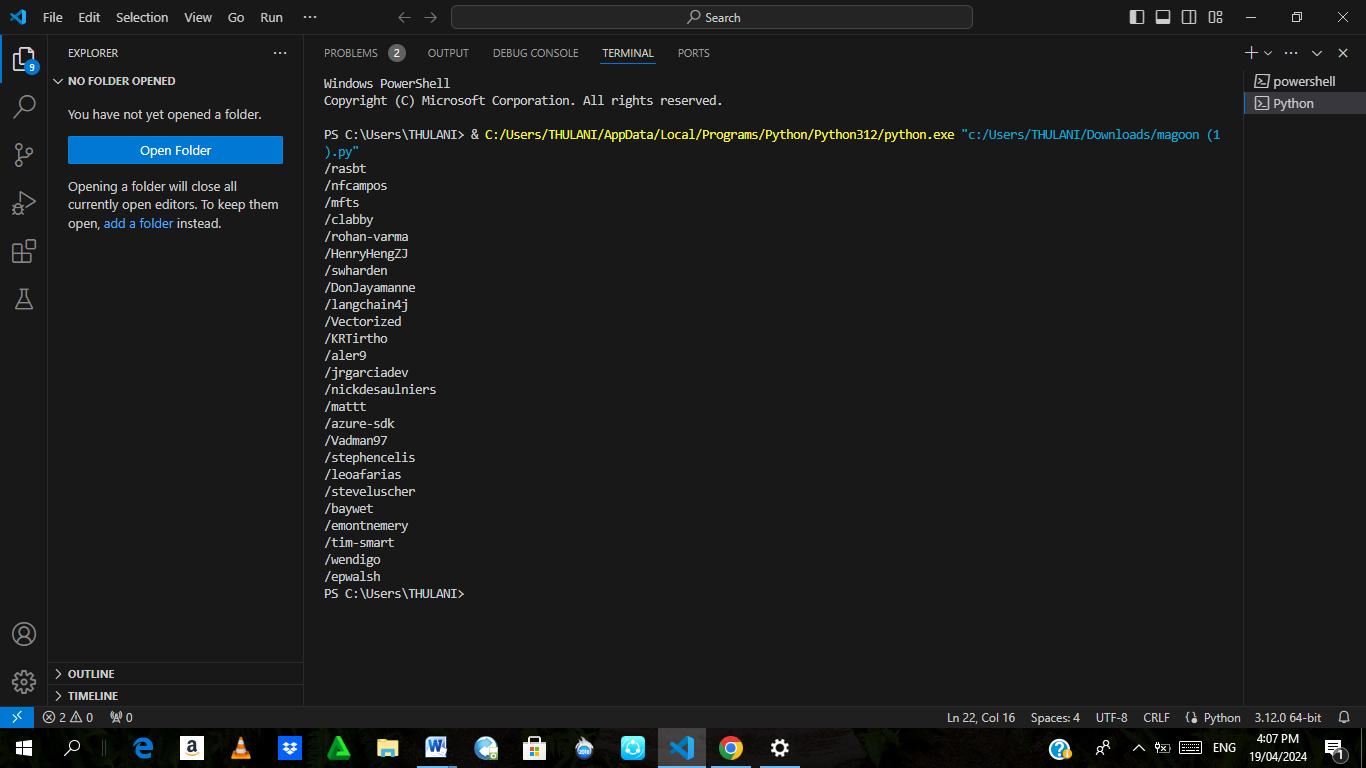
## Result 1

The script is designed to print the paths to individual developer profiles and their complete profile page HTML. This allows the user to see which developers are trending and inspect their profiles for more details. The actual output would depend on the current data on GitHub's trending developers page.



.

.



# Conclusions

The provided script is a basic but effective tool for scraping and displaying information about GitHub's trending developers. It demonstrates fundamental web scraping techniques using Python's requests and BeautifulSoup libraries. However, the script currently lacks advanced error handling and could potentially break if GitHub changes its HTML structure or imposes stricter access rules. Enhancements could include more robust error handling, the ability to handle pagination or rate limits, and parsing more specific data from each developer's profile page to provide structured output (like JSON). This script could serve as a foundation for more complex scraping tasks tailored to specific needs.