Major Software Assignment

Carter Fitzgerald: u3240494

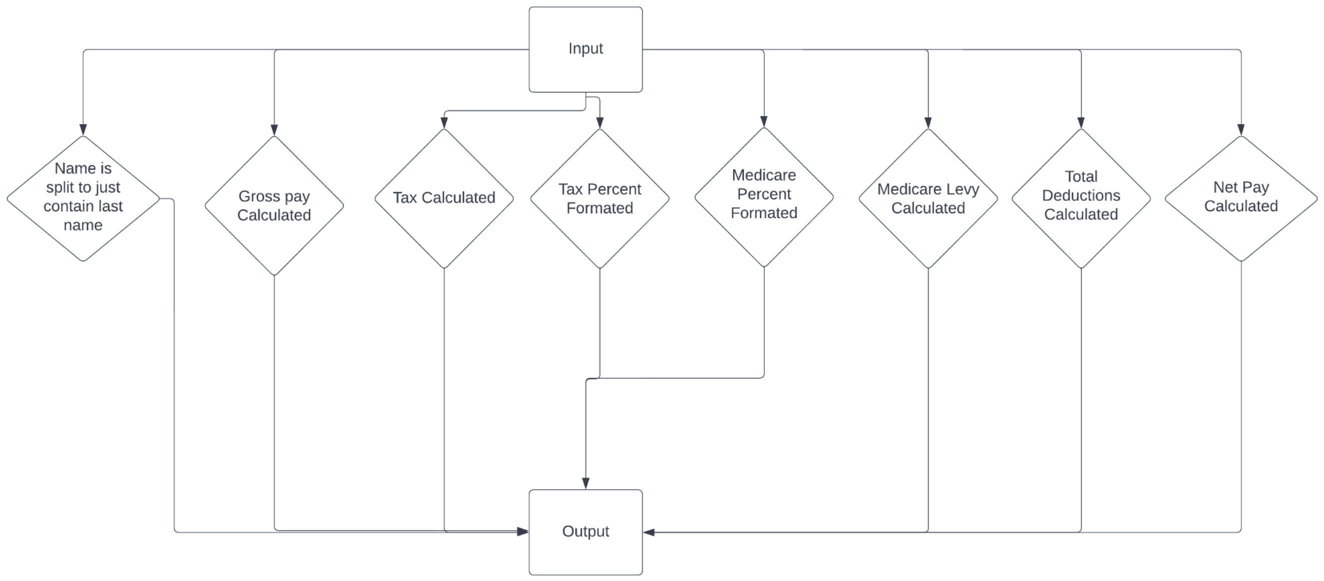
Part A

Project 1: Payroll Calculator

Requirements Analysis:

For this project the goal is to take an input of an employee's payroll information and tax information and use that input to calculate the employee's gross pay, deductions including, ato tax, medicare levy and total deductions and finally calculate the net pay of the employee and display this to the user using both console and GUI programs.

Algorithm Design:



IPO Chart/Hierarchy Chart / UML:

Input	Processing	Output
Name Hours worked Hourly rate Ato withholding rate Medicare rate	Name split into first and last name using .split() Gross pay calculated by multiplying the hours worked with the hourly rate Tax calculated by multiplying gross pay by ato tax withholding rate Medicare levy calculated by multiplying the gross pay by the medicare rate Medicare percentage calculated by multiplying medicare rate by 100 tax percentage calculated by multiplying ato withholding rate by 100 Total deductions calculated by adding the medicare levy and tax Net pay calculated by subtracting total deductions from gross pay.	Employee last name Hours worked Pay rate Gross pay Deductions ATO tax Medicare levy Total deductions Net pay

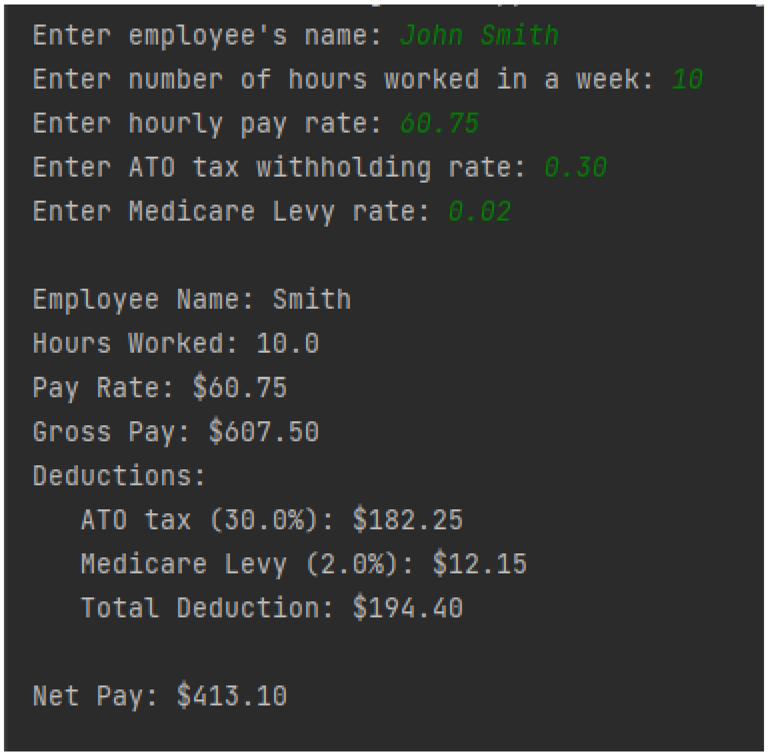
Brief Code Walkthrough - Console and GUI Program:

Console Code:
The console code is relatively simple, it first asks the user for a range of inputs including, Name, hours worked, hourly rate, ato withholding rate and medicare levy rate. Then I have taken that data, split the name to get just the last name. I calculate the gross pay, tax, medicare levy, total deductions and net pay. In the same section I format the percentage inputs given to be correct for the output. Then an f string is created which holds all the information regarding the employees payroll data. The f string is then printed.

GUI Code:

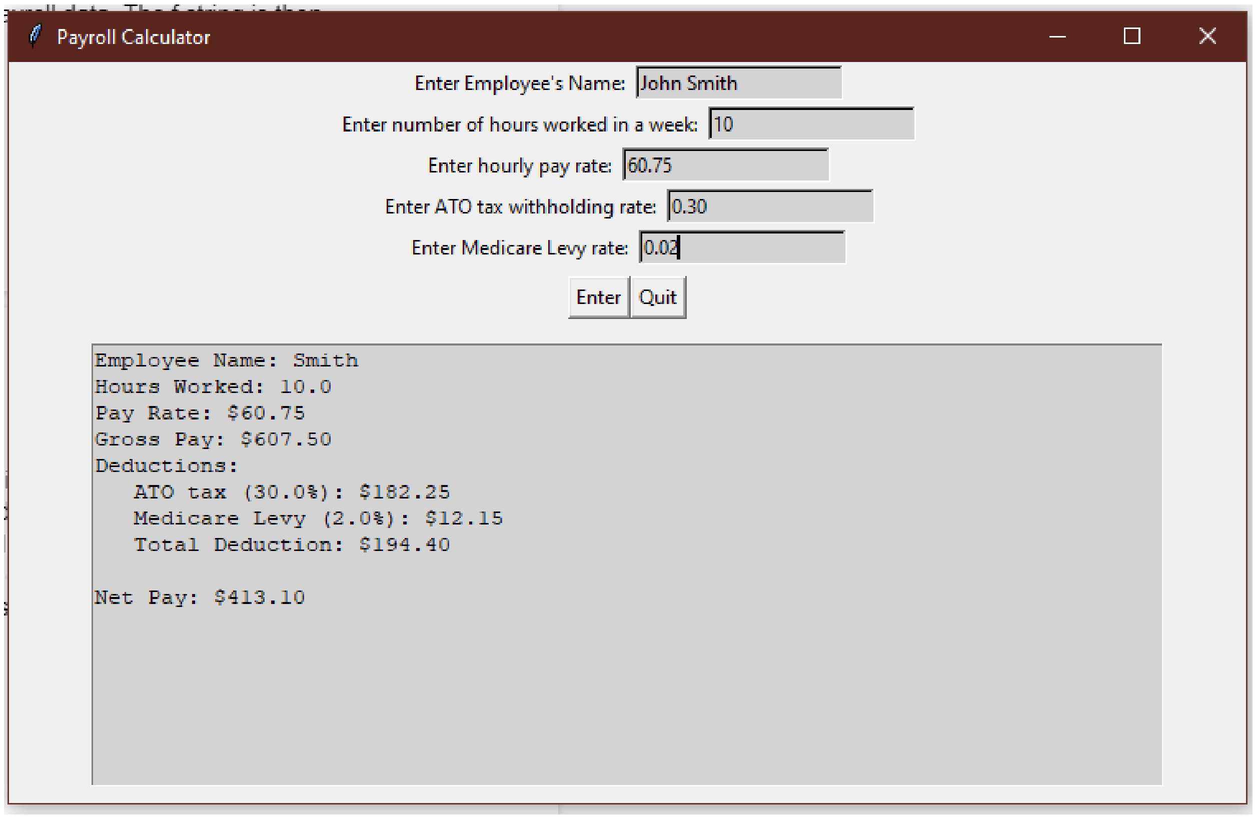
The GUI code starts by importing tkinter, then there is a function called result which contains all the console code that has been tweaked to get the input from the entry widgets. Then the main __init__ method contains all the GUI setup and frames, then there are a series of labels and entry widgets which collect the users input, finally there are 2 buttons an enter and a quit, which runs the result function which then outputs the result to a text area.

