# EasyA
# Programmer Documentation

Spike Chen (sc), Joey Le (jl), Andrew Liu (al), Peter Nelson (pn), Krishna Patel (kp) - 2-05-2023 - v1.0

## Table of Contents:



## 1. Programmer Documentation Revision History

| Date | Author | Description |
|---|---|---|
| 2-2-2023 | kp | Created the initial document. |
| 2-2-2023 | pn | Added scraper.py information |
| 2-4-2023 | kp | Added information for parser.py, search.py, and addData.py |
| 2-5-2023 | jl | Added information for graphGen.py |
| 2-5-2023 | pn | Added information for scraper.py |
| 2-5-2023 | sc | Added information for test_search.py |

## 2. Main Program Files - Overview

All of the following files make up the entirety of the EasyA program, a description of each file is contained below.

1. EasyA.py: Module that drives the whole program. When it runs, it calls the student interface which takes over calls to the other modules.
2. Parser.py: Takes the .js file provided (after the code is removed leaving a JSON object) and converts it to a CSV file for use by the search module. Does not interact directly with the rest of the program other than generating a file.
3. StudentInterface.py: This file is responsible for controlling the user interface as well as interacting with the search and graph generation modules. It will take input from the user,

call the search module, and pass that response to the graph module to display relevant info to the user.

4. GraphGen.py: Interacts with StudentInterface.py in order to receive data to produce graphs. Once the graphs are generated they are sent back to StudentInterface.py for display.
5. Search.py: Interacts with StudentInterface.py by returning the grade data relevant to what the user requested.
6. addData.py: This file is designed to run independently from the EasyA program. It handles the use case where an administrator would want to replace the grade data provided in the system. This file will overwrite the existing data.
7. Scraper.py: This file is designed to scrape the names of faculty members from HTML files downloaded from the Wayback Machine web archive of the UO department websites. It will output a file containing names formatted as "first,last."

# 3. Main Files Breakdown

- ## 3.1 EasyA.py
  - ### 3.1.1
- ## 3.2 Parser.py
  - ### 3.2.1 formatLine (dataEntry, className)
    - Function that handles the concatenation of a line of CSV output. Indexes into the JSON object at the index passed by dataEntry to obtain the data.
  - ### 3.2.2 processFile (inputFile, outFile)
    - This function opens the JSON file and the output (CSV) file. The JSON file is loaded into a dictionary which then is parsed into a CSV line one class at a time by calling formatLine. The line is then written to the file.
- ## 3.3 StudentInterface.py
  - ### 3.3.1 StoredGraph
    - Encapsulating class for graphs and graph figures. Holds data for graphs necessary for rendering and managing tkinter frames and matplotlib figures.
      - **3.3.1.1 setGraph(options, faculty_only, x_label)**
        - This method loads the data for the graph into memory, without generating or rendering it, including whether the graph should display faculty only as well as the x-axis label
      - **3.3.1.2 pushGraph(dataframe, row, column, scale, max_width, max_height)**

        - This method renders the stored graph on the given frame with the given dimensions
      - **3.3.1.3 hardExit()**
        - This method completely removes this graph from memory and forcefully re-renders the parent frame.
      - **3.3.1.4 softExit()**
        - This method completely removes this graph from memory but does not forcibly re-render the parent frame.
  - **3.3.2 renderGraphs(stored_graphs, dataframe, max_width, max_height)**
    - A function that renders all graphs in a list onto the provided tkinter frame, capping itself at the given maximum dimensions.
  - **3.3.3 processInputAndRender(max_width, max_height, faculty_list , window, subject, input_number, search_type, search_by_a, faculty_only, stored_graphs)**
    - A function that takes and sanitizes all user input before passing it to the class/instructorSearch functions depending on the parameters provided. Outputs errors onto the GUI depending on relevant cases. If the search was successful, it uses a StoredGraph to store and generate the graph output, and renders all existing graphs onto the GUI.
  - **3.3.3 createDropdown(frame, options, row, column)**
    - A helper function to generate a tkinter dropdown element with the given parameters.
  - **3.3.3 createTextEntry(frame, text, row, column)**
    - A helper function to generate a tkinter text entry box, with a title/prompt to its left.
  - **3.3.4 createDropdown(frame, options, row, column)**
    - A helper function to generate a tkinter text entry box, with a dropdown to its left to signify input type.
  - **3.3.5 initGui()**
    - This function effectively serves as the entrypoint to the entire EasyA program. It initializes the GUI with its basic input options.
- **3.4 GraphGen.py**
  - **3.4.1 graphGen(data_dictionary, options_list, faculty_only, label)**

