CMPSC 487W: Software Engineering & Design (Fall 2022)

CMPSC 487W: Software Engineering & Design

Hansi Seitaj, Eni Vejseli, Almaz Akhunbaev, Lukas Belashov

Final Project Report
November 22, 2022

Project Name: **Interactive Flow-Chart**

Dr. Alejandro Trofimoff

Submitted On:
12/08/2022

TABLE OF CONTENTS

1. INTRODUCTION

The goal of this project is to develop an application that allows students to schedule their courses and solve the problem of creating a user-friendly visualization of a student's academic path in a Criminology major. The interactive application should allow students to divide courses into four sections and allow the users to customize their version of the app, for ex., change course information. Implement a user-friendly course attribute change. Implement the ability of personal versions and uploads based on student's personal data and display the amount of credits students have taken so far. Finally, allowing the user to create customized versions of the program and have a way to upload them if needed.

2. BACKGROUND

The Software Development Life Cycle (SDLC) involves several distinct stages, including planning, analysis, design, building, testing, deployment, and maintenance.

In the Agile model, fast failure is a good thing. This approach produces ongoing release cycles, each featuring small, incremental changes from the previous release. At each iteration, the product is tested.

- **Planning**:

We tried to figure out what was required for the project by meeting with the client. We also brainstormed different methods and techniques we could use to satisfy the project requirements along with our client's needs. We came up with the idea to make it a web application but after discussing it further with our professor we switched it to a python application. Then afterward set up deadlines for our work and set up other meetings with the client to get her feedback.

- **Analysis**:

We came up with the idea to edit courses instead of just swapping them around and decided we need to create algorithms to populate our program, process different files, and edit courses. To help visualize the flow of code we created the class diagram for the project.

- **Design:**

After numerous meetings, we decided that there will be two main windows for the program, the main one and a second one for editing courses. Then we finalized the look of the main window which is divided into 4 years and 10 courses for each year.

- **Building:**

Each person had different methods to do by a certain timeline and we had iterations together to discuss the progress. We had different demos to show to our client. With client feedback, we managed to improve our design and software so it suits her needs.

- **Testing:**

We did different types of testing like unit testing, integration, and functional testing as we were working on the project. There were different bugs during the project that we identified from testing and fixed. The final testing was the acceptance testing where the software had all the functionalities.

- **Deployment:**

Created an executable and delivered it to the client. We also published a GitHub link for the project.

- **Maintenance:**

Set up a contact with the client so that for every problem, she can contact us and we will work on it.

3. DESIGN & IMPLEMENTATION

Data dictionary and data structures

Variable	Variable Name	Measurement	Allowed Values	Description
Button Array Index	buttonIndex	Numeric	0-39	Keeps track of the index of which button was clicked as an integer.
Button Array List	buttonArray	List/Array	0-39	List which holds all of the buttons on the primary window GUI.
Color Of Class	classColor	String	Stylesheet	Stores stylesheet of button as string.
Name Of Class	className	String	String Of Class Name	Stores name of class as a string.
User’s Name	name	String	Name	Stores the name of the user as a string.
Name Of File	fileName	String	Desired File Name	Stores the name which the file will be given, stored as a string.
Congratulations Label	congratulations	String	Desired Statement	Maintains label for updating the number of credits taken, and congratulating the user when necessary all classes are complete.

Number Of Credits	numberOfCredits	Numeric	Any Positive Integer	Stores the required number of credits for a class as an integer.
Total Number Of Credits	totalNumberOfCredits	Numeric	Any Positive Integer	Remembers the total number of credits for all the classes, stored as an integer

Our interactive flow-chart was designed to be user-friendly for an audience that is not particularly tech-savvy. In order to implement such basic functionality along with a simple and reassuring atmosphere, we opted for a Graphical user Interface (GUI). In this GUI we have buttons for the user to interact with, which brings us to our first algorithmic method, the *Brute Force Technique*.

Initially, this may sound daunting and ineffective, however, we found that when circumstances only deal with minor amounts of data, the brute force technique is sufficient. Since there are not many buttons to sift through, we simply implemented a linear search, traversing each button in the button array (40 buttons). Optimizing this search would yield little to no noticeable efficiency due to the small amount of data being used. Though there are numerous files the user can manage, there is always only ever one file being used for storage, thus making this algorithmic method productive.

Linear Search

For the majority of the implementation, whenever we encountered any scenario that required an extensive or thorough search, we opted for the linear search since all of the searches did not require significant amounts of comparisons. Implementing a *Greedy Algorithm* or even choosing to use a *Divide and Conquer Approach* would do little to improve the efficiency of the program. The improvement would be essentially negligible, thus we opted for more familiar methodologies.

Applicable Functions

In order to build logic into the functions that we used, there had to be effective algorithms implemented to not only improve the efficiency of the functions but to reduce any unnecessary code, thus eliminating any clutters and creating an overall more clean, professional, and tidy coding atmosphere. To accomplish such a task of cleanliness, there had to be a thorough evaluation of each function. After such an evaluation, the blocks of code were analyzed and then either reformatted or revised to improve cleanliness, reducing the amount of hard code.

Functions that dealt with checking what button was pressed, keeping track of the button index, saving user-inputted data to the file, and so on, all involved comparisons. Rather than hard-coding each and every line, we opted for algorithms to simply parse through each element, updating the necessary one when found. Once again, such a task can be accomplished with a linear search; which was perfect in our case since we had to parse through each element in order to save it anyway. Thus, using a *Divide and Conquer Approach* would be problematic, as it would eliminate half the data, leading to all sorts of data issues.

Intermediate Variables

Intermediate variables:

- ButtonIndex
- ButtonArray
- ClassColor
- ClassName
- AdditionalCredits
- Name
- FileName
- Congratulations
- NumberOfCredits
- TotalNumberOfCredits

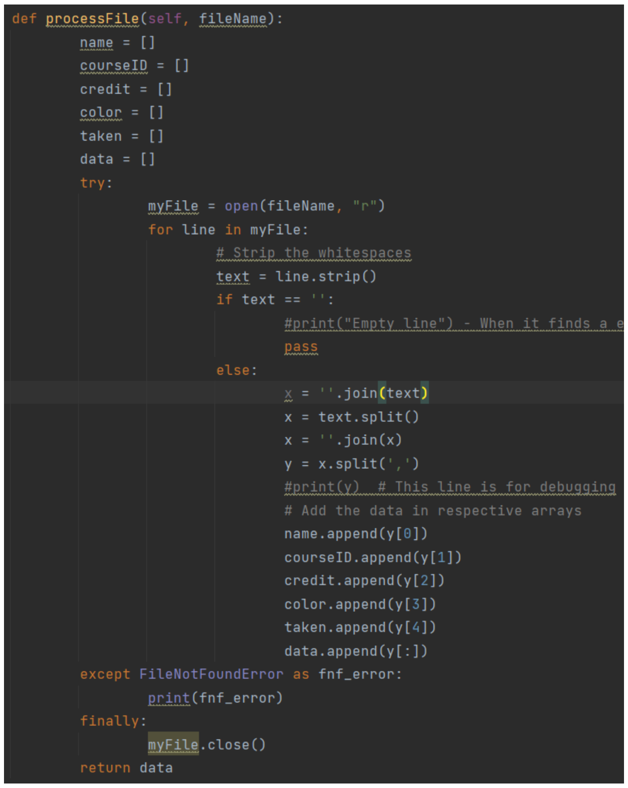
These are the intermediate variables that were an integral part of the entire project, as they are threaded throughout the entirety of the application. This is simply a brief overview of the intermediate variables, they will be further discussed and analyzed in the section regarding the data dictionary.

However to at the very least scrape the surface of the functionality of these variables, we will state the structures used for them. ButtonArray is a global list, which is declared initially after the imports, and then later populated. ButtonIndex and AdditionalCredits are integers, and ClassColor, ClassName, Name, and FileName are empty strings initially, which are populated later on.

Algorithm pseudo-codes and description:

- **processFile algorithm and pseudo-code:**

```
processFile(filename):
    Lists to save the information in the format:
    Course Name, ID, Credit, Color, Frame,
    2 Dimensional array

    Open the file.
        For line in the file:
            If there is a empty line:
                Pass it.
            Else:
                Enter the inputs into separate mentioned above lists.
    Close the file.
    Return the 2-Dimensional array
```



```
def processFile(self, fileName):
        name = []
        courseID = []
        credit = []
        color = []
        taken = []
        data = []
        try:
                myFile = open(fileName, "r")
                for line in myFile:
                        # Strip the whitespaces
                        text = line.strip()
                        if text == '':
                                #print("Empty line") - When it finds a e
                                pass
                        else:
                                x = ''.join(text)
                                x = text.split()
                                x = ''.join(x)
                                y = x.split(',')
                                #print(y)  # This line is for debugging
                                # Add the data in respective arrays
                                name.append(y[0])
                                courseID.append(y[1])
                                credit.append(y[2])
                                color.append(y[3])
                                taken.append(y[4])
                                data.append(y[:])
        except FileNotFoundError as fnf_error:
                print(fnf_error)
        finally:
                myFile.close()
        return data
```

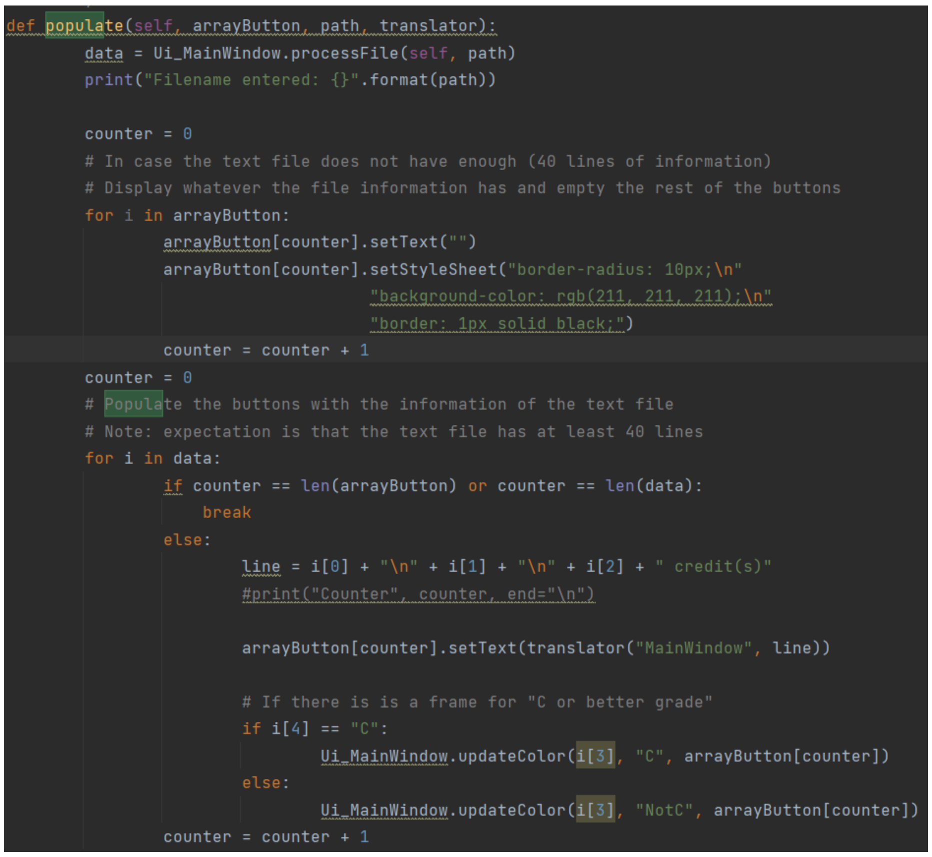
Algorithm description:

Saves the information from a text file (.txt) into 6 list data structures. The algorithm ignores all the blank lines in the text file. Also, it needs to input a filename and does exception handling. Finally, creates a two-dimensional array named ‘data’ that has all the other array information. Quality metric: high precision and accuracy.

