**MuscleMax**

**A solution for tracking fitness and nutrition**



Name: Armaan Khaitan

Candidate number: 7385

Centre name: Greenford High School

Centre number: 12432

Contents

[ANALYSIS 2](#_Toc1204449185)

[Problem Description 2](#_Toc1986037490)

[Background: 2](#_Toc41081387)

[Subject Research 7](#_Toc225605122)

[Fitness pal 8](#_Toc1542111177)

[Home workout 8](#_Toc88450457)

[Muscle booster 8](#_Toc276165663)

[Description of current system or similar current products/games 9](#_Toc107428509)

[The problems with the current systems 9](#_Toc1384764783)

[Analysis of snippets of code 9](#_Toc1771934210)

[Data flow diagram of an existing system 10](#_Toc901272785)

[Additional user requirements/limitations 11](#_Toc339519845)

[My systems’ designs 12](#_Toc862632480)

[Context diagram (Data Sources and Destinations) 13](#_Toc933124947)

[ERD diagram 13](#_Toc1582895609)

[Data flow diagram for my system 13](#_Toc283302237)

[Database overview - data dictionary for essential entities 14](#_Toc1709828774)

[Specific objectives of the proposed system 14](#_Toc1000625241)

[Limitations of the proposed system 15](#_Toc1711952812)

[Limited time: 16](#_Toc335085439)

[My programming skill 16](#_Toc139258115)

[Lack of powerful processing power 16](#_Toc1232031702)

[Type of application: 16](#_Toc1260673448)

[Proposed solution analysis 16](#_Toc341131366)

[Overview / breakdown of problems 16](#_Toc2076894761)

[Implementation methods 19](#_Toc86864001)

[Tkinter 20](#_Toc483761471)

[Pygame 20](#_Toc37596453)

[OOP classes identified 20](#_Toc1196317809)

[Data sources 21](#_Toc840702564)

[Modules 21](#_Toc956678642)

[Exercise API 21](#_Toc1554110909)

[Nutrition API 21](#_Toc1323937776)

[Internet scraping 22](#_Toc107718334)

[Beautiful soup 22](#_Toc1786647249)

[Requests 22](#_Toc1545833724)

[Sqlite3 22](#_Toc152725574)

[MySQL 23](#_Toc1497376494)

[NumPy 23](#_Toc712237514)

[OS 23](#_Toc454846880)

[Pathlib 23](#_Toc206060289)

[Potential data structures which may be used 24](#_Toc1192808888)

[Stack 24](#_Toc836071578)

[DESIGN 25](#_Toc1286618055)

[This design section is the structures and ideas I’ve have decided to implement based on research conducted in the Analysis section: 25](#_Toc46361960)

[Outline system design 25](#_Toc849306089)

[System flowcharts 25](#_Toc936977248)

[User Interface Design 26](#_Toc569896929)

[Hardware specification 27](#_Toc896798353)

[Program Structure 27](#_Toc1778882986)

[Design Data Dictionary 29](#_Toc422877567)

[Object diagrams and class definitions 29](#_Toc2049102120)

[Data Structures 29](#_Toc1681697374)

[File Organisation 29](#_Toc1081042680)

[APIs used 29](#_Toc909603193)

[Entity-relationship diagram 29](#_Toc860104829)

[Normalised database tables 29](#_Toc611312364)

[Algorithms 29](#_Toc1239676916)

[Top-down description of modules 29](#_Toc1993134039)

[Detailed test data 29](#_Toc473757995)

[TECHNICAL SOLUTION: 29](#_Toc752267929)

[Basics file 29](#_Toc1044723082)

[Login system: 29](#_Toc1892053162)

[Constants file 29](#_Toc1550365411)

[GUI basics file 29](#_Toc381739454)

[Profile 29](#_Toc192291657)

[Calendar 29](#_Toc1276174418)

[Dynamic workout display 29](#_Toc647957629)

[Caloric section 29](#_Toc1267444211)

[Treeview class 29](#_Toc333661855)

[Graphing 29](#_Toc961984391)