Evidence of Testing:
Console Testing:



```
Enter employee's name: John Smith
Enter number of hours worked in a week: 10
Enter hourly pay rate: 60.75
Enter ATO tax withholding rate: 0.30
Enter Medicare Levy rate: 0.02

Employee Name: Smith
Hours Worked: 10.0
Pay Rate: $60.75
Gross Pay: $607.50
Deductions:
   ATO tax (30.0%): $182.25
   Medicare Levy (2.0%): $12.15
   Total Deduction: $194.40

Net Pay: $413.10
```

GUI Testing:


Payroll Calculator
Enter Employee's Name: John Smith
Enter number of hours worked in a week: 10
Enter hourly pay rate: 60.75
Enter ATO tax withholding rate: 0.30
Enter Medicare Levy rate: 0.02
Enter
Quit

```
Employee Name: Smith
Hours Worked: 10.0
Pay Rate: $60.75
Gross Pay: $607.50
Deductions:
   ATO tax (30.0%): $182.25
   Medicare Levy (2.0%): $12.15
   Total Deduction: $194.40

Net Pay: $413.10
```

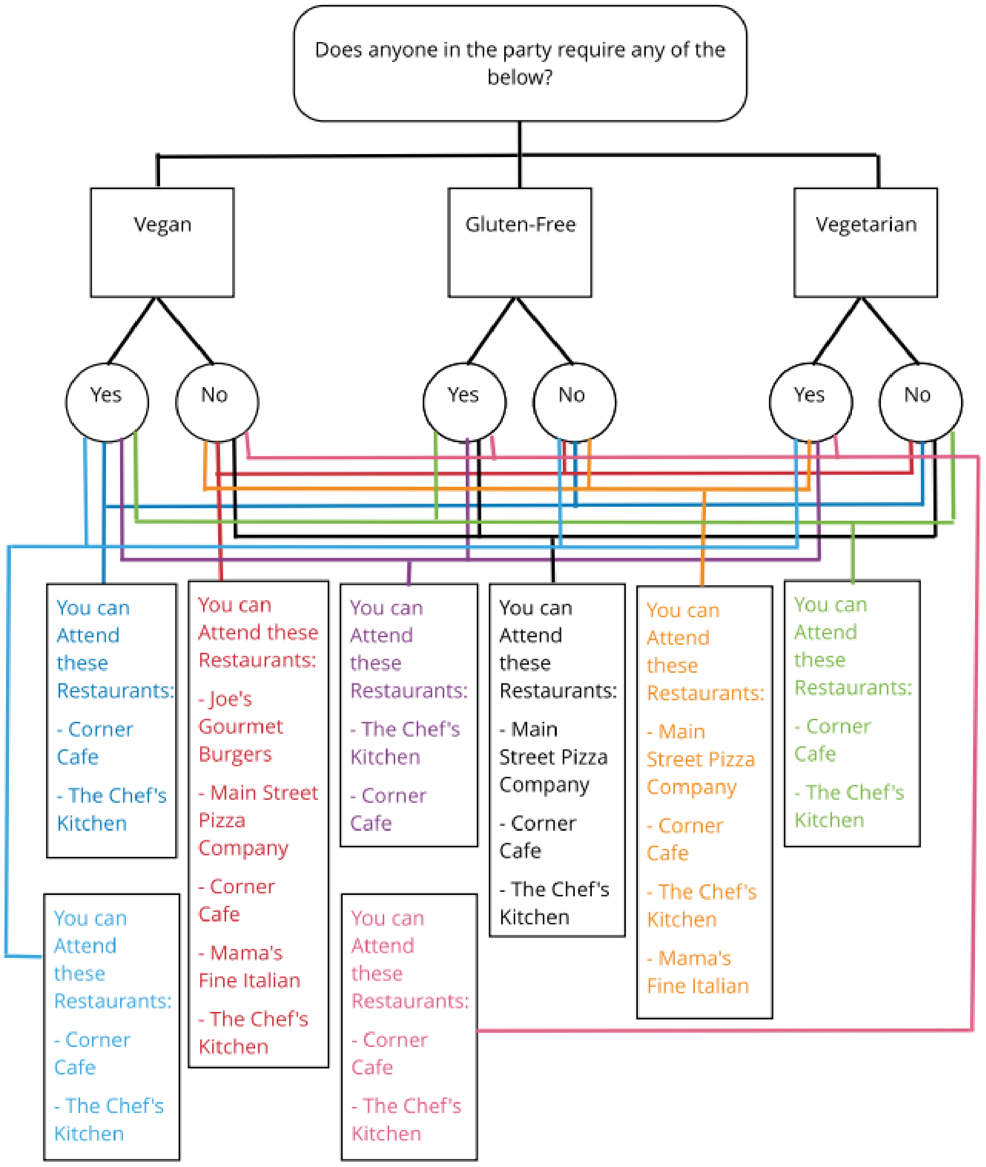
Special Features:
Due to poor time management I was unable to include any special features, but if I was to do the project again, I would include input validation that notifies the user when the data inputted is unacceptable. I also tried to get a scrollbar in the text area of the GUI program but I couldn't get it to work in time.

Reflection:
After completing this project I was very happy, I felt the project was simple, easy to understand and was made up of simple code. The transition from console code to GUI code was interesting as a lot of small things can cause the console code not to work, and things like having to change the way the program accesses the users input.

Project 2: Restaurant Selector

Requirements Analysis:
For this project the user has to provide whether anyone in their group or party are vegetarian, vegan or gluten-free. Then there are a list of restaurants each that cater to different food constraints, and some to none or all. It is the program's job to determine which restaurants the group/party can attend that will satisfy everyone's needs.

Algorithm Design:


Does anyone in the party require any of the below?
Vegan
Gluten-Free
Vegetarian
Yes
No
Yes
No
Yes
No
You can Attend these Restaurants: - Corner Cafe - The Chef's Kitchen
You can Attend these Restaurants: - Joe's Gourmet Burgers - Main Street Pizza Company - Corner Cafe - Mama's Fine Italian - The Chef's Kitchen
You can Attend these Restaurants: - The Chef's Kitchen - Corner Cafe
You can Attend these Restaurants: - Main Street Pizza Company - Corner Cafe - The Chef's Kitchen
You can Attend these Restaurants: - Main Street Pizza Company - Corner Cafe - The Chef's Kitchen - Mama's Fine Italian
You can Attend these Restaurants: - Corner Cafe - The Chef's Kitchen
You can Attend these Restaurants: - Corner Cafe - The Chef's Kitchen
You can Attend these Restaurants: - Corner Cafe - The Chef's Kitchen

IPO Chart/Hierarchy Chart / UML:

Input	Processing	Output
Vegetarian? Vegan? Gluten-free?	Using if else statements determine if the input for vegetarian was yes or no and set that to a 1 if yes 0 if no Using if else statements determine if the input for vegan was yes or no and set	Restaurant choices: Lists all possible restaurants which suit group/party

	that to a 1 if yes 0 if no Using if else statements determine if the input for gluten-free was yes or no and set that to a 1 if yes 0 if no Using an elif statement all combinations of values are checked using the 0 or 1 for each input and this creates a string which contains all the possible restaurants to attend	

Brief Code Walkthrough - Console and GUI Program:

Console Code:

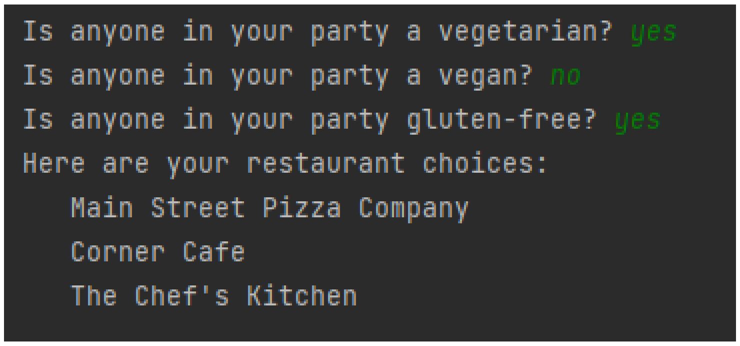
First it asks the user to answer yes or no to if anyone in the group is vegetarian, vegan or gluten free. Then using an if statement if checks to see if yes was answer if true it sets a variable to 1 if false it sets it to 0 and this is done for all inputs. Then using an elif statement every possible combination of dietary needs in listed and then sets a variable result in a string which contains all the valid restaurants which meet everyone's needs. Then it prints the result string.

GUI Code:

The GUI code starts by importing tkinter, then there is the class MyGUI which contains 2 Methods, the result method and the __init__ method, the result method is mainly the console code with a few tweaks, and removing the if else statements to change yes and no to 1 and 0. The __init__ method again contains the setup for the GUI, the frames, this time i used checkbuttons to let users provide input as it avoids input validation, and can be set to output 0 or 1 depending on whether the button is highlighted or not. Then there are the enter and quit buttons and the text area which outputs the valid dietary restaurants which meet your group's requirements.

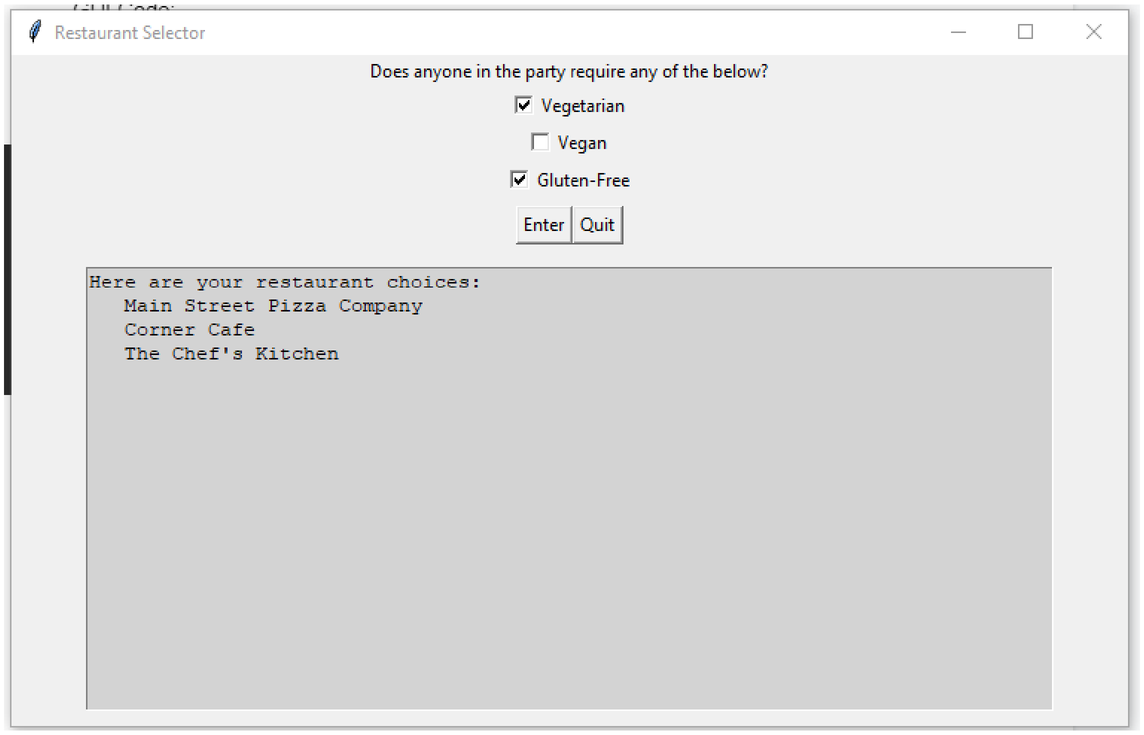
Evidence of Testing:

Console Testing:



```
Is anyone in your party a vegetarian? yes
Is anyone in your party a vegan? no
Is anyone in your party gluten-free? yes
Here are your restaurant choices:
   Main Street Pizza Company
   Corner Cafe
   The Chef's Kitchen
```

GUI Testing:



```
Here are your restaurant choices:
   Main Street Pizza Company
   Corner Cafe
   The Chef's Kitchen
```