- **Populate algorithm and pseudo code:**

```
populate(arrayButton, path, translator):
    Calls processFile method to access the data.

    For button in arrayButton:
        Display whatever the file information has and empty the rest of the buttons.

    For element in data:
```

```
If we populated 40 buttons:
    Break
Else:
    Populate buttons text, color, frame.
```



```
def populate(self, arrayButton, path, translator):
    data = Ui_MainWindow.processFile(self, path)
    print("Filename entered: {}".format(path))

    counter = 0
    # In case the text file does not have enough (40 lines of information)
    # Display whatever the file information has and empty the rest of the buttons
    for i in arrayButton:
        arrayButton[counter].setText("")
        arrayButton[counter].setStyleSheet("border-radius: 10px;\n"
                                     "background-color: rgb(211, 211, 211);\n"
                                     "border: 1px solid black;")
        counter = counter + 1
    counter = 0
    # Populate the buttons with the information of the text file
    # Note: expectation is that the text file has at least 40 lines
    for i in data:
        if counter == len(arrayButton) or counter == len(data):
            break
        else:
            line = i[0] + "\n" + i[1] + "\n" + i[2] + " credit(s)"
            #print("Counter", counter, end="\n")

            arrayButton[counter].setText(translator("MainWindow", line))

            # If there is is a frame for "C or better grade"
            if i[4] == "C":
                Ui_MainWindow.updateColor(i[3], "C", arrayButton[counter])
            else:
                Ui_MainWindow.updateColor(i[3], "NotC", arrayButton[counter])
        counter = counter + 1
```

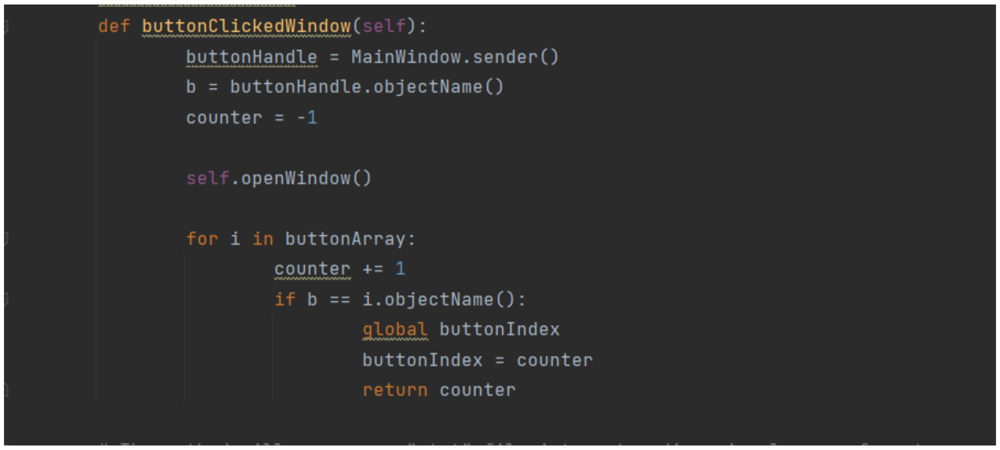
Algorithm description:

Populate method takes the processed file and displays the data to the main Window interface and populates the button titles, colors, and frames. In other words, the method inputs the button array, the file path, and a translator method. In case the file entered does not have enough information for 40 information it will leave it blank and ready for customization. Quality metric: high precision and accuracy.

- **buttonClickedWindow algorithm and pseudo code:**

```
buttonClickedWindow():
    Initialize MainWindow sender to detect which object is clicked
    Declare variable and set it equal to object name
    Initialize a counter to -1
```

```
Open secondary window

For button in buttonArray: (To find button index)
    Increment counter
    If button clicked is found in button array:
        Global declaration of buttonIndex variable
        Assign button index to counter
        Return counter
```



```python
def buttonClickedWindow(self):
        buttonHandle = MainWindow.sender()
        b = buttonHandle.objectName()
        counter = -1

        self.openWindow()

        for i in buttonArray:
                counter += 1
                if b == i.objectName():
                        global buttonIndex
                        buttonIndex = counter
                        return counter
```

Algorithm description:

The algorithm performs a linear search through each button in the button array. First a sender for the main window is created to grab the object which activated the buttonClickedWindow function, however in order to translate that syntax into something legible, the object name of that object must be assigned to the variable. Now that variable which holds the name of the button that triggered the function call is compared to every button name in the array, once a match is found, that iteration is returned as the index.

- **UpdateColors algorithm and pseudo code:**

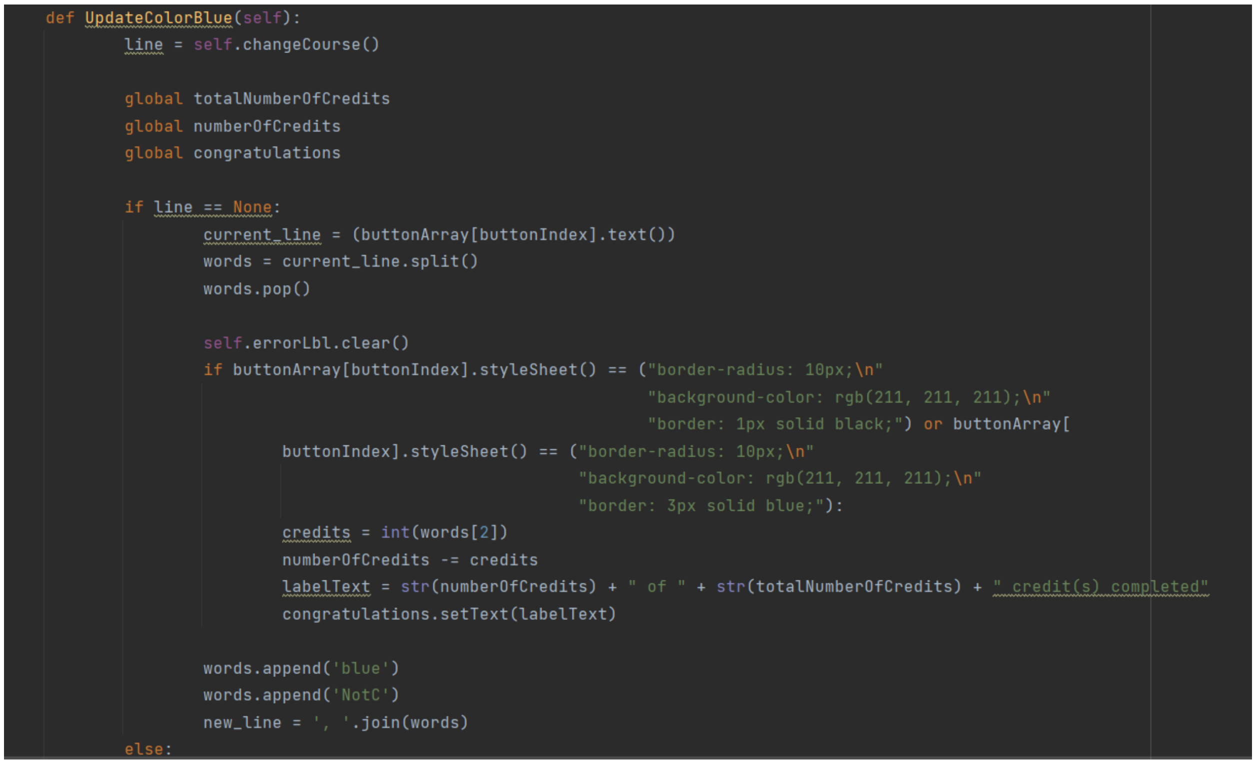
```
UpdateColorBlue():
    Assign string to equal input from secondary window.

    If no user input:
        Keep current buttons text, but switch color to blue
    Else:
        Create new string, combing the user input and assigning blue to it.

    Open default template file in read mode,
```

Parse file and assign data to list
Close file

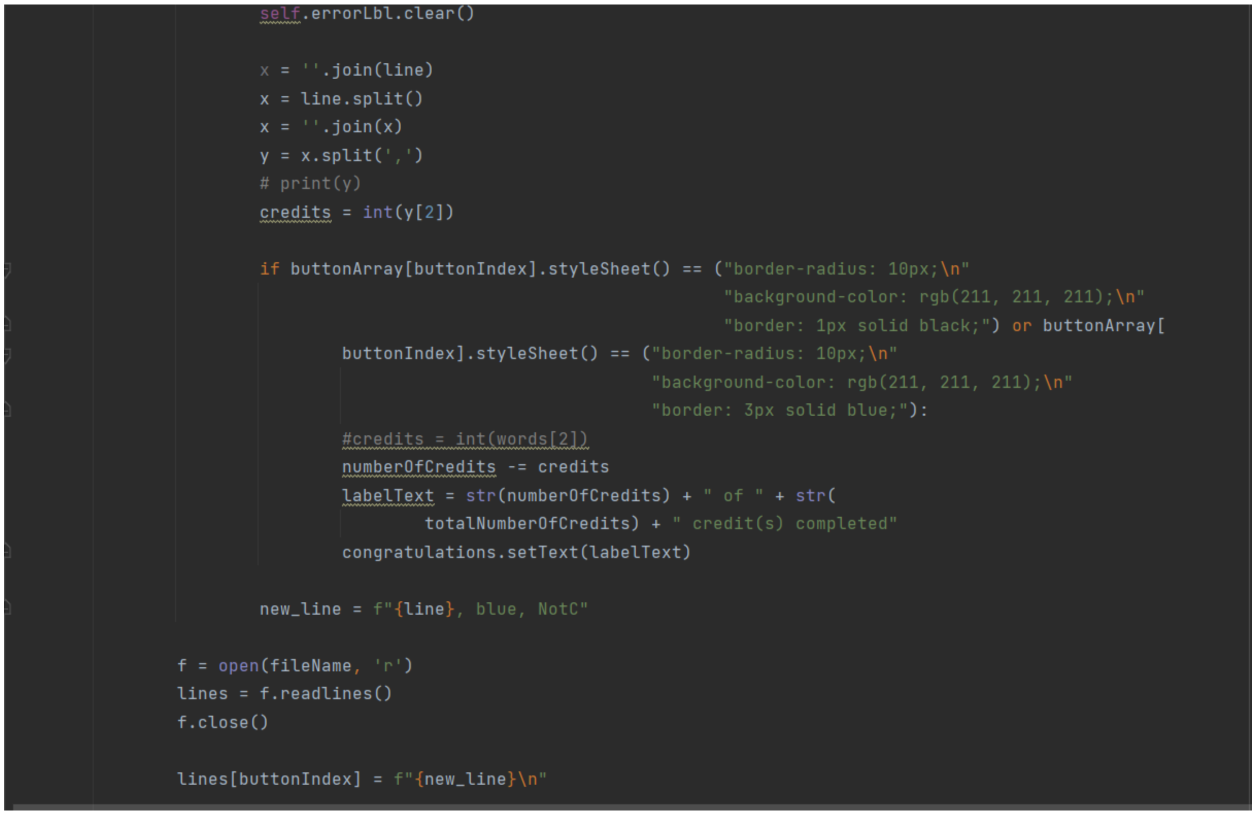
Using the button index from the previous function, pass the button index into the list and overwrite that index with the new updated line.