[Search for Images file 29](#_Toc1425199184)

[History file 29](#_Toc1615956462)

[Main file 29](#_Toc2064865772)

[GUI Login file 29](#_Toc370613767)

[Start file 29](#_Toc460937040)

[Database Creation file 29](#_Toc364997085)

[Implementation techniques: 29](#_Toc1691916846)

[Inheritance 29](#_Toc252385179)

[Polymorphism 29](#_Toc1369621570)

[Composition 29](#_Toc1469259416)

[TESTING 29](#_Toc1383880438)

[Mistakes encountered while coding 29](#_Toc601963751)

[Video link to testing of application 29](#_Toc408667099)

[Evaluation 29](#_Toc1982652714)

ANALYSIS

# Problem Description

My application is a fitness tracking and progress app to help people keep track of their different workouts, suggest suitable ones, and create a plan for helping them achieve a goal. I have a friend who’s planning to open his own gym that thinks that a desktop app would be particularly useful to have for his customers in his gym.

It is also a place where they can enter their food for the day which will be split into ingredients and calories calculated and a chart generated displaying their split of different food groups.

# My client asked me to write a program that could record the user’s profile and record peoples' workouts, suggesting workouts to beginners and allowing more experienced people the option to edit their workouts and display and save their progress. Also, my client asked me to add a feature that could take their current meal and split it into ingredients, with the calories collated and the different food groups displayed in a chart format.Background:

Many people find it difficult to keep track of their progress in the gym and at home when exercising. They miss having a clear display with all the in-built plans and processing done for them, allowing them to focus on their workouts. On Google Play store and Apple App Store there were different apps to measure progress in gym and home workouts and track one’s diet too. However, no app could be found for free that could meet these two criteria fully within the same app. Often the principal problem people find on their fitness journey is recording their progress and consistency. My computer application will help keep track of their workouts and nutrition and show their progress to encourage them to break their records.

End User Investigation

**I formed these questions to gauge a sense of my user’s requirements and how well I can fulfil these requirements with the time, resources and skills I have. For the options of workouts and nutrition I did some research and added some of the topmost popular workouts tracking and nutrition tracking apps as part of the multi choice questions.**

**For example, the “Sworkit Fitness & Workout App” is ranked #1 fitness app by ACSM – American College of Sports. It has over 135,000 5-star reviews and an average rating of 4.7 on Google Play Store.**

**Questionnaire:**

1. Do you work out regularly?

A blue and red pie chart

Description automatically generated

1. Where do you workout? If you workout in both the gym and home, select whichever one you workout in most of the time. If you workout elsewhere, please specify in other.
2. A screenshot of a graph

   Description automatically generatedIf you record your workout, please select which one and write in other if not listed.

A pie chart with numbers and a diagram

Description automatically generated with medium confidence

A screenshot of a fitness app

Description automatically generated

1. What would you say you like about the way you record your workouts, be it personal trainer, app or other?

A screenshot of a white box

Description automatically generated

1. What would you say is missing from the way you record your workouts, be it personal trainer, app or other?

A screenshot of a phone

Description automatically generated

1. Food: If you record your nutrition, please select which one and write in other if not listed:

A screenshot of a graph

Description automatically generated

A group of colorful circles with text

Description automatically generated

7) What would you say you like about the way you record your nutrition, be it personal trainer, app or other?

Description automatically generated

8) What would you say is missing from the way you record your nutrition, be it personal trainer, app or other?A screenshot of a white box

Description automatically generated

Would you like it if there was an app to conveniently and seamlessly record both your workouts and nutrition?

A screenshot of a graph

Description automatically generated

**Firstly, all the people who work out regularly have said they think my app would be useful. There was a lot of variances in the type of workouts used ranging from Sworkit Fitness to Personal trainer. 75% of people said they use Fitness pal as a nutrition tracking app with the other 25% using Nutrisense. This goes to show how popular and culturally diverse Fitness pal is. It is also said to be the number one health and fitness tracking app in the world – just for nutrition, it’s capabilities are limited to just food.**

# 

# Subject Research

I have researched 3 applications to explore what potential applications are already available and understand their strengths and weaknesses to mold my application accordingly:

Fitness Pal – It is the second most and popular fitness app.

