[Hanna Abrahem](mailto:habrahem@haverford.edu), [Fejiro Anigboro](mailto:fanigboro@haverford.edu), [Mohammad Fanous](mailto:mfanous@haverford.edu), [Phoebe Hulbert](mailto:phulbert@haverford.edu)

Jeová Farias Sales Rocha Neto

CS105

December 16th, 2022

Project Report

**Introduction**:

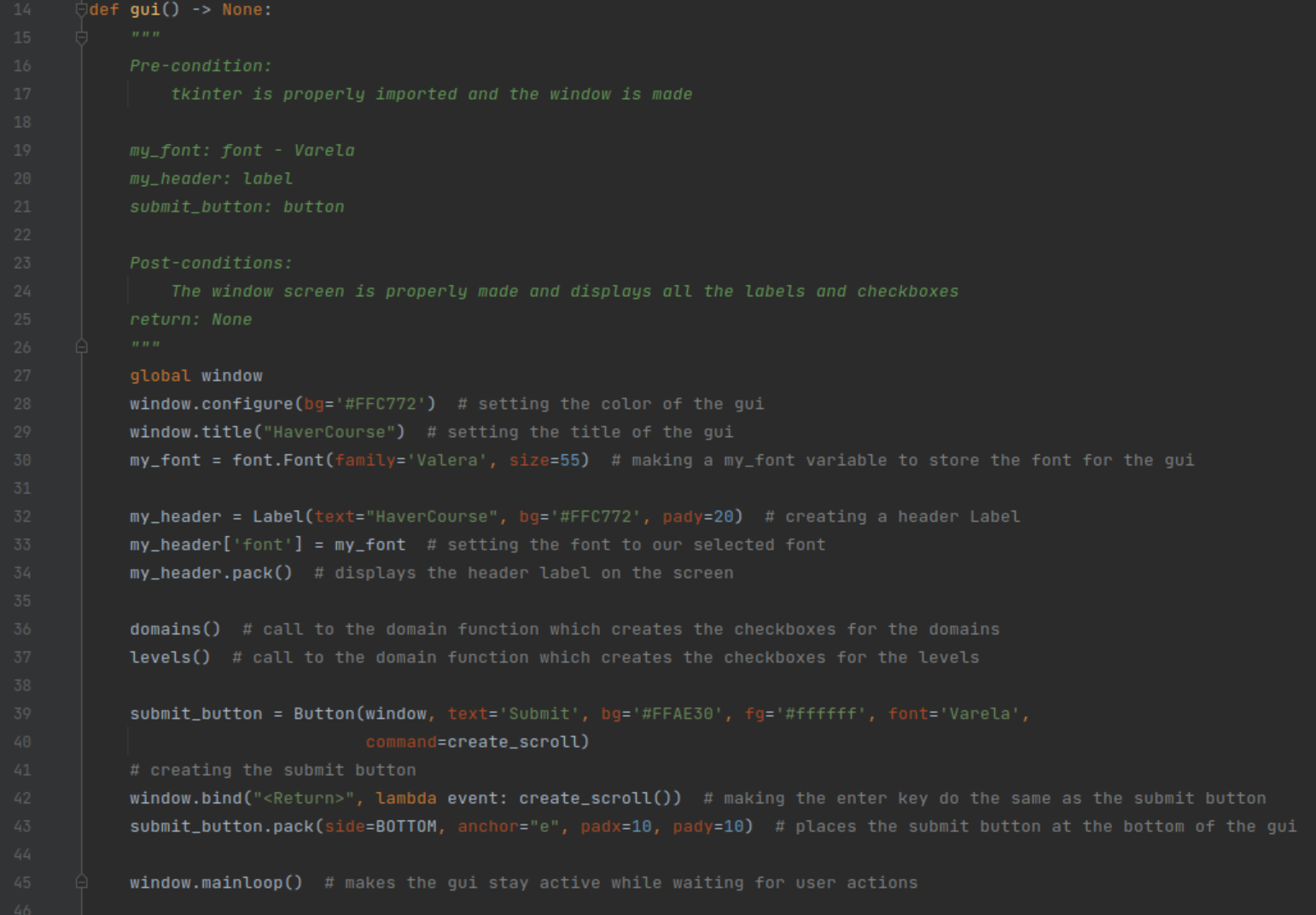
For our project, we wanted to create a program that would allow students to input their interests (departments, domains, professors, etc.) and the program would output a list of classes based on relevance to their inputted interests. Throughout working on our project though, we realized given our time, knowledge, and resources this wasn’t feasible. Our final product is a program that will generate course suggestions for students at Haverford, based on requirements they need to fill (domain and coure level). Our motivation for this project is that we’ve all had trouble finding classes that fulfill requirements we need (especially domain courses as Bionic does not allow students to filter courses based on domains) and we wanted to create a program that addresses this issue. We wanted to implement unique features that fit what we would want on a course recommendation website. This project helped us learn to create a graphical user interface while solving our problem by creating a program that generated courses based on requirements we need to fill.