Open new file in write mode
Write new updated list (data) to file
Close file

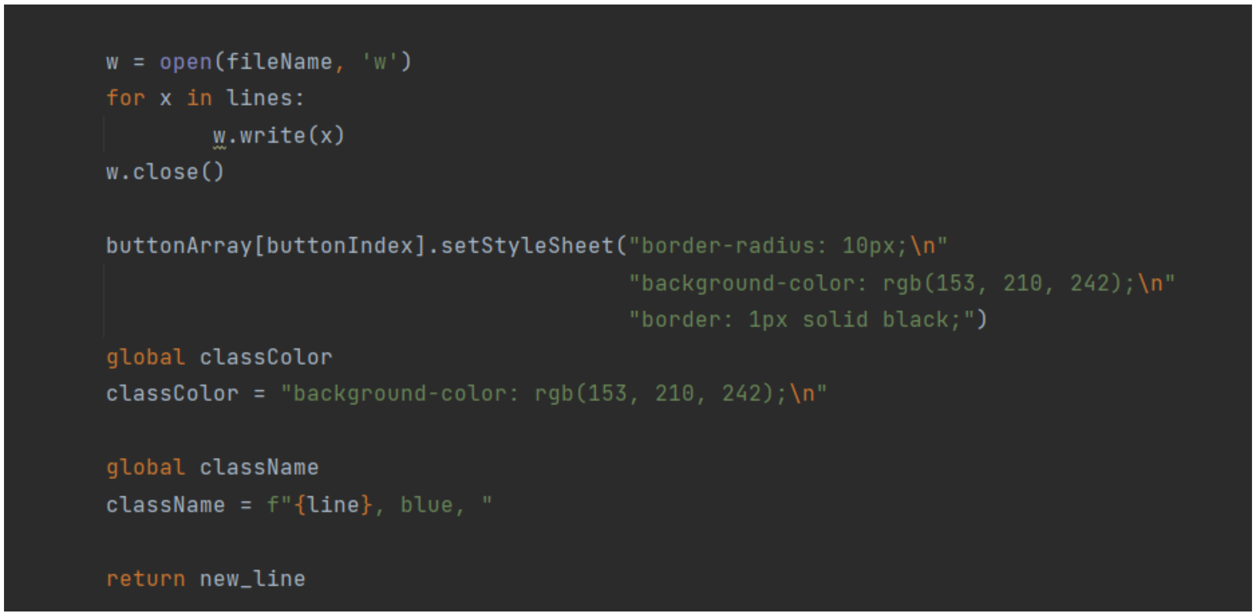
Update stylesheet of button clicked with new/same line but with a blue background
Global declarations of classColor and className
Initialize classColor and className



```python
def UpdateColorBlue(self):
        line = self.changeCourse()

        global totalNumberOfCredits
        global numberOfCredits
        global congratulations

        if line == None:
                current_line = (buttonArray[buttonIndex].text())
                words = current_line.split()
                words.pop()

                self.errorLbl.clear()
                if buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                                             "background-color: rgb(211, 211, 211);\n"
                                                             "border: 1px solid black;") or buttonArray[
                        buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                                      "background-color: rgb(211, 211, 211);\n"
                                                      "border: 3px solid blue;"):
                        credits = int(words[2])
                        numberOfCredits -= credits
                        labelText = str(numberOfCredits) + " of " + str(totalNumberOfCredits) + " credit(s) completed"
                        congratulations.setText(labelText)

                words.append('blue')
                words.append('NotC')
                new_line = ', '.join(words)
        else:
```



```
                self.errorLbl.clear()

                x = ''.join(line)
                x = line.split()
                x = ''.join(x)
                y = x.split(',')
                # print(y)
                credits = int(y[2])

                if buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                                             "background-color: rgb(211, 211, 211);\n"
                                                             "border: 1px solid black;") or buttonArray[
                        buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                                      "background-color: rgb(211, 211, 211);\n"
                                                      "border: 3px solid blue;"):
                        #credits = int(words[2])
                        numberOfCredits -= credits
                        labelText = str(numberOfCredits) + " of " + str(
                                totalNumberOfCredits) + " credit(s) completed"
                        congratulations.setText(labelText)

                new_line = f"{line}, blue, NotC"

        f = open(fileName, 'r')
        lines = f.readlines()
        f.close()

        lines[buttonIndex] = f"{new_line}\n"
```



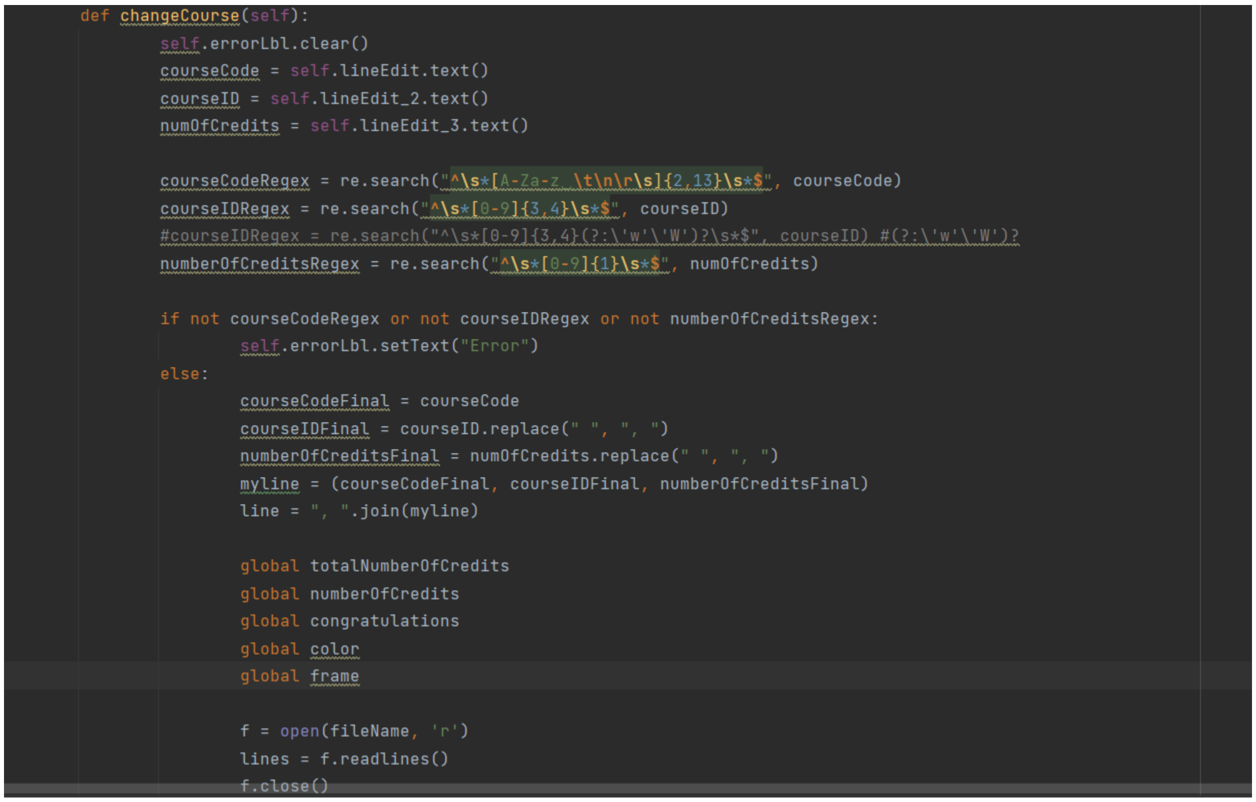
```
w = open(fileName, 'w')
for x in lines:
        w.write(x)
w.close()

buttonArray[buttonIndex].setStyleSheet("border-radius: 10px;\n"
                                       "background-color: rgb(153, 210, 242);\n"
                                       "border: 1px solid black;")
global classColor
classColor = "background-color: rgb(153, 210, 242);\n"

global className
className = f"{line}, blue, "

return new_line
```

Algorithm description:

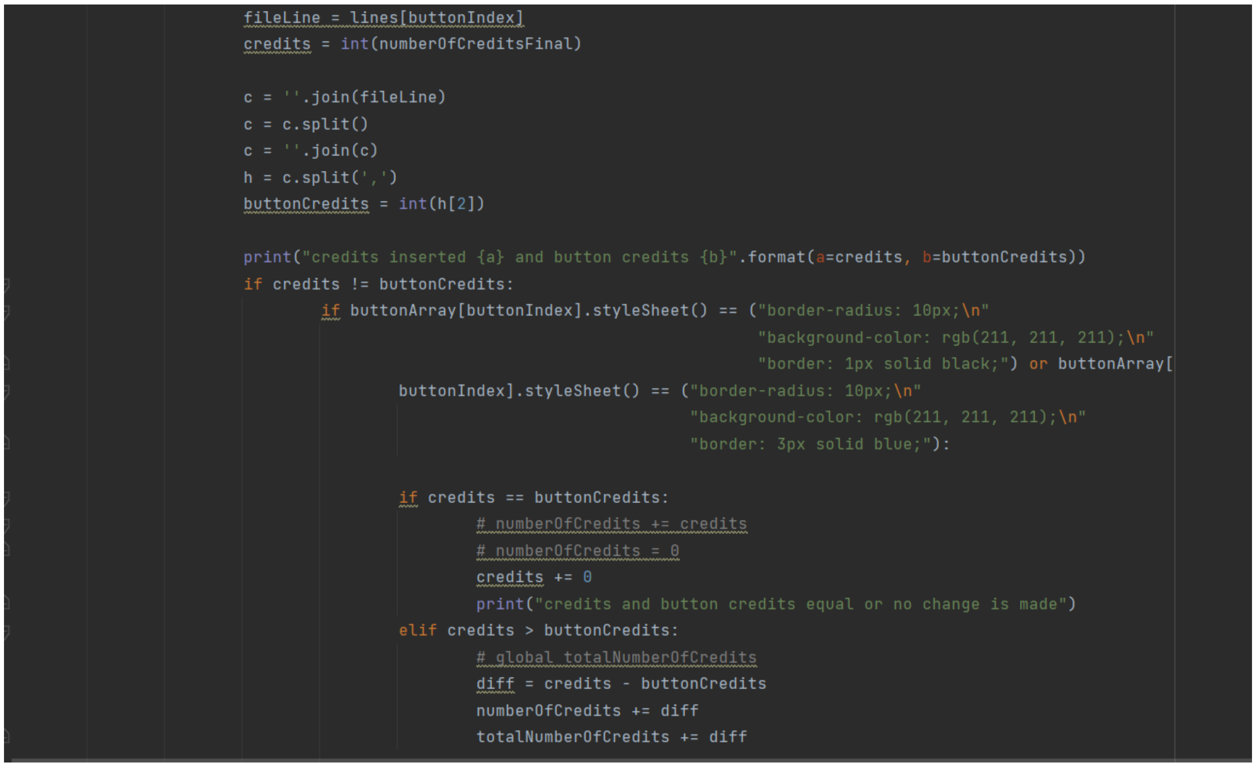
The algorithm takes a line from the user and based on the input, updates the contents of the front-end button, and back-end variable.

- **changeCourse algorithm and pseudo code:**

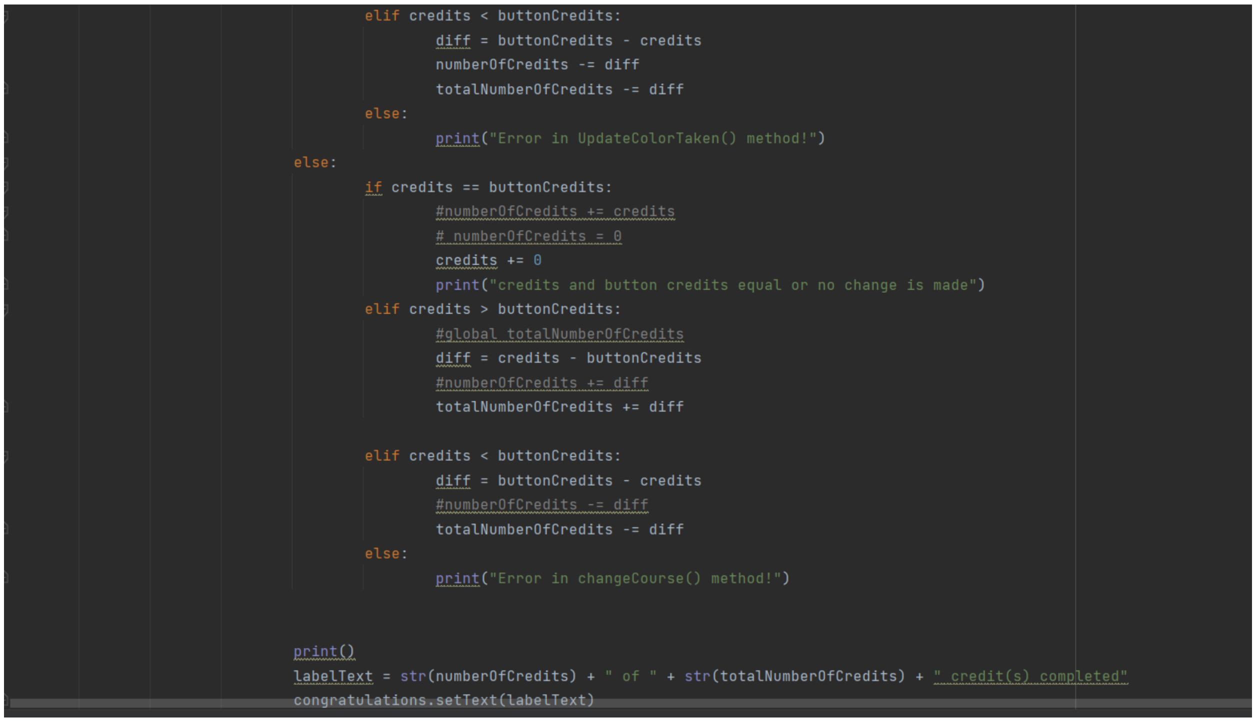
```
changeCourse():
        Clear the error label that pops up if the user does not follow course information input
restrictions.
        Take the course information: course code, course id, and the number of credits.
        Evaluate each input using regex.

        If the course information is correct:
                Open the txt file and change the course information.
        Else:
                Display the error message to the user.

        Multiple if statements to update the congratulations label in the right format.
        Return line
```



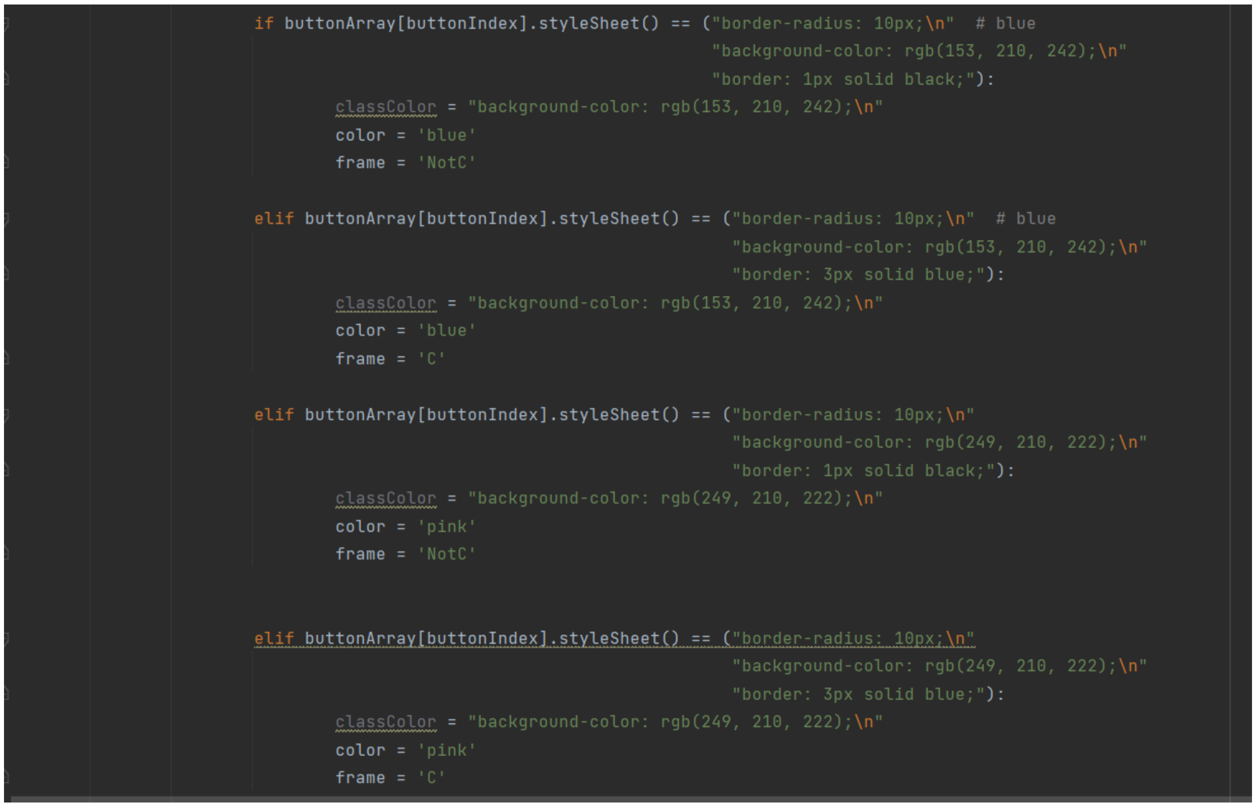
```python
    def changeCourse(self):
        self.errorLbl.clear()
        courseCode = self.lineEdit.text()
        courseID = self.lineEdit_2.text()
        numOfCredits = self.lineEdit_3.text()

        courseCodeRegex = re.search("^\s*[A-Za-z \t\n\r\s]{2,13}\s*$", courseCode)
        courseIDRegex = re.search("^\s*[0-9]{3,4}\s*$", courseID)
        #courseIDRegex = re.search("^\s*[0-9]{3,4}(?:\'w'\'W')?\s*$", courseID) #(?:\'w'\'W')?
        numberOfCreditsRegex = re.search("^\s*[0-9]{1}\s*$", numOfCredits)

        if not courseCodeRegex or not courseIDRegex or not numberOfCreditsRegex:
            self.errorLbl.setText("Error")
        else:
            courseCodeFinal = courseCode
            courseIDFinal = courseID.replace(" ", "", ")
            numberOfCreditsFinal = numOfCredits.replace(" ", "", ")
            myline = (courseCodeFinal, courseIDFinal, numberOfCreditsFinal)
            line = ", ".join(myline)

            global totalNumberOfCredits
            global numberOfCredits
            global congratulations
            global color
            global frame

            f = open(fileName, 'r')
            lines = f.readlines()
            f.close()
```