Home Workout – It was the number one most downloaded fitness app in 2021 – [28 million times](https://www.businessofapps.com/data/fitness-app-market/).

Muscle Booster – It was the 5th most downloaded fitness app in 2021 – [18.5 million times.](https://www.businessofapps.com/data/fitness-app-market/)

## Fitness pal

Fitness pal records the calories in meals. The user chooses how many calories they want to eat per day based on several questions that find their fitness goals.

It allows the user to pick between four meals: breakfast, lunch, dinner, snack

Fitness pal pictures

Once they have picked the meal, they can search through a database of food that will tell them about the calories for the food items, accounting for the number of portions eaten. The total calories are then subtracted from the user’s daily calorie goal.

|  |  |
| --- | --- |
| Pros | Cons |
| Effectivelytracks calories consumed | Cannot track workouts for which a separate app is needed, which is inconvenient. |
| Displays the calories in a pie chart | User cannot manually enter calories if not in search bank |
|  | Cannot track or record workouts |

## Home workout

This app has pre-suggested workouts and training plans that the user can select. However, they are only for exercise at home and not for the gym. Also, the user cannot add their own workouts or edit the existing workouts. It is useful in having a set plan for beginners to start exercising and stick to a regime. However, for someone seriously interested in fitness, it is not useful because the options are too limited and restrictive.

|  |  |
| --- | --- |
| Pros | Cons |
| Offers a training plan | Does not record calories or offer diet plans, which is very inconvenient |
| Displays what muscle/area is being worked | As the name suggests, it only records home workouts and so is useless for people who also go to the gym |

## Muscle booster

This app asks the user questions to gauge their level of fitness.

In my opinion the questions are slightly tailored to the gym so people who work out at home with no equipment may be judged as amateurs even if they’re not - however this point may be unavoidable in my code too.

It has a quite nice display of different muscle groups the user can train, with the option to select single or multiple muscles to train at one time; this is something that I might incorporate into my program.

The layout of the workouts with a separate frame appearing when the workout is clicked is also nice, which I may choose to adopt.

It also splits the workouts into morning workouts and evening, lengthy workouts in a calendar format that I also would possibly adopt using the “calendar” module of python. This app is one of the best apps I’ve seen so far for recording workouts.

# Description of current system or similar current products/games

## The problems with the current systems

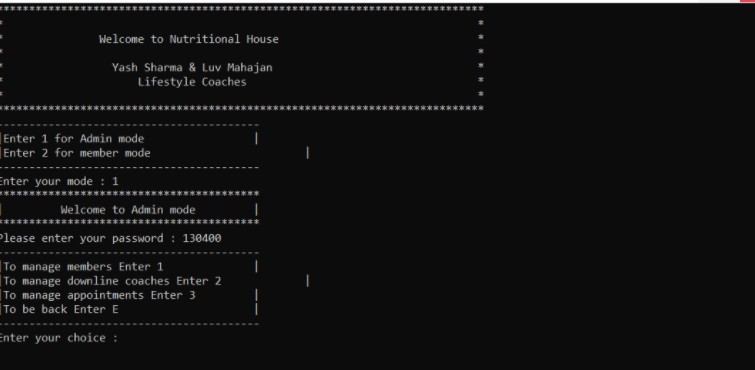
They don’t record both calorie analysis and workout analysis.

Most apps need manual entering to create entire workouts.

Many apps are tailored too much to one side of working out whether that be gym, home, aerobics, calisthenics (a form of strength training consisting of a variety of movements that exercise large muscle groups. These exercises are often performed rhythmically and with minimal equipment).

This restricts variance in workouts as a combination of these is usually the most optimal workout to do. My application might possibly solve this problem.

## Analysis of snippets of code



1. The idea of admin mode and user mode is an interesting concept I may choose to implement. The user enters a password that allows them access to either the owner of the app status or just a guest status and according to distinctive features will be unlocked.