**Methodology**:

HaverCourse required different functions and the GUI aspect of the code demanded a special python library called tkinter. Tkinter allows users to add buttons, labels, checkboxes to build a user interface. We also used two other libraries called typing and csv. The library typing in Python provides support to type hints. Type hints is a tool to help catch errors by providing information about the expected types of values in the code. Lastly, the CSV library allowed us to read and use and sort through our CSV file of Haverford courses. We used 3 different modules for this project, a main module, a sort\_functions module, and a display\_screen\_GUI. In the main.py module we have the function GUI where we have set up the colors, fonts, the labels, the checkboxes, and where everything is placed in the GUI. The Sort\_Function.py reads the CSV file then filters the data, removing unwanted courses. Finally, it filters the data into domains then sub divides the domains into levels. The display\_screen\_GUI.py module contains the code for displaying the results for the user. We had a total of 13 functions which helped with readability and easy debugging.

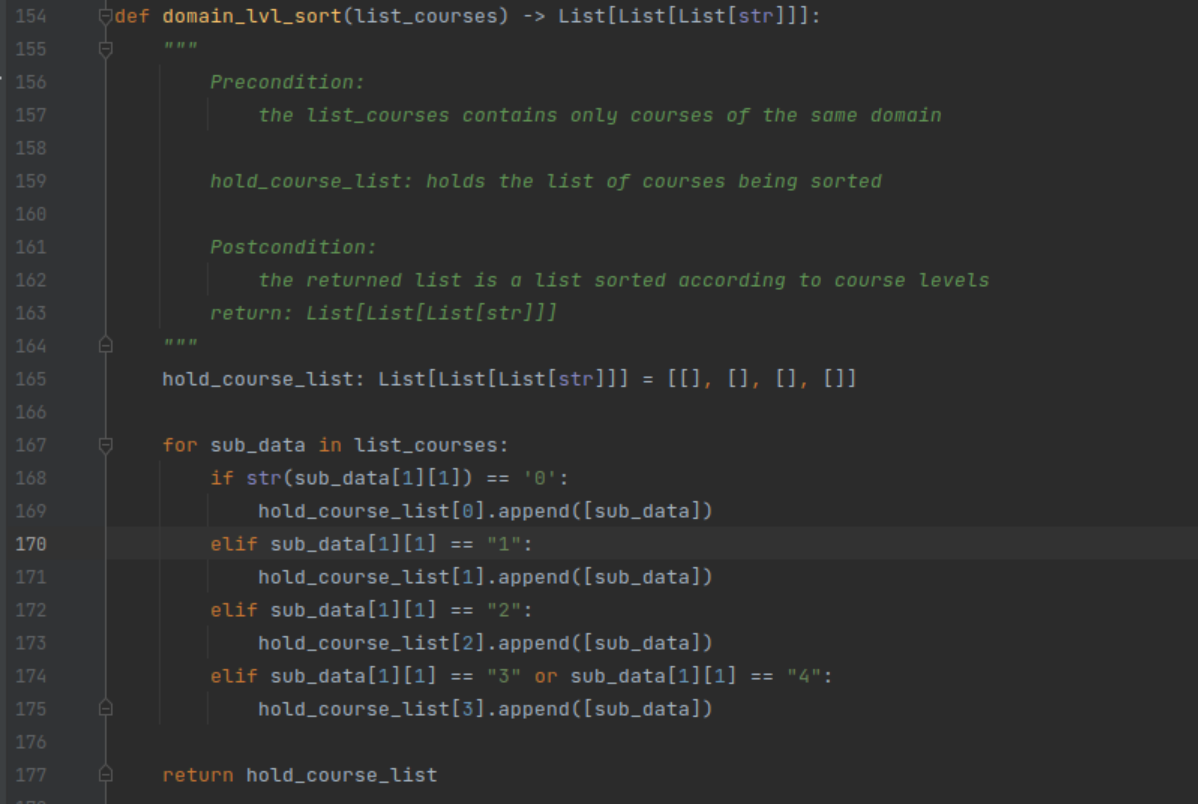
**Results**: screenshots of the code in action and add short comments

