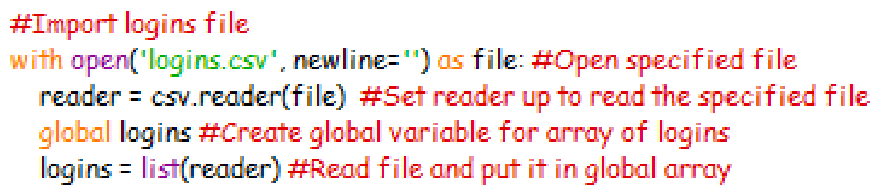
Development

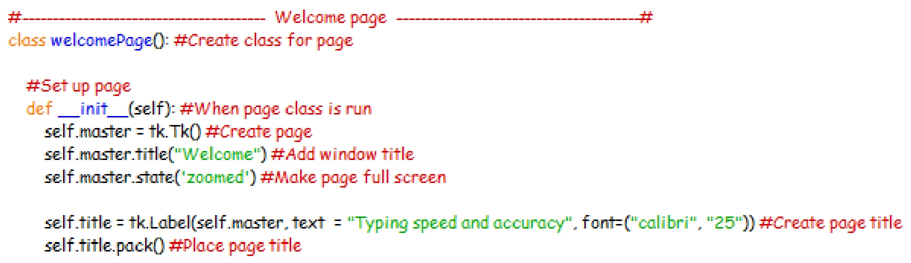
What modules?

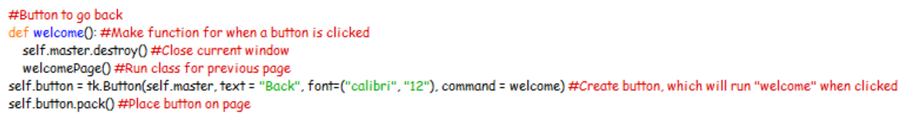
- I will be using “tkinter,” as it allows for easy, simple GUI creation, and I have some experience in using it in the past
- I will also be using the “random” module, to randomise which words are presented to the user
- The python CSV module has lots of use in my program, to read from and save to CSV files, for my data storage
- “datetime” will also be used, to save when exactly the tests were done, to them be presented on the progress checking page
- The “time” module will be used to time how long it takes the user to type the different words

Techniques used

Technique	Justification

File importing

`#Import logins file`
`with open('logins.csv', newline='') as file: #Open specified file`
`    reader = csv.reader(file)  #Set reader up to read the specified file`
`    global logins #Create global variable for array of logins`
`    logins = list(reader) #Read file and put it in global array`
 	
The beginning of the program includes all the necessary imports of the CSV data storage files. This is done using the CSV reader. It creates a global array for each of the CSV files to be manipulated easer within the program.
 	
Reference: https://docs.python.org/3/library/csv.html
	By using a 2D array to store my data, it will be very easy to access data from it, add things, remove things and change things from the CSV files, considering the data is already saved in rows and columns in the file.

Initialising pages

`#-------------------------------------- Welcome page --------------------------------------#`
`class welcomePage(): #Create class for page`
 	
`    #Set up page`
`    def __init__(self): #When page class is run`
`        self.master = tk.Tk() #Create page`
`        self.master.title("Welcome") #Add window title`
`        self.master.state('zoomed') #Make page full screen`
 	
`        self.title = tk.Label(self.master, text  = "Typing speed and accuracy", font=("calibri", "25")) #Create page title`
`        self.title.pack() #Place page title`
 	
Each page is its own class, which when initialised will create a tkinter window.
	By separating each part into separate classes, variables can be kept separate easily and when working on bugs, the code is easily readable due to the segmentation of my code.

Navigating pages

`#Button to go back`
`def welcome(): #Make function for when a button is clicked`
`    self.master.destroy() #Close current window`
`    welcomePage() #Run class for previous page`
`self.button = tk.Button(self.master, text = "Back", font=("calibri", "12"), command = welcome) #Create button, which will run "welcome" when clicked`
`self.button.pack() #Place button on page`
 	
This allows the user to get between each page.
	To get between pages, a button is used which closes the current tkinter window and goes to the class for the next one.