2. Instead of the “downline coaches” and “appointments”, as seen in

the snippet, I will have workouts, split, calories, charts, etc. and use a user interface that will most likely be tkinter, but turtle and pygame are highly unlikely but available options

## Data flow diagram of an existing system

A diagram of a mind-muscle management system

Description automatically generated  
This diagram represents the flow of data split into different sections representing distinct parts of an existing application. The structure seemed intuitive and clever, and I plan to use some of the structure from here, for example the log-in, the members, the machines and weights. Obviously, this application is built for slightly differing purposes than mine so some parts of this diagram and not relevant for me.

## Additional user requirements/limitations

My app doesn’t particularly have a specific scope, able to be used by anyone who pleases, ranging from beginners to very advanced and seasoned people. Tracking and recording are helpful at all experience levels.

A graph of blue and grey bars

Description automatically generated

This diagram shows that the main demographic of people who use fitness apps are 30-39 years of age. However, for all age groups more than 20% use fitness apps. This is why my

application is not targeted towards a specific age group. This [link](https://www.bbc.co.uk/news/technology-55318822) says that coronavirus saw a drastic increase in fitness apps being used, for example Strava, an application I recently downloaded, which has been recommended to me by a friend. I quite like the idea of a login and profile page for the user. The login and signup process will be short. Either one should only take a few seconds maximum to complete. The client will only need to enter their details in designated and very self – explanatory boxes, then press a button Login / SignUp depending on which page their on. The login or signup page will be pre – decided by the code. After the client enters their details and presses the button, the rest will be done by the code, and they will be able to access the application. A status of experience will be decided in the beginning based on questions which will be updated from beginner, to intermediate to advance to display their fitness journey too. Access to the source code will be restricted to avoid unintentional or malicious tampering or avoid people editing their workouts to make it seem like they’re much more advanced than they are. To help me understand my end user requirements I conducted an interview with students in my school, asking questions about the current system to gain an understanding of the needs of my target users

# My systems’ designs

## Context diagram (Data Sources and Destinations)

## A diagram of a user profile

## ERD diagram

A diagram of a program

Description automatically generated

## Data flow diagram for my system

A diagram of a calorie tracker

Description automatically generated

## Database overview - data dictionary for essential entities

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data type** | **Characters** | **Status** |
| **Weight** | **Integer** | **2** | **None** |
| **Reps** | **Integer** | **2** | **None** |
| **Name** | **Alphanumeric** | **10-20** | **Not null** |
| **Exercise** | **Text** |  |  |
| **Exercise\_ID** | **Integer** | **1-2** | **Primary Key** |
| **Equipment** | **Alphanumeric Text** | **5-10** | **None** |
| **Difficulty** | **Alphanumeric Text** | **5-15** | **None** |
| **Day** | **Alphanumeric Text** | **7** | **None** |

# Specific objectives of the proposed system

Based on my research and concept diagrams I have summarised the functionalities of my application into the following objectives:

1. A login page will load when the app is started. This will include a sign in page and a registration page. The registration page with collect name, age, gender, password while the sign in page will allow the user to enter their username and password to access the system
2. There will be a set of questions to decide the experience level based on the user’s answer to sthe questions
3. A page will load showing the user’s profile of experience, age, gender, name with some colour and a picture if they want, else a default will be provided. The user interface will load the profile of the user within a few seconds / immediately of logging in or signing up. All other functionality of the application will also respond within a few seconds / immediately of clicking on any feature of the application.
4. There will be a separate page for 6 different pre-made workouts but I could possibly build in the functionality of being able to change the number of pre-made workouts. Each workout will have editable and removable options and a dropdown menu of all the workouts available. There will also be a choice to add exercises if not available.
5. A dynamic workout saved database and workout history will be in place. The database will update instantly as soon as the user implements any of the update functionalities
6. There will be a separate page that appears when a certain workout is picked showing the exercises to be completed with the weight, reps, sets and equipment, muscle and difficulty level all there. It will be a checkbox beside each one and a master checkbox for every workout that will increase the number of workouts completed when clicked.
7. When a certain exercise is pressed there will be an image and the relevant information there
8. There will be a button to view a graph showing the progress of workouts info: reps against set or reps against weight etc...
9. There will be a separate tab for calories.
10. There will be an entry box to either search up the meal in a databank of meals or manually enter the calories.
11. There will be a circle showing how many calories have been eaten and are left. There will be a clickable button to show a pie chart of the split of calories into different food group categories with different colours representing each category.
12. There will be a history of nutrition showing the meals each day.