Screenshot of the main GUI function code



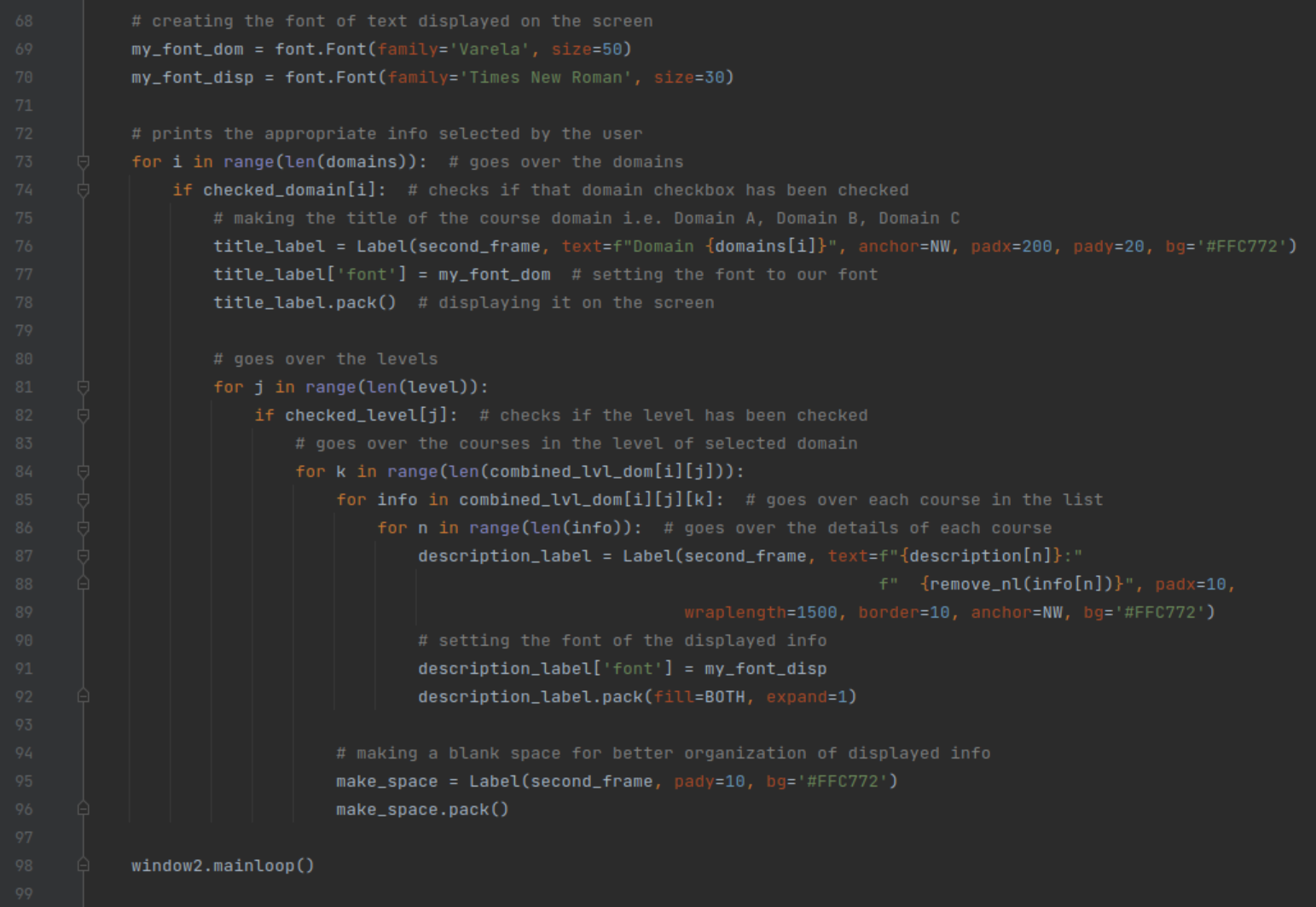
**Short comment** - The GUI function displays the labels containing the headings – HaverCourse, Domains, and Levels – and the checkboxes under the domains and level labels.