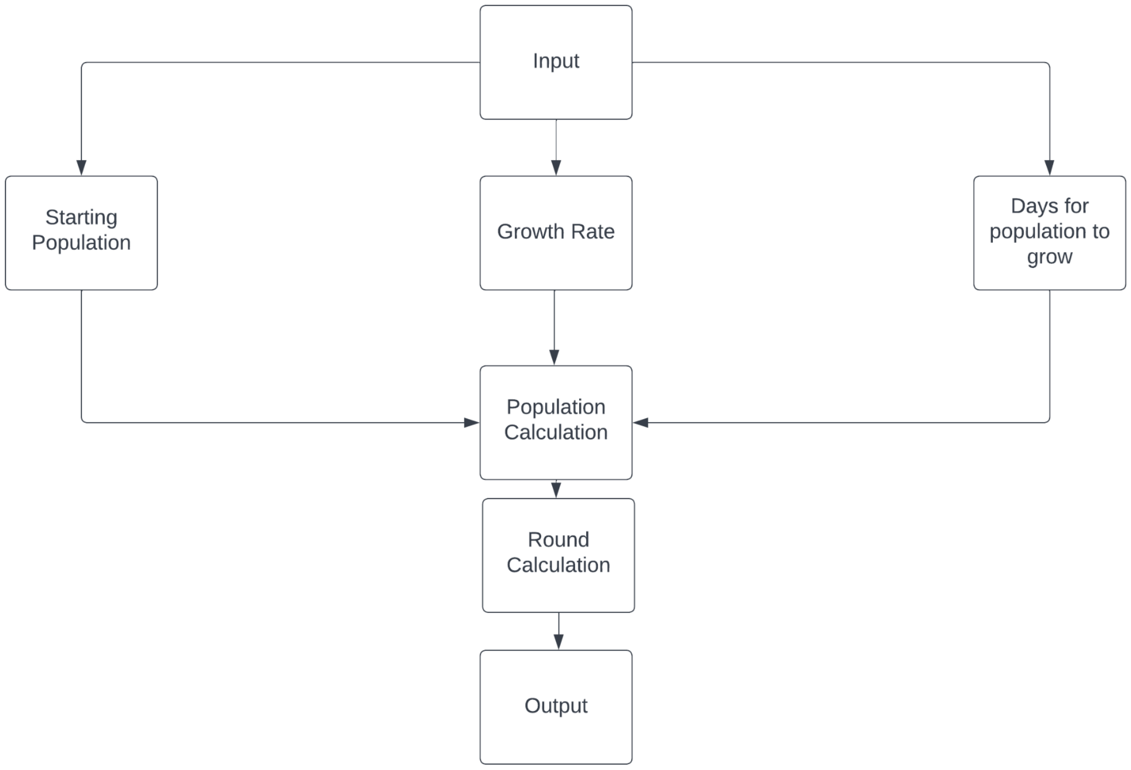
Special Features:
It doesn't really have any special features since the use of checkbuttons to get user input avoids the need for input validation.

Reflection:
I was really happy with this project because I hadn't used checkbuttons before so it was a valuable learning experience and I think it turned out really well. The annoying thing about this project is making sure that you have all the possible combinations that the users could input and then match that to a list of restaurants. I did mine manually using the flowchart above and then using a variable set to a string containing all the correct restaurants for each combination. But I think if I was to do this again I would have something that determines which restaurants they can go to without having to manually input the strings myself.

Project 3: Population Tracker

Requirements Analysis:
This program requirement is effectively to build a compound interest calculator, since you take a starting value then you exponentially grow that value using an input percentage and it compounds for a user input amount of time. Except instead of money it is population

Algorithm Design:



IPO Chart/Hierarchy Chart / UML:

Input	Processing	Output
Starting Population Growth Rate Days to run	Calculate new growth rate into more useful value by dividing by 100 then adding 1 Using a for loop, loop for through the range of 1 to days to run and calculate new growth by multiplying the starting pop and new growth rate then subtracting the starting population Also inside the loop add the new growth onto the starting population Inside the loop round the population to 5dp Finally inside the loop print	A table containing the days in the left column and the population in the right column

	both the current iteration of the loop(current day) and the rounded current population.	

Brief Code Walkthrough - Console and GUI Program:

Console Code:

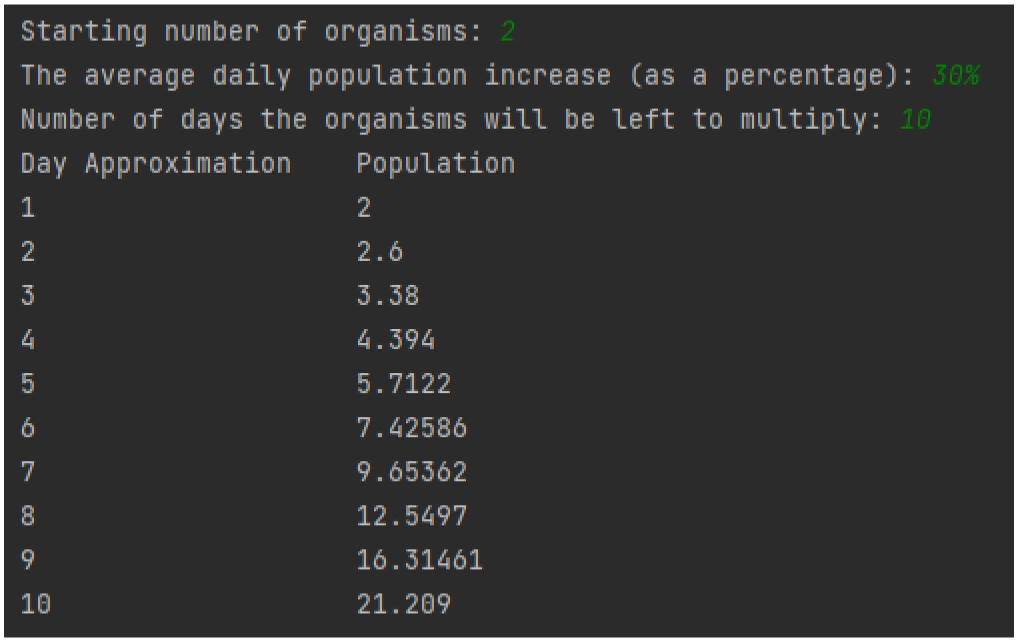
The console code for this program is really simple, 12 lines. The first 3 are input functions which gather the input data from the user. Then the growth rate input is stripped of its % sign and divided by 100 and has 1 added to it to convert it from 30% to a decimal representing 130% or 1.3 if input is 30%. Then the print begins and it does the first day outside of the loop. Then using a for loop where it iterates through all the way till the user input value for days. Inside the loop the population increase is calculated and added to the current population which is then displayed for each iteration.

GUI Code:

The GUI code is slightly different it imports tkinter, then it has the class MyGUI and inside there are 2 methods: the result method and the __init__ method. The result method is a variation on the console code. I get the input values from the entry widgets and then do the same conversion for the growth rate. Then there is a list for days which is filled using a loop which loops through using a range all the days and on each iteration appends the current iteration to the day list. Then there is a population list which does the same calculation as the console loop and appends the population to the list. Then both lists are inserted into separate text areas which are located side by side.

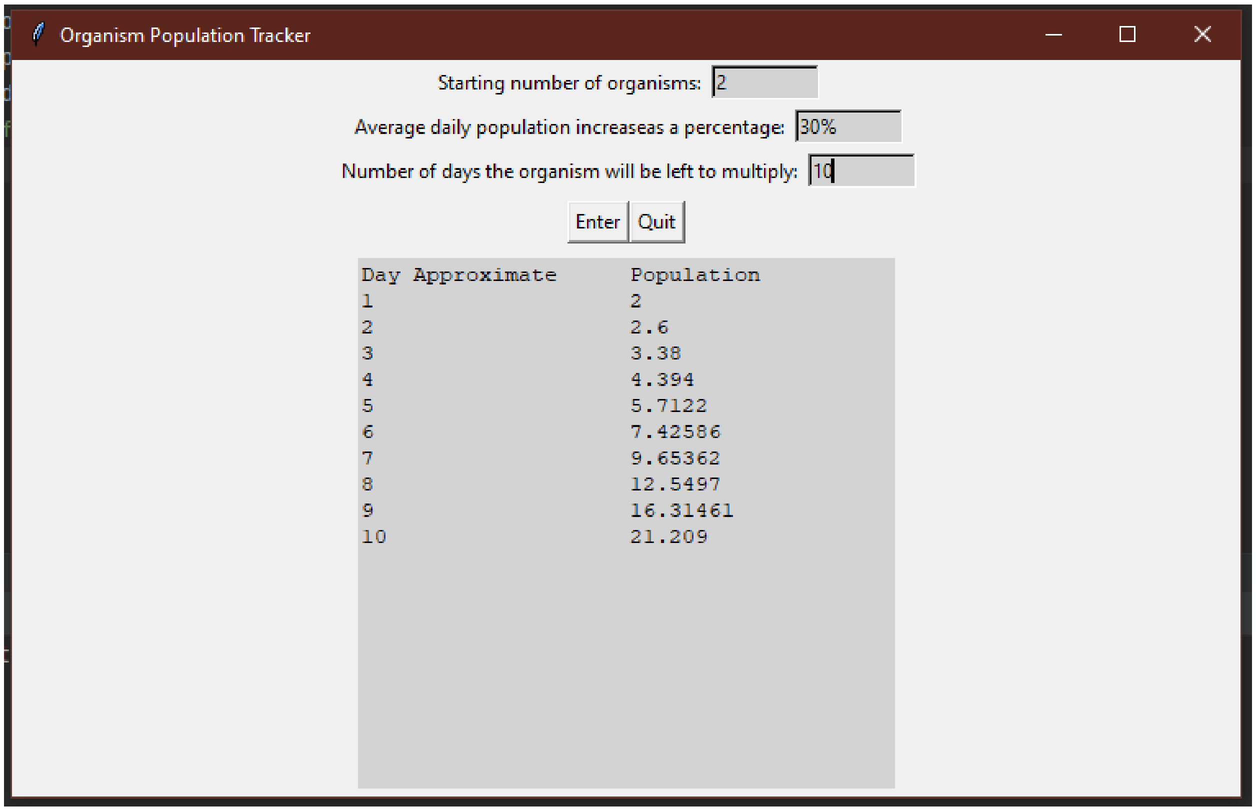
Evidence of Testing:

Console Testing:



```
Starting number of organisms: 2
The average daily population increase (as a percentage): 30%
Number of days the organisms will be left to multiply: 10
Day Approximation     Population
1                     2
2                     2.6
3                     3.38
4                     4.394
5                     5.7122
6                     7.42586
7                     9.65362
8                     12.5497
9                     16.31461
10                    21.209
```

GUI Testing:



```
Starting number of organisms: 2
Average daily population increaseas a percentage: 30%
Number of days the organism will be left to multiply: 10
```

```
Day Approximate      Population
1                    2
2                    2.6
3                    3.38
4                    4.394
5                    5.7122
6                    7.42586
7                    9.65362
8                    12.5497
9                    16.31461
10                   21.209
```

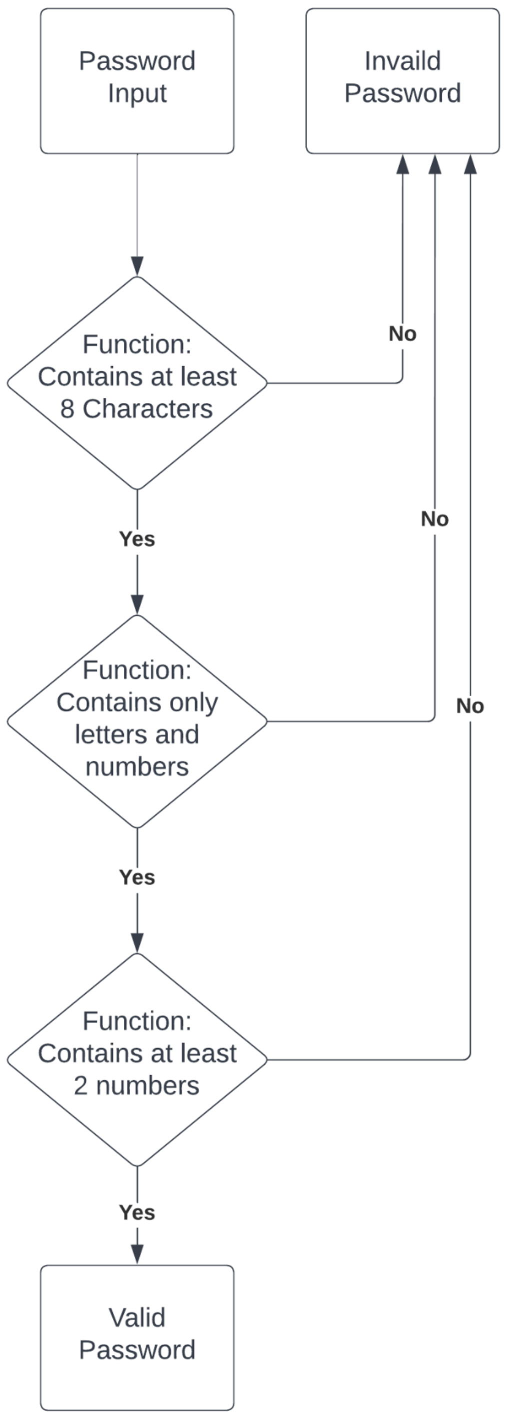
Special Features:
Unfortunately again due to poor time management on my part I was unable to include input validation in this program although it would have been a nice feature to have. Although in my GUI program I did use lists which were outside the learning objective.