# Limitations of the proposed system

## Limited time:

I have about 2.5 months to complete the technical solution to my problem, which means although I will try my best it could possibly not be the absolute best I could produce because of the time limit

## My programming skill

I am trying to implement GUIs, APIs, databases and website scraping. I have not used APIs, databases or website scraping many times before so it will prove challenging when programming it. This means I could produce a relatively more simplistic version of what could be achieved with these tools.

## Lack of powerful processing power

Because I am programming on a cloud-based platform called Replit, or the IDLE shell of python, I have limited processing power and therefore need to optimize my code to run smoothly and efficiently.

## Type of application:

The application is a computer app meaning it can only be used on laptops and PCs, not mobile phones. A mobile or web application could have been other options, but I have chosen to create computer app. This could be used by users on their own computers or the gym computer

# Proposed solution analysis

## Overview / breakdown of problems

After analyzing practical solutions (Fitness pal, Home Workout and Muscle Booster), I have chosen to create my own solution in the programming language of python:

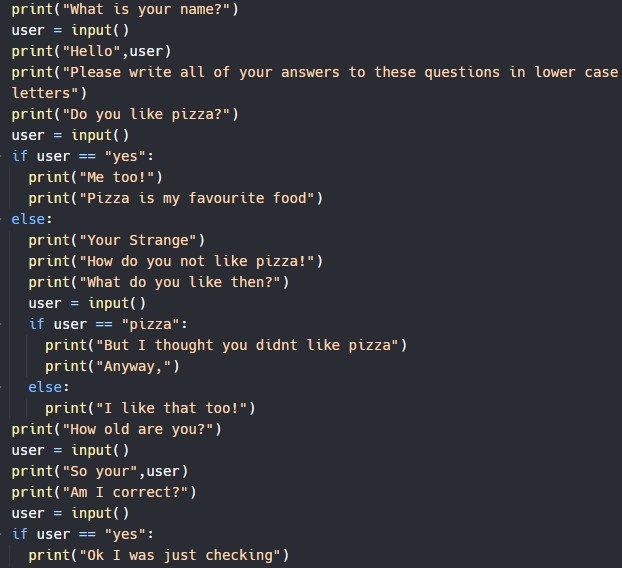
I have chosen to write in python because I have been using and coding with python for approximately 3-4 years now, so time will not be wasted learning a new programming language like JavaScript, Java, Rust, or C++.

Python is a mainly object-oriented dominant language, not as much as Java but still mainly written in object orientation.   
It supports many database types like MySQL, SQLite3, Oracle, Sybase, PostgreSQL etc...…

This therefore means that I can create a relational database with tables for my workouts, calories and recording current and history of workouts.

Moreover, I can then use classes to call this data that has been stored in secondary storage and link a GUI with them to interact with the user and be able to edit and update this database.

Python



The object orientation of python means I can create functions that I can assign buttons, frames, and check buttons to.

Python is a widely used programming language nowadays alongside JavaScript and java which means there are many official websites with free help I can use like, “Stack Overflow”, “GitHub”, and other discussion forums like “Student Room”

Python also has an intuitive interface with easy to add breakpoints and an inbuilt debugger that shows the status of the variables and functions at the point you stop it. This allows for easier and effective debugging by going through each line slowly and checking where error occurs.

