Criterion C: Development

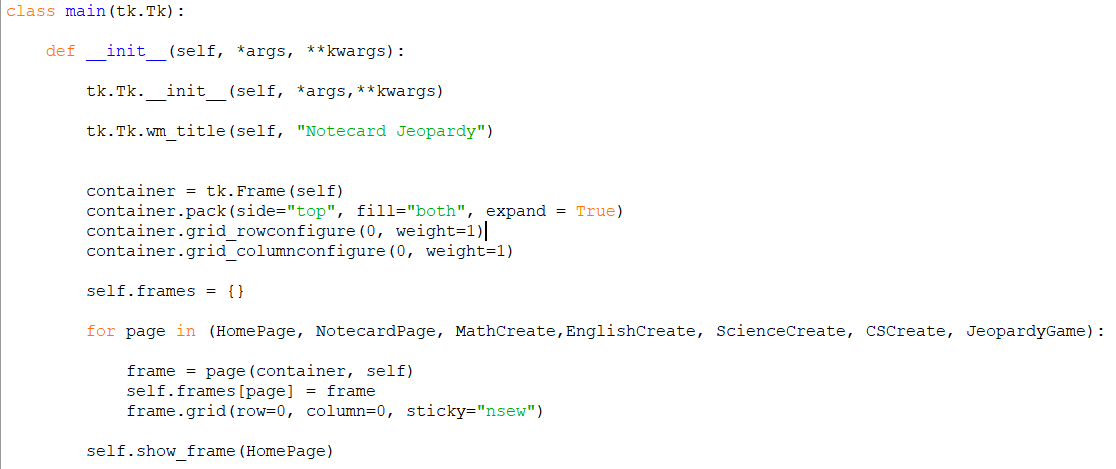
**Classes**:

* 
* 
* 
* 
* 
* 
* 
* 

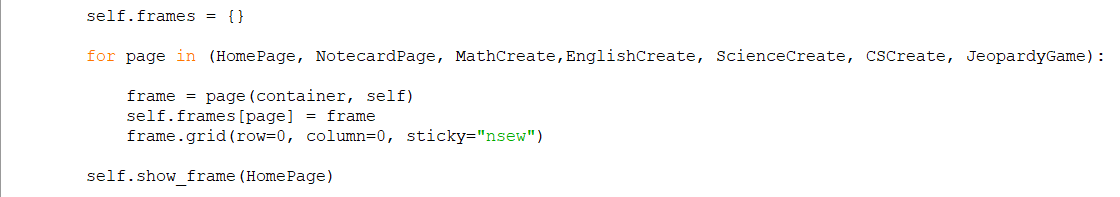
**List of the Technique:**

1. All of the classes have (tk.Frame) or tk.Tk as a parameter, which means the classes are inheriting from tk.Frame or tk.Tk. They are both built-in functions of the tkinter package.
2. Techniques used in main class allow for highly efficient page navigation
3. main class frame navigation combined with methods used in JeopardyGame to allow for flexibility in saving notecards and editing them for the game.
4. Various methods and functions are used, both functions included in the tkinter package (Button, Entry, etc) and also methods that I created.
5. Use of lambda, and global and local variables

**main class:**

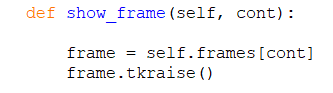


This is the main class, inherited from tk.Tk class from tkinter. In the constructor (def \_\_init\_\_), I first initialized the inherited class with tk.Tk.\_\_init\_\_ so that it can be used. Then, created container variable. It contains the frames, which are the various pages that will be added later on. This sets the base for efficient navigation of tabs



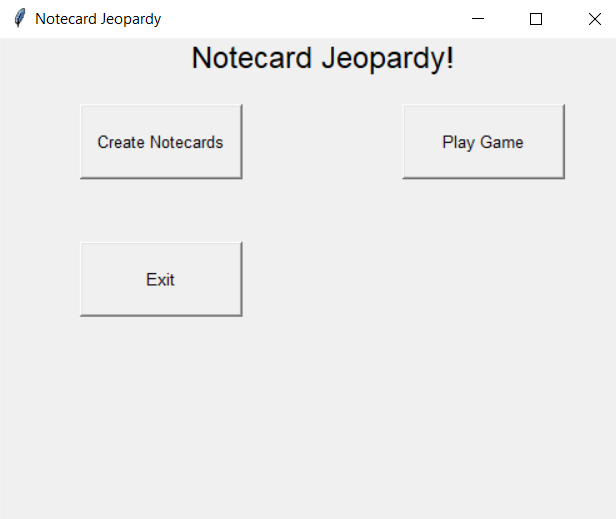
self.frames hold all of the pages of the program in the list in the for loop.