Checking usernames/passwords

```
#Check username and password match known username and password
for i in range(1, len(logins)):
    if logins[i][0] == self.loginUsername: #Check username is correct
        if logins[i][1] == self.loginPassword: #Check password is correct
            global user
            user = i

            #Go to next page
            self.master.destroy()
            menuPage()

        else:
            self.messageText.configure(text = "Incorrect username or password")
    else:
        self.messageText.configure(text = "Incorrect username or password")
```

Usernames and passwords are checked against a database. This is done by looping through each username to check if it is correct, and if it it is checks if the password is the same as the user inputted as well. If they both do then it will go to the next page, if not it will give the user a message of what was incorrect.

This technique allows usernames and passwords to be checked as well as giving the user feedback for what they may have inputted incorrectly.

Showing progress data

```
#Add to list of previous user data
for i in range(1, len(evaluation)):
    currentRow = str(evaluation[i][0])+" - "+evaluation[i][1]+" - "+str(int(evaluation[i][2]))+"% - "+str(evaluation[i][3])
    progressList.insert("end", currentRow)
```

On each of the progress check pages, information is taken from the pages and put into a list. The program will loop through each of the rows, adding them every time.

This allows the entire file to be shown to the user through a list, through which they can scroll.

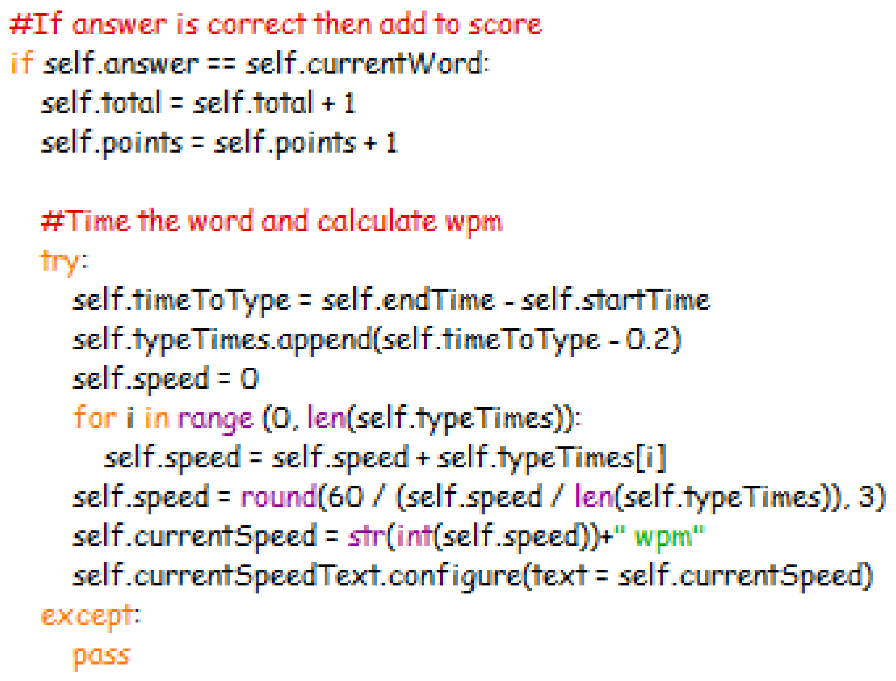
New backspace key functionality

```
#When backspace key pressed
def backspacePressed(self, event):
    self.answer = self.textBox.get()
    if self.answer != "":
        self.total = self.total + 1

        #Add data to database
        self.split = accuracy[self.currentWordID][user].split(" of ")
        self.split[1] = int(self.split[1])+1
        accuracy[self.currentWordID][user] = str(self.split[0]) + " of " + str(self.split[1])

    #Clear box
    self.textBox.delete(0, "end")
```

As is mentioned in the user instructions in the program, the backspace button clears the entire box and counts as a word being entered incorrectly.

This ensures that users cannot get a perfect accuracy score through deleting an incorrectly typed letter and retyping it, making the data collected about their accuracy accurate.

Calculting data to be stored



```
#If answer is correct then add to score
if self.answer == self.currentWord:
    self.total = self.total + 1
    self.points = self.points + 1

    #Time the word and calculate wpm
    try:
        self.timeToType = self.endTime - self.startTime
        self.typeTimes.append(self.timeToType - 0.2)
        self.speed = 0
        for i in range (0, len(self.typeTimes)):
            self.speed = self.speed + self.typeTimes[i]
        self.speed = round(60 / (self.speed / len(self.typeTimes)), 3)
        self.currentSpeed = str(int(self.speed))+" wpm"
        self.currentSpeedText.configure(text = self.currentSpeed)
    except:
        pass
```

For the evaulation game mode, data must be stored in the databases about the user's progress, and so when the enter key is pressed, and the user has entered the word correctly data must be recorded.