|  |  |
| --- | --- |
| **Advantages of using python** | **Disadvantages of using python** |
| Easy to learn and understand syntax and “grammar” | Slow speed |
| Vast collection of varied libraries | High memory usage |
| Free, open-source and a vast community | Garbage collection leads to potential memory losses which could lead to unintended eradication of widgets that weren’t meant to be destroyed. |
| Interpreted language which is helpful due to the inevitable plethora of errors and bugging during creation | Multi-threading enables you to write in a way where multiple activities can continue concurrently in the same program.  Python doesn't do this very well although it does support basic threading. |
| It is the second highest paid computer | The data types of variables in Python can |
| language which means that skill in this language especially for a project of this size is useful | change suddenly, as it is a dynamically typed language. A variable holding a string may hold an integer later, and this can lead to runtime errors. |
| Used in data science, machine learning, and server-side web development | Code can’t be blocked off like in jupyter notebooks and VS studio code reducing the readability of it, |
| Highly scalable as it is suitable for building large-scale web applications. | The presentation and analysis of massive data volumes by a Python app, |
|  | Third-party libraries must be used to compensate for the lack of ability of python. [This won’t affect my project but is an overall disadvantage of python]. |
|  |  |

## Implementation methods

### Tkinter

A python module used to create GUI projects

Tkinter is a graphical user interface that allows users to interact with computers through icons like labels, buttons, entry boxes etc.....

It’s simple and easy to code, interact and learn with

It uses labels to display text

It has many secondary functions which allow for direct typing into GUI window, like checkboxes, buttons, entry boxes, radio buttons, frames, text widgets

It helps supply the visual representation of my code to a layperson person using my program.

It is one of the most used UIs so therefore makes sense that it is easy to use and interact with

It has three geometry managers: place (), pack (), grid () which are powerful and easy to use.

### Pygame

More specifically a game programming module used primarily for graphics and mouse binding events like arrow keys

It is a free and open-source cross-platform library for the creation of complex projects using Python

It has many different modules like music, images, shapes, text, buttons, keys, sound, videos and more.

### OOP classes identified

Create database

Insert into database

Display Workouts

Display Profile

Display specific exercises

Display database in a comprehensible view and allow editability of the view and database where coder allowed

Display meals, and caloric split

Add workouts

Add increment of weights and reps to status table and history of workouts take

Add manual suggestion of workouts

Add split of calories from the meal inputted using an API

Add each workout checked with checkbox to history of workouts table

Show the pie chart for the calories and the line graph or something for the progression of weight and reps and sets [including home workouts] throughout your journey in working out.

# Data sources

### Modules

API - application programming interface

A software that allows communication between two servers/clients acting as the connection between the two seeming to the programmer as a flawless direct connection

### Exercise API

Supplies access to a comprehensive list of hundreds of exercises targeting every major muscle group

Supplies the name, type, muscle, biceps, equipment, difficulty, instructions of each exercise

I can therefore extricate the necessary exercises I require irrelevant information from each one and load them into a relational database I create that I can then manipulate and use to suit my needs

### Nutrition API

Extracts nutrition information from text using natural language processing.

From food blogs to menus to recipes, it can read any text and calculate the corresponding nutrition data.

An intelligent feature of this API is custom portioning: if your text specifies quantities of individual food items or ingredients, the algorithm will automatically scale the nutrition data in the result accordingly.

This will allow me to split the user’s inputted meals into separate categories of calories, enabling me to display both their total amalgamation of calories and their split into:

Total fat

Total saturated fat

Total protein

Total sodium

Total potassium

Total cholesterol

Total carbohydrates

Total fiber

Total sugar

### Internet scraping

Some information that I need that would be inefficient to manually enter

However, if it cannot be found on the APIs I have might need to come from the internet.

I could use several python libraries to seamlessly extract information from certain websites

Then manipulate my code to gather the requisite information I need.

### Beautiful soup

A python library for pulling data out of HTML files

It supplies simple ways of searching, and editing the parse tree

The parse is a visual representation of the syntactic structure of a piece of source code, as produced by a thing called a parser

Parsers are used when there is a need to represent source code abstractly as a data structure/ in a way that is more interpretable to the human brain)

It is all about visualizing the structure of the syntax of the website

It shows the hierarchy of the elements in the code and the relationships between them.