Reflection:
I really liked this project especially because I really like finance and math and this was very similar to a compound interest calculator. The program wasn't terribly hard to make although I did suffer through some difficulty trying to get the first day to be different, then for the GUI version I had difficulty outputting the results in the form of a table. Another thing I thought was cool about this project was the way in which I converted the input value of 30% into 1.3. I thought the process was really cool and effective.

Project 4: Password Checker

Requirements Analysis:
The requirements for this project were very simple, with more and more websites these days, requiring more complex passwords to keep their users safe. This program could be used by a website to determine if the password entered by the user meets their requirements which are that the password must have at least 8 characters, contain only numbers and letters and contain at least 2 digits. Then using input validation and especially functions check to see if the password is valid and return the result to the user.

Algorithm Design:


Password Input
Invaild Password
Function: Contains at least 8 Characters
No
Yes
Function: Contains only letters and numbers
No
Yes
Function: Contains at least 2 numbers
No
Yes
Valid Password

IPO Chart/Hierarchy Chart / UML:

Input	Processing	Output
Any Password	Length Function - Determines if the password is longer than 8 characters by looping the len of password unless value is > 8 Symbol function - Determines if password is made up of only numbers and letters using .isalnum then passes through an if statement to check if valid or invaild Number function - Determines if there are 2 or more numbers in password By using an if statement and loop that checks if .isdigit is less than 2 Validity function - This function checks to see if all previous functions have returned invaild or valid results and determines with an elif statement whether the password is valid or not. Main Function - This function takes the value of the validity function and uses an if statement to convert it into a string which conveys to the user whether their password is valid or invalid.	Password Statement about validity of password in regards to requirements.

Brief Code Walkthrough - Console and GUI Program:

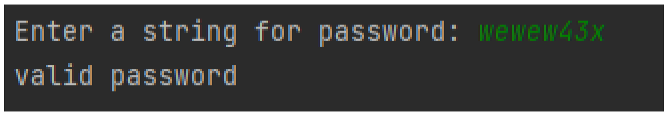
Console Code:

The console code is made up of the password input and 5 functions. The functions include a length function which determines the length of the password and returns invalid if under 8 characters. The symbol function which uses the .isalnum to check whether the password contains characters other than letters or numbers. Then the number check function which checks using a loop to see if the amount of numbers in the password is less than 2. Then there is the validity check function which looks at the values of all the previous functions and determines if the password is valid or invalid.

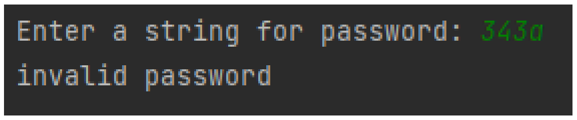
Finally the main function checks the value of the validity function and uses an if else statement to report the validity of the password back to the user through a string.

GUI Code:
GUI Code is very similar. It imports tkinter, then it has the __init__ method in the class MyGUI which sets up the GUI and contains the label and entry widget which allow the user to input their password to be checked. Then there are 2 buttons: enter and quit. The enter runs the main function from the console code which then inserts the output statement into a text area in the GUI. I had to adjust all the functions which used the password variable to declare it locally in each instead of globally as it was causing issues. But it's overall very similar and uses all the same functions.

Evidence of Testing:
Console Testing:

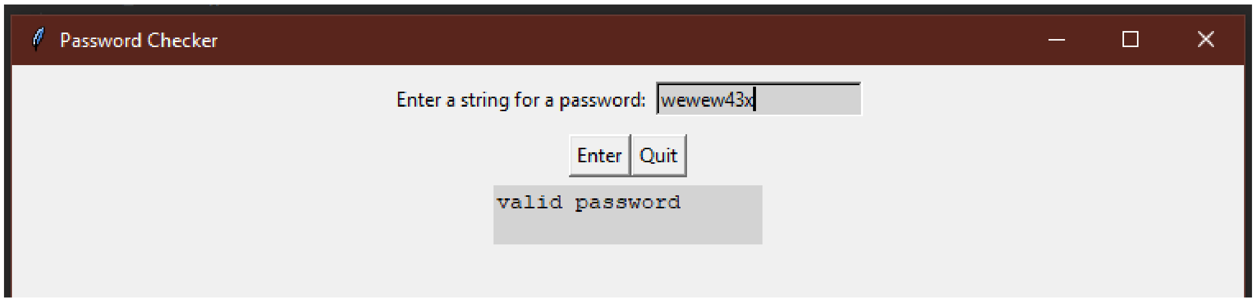


```
Enter a string for password: wewew43x
valid password
```



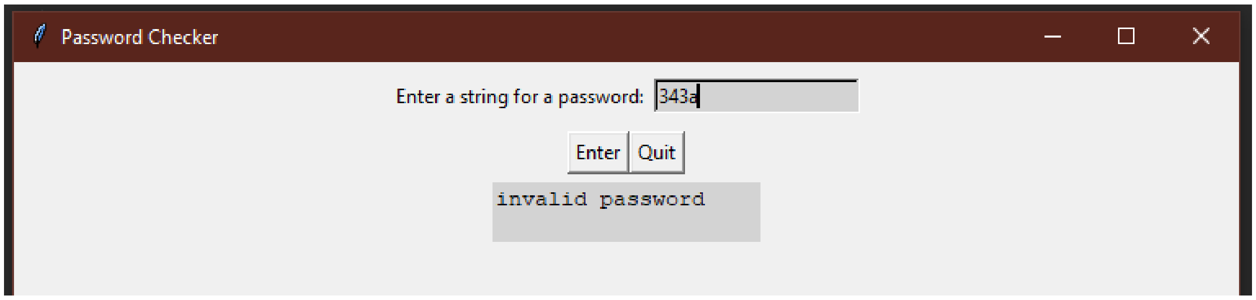
```
Enter a string for password: 343a
invalid password
```

GUI Testing:



Enter a string for a password: wewew43x

```
valid password
```



Enter a string for a password: 343a

```
invalid password
```

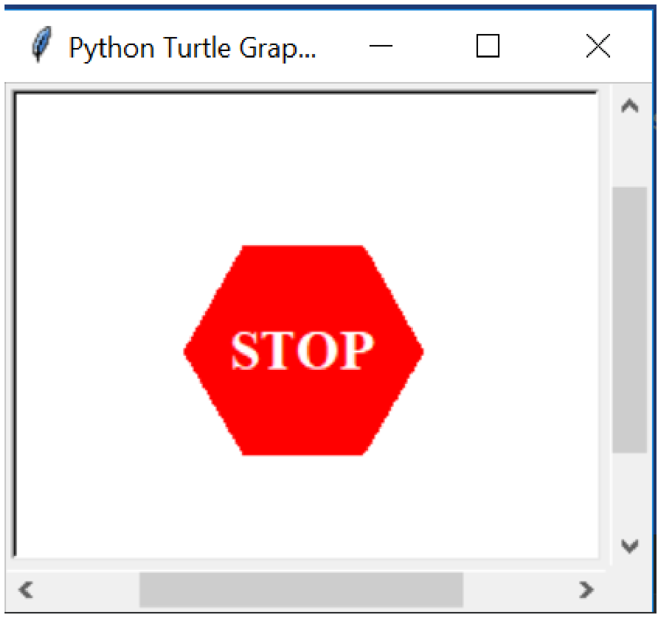
Special Features:
This program does have input validation because that is effectively what the program is designed to do. Although again due to time management I couldn't elaborate on the program to provide the user with the exact requirement which meant their password failed.

Reflection:

I struggled with this program a little bit as I hadn't had much practice with input validation, I especially had trouble trying to determine whether the password had 2 or more numbers in it. As mentioned above if I had time I would go back and add in why the user input an invalid password in the output statement. I also struggled converting my functions over into the GUI program as I was using a global variable, which I changed in the GUI to be a local variable in every function that needed it.

Project 5: Turtle Graphics

Requirements Analysis:
This project is very simple all you have to do is to recreate the image below using the turtle graphics library, making sure to use the same colors.



Algorithm Design:
This project doesn't involve an algorithm since it doesn't take any user input.