        - A function that creates a graph based on the data dictionary and display options provided for that graph.
    - **3.4.2 compileBarData(data, view_levels, view_as, view_count, view_only_faculty)**
        - A helper function for graphGen() that takes a provided dictionary and converts it to two lists to be used for a single graph.
    - **3.4.3 formatName(name)**
        - A helper function for compile_bar_data() that formats an instructor's full name to provide just the last name.

- **3.5 Search.py**
    - **3.5.1 buildMaps(filename, subjectMap, instructorMap)**
        - This function builds the dictionaries needed to perform the search to find the relevant grade data. It builds two maps, one for subjects that keep track of how many courses an instructor has taught, and one for keeping track of the grades an instructor has assigned for every course they have taught. To accomplish this the function opens a CSV file containing data and then reads through it, adding to the maps as it runs through the file.
    - **3.5.2 AddList (gradeList, data)**
        - This function takes two lists of grades and adds them to the existing one (gradeList). This is done so that they can be averaged later.
    - **3.5.3 getFacultyList (fileName)**
        - This function opens the file containing names of faculty produced by Scraper.py, sorts the names as "last, first", and adds them to a list of faculty in all lower case characters.
    - **3.5.4 isFaculty(instructorName, facultyList)**
        - Function returns a bool determining if instructorName is in facultyList.
    - **3.5.5 avgList(gradeList, count)**
        - This function averages each percentage in the gradeList by dividing the percentages over count. Count is the number of courses.
    - **3.5.6 instructorSearch(subject, level, classNum, facultyList)**
        - This function carries out the main search needed to find averages by instructor for a given level, subject, or course number. The subject is required and level and course number can be skipped if -1 is the value of the parameters when called. This function will return a dictionary containing names, grades, number of classes taught, and if an instructor is a faculty member.

- ### 3.5.7 classSearch(subject, level, facultyList, facultyLookup)
    - This function carries out the search needed to find averages of grades by course number in a subject. Both the level and subject are required otherwise an error will be returned. The function will return a dictionary containing course numbers, grades, number of classes the data represents, and whether or not the search was only for faculty. The bool in the return dictionary is to keep the format consistent for graph processing.

## 3.6 addData.py

- ### 3.6.1 replaceData(newDataFile,currentDataFile, newFacultyFile,currentFacultyFile)
    - This function goes through the new data and faculty file and writes them to the existing files.
- ### 3.6.2 userDataUpdate ()
    - This function drives the administrator use case. It will ask display requirements on the terminal, and ask the user for input. If the file user wants to read from is not valid, the user will be prompted again. Once all files are valid, the user will be warned that this operation cannot be undone. Once the user confirms, it will call replaceData and finish the replacement of the data in the EasyA system.
- ### 3.6.3 clearTerminal()
    - This function clears the screen after lots of text is displayed to improve readability.

## 3.7 Scraper.py

- ### 3.7.1 scraper.py
    - This function opens html files and uses them to generate a CSV text file of faculty names in abbreviated or full name format.

## 3.8 test_search.py

- ### 3.8.1 search_faculty()
    - In this class, it calls the function instructorSearch() from the search.py file to test if the instructorSearch() gets the right return which can be used for the EasyA.py to draw the graph.
- ### 3.8.2 Search_professor_inlist()
    - In this class, it is also for the function instructorSearch() from the search.py file. Since the the instructorSearch() need to check that if the instructor is in the faculty list, for easily to see it, in this class just use the one instructor instead of the faculty list as the input to see, if it just this

instructor is true and other instructor for this class are false that means this part is success.

- ### 3.8.3 classSearch()
  - In this class, it calls the function classSearch() from the search.py file to see if the classSearch() gets the right return. Two functions in this class are for testing if the user wants to search the class in the same level and same subject, if the return is divided by the faculty or non faculty.
- ### 3.8.4 search_error()
  - In this class, it is for the search.py, because the search.py should read the user's input and use the input to do the class search or the instructor search. This class is to test if the user gets the wrong input, thesearch.py can return the right error number back, and that to let the user know, where is wrong for the user's input.