I could use this to pull data from the HTML source code of websites or the homepage of the google browser

### Requests

It is a python module used to send a protocol for fetching resources such as HTML documents [HTTP requests] across a browser.

I can use this in tandem with beautiful soup and selenium to pull data from the internet for my use like researching workouts and calories in the case of this application.

#### Sqlite3

This could possibly be one of the most useful modules.

Sqlite3 is a module that allows me to create, insert into, update, remove and select information to and from a custom database of my creation.

This module is so useful because it allows me to save data on there that will stay there even after the program stops running and can be recalled and updated when the program requires

This drastically increases efficiency and decreases processing time.

Very quick and easy to implement

Lacks user management and security features

Not easily scalable

Is not suitable for big databases

Cannot be customized

### MySQL

MySQL is also easy and quick to implement

It cannot be used on large database sizes and slows down in such cases

Superior SQL databases are Oracle, SQL server, PostgreSQL.

However, they could possibly require many complicated downloads and complications that I might not want to waste time on

### NumPy

A python library primarily used for working with arrays, linear algebra, matrices, complex mathematical concepts like Fourier transformations and calculus

I may need to use this module and my knowledge of product moment correlation coefficient, linear regression, and averages to calculate the values for the charts of progression of weights and fluctuation in amount of certain categorical intake in meals for example:

I could use it to calculate how saturated fat intake has increased/decreased

I could use it to calculate how protein intake has increased/decreased

NumPy arrays are faster and more compact than Python lists:

NumPy uses much less memory to store data and it supplies a mechanism of specifying the data type

Some capabilities of the NumPy module include:

String manipulation

Powerful n-dimensional arrays

Image manipulation capabilities

Easy to use.

### OS

This module is a way for programmers to interact with the operating system.

This module allows them to create new files, folders, handling and changing existing or new directories, transferring files into different directories, checking current path or current directory and other things like this…

This could allow me to save sound files to the current folder that can be listened to by the user, perhaps an audio of the instructions of an exercise.

### Pathlib

It is thought that managing the files and folders is very inconvenient in os

Perhaps “Pathlib” could be a better choice, another module in python used for interacting with the operating system and handling file management

# Potential data structures which may be used

#### Stack

Perhaps a stack will be used for the history of workouts part and progression of weights and reps

The last item checked will be the one displayed at the top, so in essence the first one displayed.

A database will be used to store the workouts

Perhaps a queue can be used to store the calories each day:

Displaying the first set of meals you had on the day you started the application and the last one being the one you entered just before closing the application.

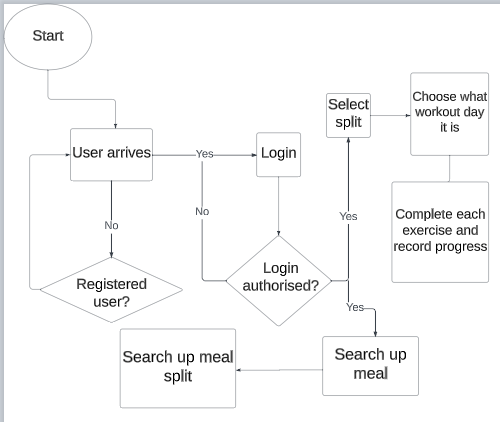
Perhaps an array can be used to store the user’s profile and preferences which can then be transferred to a user’s profile that can be called from and displayed.

# DESIGN

# This design section is the structures and ideas I’ve have decided to implement based on research conducted in the Analysis section:

# Outline system design

## System flowcharts



The user runs the program.

They are taken straight to the login page if they are already a registered user and to the sign in page if they are not.

In the registration they enter their details and are then taken into the rest of the program

In the signing in, the user is given 3 opportunities to enter the right details. If they get it wrong thrice they are told to reset their password.