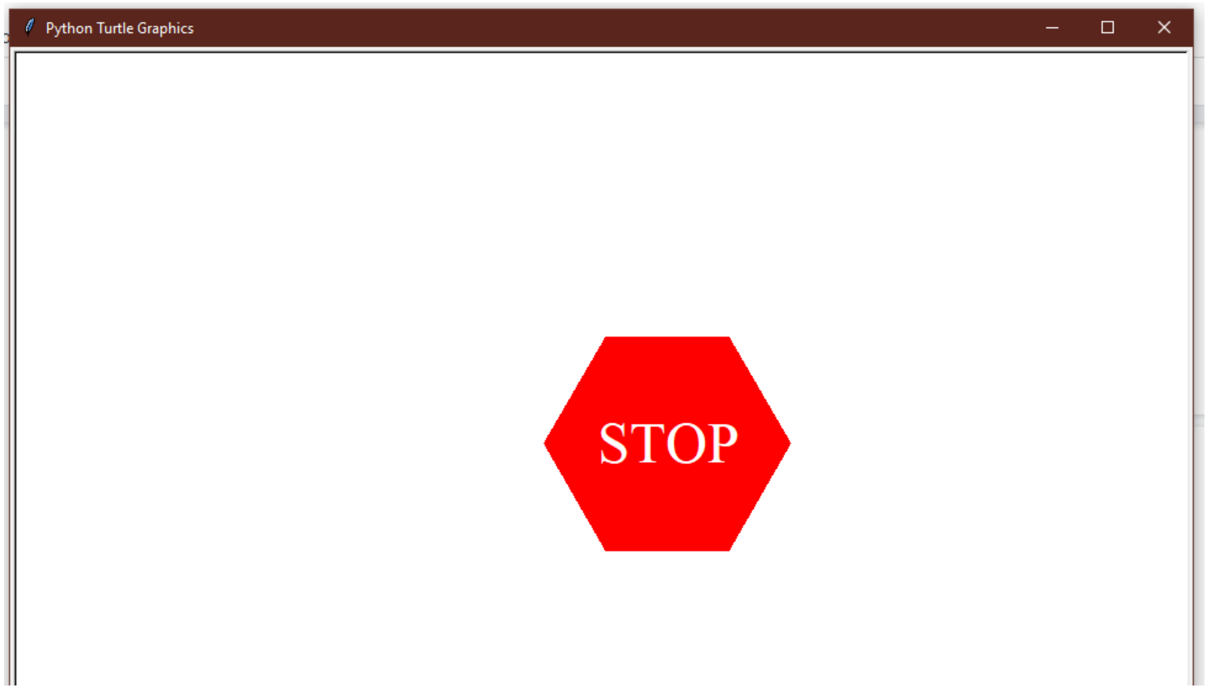
IPO Chart/Hierarchy Chart / UML:

Input	Processing	Output
None	Using a loop to create the hexagon Using the fill function to make the sign red Using the pencolor function to make the red sign Using the hideturtle function to remove the turtle from the	A red stop sign made using the turtle graphics library

	screen Using the write function to create the text	

Brief Code Walkthrough - GUI Program:
GUI Code:
First I imported the turtle graphics library, then I hid the turtle, set the pen color to red and fill color to red, began the fill and used a while loop that iterates 6 times to create the pentagon of the stop sign. Then I end the fill and move the turtle to the left left corner of the text, where I change the pen color to white and use the write function to write stop then close with the done function.

Evidence of Testing:
GUI Testing:



Special Features:
This program doesn't include any special features, although I did change the font to match the one used in the example.

Reflection:
This project was quick and fun, I really enjoy using the turtle graphics library because you can see your changes quickly and clearly. The only real difficulty in this project was fine tuning the location of the turtle to start writing text, so that it looked centered on the sign.

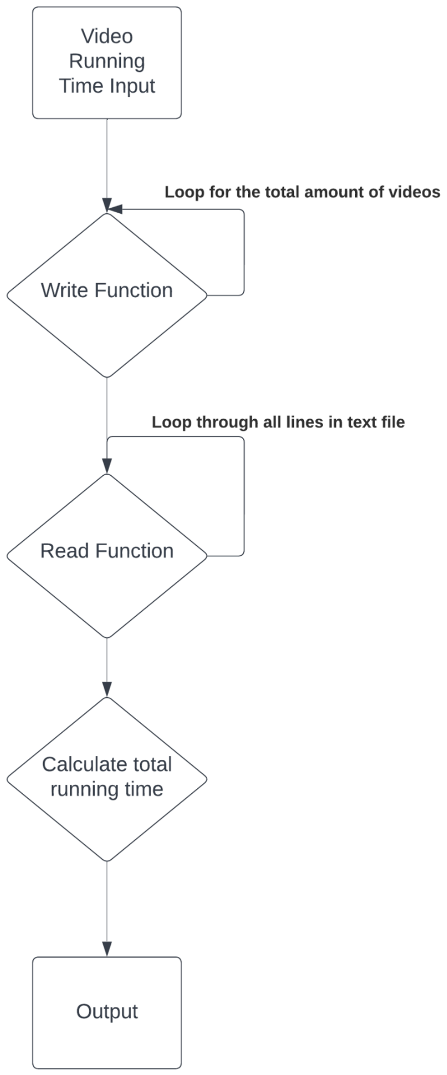
Part B

Project 1: Video Time Logger

Requirements Analysis:

The requirements for this program are to create a program that takes a user input about the amount of videos taken and then asks for the running time of each video. Which is then saved onto a text file. The program then reads the text file outputs all the times and compiles them into a total running time and outputs that to the user.

Algorithm Design:


Video Running Time Input
Loop for the total amount of videos
Write Function
Loop through all lines in text file
Read Function
Calculate total running time
Output

IPO Chart/Hierarchy Chart / UML:

Input	Processing	Output
Number of Videos Length of each video in seconds	Write Function - This function is used to loop through the videos running time inputs and write them to a file which can be read later. Read Function - This function reads the file created earlier and collates all the running times together and outputs a list of all the videos and their running times as well as the total running time of all the videos.	Running time for each video Total combined running time

Brief Code Walkthrough - Console and GUI Program:

Console Code:

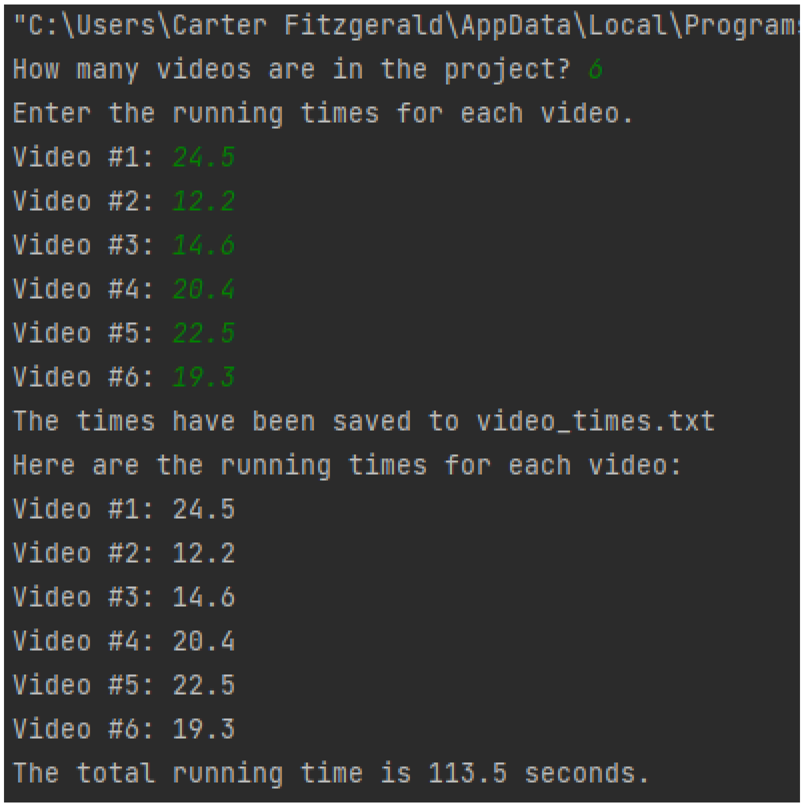
My Console code is made up of 2 Functions: the save function which writes to the text file and the read function which reads the data from the text file. The save function, asks for the input from the user inside a loop and then prompts them to enter the running times for the amount of videos they entered. The function then at the end of the loop writes the running time to the text file to be accessed later. Then in the read function it opens the text file, then runs a loop which prints each line of the file as an output statement and adds to a total running time variable. Then the function closes the file and prints the total running time output statement.

GUI Code:

My GUI code is a bit better and makes it much simpler for the user, they are provided with an entry widget for the amount of videos, then using the simple dialog and message box modules I am able to prompt the user for the running time of each video. Then they press the display button and all the data is displayed. This code includes the usual class MyGUI and method __init__ which contains all the setup, frames, buttons, labels, entry and text areas. Then the 2 functions have been modified to use tkinter entry widgets and the message box and simple dialog module and they also now include exception handling, but the code inside is very similar.

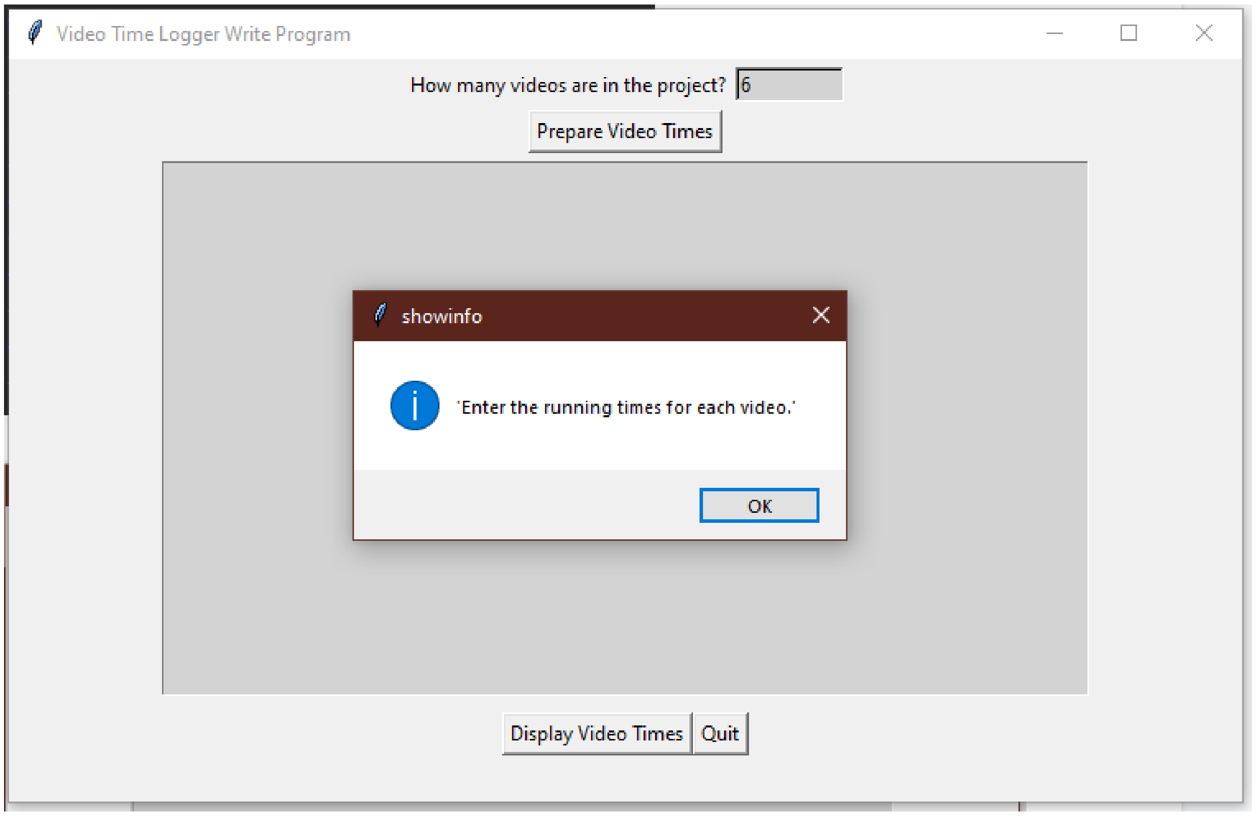
Evidence of Testing:

Console Testing:



```
"C:\Users\Carter Fitzgerald\AppData\Local\Program
How many videos are in the project? 6
Enter the running times for each video.
Video #1: 24.5
Video #2: 12.2
Video #3: 14.6
Video #4: 20.4
Video #5: 22.5
Video #6: 19.3
The times have been saved to video_times.txt
Here are the running times for each video:
Video #1: 24.5
Video #2: 12.2
Video #3: 14.6
Video #4: 20.4
Video #5: 22.5
Video #6: 19.3
The total running time is 113.5 seconds.
```