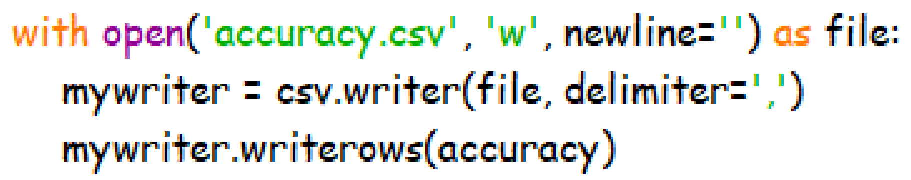
The number of times it took the word to be answered is calculated through adding 1 to the total each time the user enters either a correct or incorrect answer (self.total) and adding 1 to the number of points the user has (self.points) when the word is answered correctly.

The time for the word to be typed is calculated by starting a timer when the word appears and stopping the timer once it has been answered correctly and calculating the time difference (self. timeToType).

0.2s is taken off from this value, due to human reaction times, and then an average wpm is found by adding all the previous times together and dividing by the total.

60s is then divided by this number to give the average speed in wpm.

Saving data to databases



```
with open('accuracy.csv', 'w', newline='') as file:
    mywriter = csv.writer(file, delimiter=',')
    mywriter.writerows(accuracy)
```

Whenever these CSV files need to be saved, the CSV writer is used to write each row of the array.

Reference: https://docs.python.org/3/library/csv.html?highlight=csv%20writer#csv.writer

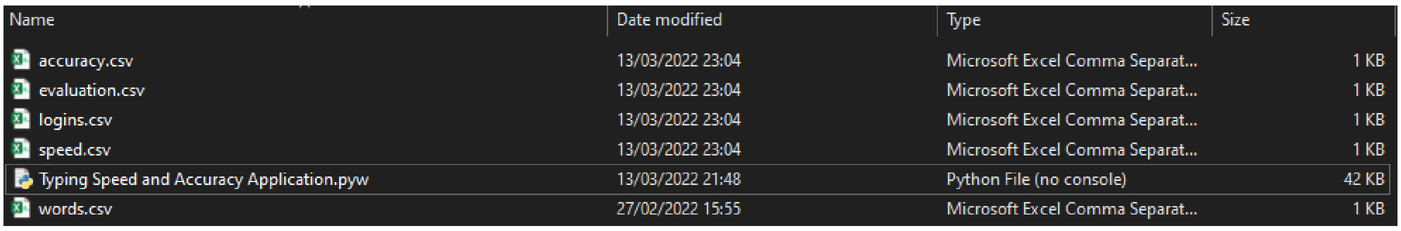
To store the data in a non-volatile way, in a CSV file, the CSV writer needs to be used.

The CSV writer opens the file, and then writes an array onto it. This file is then saved.

Choosing next word
 	
`#Next word`
`self.currentWordID = random.choices(self.speedValues, self.speedProbabilities)[0]`
 	
Suggesting words based upon the user's scores at speed/accuracy. The random.choices method is used for this.
 	
Reference: https://www.w3schools.com/python/ref_random_choices.asp
	This method allows an item from a list to be chosen with the probability of each item being chosen dependent on a separate list of values.
	
	The list of values used is from the databases of accuracy/speed scores.

