EasyA - Programmer's Documentation
Group 4
February 4, 2023
Author: Juan Rios

1. Introduction

The Programmer's Documentation describes the important aspects of how the EasyA system functions at a high level. This provides any person(s) who wants to make future modifications to the source code with the guidance to do so.

The following document will provide an overview of the source files. Along with this, it provides a rundown of the main functions in the code for scraping data, the GUI, and graph generation.

All code is written in Python 3.8+.

2. EasyA Source Files

2.1. Data Scraper

2.1.1. scrape_data.py

- A script that sets up class_database.json by scraping the .js data file linked in the SRS and the regular faculty names from the UO's old website on the wayback machine.

2.1.2. class_database.json

- A .json file that serves as a database for the UO class records scraped in scrape_data.py.

2.1.3. regular_faculty_names.txt

- A .txt file that has the name of each regular faculty member in UO's "Natural Sciences" departments from the old UO website on the wayback machine, scraped by scrape_data.py, one name per line.

2.2. GUI

2.2.1. easyA.py

- Holds the GUI code for user interaction in the EasyA system. IT allows input from both Student and System Admin users.

2.2.2. create_graph.py

- Called by the GUI file, *easyA.py*, when a Student user clicks the *Generate Graph* button on the Student User Page. The function will collect the data from this page and then display it on a graph.

3. Data Scraping Functions

Functions can be found under scrape_data.py.

3.1. extract_json(text)

- Removes the outer brackets in the *text* file (original .js file) that is provided for grade data so the file can be processed as JSON.

3.2. get_class_dict()

- Uses *requests.get()* to load the webpage containing the grade data. It then uses *BeautifulSoup* to parse the webpage. Finally, *extract_json()* formats the data so that it can be returned as a Python dictionary.

3.3. get_faculty_names()

- Uses *BeautifulSoup* to scrape the links for the departments in the Natural Sciences from *web.archive.org*. It then calls *scrape_faculty_names()* for each department to get the list of regular faculty in that department.

3.4. scrape_faculty_names(dept_page_url)

- Uses *BeautifulSoup* to scrape *dept_page_url* for a list of all regular faculty teaching in that department.

3.5. add_regular_faculty_to_db(faculty_file_name, databse_file_name)

- Reads the given *faculty_file_name* (regular_faculty_names.txt) file and updates the *databse_file_name* (class_database.json) database with the boolean *fregular* to indicate which instructors are regular faculty (True) or not regular faculty (False).

4. GUI Functions

Functions can be found under easyA.py. All are contained in the GUI class. The GUI was created with tkinter.

4.1. __init__(self)

- Constructor for the GUI class. Initializes the Welcome Screen and creates frames for the Landing Page, Student User Page, and SysAdmin User Page.

4.2. generateGraph()

- Creates the grade data graph by calling the *class_search()* function imported from *create_graph.py*. Also holds error checking that requires the user to input a department and a course level and/or specific course number.

4.3. studentPage(self)

- Loads up the Student User Page and creates frames for the Course Selection and Graph Filter sections. Also adds functionality to the Generate Graph and Back buttons.

4.4. enter_data(self)

- Takes in a JSON file that contains new grade data as input from the user. Uses *json.load()* to convert the file and use the data to either add to or modify *class_databse.json*.

4.5. adminPage(self)

- Loads up the System Admin User Page and creates a frame for the Data Entry section. Adds functionality to the Enter Data button so an admin can upload data.

4.6. LandingPage(self)

- Loads the Landing Page of the GUI and displays buttons to allow for user selection. Displays the citations of where the data is from and why the data might be limited.

5. Graph Functions

Functions can be found under create_graph.py.

5.1. class_search(code, aorf, faculty, num_classes)

- Takes input from the *generateGraph()* function from *easyA.py* file to generate and display the grade data graph. The parameters are as follows:
  - *code* (int): The number of integers in this parameter indicates the user's choice of graph. A size of 1 means the system searches for all courses in a given department. A size of 0 or 3 tells the system to search for each course in a given class code.
  - *aorf (*bool): If True, display percentage of D's or F's. Otherwise, the percentage of A's is shown.
  - *faculty* (bool): Checks the *fregular* boolean from *scrape_data.py*. If True, only fetch grade data for regular faculty. Otherwise, show grade data for all instructors.
  - *num_classes* (bool): If True, show the number of classes taught by the instructor and add it in parenthesis next to their name. Otherwise, do nothing.

6. Major Data Structures

The primary data structures that were used in this system were JSON objects and Python dictionaries. Both were useful in handling the large sets of data that were initially provided for the system. Once converted from the .json database to an internal Python data structure, the exact structure of our class data within Python is a: dictionary where class codes e.g. "ART111" are keys, whose values are lists of dictionaries, where each dictionary list element corresponds to a single class for a single term, and has keys that correspond to class attributes such as "instructor", "aperc", etc.

7. Known Issues

A recurring error is given by the requests module when scraping from the web archive page for the regular faculty in a department. We figured this was due to the web archive site blocking the continuous requests from the scraper. A simple workaround for this was to temporarily suspend the execution of the program for one to three seconds with *time.sleep()*. Implementing *time.sleep()* helped to reduce most of these errors, but a few of the department pages still block the scraper.

8. Acknowledgements

EasyA was created by Alder French, Blake Skelly, Connor MacLachlan, Garrett Bunkers, and Juan Rios at the University of Oregon for the course CS 422: Software Methodologies, taught by Anthony Hornof.