First instance of algorithmic thinking:

This loop iterates through each class/frame, stores them as key-value pairs. This way, all of the pages are initialized as soon as the program begins running. 

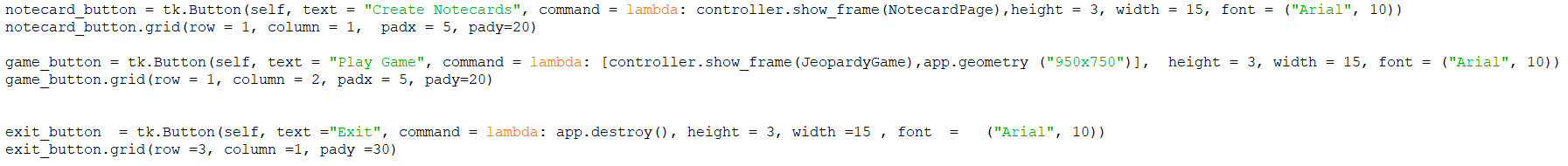
Then, show\_frame raises a page to the top with tkraise(). This navigation method is used for all page except JeopardyGame.

**Main Window:**



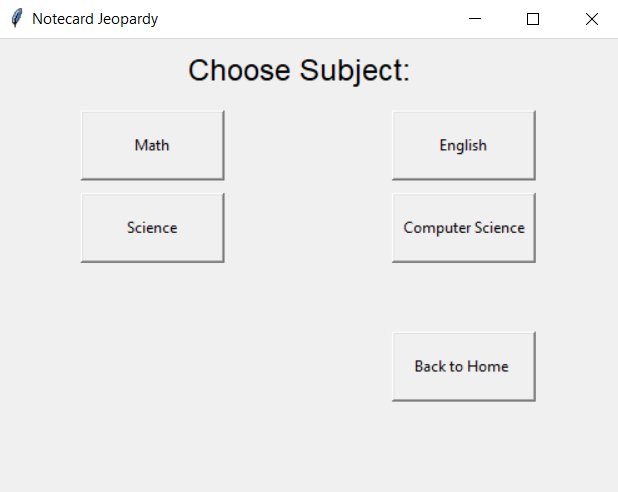
This is the main window, with each of the buttons leading to their corresponding frame /page, and the Exit button exiting program.

Use of tkinter controls:



The functions are done with lambda in the command parameter. controller.show\_frame() is used again. Lambda is throwaway function that allows parameters to be passed (JeopardyGame, NotecardPage). Value is not saved, only used once with Lambda.

After clicking notecard creation, the user can then choose which subject to create notecards for;

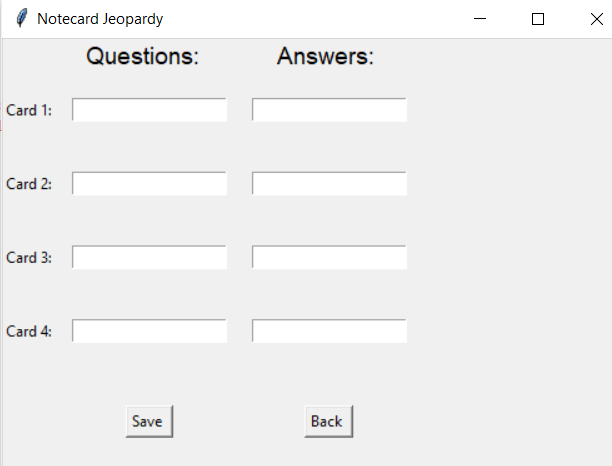


This page increases the ease of access and organization of notecard creation for each subjects

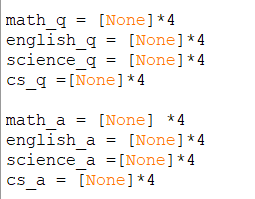
Back button goes back to HomePage.

Math button goes to the MathCreate page where they can type in their notecards for math.

These buttons all follow the same format as the previous navigation buttons with controller.show\_frame(Frame).



