Web Scraping and Cracking the Web

Wouldn’t it be so much easier to crack passwords if you didn’t have to enter them all manually? And wouldn’t it be nice if there was some way to automate going to websites and trying emails and passwords?

Or what about that GPU you wanted to buy before the scalpers took them all? Those Taylor Swift concert tickets? If only there was some way to automatically do these things…

**Goal:**   
You have some options with this project. You can create a program that automatically tries various email/password combinations on a website to login. Another option is creating a web scraper to automatically gather information for you from a website, notify you, and (possibly) *act* on it.

**Note:**The “how” or “implementation” of this project is intentionally vague. Example of different things you can do with a web scraper are given below, however, you cannot use any of the example projects. Additionally, the types of web Scrapers given below are not the only types that exist. Be creative, research, and think of something you might actually find useful. You are encouraged to look up sources on how to do things – Google and StackOverflow are your friends. You are the architect of this program. I encourage you to think first, plan a lot, and code last. Feel free to use any programming language for this project and import any necessary packages.

**Examples**

1. Compile Information: Compile information about available items and cost on IMDB or Amazon

2. Extract information: Pull and rewrite text from articles that ignores adds to improve readability

3. Create Information: Pull data from weather service and calculate average temperature for a specific time period

3. Website Cracker: create code to brute force hack into someone’s Facebook account.

4. Automate Website Navigation: Order a pizza from Mod Pizza using CLI

5. Automate Tedious Tasks: Delete emails in gmail with a keyword

6. Scalper: Create code that refreshes to check availability of rare items on Amazon and then updates user or adds them to the cart.

7. Other: There are so many more options, but I want y’all to think and be creative so I’m going to stop putting examples now. But consider places that frequently see bots or captcha to prevent bots. Why are there bots being used or created? What reward are they getting for the trouble?

**Extensions**

* Automatically sends alerts (email, text) when some alert is triggered
* Program automatically runs on a set schedule
* Create own custom server with database of valid user accounts that is part of demonstration

Have an idea? Ask!

**Requirements/Rubric**

|  |  |
| --- | --- |
| Task | Points |
| Writeup:   * Research * Works Cited (MLA/APA) | 8  2 |
| Presentation:   * Well-Designed Slides * Engaging (does not read off slides, looks at audience, inflective voice) * Responds to questions well | 5  10  5 |
| Program:   * Version Control (regular usage of git) * Demonstration (program meets requirements) * Run via command line and add arguments | 4  10  4 |
| README.md   * Dependencies * Commands and arguments to run | 1  1 |
| Extensions:   * See later in document | +2/each |
| Penalties:   * Spelling, Punctuation, Grammar, Capitalization * Not following programming conventions (modular code, documentation, comments) | - 1/each |
| TOTAL | / 50 |

**Presentation:**

**Directions:** At least one slide must be present to address the following. You can use more than one slide for each if needed.

**1. Title Slide:** Project Name, Title, and developer (your name)

**2. Goal of Program and Why:** What is the program supposed to do and why did your team choose to do this?

**3. Website Used:** What website did you use? What is the purpose of this website? Why did you choose this website?

**4. Design Process:** What did you research, what was your step-by-step process, what changes did you make?

**5. Programs Used:** What important programs or libraries you used. Describe them and why you used them.

**5. Difficulties and Changes:** What were some difficulties you found and how did you address/change them. What are some limitations of your program?

**6. Code:** What language or languages did you use? Why did you use them? Place snips of important code pictures and describe what they are doing and why. (Basically break down the code, but be more prepared for it then we usually are)

**7. Demonstrations:** an embedded VIDEO of your program running. Ensure output and input are clearly visible. (We’ve wasted a lot of time in presentations trying to show our programs running, use a screen recorder such as power point to show your program running). If your program takes longer than a minute to run, use a video editor to edit and create a video that has only the following; input → 30 second or less snippet of running time → Output

**8. Project Reflections:** What did you get out of this project either through knowledge or practice?

**9. Class Reflection:** What did you get out of other projects? What was your favorite, most useful, project? What did LaFoy do well this year? What should he change? Was the class worthwhile. (“I actually want the feedback, don’t be a glazer just to look good)

**10. Works Cited and Questions:** Show works cited

**11. Questions:** slide to prompt questions from classmates

**Writeup:**

Research your chosen topic.

Are you creating a password cracker for a website that will try out multiple combinations of passwords? How do you interact with a website via a program? Or are you creating your own scalping bot? What programming language and packages do you use? Ran into a roadblock? What research have you done to investigate overcoming or circumventing this roadblock?

Anything you attempt to try should have research to back it up BEFORE you try it.

As a general guideline, this should be a minimum of one page, 12-pt font, Times New Roman. You should have at least 5 sources.

**Testing Your Code and Important Considerations**

* You may need to use a virtual machine to get access to “Inspect element”.
* Before you try this out and accidentally commit a DoS attack, make sure you set a sleep/wait time between your requests.
* Make sure you read the robots.txt file to see what is allowed.
* Captcha got you flummoxed? Research or choose a different site.
* **ONLY ATTEMPT CRACKING YOUR OWN PASSWORD**

**Scaffolded Project**

1. **Decide:** Are you going to try to crack passwords or are you going to do something involving web scraping? Do some research, look around, tinker, then decide.
2. **Git and GitHub:** Track your project from the very beginning using version control. I recommend git. GitHub allows you to work on the program from multiple computers and between multiple people. Look up how to write good git commit messages.
3. **Password Cracking Route (considerations): ONLY ATTEMPT CRACKING YOUR OWN PASSWORD.** Find a site, encryption mechanism?, CAPTCHA?, rate-limiting password attempts?
4. **Web-scraping Route (considerations):** Find a site, robots.txt?, CAPTCHA?, rate-limiting?, requires login?
5. **Document:** Keep a document of what you try (and maybe why you tried that) and the outcome. This document will help you when you look back at what may have worked or did not work.