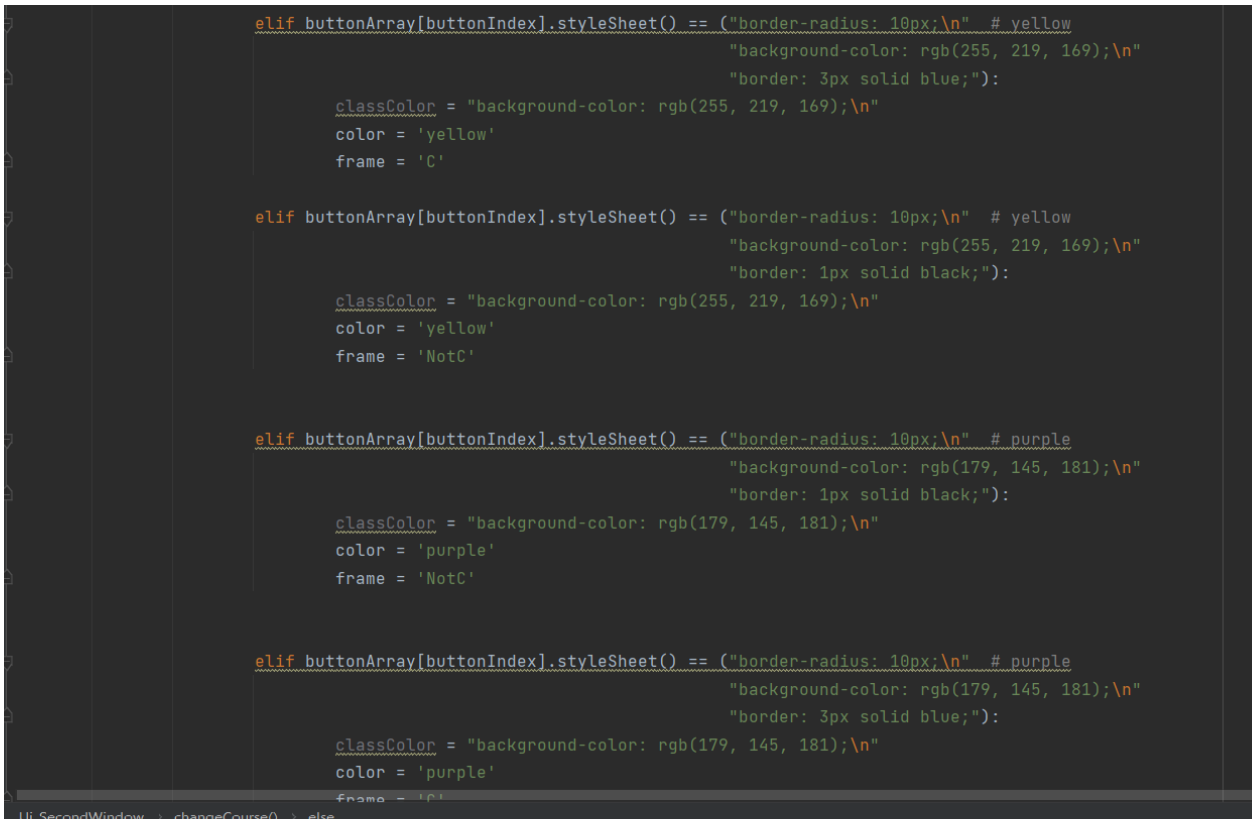
```
fileLine = lines[buttonIndex]
credits = int(numberOfCreditsFinal)

c = ''.join(fileLine)
c = c.split()
c = ''.join(c)
h = c.split(',')
buttonCredits = int(h[2])

print("credits inserted {a} and button credits {b}".format(a=credits, b=buttonCredits))
if credits != buttonCredits:
        if buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                                     "background-color: rgb(211, 211, 211);\n"
                                                     "border: 1px solid black;") or buttonArray[
                buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                              "background-color: rgb(211, 211, 211);\n"
                                              "border: 3px solid blue;"):

                if credits == buttonCredits:
                        # numberOfCredits += credits
                        # numberOfCredits = 0
                        credits += 0
                        print("credits and button credits equal or no change is made")
                elif credits > buttonCredits:
                        # global totalNumberOfCredits
                        diff = credits - buttonCredits
                        numberOfCredits += diff
                        totalNumberOfCredits += diff
```



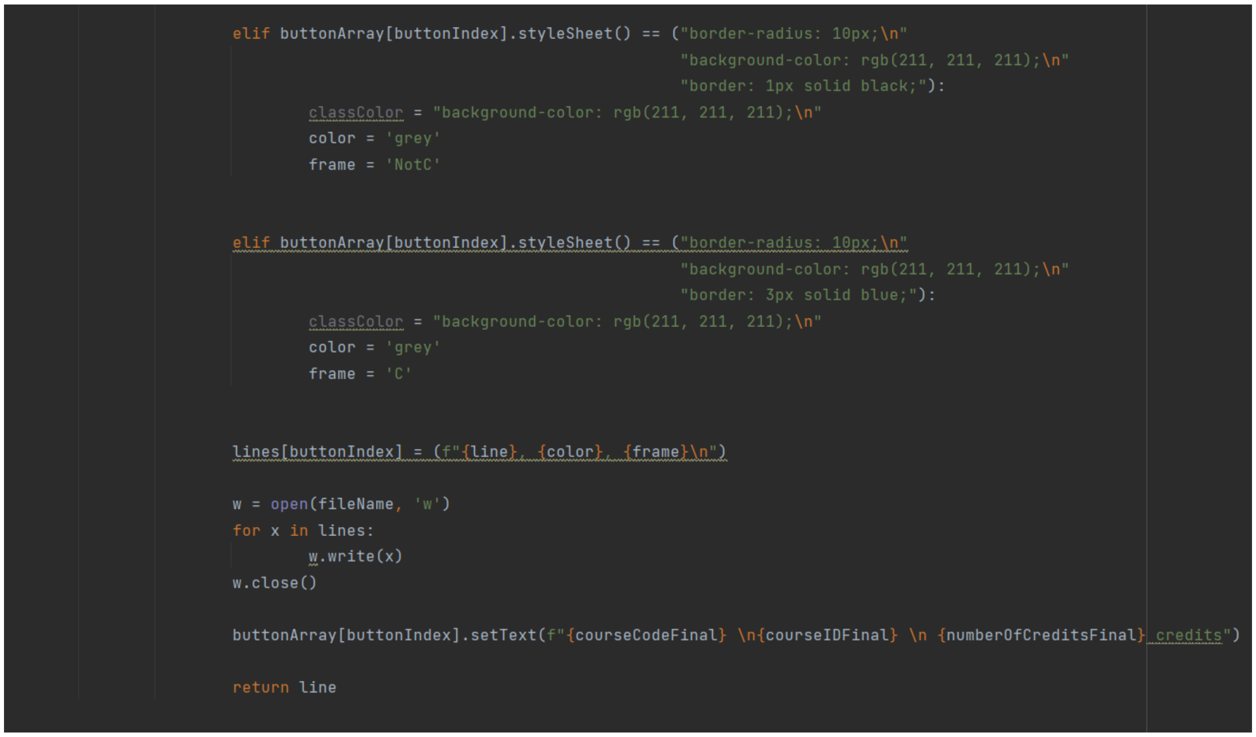
```
                elif credits < buttonCredits:
                        diff = buttonCredits - credits
                        numberOfCredits -= diff
                        totalNumberOfCredits -= diff
                else:
                        print("Error in UpdateColorTaken() method!")
        else:
                if credits == buttonCredits:
                        #numberOfCredits += credits
                        # numberOfCredits = 0
                        credits += 0
                        print("credits and button credits equal or no change is made")
                elif credits > buttonCredits:
                        #global totalNumberOfCredits
                        diff = credits - buttonCredits
                        #numberOfCredits += diff
                        totalNumberOfCredits += diff

                elif credits < buttonCredits:
                        diff = buttonCredits - credits
                        #numberOfCredits -= diff
                        totalNumberOfCredits -= diff
                else:
                        print("Error in changeCourse() method!")


        print()
        labelText = str(numberOfCredits) + " of " + str(totalNumberOfCredits) + " credit(s) completed"
        congratulations.setText(labelText)
```



```
if buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"  # blue
                                             "background-color: rgb(153, 210, 242);\n"
                                             "border: 1px solid black;"):
    classColor = "background-color: rgb(153, 210, 242);\n"
    color = 'blue'
    frame = 'NotC'

elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"  # blue
                                               "background-color: rgb(153, 210, 242);\n"
                                               "border: 3px solid blue;"):
    classColor = "background-color: rgb(153, 210, 242);\n"
    color = 'blue'
    frame = 'C'

elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                               "background-color: rgb(249, 210, 222);\n"
                                               "border: 1px solid black;"):
    classColor = "background-color: rgb(249, 210, 222);\n"
    color = 'pink'
    frame = 'NotC'


elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                               "background-color: rgb(249, 210, 222);\n"
                                               "border: 3px solid blue;"):
    classColor = "background-color: rgb(249, 210, 222);\n"
    color = 'pink'
    frame = 'C'
```



```
elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"  # yellow
                                               "background-color: rgb(255, 219, 169);\n"
                                               "border: 3px solid blue;"):
    classColor = "background-color: rgb(255, 219, 169);\n"
    color = 'yellow'
    frame = 'C'

elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"  # yellow
                                               "background-color: rgb(255, 219, 169);\n"
                                               "border: 1px solid black;"):
    classColor = "background-color: rgb(255, 219, 169);\n"
    color = 'yellow'
    frame = 'NotC'


elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"  # purple
                                               "background-color: rgb(179, 145, 181);\n"
                                               "border: 1px solid black;"):
    classColor = "background-color: rgb(179, 145, 181);\n"
    color = 'purple'
    frame = 'NotC'


elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"  # purple
                                               "background-color: rgb(179, 145, 181);\n"
                                               "border: 3px solid blue;"):
    classColor = "background-color: rgb(179, 145, 181);\n"
    color = 'purple'
    frame = 'C'
```