GUI Testing:

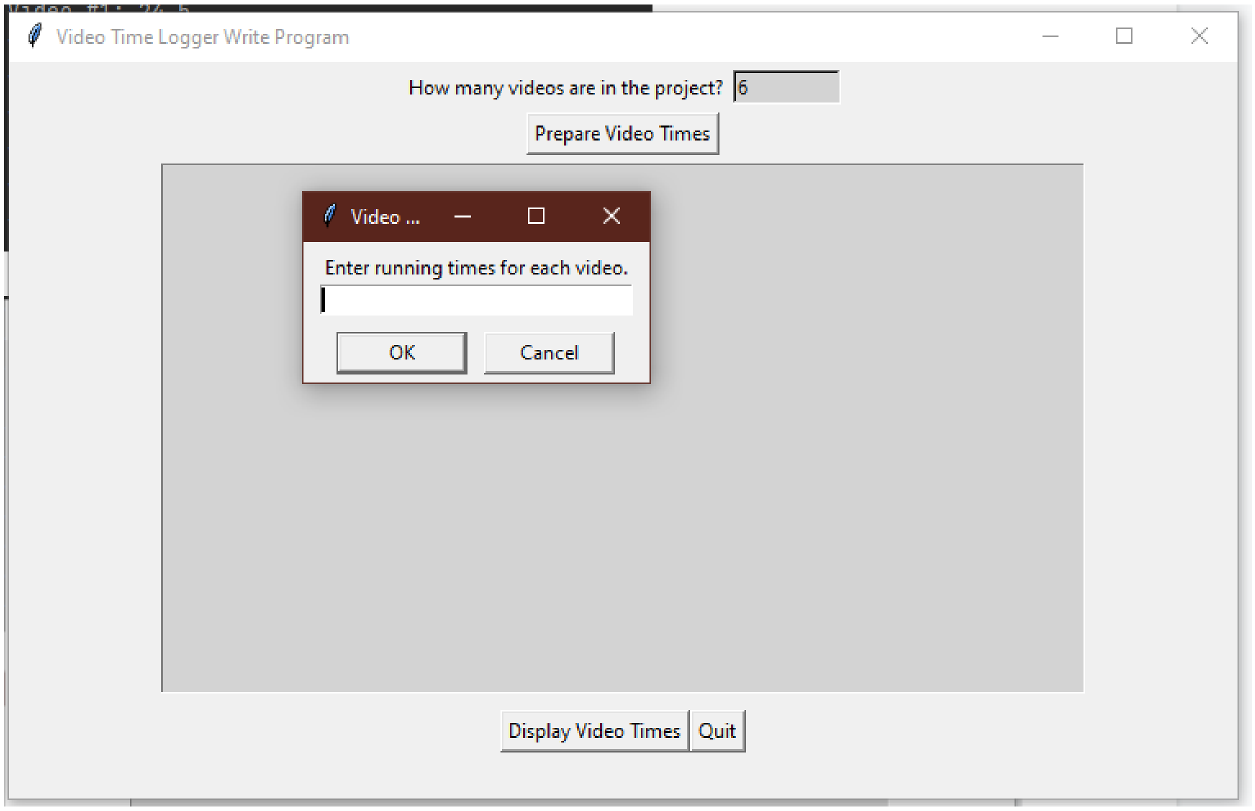


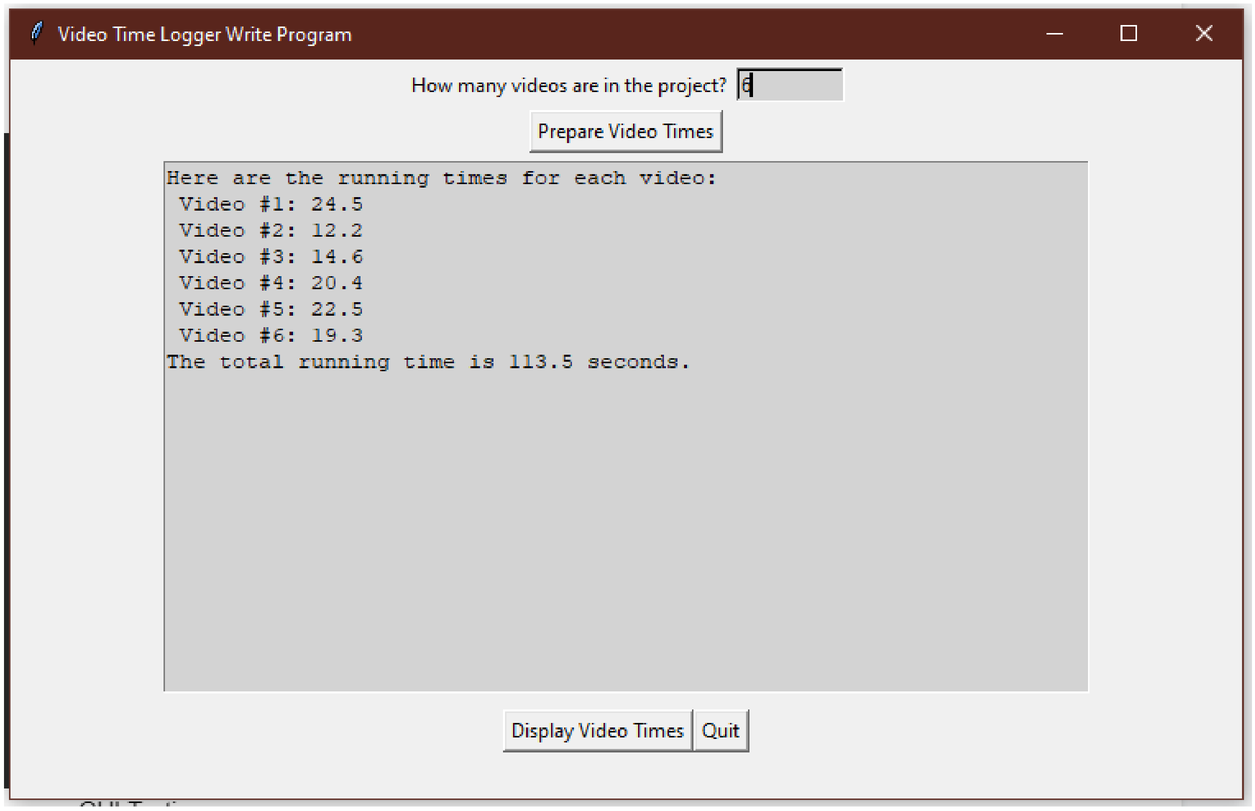
Video Time Logger Write Program

How many videos are in the project? 6

showinfo

'Enter the running times for each video.'


Video Time Logger Write Program
How many videos are in the project? 6
Prepare Video Times
Video ...
Enter running times for each video.
OK
Cancel
Display Video Times
Quit


Video Time Logger Write Program
How many videos are in the project? 6
Prepare Video Times
Display Video Times
Quit

```
Here are the running times for each video:
 Video #1: 24.5
 Video #2: 12.2
 Video #3: 14.6
 Video #4: 20.4
 Video #5: 22.5
 Video #6: 19.3
The total running time is 113.5 seconds.
```

Special Features:

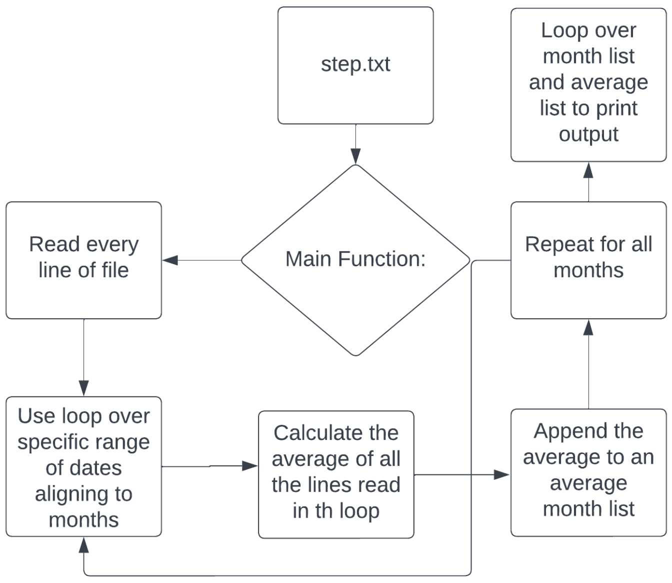
The GUI program for this project includes exception handling for when the file can't be opened and input validation if a number isn't entered into the entry at the top of the program.

Reflection:
Reflecting on this project, this was one of my first times using the write and read functions in python and outputting data to a file. Therefore this project was full of challenges for me and pushed me to get right. Looking back I would like to have added the exception and input validation on the console program, but again my time management was poor and I was unable to do this.

Project 2: Fitness Tracker

Requirements Analysis:
The requirement for this project involved taking the data provided in the file steps.txt which was 365 lines of daily step values. The program has to read these values and then calculate the average steps taken over each month.

Algorithm Design:


step.txt
Main Function:
Read every line of file
Use loop over specific range of dates aligning to months
Calculate the average of all the lines read in th loop
Append the average to an average month list
Repeat for all months
Loop over month list and average list to print output

IPO Chart/Hierarchy Chart / UML:

Input	Processing	Output
step.txt	Average Function - Determines the average value of a list	The output is the average amount of steps taken for each month in the year from

	Main Function - Reads file and uses for loops to loop each month using a range function with values corresponding to each month, then appends every value in range to a list which is passed to average function, which the result is then appended to an average list. (This is done for every month) This function also includes exception handling for if the file couldn't be accessed and indexing errors	the input of steps.txt

Brief Code Walkthrough - Console and GUI Program:

Console Code:

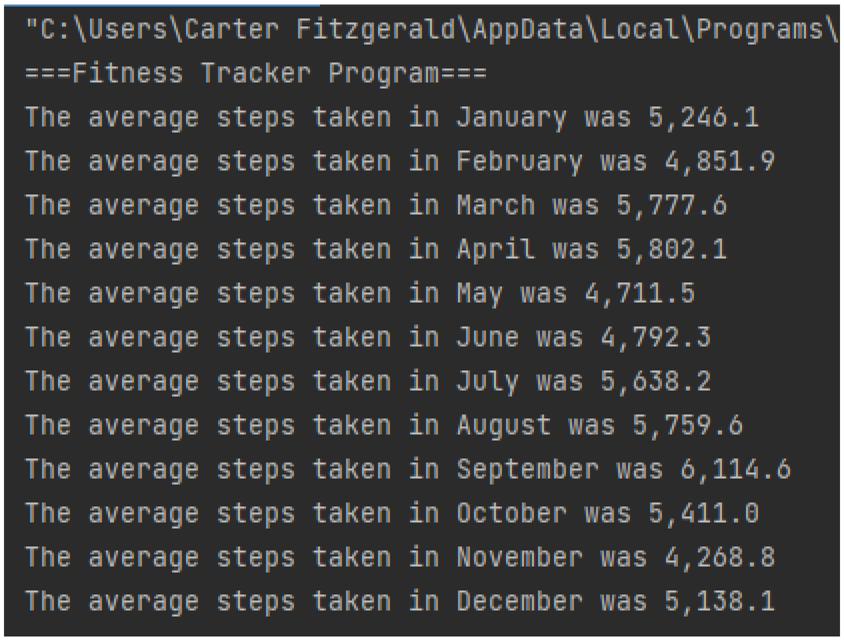
At the beginning of the console code is the average function which I reference to to get the average of the contents of lists. Then I have the main function which the first thing is a list of all the months in a year, this is followed by an initialisation of an individual list for each month which is empty. Then The try function is used which contains the reading of the steps.txt file and setting that to a variable. Then each month has a loop where it passes through a specific range and number of lines in that variable which contains a list from steps.txt and it compiles all the days into a total steps for the month the empty list specific to each month, this is then passed to the average function which determines the average, which then gets passed into a list for the average of each month. Then after all the loops another loop that iterates 12 times creates the output statement using the month list and average list. Then the exception handling is below that and the main function is called.