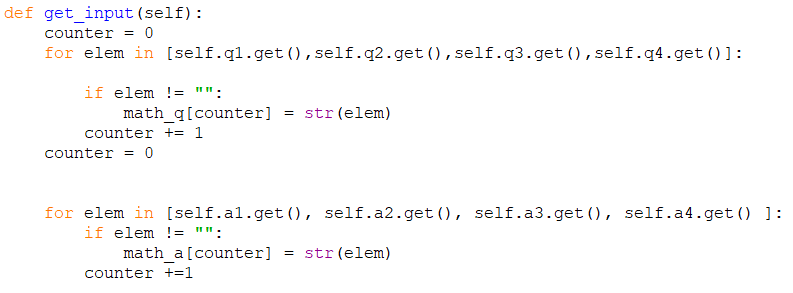
User inputs notecards, clicking save will save entry box input into question and answer arrays. These arrays are global variables that were declared at beginning so all classes could access them.



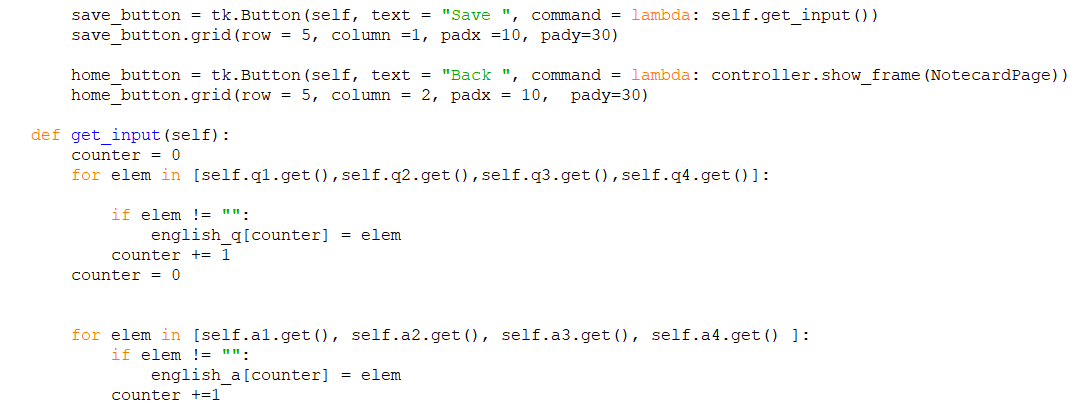
\*4 sets length to 4, for 4 notecards per subjects. This is the most efficient, about 6 times faster than using range(4)



The save button calls the get\_input() method, where I used algorithmic thinking.



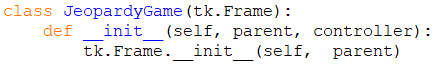
First create counter. Q1-q4 each represent one math question, and counter is the index of math\_q array. Same for a1-a4. After the question loop is finished iterating, the counter is reset to zero for answer loop . Inside both of the loops, a conditional statement used to check if the user input in the entry box is empty. If not empty, then the user input will be saved into the corresponding position in the math\_q or math\_a array as a string. String normalizes format, less room for error later on. Algorithmic thinking here again.



The same process is used for the other subjects (English, Cs, Sciencee)..

There was no need for an edit button, because the save button is able to perform the task. Every time Save button is pressed, get\_input() will replace current data with new data in the entry boxes.

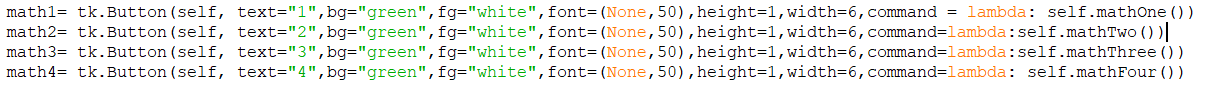
Finally, the JeopardyGame Class is the one for the Jeopardy page.



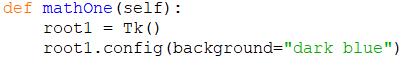
It also begins by inheriting from the tk. Frame class and initializing the class within the constructor method.

**Navigation methods used in JeopardyGame combined with the main class navigation method for the rest of program to allow for the flexibility in saving notecards and accessing them for the game.**

After this, the buttons for each of the four questions under the subject are added.

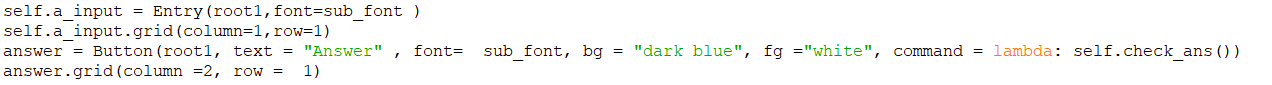


Here, the way of navigation changes. Instead of using a controller and bringing a page to the front with controller.show\_frame(), I call a method within the same Jeopardy Game class. As an example, I call the mathOne()::