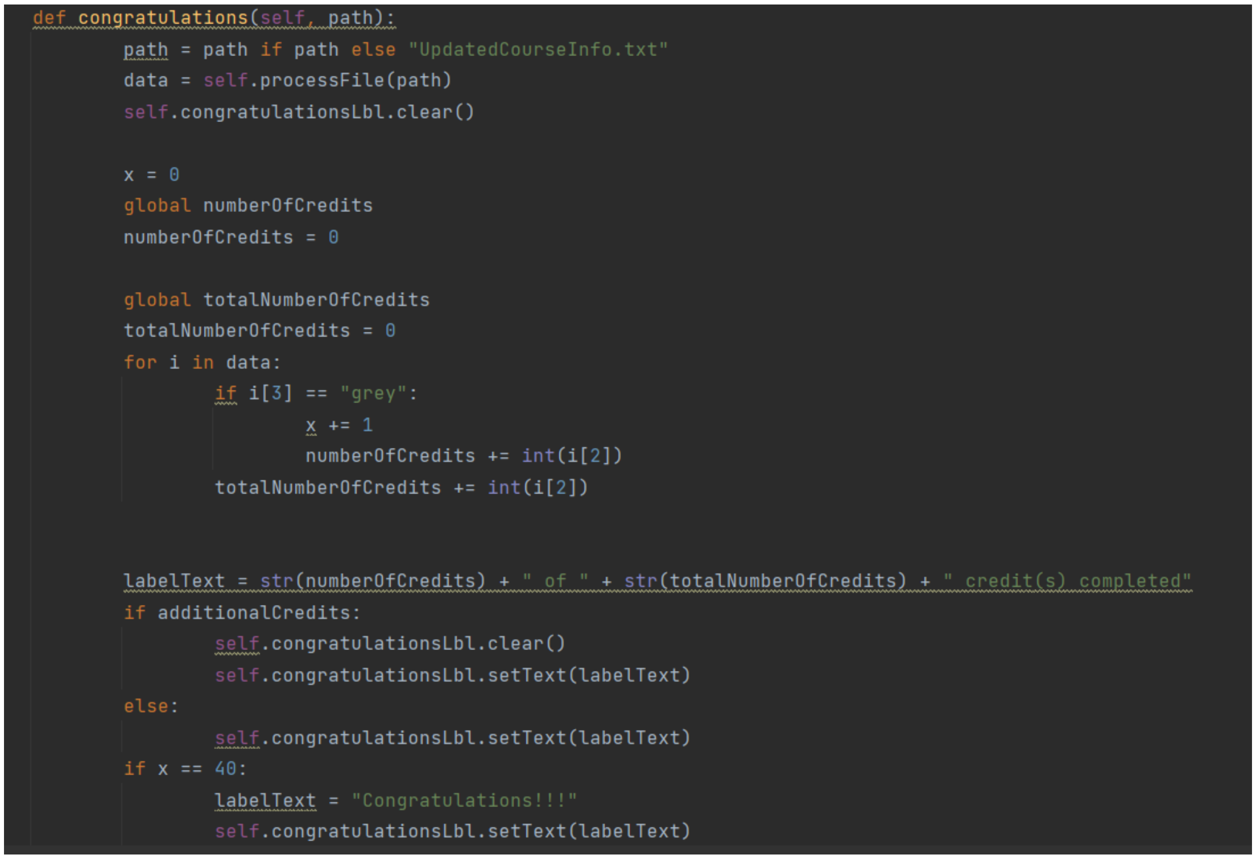


```
elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                               "background-color: rgb(211, 211, 211);\n"
                                               "border: 1px solid black;"):
        classColor = "background-color: rgb(211, 211, 211);\n"
        color = 'grey'
        frame = 'NotC'


elif buttonArray[buttonIndex].styleSheet() == ("border-radius: 10px;\n"
                                               "background-color: rgb(211, 211, 211);\n"
                                               "border: 3px solid blue;"):
        classColor = "background-color: rgb(211, 211, 211);\n"
        color = 'grey'
        frame = 'C'


lines[buttonIndex] = (f"{line}, {color}, {frame}\n")

w = open(fileName, 'w')
for x in lines:
        w.write(x)
w.close()

buttonArray[buttonIndex].setText(f"{courseCodeFinal} \n{courseIDFinal} \n {numberOfCreditsFinal} credits")

return line
```

Algorithm description: The algorithm takes the course information from the user and updates the course information in the txt file; it also, updates the number of credits and the total number of credits in the main window.

- **congratulations algorithm and pseudo code:**

```
congratulations(path):
    Take the txt path.
    Read the content in the txt file and save it to the data variable.
    Clear the congratulations label.
    for i in data:
        if a course color is grey:
            count the number of credits.
            count the number of courses with color grey
            count the total number of credits
    Display the number of credits already taken our of total

    if number of courses with color grey is 40:
        all courses are taken
        display a congratulation label.
```



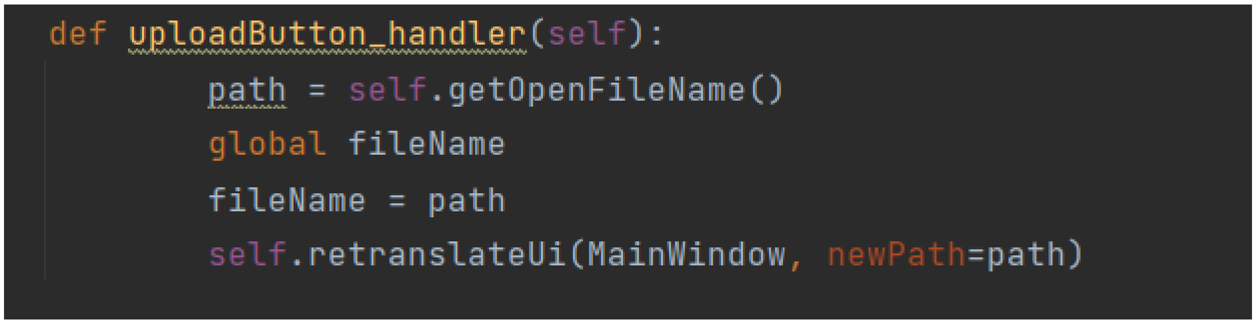
```
def congratulations(self, path):
        path = path if path else "UpdatedCourseInfo.txt"
        data = self.processFile(path)
        self.congratulationsLbl.clear()

        x = 0
        global numberOfCredits
        numberOfCredits = 0

        global totalNumberOfCredits
        totalNumberOfCredits = 0
        for i in data:
                if i[3] == "grey":
                        x += 1
                        numberOfCredits += int(i[2])
                totalNumberOfCredits += int(i[2])


        labelText = str(numberOfCredits) + " of " + str(totalNumberOfCredits) + " credit(s) completed"
        if additionalCredits:
                self.congratulationsLbl.clear()
                self.congratulationsLbl.setText(labelText)
        else:
                self.congratulationsLbl.setText(labelText)
        if x == 40:
                labelText = "Congratulations!!!"
                self.congratulationsLbl.setText(labelText)
```

Algorithm description:

The algorithm shows the user the number of credits taken already. If all courses are taken, it displays the congratulations label.

❖ **uploadbutton_Handler algorithm and pseudo code:**

```
Call the getOpenFileName()
        Get the file path from it
        Re-translate the information to the window
```



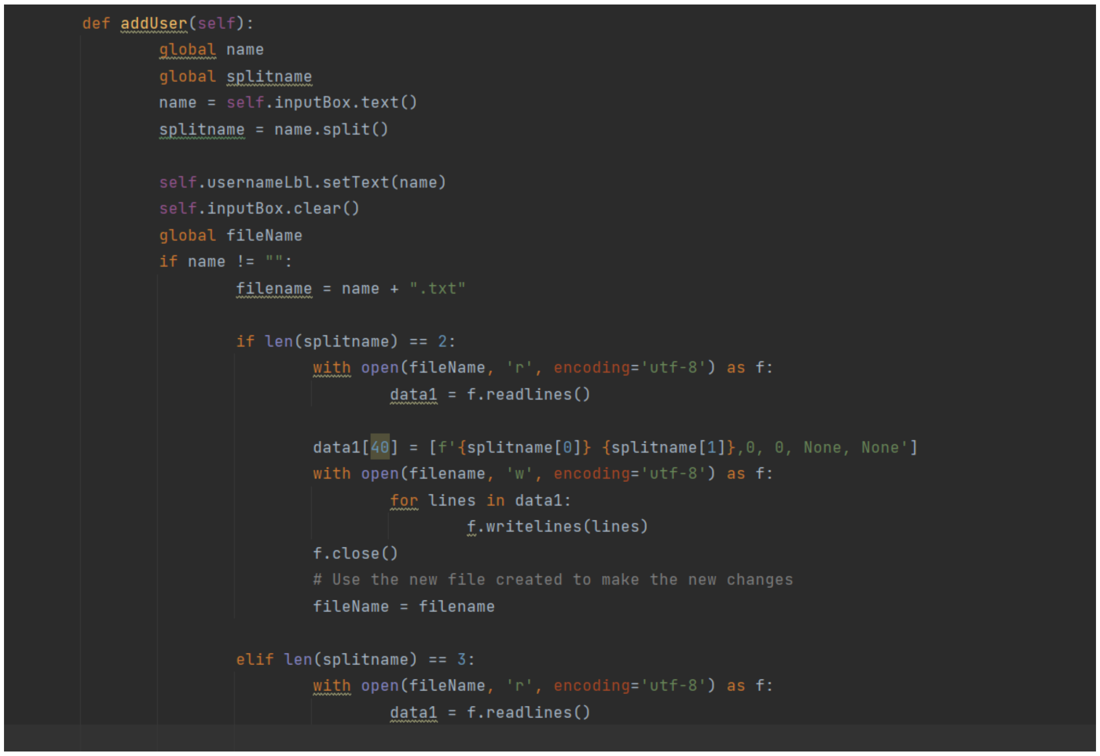
```
def uploadButton_handler(self):
        path = self.getOpenFileName()
        global fileName
        fileName = path
        self.retranslateUi(MainWindow, newPath=path)
```

Algorithm description:

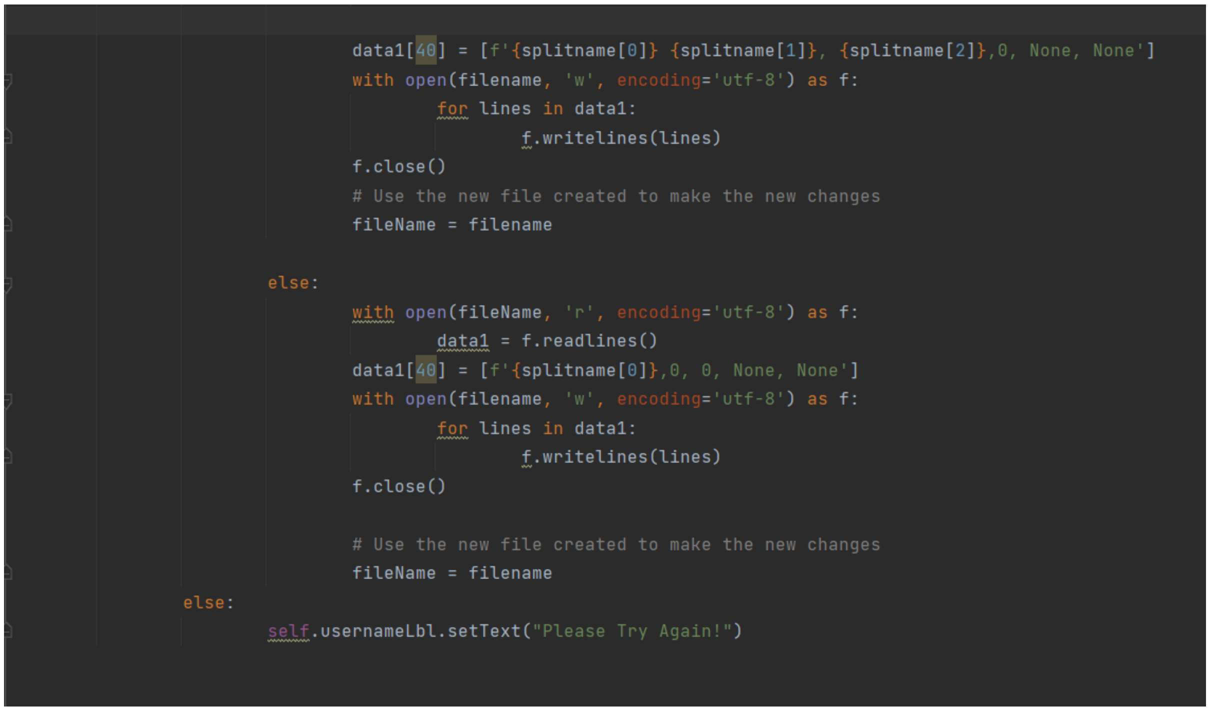
The algorithm makes it possible for the user to upload a new text file into the program and displays the information from that text file.