GUI Code:

It's almost identical to the console code, there is the __init__ method at the top which sets up the GUI and displays the output when the display button is pressed, since this program needs no input from the user. That is about all that is different and the way that the data is output is also slightly different.

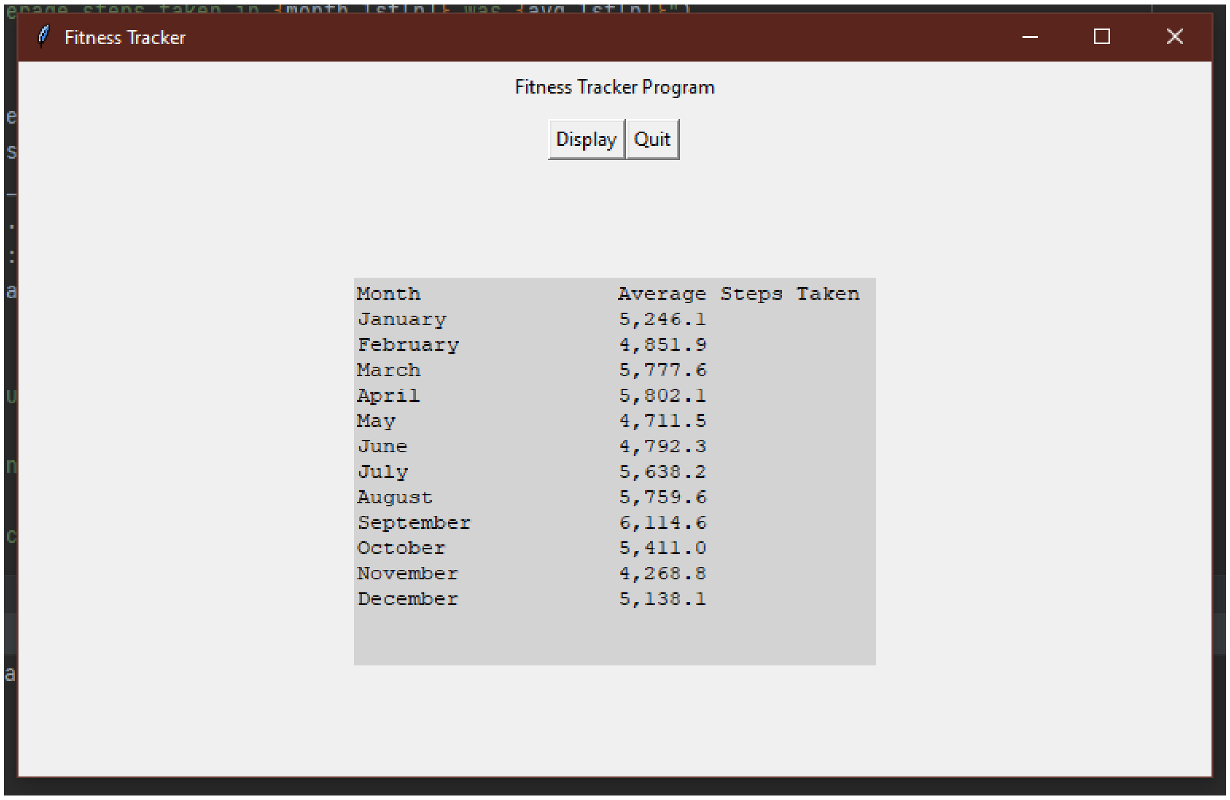
Evidence of Testing:

Console Testing:



```
"C:\Users\Carter Fitzgerald\AppData\Local\Programs\
===Fitness Tracker Program===
The average steps taken in January was 5,246.1
The average steps taken in February was 4,851.9
The average steps taken in March was 5,777.6
The average steps taken in April was 5,802.1
The average steps taken in May was 4,711.5
The average steps taken in June was 4,792.3
The average steps taken in July was 5,638.2
The average steps taken in August was 5,759.6
The average steps taken in September was 6,114.6
The average steps taken in October was 5,411.0
The average steps taken in November was 4,268.8
The average steps taken in December was 5,138.1
```

GUI Testing:



```
Month                     Average Steps Taken
January                   5,246.1
February                  4,851.9
March                     5,777.6
April                     5,802.1
May                       4,711.5
June                      4,792.3
July                      5,638.2
August                    5,759.6
September                 6,114.6
October                   5,411.0
November                  4,268.8
December                  5,138.1
```

Special Features:
Both my console and GUI program include exception handling which can deal with IO errors, indexing errors and will throw an error if anything else goes wrong.

Reflection:
I was really happy with how I was able to complete this project, I thought it was

interesting and it challenged me. However, if I was to do it again I think I would try and use some more functions to reduce the complexity and clutter of the loops I created.

Project 3: Employee Management System

Requirements Analysis:
For this project we have to write a program that is an employee management system, it needs to have a class called Employee and has the attributes, name, ID Number, Department and Job Title. Then it wants you to draw a UML diagram on the class and write a program that creates 3 objects of the employee class, with the information from a table and displays their details. Then you need a subset class of Employee called ShiftEmployee which adds 2 new attributes, Shift number and hourly rate. Then you have to write a program that creates an object of the ShiftEmployee class and prompts the user to enter data for each attribute. Demonstrate by creating at least 2 ShiftEmployee Objects. Create another subset of Employee called Contractor which has 3 new attributes: Contract End date, ABN and fixed contract salary, demonstrate by creating 2 contractor objects.

Algorithm Design:
Unfortunately due to my poor time management I was unable to finish this project and I had trouble understanding it, so I couldn't create an algorithm design for this project.

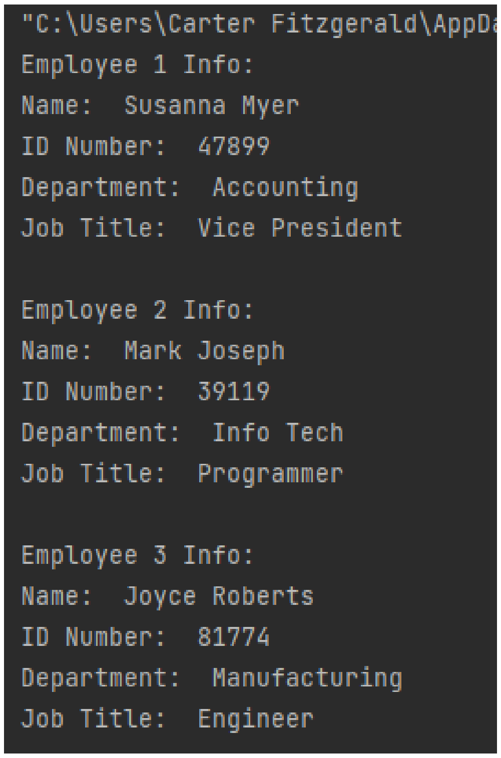
IPO Chart/Hierarchy Chart / UML:
Again, like above due to not finishing this project and a lack of understanding I am unable to create an IPO Chart/Hierarchy Chart / UML for this project.

Brief Code Walkthrough - Console and GUI Program:
Console Code:
My incomplete console code is made up of 4 different files, this project was very confusing for me, but I gave it my best attempt with the limited time I had. The employee.py file is where all the classes are created and they contain heaps of functions, that are all related to the Employee class, ShiftEmployee class and the Contractor class. The Shift Employee Object Tester file is where I created the user input for the ShiftEmployee class. Within this file I also created 2 example objects for the ShiftEmployee class. The contractor Employee Object Tester is where I have 2 example contractor objects stored, which can also be displayed. Finally the Employee Console file is where I started creating the menu for using the employee management system before I started getting errors and ran out of time to finish the project.

GUI Code:
Ran out of Time couldn't write the GUI code

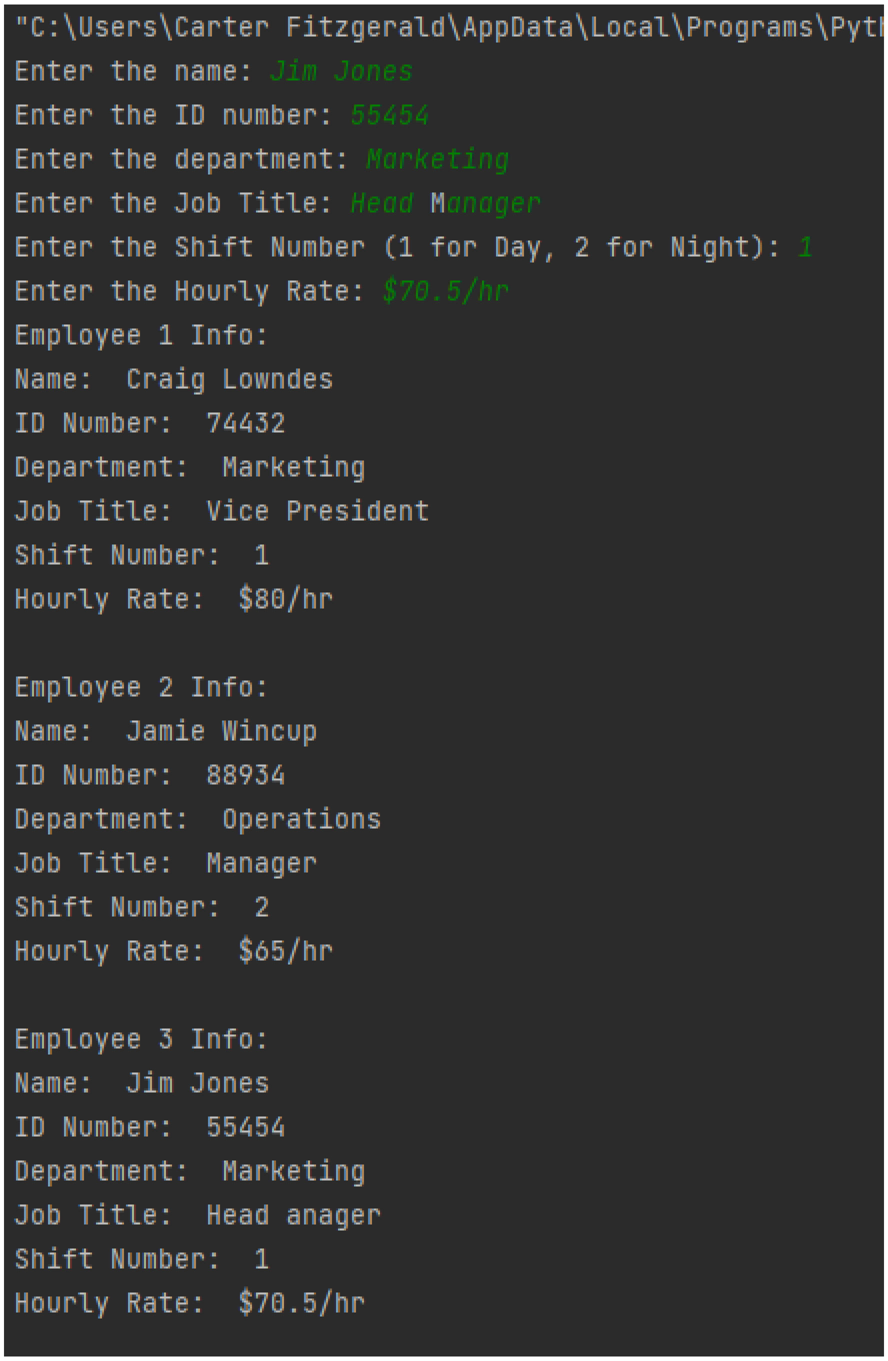
Evidence of Testing:
Console Testing:

Below is the objects I created for the Employee Class which matched the table



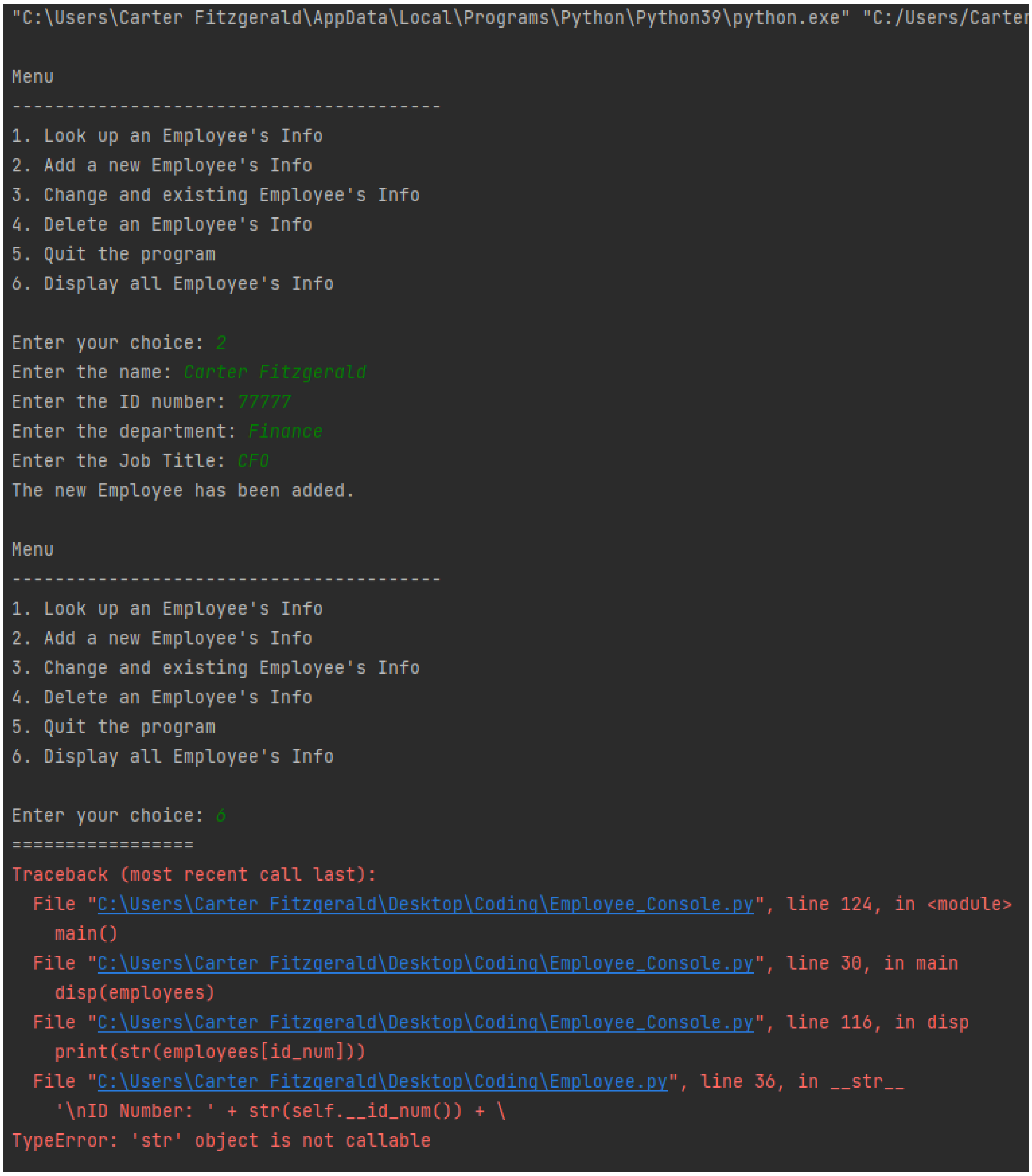
```
"C:\Users\Carter Fitzgerald\AppDa
Employee 1 Info:
Name:  Susanna Myer
ID Number:  47899
Department:  Accounting
Job Title:  Vice President

Employee 2 Info:
Name:  Mark Joseph
ID Number:  39119
Department:  Info Tech
Job Title:  Programmer

Employee 3 Info:
Name:  Joyce Roberts
ID Number:  81774
Department:  Manufacturing
Job Title:  Engineer
```

Below is the ShiftEmployee Object Test with User Input



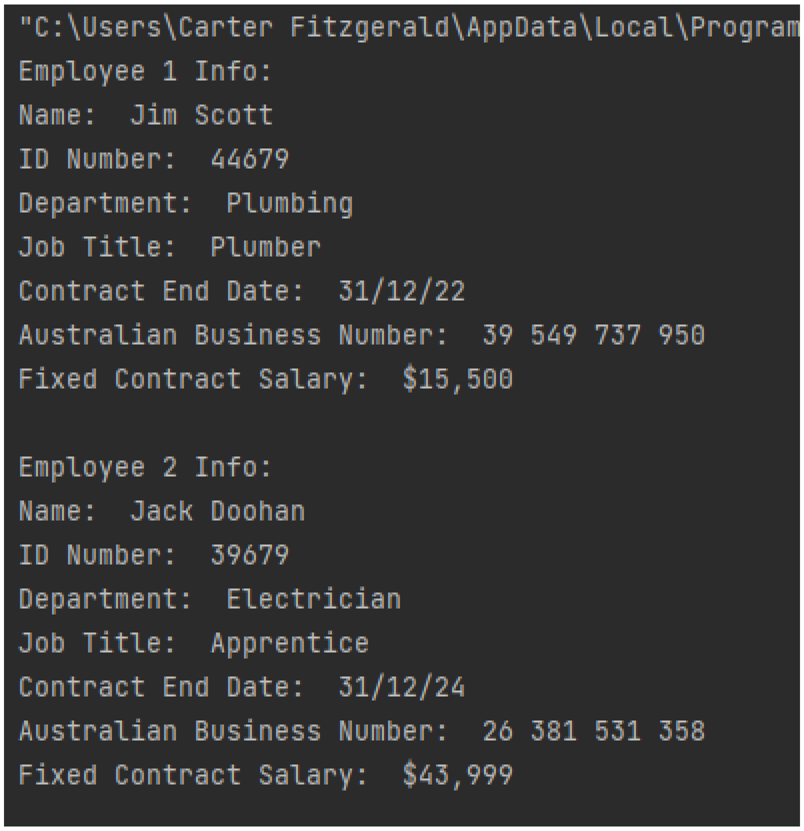
```
"C:\Users\Carter Fitzgerald\AppData\Local\Programs\Pyt
Enter the name: Jim Jones
Enter the ID number: 55454
Enter the department: Marketing
Enter the Job Title: Head Manager
Enter the Shift Number (1 for Day, 2 for Night): 1
Enter the Hourly Rate: $70.5/hr
Employee 1 Info:
Name:  Craig Lowndes
ID Number:  74432
Department:  Marketing
Job Title:  Vice President
Shift Number:  1
Hourly Rate:  $80/hr

Employee 2 Info:
Name:  Jamie Wincup
ID Number:  88934
Department:  Operations
Job Title:  Manager
Shift Number:  2
Hourly Rate:  $65/hr

Employee 3 Info:
Name:  Jim Jones
ID Number:  55454
Department:  Marketing
Job Title:  Head anager
Shift Number:  1
Hourly Rate:  $70.5/hr
```

Below is the main console file and the error I was getting



```
"C:\Users\Carter Fitzgerald\AppData\Local\Programs\Python\Python39\python.exe" "C:/Users/Carter

Menu
----------------------------------------
1. Look up an Employee's Info
2. Add a new Employee's Info
3. Change and existing Employee's Info
4. Delete an Employee's Info
5. Quit the program
6. Display all Employee's Info

Enter your choice: 2
Enter the name: Carter Fitzgerald
Enter the ID number: 77777
Enter the department: Finance
Enter the Job Title: CFO
The new Employee has been added.

Menu
----------------------------------------
1. Look up an Employee's Info
2. Add a new Employee's Info
3. Change and existing Employee's Info
4. Delete an Employee's Info
5. Quit the program
6. Display all Employee's Info

Enter your choice: 6
=================
Traceback (most recent call last):
  File "C:\Users\Carter Fitzgerald\Desktop\Coding\Employee_Console.py", line 124, in <module>
    main()
  File "C:\Users\Carter Fitzgerald\Desktop\Coding\Employee_Console.py", line 30, in main
    disp(employees)
  File "C:\Users\Carter Fitzgerald\Desktop\Coding\Employee_Console.py", line 116, in disp
    print(str(employees[id_num]))
  File "C:\Users\Carter Fitzgerald\Desktop\Coding\Employee.py", line 36, in __str__
    '\nID Number: ' + str(self.__id_num()) + \
TypeError: 'str' object is not callable
```

Below is the Contractor Object Test program



```
"C:\Users\Carter Fitzgerald\AppData\Local\Program
Employee 1 Info:
Name:  Jim Scott
ID Number:  44679
Department:  Plumbing
Job Title:  Plumber
Contract End Date:  31/12/22
Australian Business Number:  39 549 737 950
Fixed Contract Salary:  $15,500

Employee 2 Info:
Name:  Jack Doohan
ID Number:  39679
Department:  Electrician
Job Title:  Apprentice
Contract End Date:  31/12/24
Australian Business Number:  26 381 531 358
Fixed Contract Salary:  $43,999
```

GUI testing:
No GUI Program to test
Special Features:
Attempted at using the pickle module with little success. Otherwise no special features.

Reflection:
This project was really difficult for me, this is my first time learning python and so I struggled to get my head around classes, subclasses, dictionaries and objects. I think that if I had managed my time more effectively I would have been able to complete this project successfully unfortunately that's not the case and while I did write a large portion of the console code for this program once I introduced the pickle module and .dat I struggled to understand and my code kept getting errors. With more time I will finish this project properly in my own time.