File structure

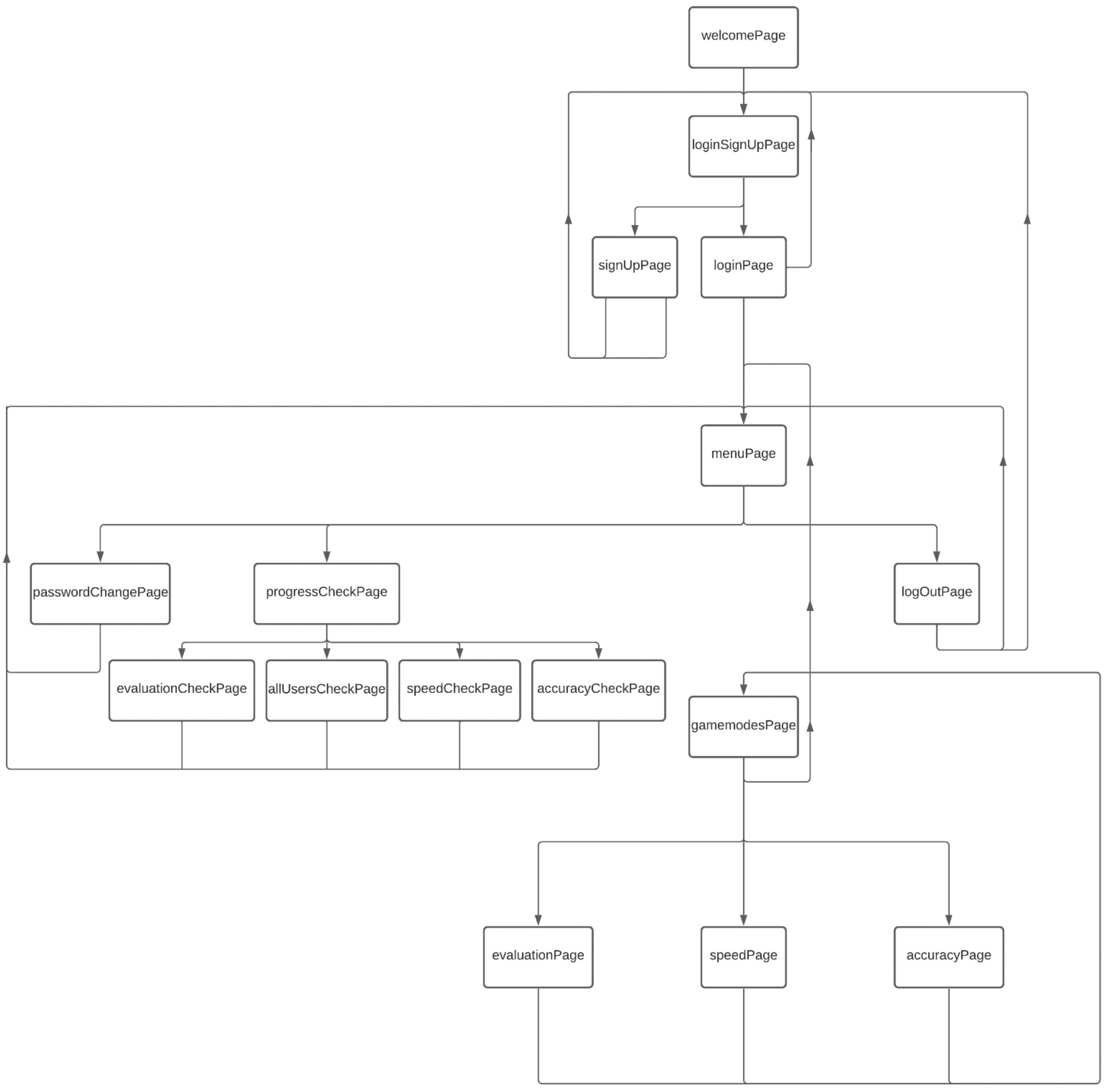
My program requires 6 files to run: The program itself, as well as 5 CSV files used to store data. These files must all be stored in the same folder for the program to work. I have saved the program as a .pyw file, as this means that the python console will not open when the program is run, as the console is not needed for anything and so would just be getting in the way of the users.



Name	Date modified	Type	Size
accuracy.csv	13/03/2022 23:04	Microsoft Excel Comma Separat...	1 KB
evaluation.csv	13/03/2022 23:04	Microsoft Excel Comma Separat...	1 KB
logins.csv	13/03/2022 23:04	Microsoft Excel Comma Separat...	1 KB
speed.csv	13/03/2022 23:04	Microsoft Excel Comma Separat...	1 KB
Typing Speed and Accuracy Application.pyw	13/03/2022 21:48	Python File (no console)	42 KB
words.csv	27/02/2022 15:55	Microsoft Excel Comma Separat...	1 KB

Usage of classes

Using a diagram like the navigation chart in section B, a class dependency diagram can be created.



List of words

The database of words which I am using in my program, are from a website which lists the most common five letter words. I chose these as five letters seems a reasonable length and I chose at least one word which began with each letter to ensure that the user would be practising typing each letter.
https://www.unscramblerer.com/common-five-letter-words/