❖ **AddUser algorithm and pseudo code:**

```
Get name from input box
        Split the name into words
        Display the name
        If input not empty :
                    Create txt file based on the student's name
                    If length(input) = 2 words
                        Open the txt file in and close when done
                        Edit line 41 so it contains the name  in the proper format for 2 words
                   elIf length(input) = 3 words
                       Open the txt file in read mode and close when done
                        Edit line 41 so it contains the name  in the proper format for 3 words
                   Else:
                        Open the txt file in read mode and close when done
                        Edit line 41 so it contains the name  in the proper format for 3 words
            Else :
                  Show message to try again
```



```python
def addUser(self):
        global name
        global splitname
        name = self.inputBox.text()
        splitname = name.split()

        self.usernameLbl.setText(name)
        self.inputBox.clear()
        global fileName
        if name != "":
                filename = name + ".txt"

                if len(splitname) == 2:
                        with open(fileName, 'r', encoding='utf-8') as f:
                                data1 = f.readlines()

                        data1[40] = [f'{splitname[0]} {splitname[1]},0, 0, None, None']
                        with open(filename, 'w', encoding='utf-8') as f:
                                for lines in data1:
                                        f.writelines(lines)
                        f.close()
                        # Use the new file created to make the new changes
                        fileName = filename

                elif len(splitname) == 3:
                        with open(fileName, 'r', encoding='utf-8') as f:
                                data1 = f.readlines()
```



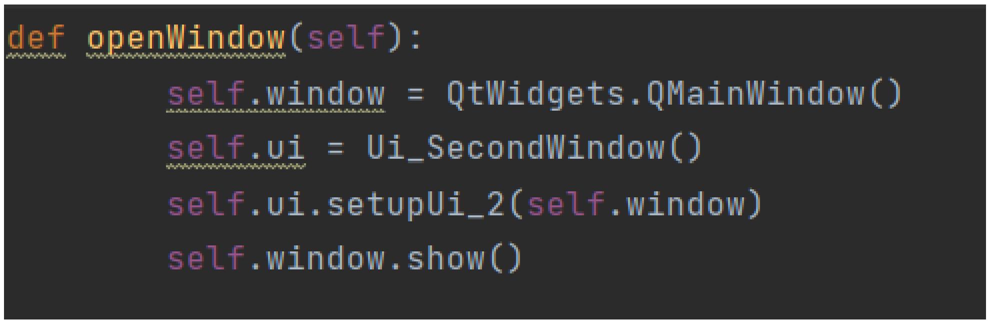
```
                data1[40] = [f'{splitname[0]} {splitname[1]}, {splitname[2]},0, None, None']
                with open(filename, 'w', encoding='utf-8') as f:
                    for lines in data1:
                        f.writelines(lines)
                f.close()
                # Use the new file created to make the new changes
                fileName = filename

            else:
                with open(fileName, 'r', encoding='utf-8') as f:
                    data1 = f.readlines()
                data1[40] = [f'{splitname[0]},0, 0, None, None']
                with open(filename, 'w', encoding='utf-8') as f:
                    for lines in data1:
                        f.writelines(lines)
                f.close()

                # Use the new file created to make the new changes
                fileName = filename
        else:
            self.usernameLbl.setText("Please Try Again!")
```

Algorithm description

This algorithm adds the name of the user to the text file based on the user's input. It saved the name on the last line of the text file in the proper format based on how long the user's name is. It generates a new file with all the correct information named after the user so this way each student has its own file. When a file is uploaded with a user's name in the last line, it displays the user's name inserted.

- **OpenWindow algorithm and pseudo code:**

```
Get all the widgets
    Setup the window
    Show the window
```

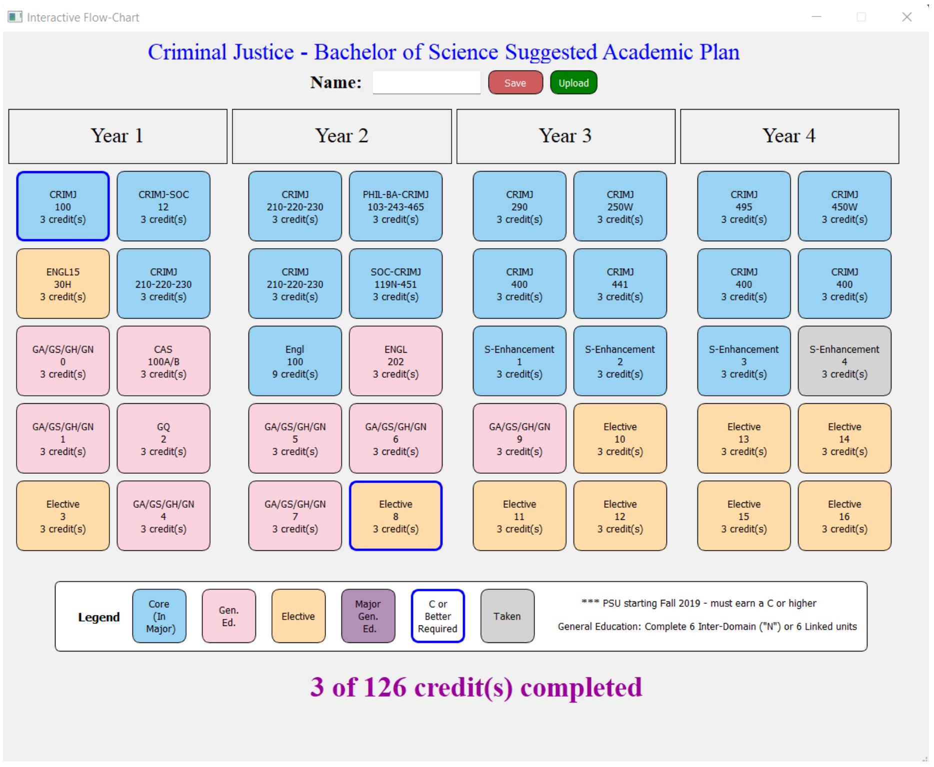


```
def openWindow(self):
        self.window = QtWidgets.QMainWindow()
        self.ui = Ui_SecondWindow()
        self.ui.setupUi_2(self.window)
        self.window.show()
```

Algorithm description

This algorithm generated the second user interface window.

4. Visual interface description



Criminal Justice - Bachelor of Science Suggested Academic Plan

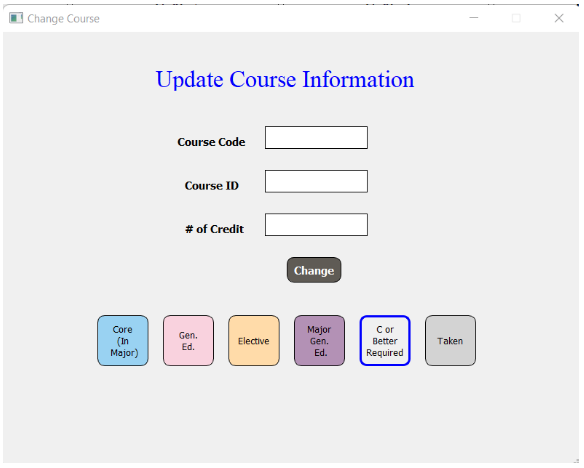
Year 1		Year 2		Year 3		Year 4	
CRIMJ 100 3 credit(s)	CRIMJ-SOC 12 3 credit(s)	CRIMJ 210-220-230 3 credit(s)	PHIL-BA-CRIMJ 103-243-465 3 credit(s)	CRIMJ 290 3 credit(s)	CRIMJ 250W 3 credit(s)	CRIMJ 495 3 credit(s)	CRIMJ 450W 3 credit(s)
ENGL15 30H 3 credit(s)	CRIMJ 210-220-230 3 credit(s)	CRIMJ 210-220-230 3 credit(s)	SOC-CRIMJ 119N-451 3 credit(s)	CRIMJ 400 3 credit(s)	CRIMJ 441 3 credit(s)	CRIMJ 400 3 credit(s)	CRIMJ 400 3 credit(s)
GA/GS/GH/GN 0 3 credit(s)	CAS 100A/B 3 credit(s)	Engl 100 9 credit(s)	ENGL 202 3 credit(s)	S-Enhancement 1 3 credit(s)	S-Enhancement 2 3 credit(s)	S-Enhancement 3 3 credit(s)	S-Enhancement 4 3 credit(s)
GA/GS/GH/GN 1 3 credit(s)	GQ 2 3 credit(s)	GA/GS/GH/GN 5 3 credit(s)	GA/GS/GH/GN 6 3 credit(s)	GA/GS/GH/GN 9 3 credit(s)	Elective 10 3 credit(s)	Elective 13 3 credit(s)	Elective 14 3 credit(s)
Elective 3 3 credit(s)	GA/GS/GH/GN 4 3 credit(s)	GA/GS/GH/GN 7 3 credit(s)	Elective 8 3 credit(s)	Elective 11 3 credit(s)	Elective 12 3 credit(s)	Elective 15 3 credit(s)	Elective 16 3 credit(s)

*** PSU starting Fall 2019 - must earn a C or higher

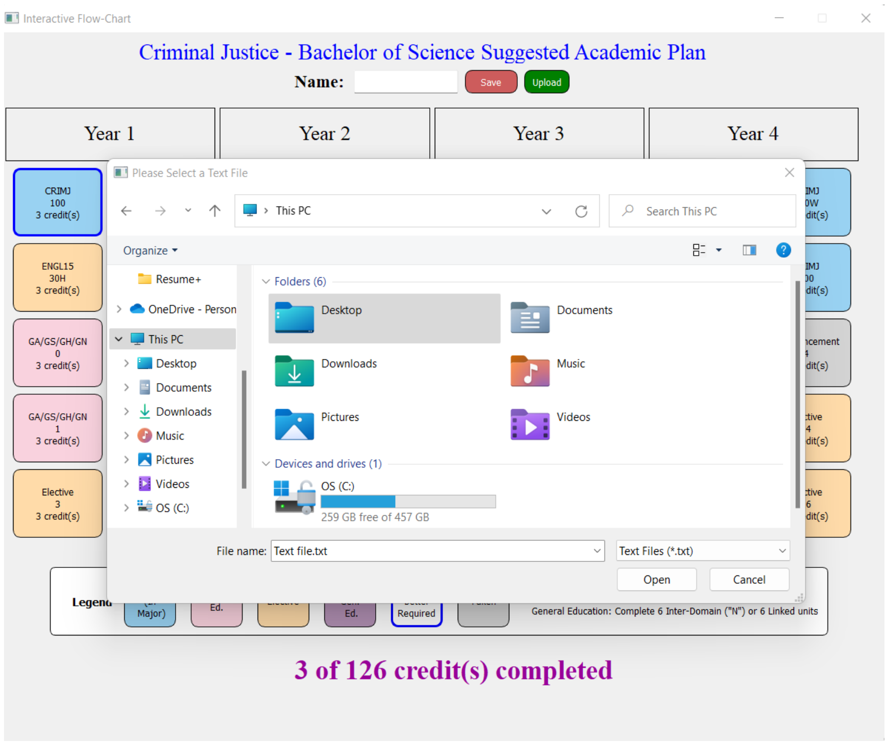
General Education: Complete 6 Inter-Domain ("N") or 6 Linked units

3 of 126 credit(s) completed

The user interface above is the main window of the program. There are 4 years in total and each year has 10 classes. Each class is represented by a visual block. Each block contains pieces of information about the courses. At the top of the main window, there is the title and below the title, we have the input box. We have two buttons: the save button and the upload button. At the bottom of the main window is the legend that explains the colors and a line that shows the number of credits taken.