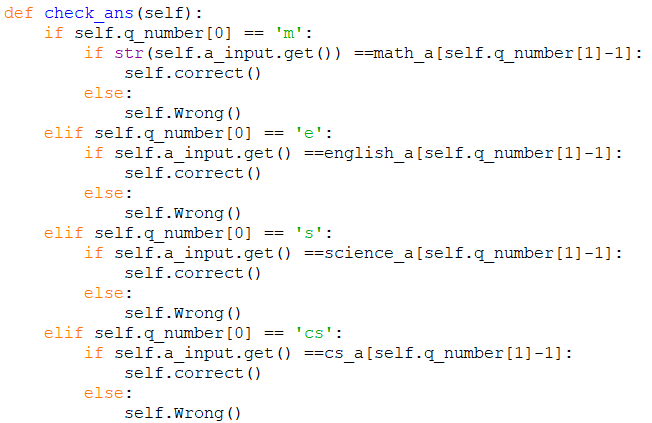
This function uses a different approach. Earlier, for ease of navigation all of the classes were initialized when the program began to run. show\_frame raises already-initialized page to top.. However, here, I use root1 = Tk(), which creates a Tkinter root widget. It creates an entirely new page that I can format.

I want the label to be the question that the user entered earlier in the corresponding notecard, so I set text =math\_q[0]. math\_q in the array that stores the math question data, and since this is math question 1, the index would be zero.

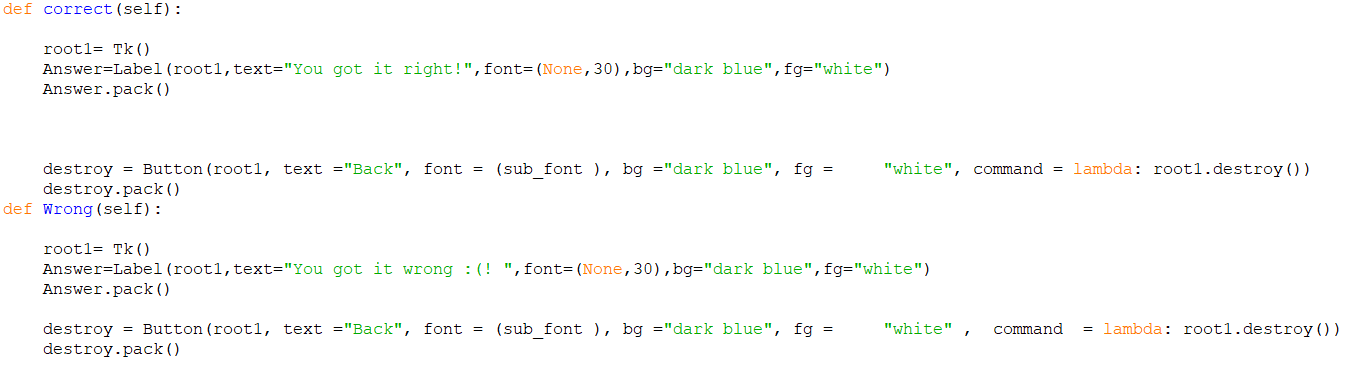


Then, answer button calls check\_ans() method. The check\_ans() function uses nested conditional statements to determine whether the answer is correct. Earlier in each of the question functions, I declared a variable called self.q\_number: 

First value represents the subject, second value represents notecard number.

check\_ans checks subject, then if entry box input matches corresponding answer data. 

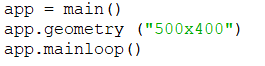
self.correct() and self.Wrong() takes user to corresponding screen.



Both of these functions are navigated to through a similar method as the questions inside the JeopardyGame class, with root = Tk() widget.

The reason I used this technique in combination with the tk.Frame navigation technique in main class is so that the data would be updated and displayed correctly in the Jeopardy Game. If the questions are initialized at the beginning, then the display would be empty, since the variables are empty at the beginning of the program.

With this way, the page is only initialized when clicked, so each time flashcards are created/edited, clicking on a question will refresh the display with updated data. This allows the Jeopardy Game aspect to work and function with the notecard creating.

Finally, the technique I used at the end is another tkinter formatting technique: 

app.geometry(“750x 669”) changes window size for Jeopardy

app.geometry(500 x400) switches back

At this point, the program is finished. Throughout this program, I used algorithmic thinking to write the functions of multiple buttons, and their interactions with entry boxes. I also used algorithmic thinking in the code for navigating between the pages. I decided to structure the program in the way that I did and used the techniques that I did because it would allow the smooth running of my program and a functioning connection between the flashcard creation and the jeopardy Game.

Word Count: 1074.