Screenshot of the domain level sorting function



**Short comment -** This function sorts through the domain courses into levels. The new sorted list has four elements, in the level order 000, 100, 200, 300+ level course.

Screenshot of the create\_scroll function that combines all the different sub ideas and functions together



**Short comment** - This function makes the scrollable GUI page and displays the results. The for loop starting on line 73 goes through the checked domains and checked level lists to determine which options the user had selected. The loops following it find the appropriate data to which is then displayed on the GUI.

Screenshot of test and results





Short comment - The above screenshot shows the user selecting Domain B and 300+ level courses and the program displays the results based on those inputs.

**Discussion**:

Although our project is finished, it is important to note that the process was not perfect as we did face some challenges throughout. Our very first challenge was finding a CSV file of the Haverford courses data (Department, title, attribute, level, etc). We looked online but could not find one so we decided to reach out to the registrar’s office to ask for a copy. After a few emails and in-person visits, we could finally have it and officially get started. The next challenge was to teach ourselves GUI, Graphical User Interface, in a short period of time and luckily we found abundant resources on the Internet to accomplish this. As we started building the GUI, we had more unpredicted problems that deepened our knowledge about how GUI works. GUI was more complex than we imagined; every time we thought we were done, we had to write a new function to account for a certain feature we overlooked or thought was automatically included. For instance, some of the features we needed to add included making the GUI course result page scrollable and get the resulting courses to be displayed on it. Moreover, we needed to have a proper positioning for the widgets/labels on the main screen of the project and to figure out the filtering and sorting functions that would generate accurate results to the users. And those functions were not enough, contrary to what we assumed, as we needed extra functions to display the appropriate filtered domain/level courses based on the user’s choice. We also needed to figure out how to connect the resulting functions together. All in all, through the challenges we encountered, we realized how precise, demanding, and detailed building a GUI is and we could get a glimpse of what it feels and looks like to be a full stack developer.

**Conclusion**:

As stated above, HaverCourse is a program that generates course recommendations for students at Haverford, based on requirements they need to fill. Although we faced challenges, such as learning how to create a graphical user interface, finding a csv file of all Haverford courses, and creating a display page that scrolled. We overcame them all and persevered through. We used new and old features and libraries in Pycharm, such as *Tkinter*, *typing*, and *csv*, which allowed us to expand our programming knowledge. This project provided us a space to work together and create a program that could actually be used by other students at Haverford. In the future, it could be interesting to add some other features and touch it up in order to allow students at Haverford to actually use it as a course recommendation system.

Code sources -

Learning GUI -

[Tkinter Course - Create Graphic User Interfaces in Python Tutorial](https://www.youtube.com/watch?v=YXPyB4XeYLA)

Making the scrollable GUI page - [Adding a Full Screen ScrollBar - Python Tkinter GUI Tutorial #96](https://www.youtube.com/watch?v=0WafQCaok6g)

Binding the mousewheel scroll to scroll the page -

<https://stackoverflow.com/questions/17355902/tkinter-binding-mousewheel-to-scrollbar>

General key binding -

<https://www.tutorialspoint.com/how-to-bind-a-key-to-a-button-in-tkinter>

Making the GUI fit to screen -

<https://stackoverflow.com/questions/30965033/python-tkinter-application-fit-on-screen>