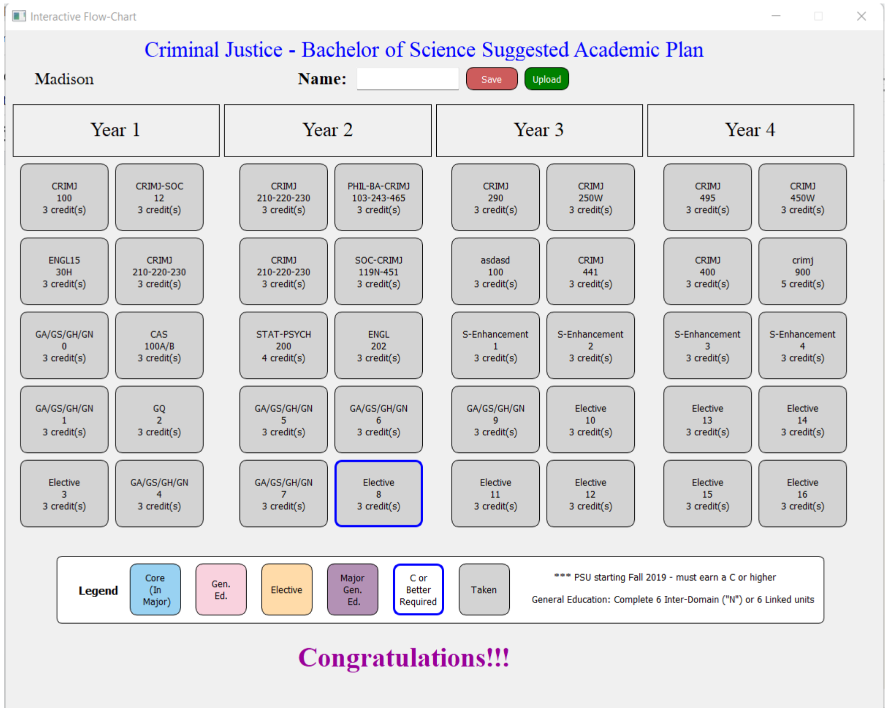


The user interface above is the second window. It opens after clicking on one of the classes and it allows the user to edit the contents of their course. The user can also click on one of the colors for the course and then click Change to save the changes.



3 of 126 credit(s) completed

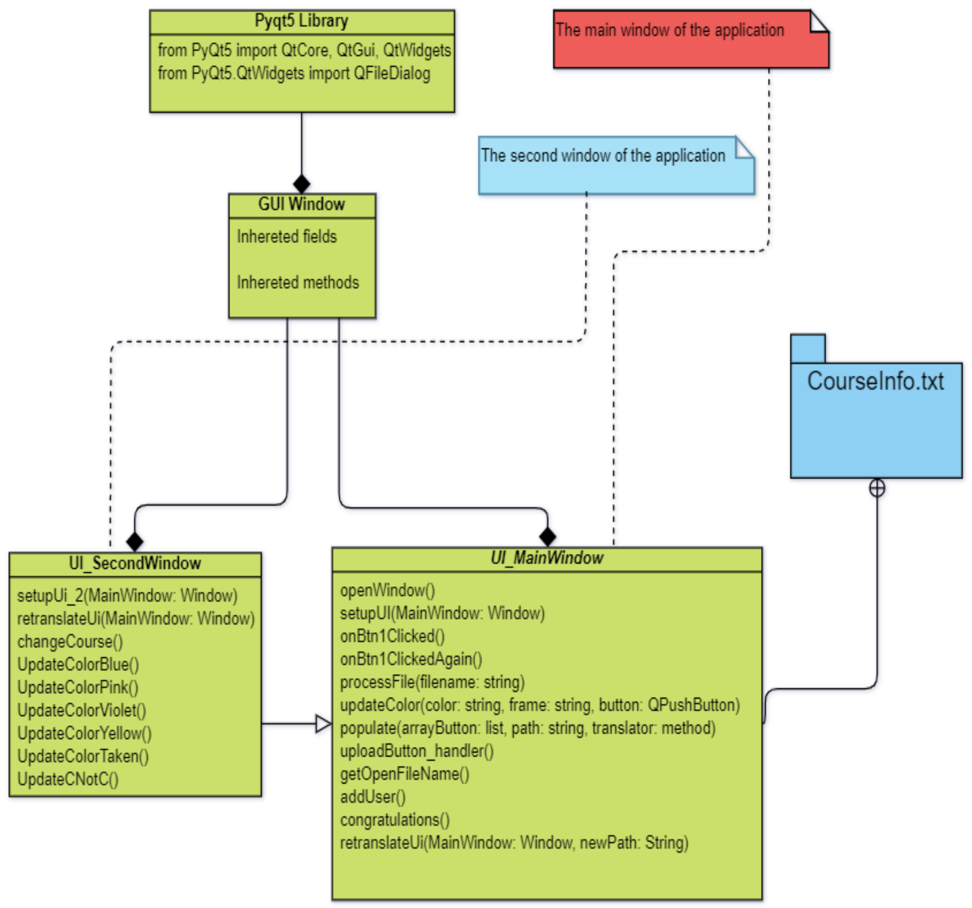
The above screenshot shows the functionality of the upload button. The user can upload txt files containing different information and the program will be populated based on the new txt file uploaded.



Congratulations!!!

This is an example of when all the courses are taken. Each class is gray as they all are taken and the Congratulations message is showing at the end. You can see the name of the student on the top left, the name appears every time a personalized file is uploaded so the advisor knows with which student they are dealing.

Class diagram:


Pyqt5 Library
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog
The main window of the application
The second window of the application
GUI Window
Inhereted fields
Inhereted methods
CourseInfo.txt
UI_SecondWindow
setupUi_2(MainWindow: Window)
retranslateUi(MainWindow: Window)
changeCourse()
UpdateColorBlue()
UpdateColorPink()
UpdateColorViolet()
UpdateColorYellow()
UpdateColorTaken()
UpdateCNotC()
UI_MainWindow
openWindow()
setupUI(MainWindow: Window)
onBtn1Clicked()
onBtn1ClickedAgain()
processFile(filename: string)
updateColor(color: string, frame: string, button: QPushButton)
populate(arrayButton: list, path: string, translator: method)
uploadButton_handler()
getOpenFileName()
addUser()
congratulations()
retranslateUi(MainWindow: Window, newPath: String)

Poster link: Poster

5. Testing

Unit testing.

Unit testing is part of the software development process in which small parts of an application, called units, are individually tested for proper operation. We tested individual methods separately. Each group member tested the methods they worked on individually.

Integration/System Testing

Integration testing is the part of software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before functional testing. Once the unit testing was done with each method. We combined all methods and tested them.

Functional Testing

After the integration tests are performed, more complex levels of tests are used. The functional testing process in which software is tested to ensure that it conforms with all business requirements and to ensure that it has all the required functionality for the software to be used by the end-user without issues. The software was tested using the list of requirements provided by the user.

Acceptance Testing

Acceptance Testing is where a system is tested for acceptability by the end user. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery into production. The client asked to use the application for a few days to check if it meets the expectations.

Demo video link:

https://psu.mediaspace.kaltura.com/media/Hansi+Seitaj%27s+Zoom+Meeting/1_41j4889b?st=0&ed=285

After meeting with the client, we came up with this demonstration so that the client, a user or a developer be able to go over it and understand in a few steps how to run the program. Furthermore, this demo contains a big picture of the project. Thank you for using the program, and for your time!

6. CONCLUSION

Client Feedback:

Interactive Degree Requirements Flow Chart Assessment by Lisa Morris - Fall 2022

Fall of 2022 I worked with Hansi Seitaj and his team Lukas Belashov, Almaz Akhunbaev, and Eni Vejseli, on the creation of an executable program to assist Criminal Justice advisors with advising students. Penn State Abington offers both a Bachelor of Arts and a Bachelor of Science in Criminal Justice and I was very interested in finding a way to make degree requirements easier. I met with Hansi a number of times throughout the semester. He was very passionate about the project and attentive to suggestions I made regarding edits. Last week I received an executable file from him and … it works! I finally found time this week to put it to the test by creating a document for a student. Although the concept is relatively simple in that once requirements are filled – the tile will change from an informative color to grey … the results definitely help with planning purposes and will aid students in seeing what degree requirements are left to complete and get them to graduation. (simple in application, but I'm sure it took a considerable amount of brainstorming and code writing to be successful.)   I am looking forward to making a number of personalized documents for my assigned advisees over winter break to be prepared for the start of the Spring semester. I would like to end by saying thank you for the opportunity to have my idea for a project come to fruition and it was a pleasure chatting with Hansi.

Our conclusions:

From the client feedback, we can conclude that we were able to achieve our goal and met all client requirements. The application provides a user-friendly visualization of a student's academic path in a Criminology major or any other major. We gained experience in working with the PyQt5 library and with the QT Designer app. We managed to create multiple algorithms to implement the logic in the back end and came up with a user-friendly front end. To make it easier for the client, we created an executable file to make it portable and delivered it to the client. This project was really helpful for us as we gained experience working together as a team and learned how to work with a real client.

7. REFERENCES

https://pypi.org/project/PyQt5/

https://blog.nfocus.co.uk/testing-methodologies-five-core-components-of-testing

Client-provided slideshow.