After correct login they are sent to the page that asks them questions to decide their level of experience. Then they enter the app which shows them their workouts and allows them to view information on a specific exercise

They can view all possible exercises in a separate tab, which also allows them to edit the values, which is where they can update their progress

Then there is the calorie tab where they can search up a meal and it will show them the calorie split

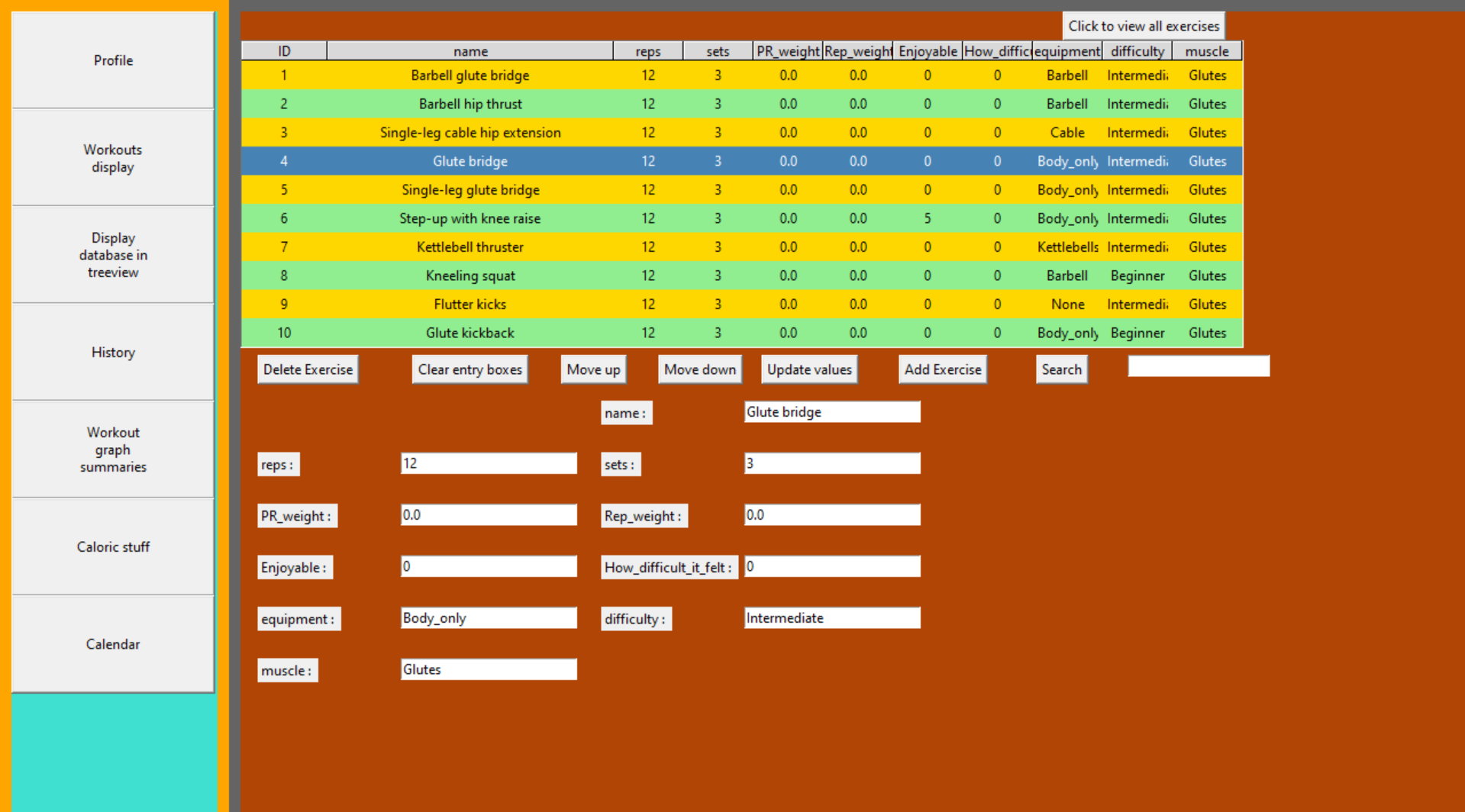
Then the calendar tab shows them a summary of their workouts on any day and their max lift and exercise name

The graph tab shows them a horizontal bar graph of their different weights and reps and exercises against one another. Since there are multiple graphs available, the user can pick which one they want to view

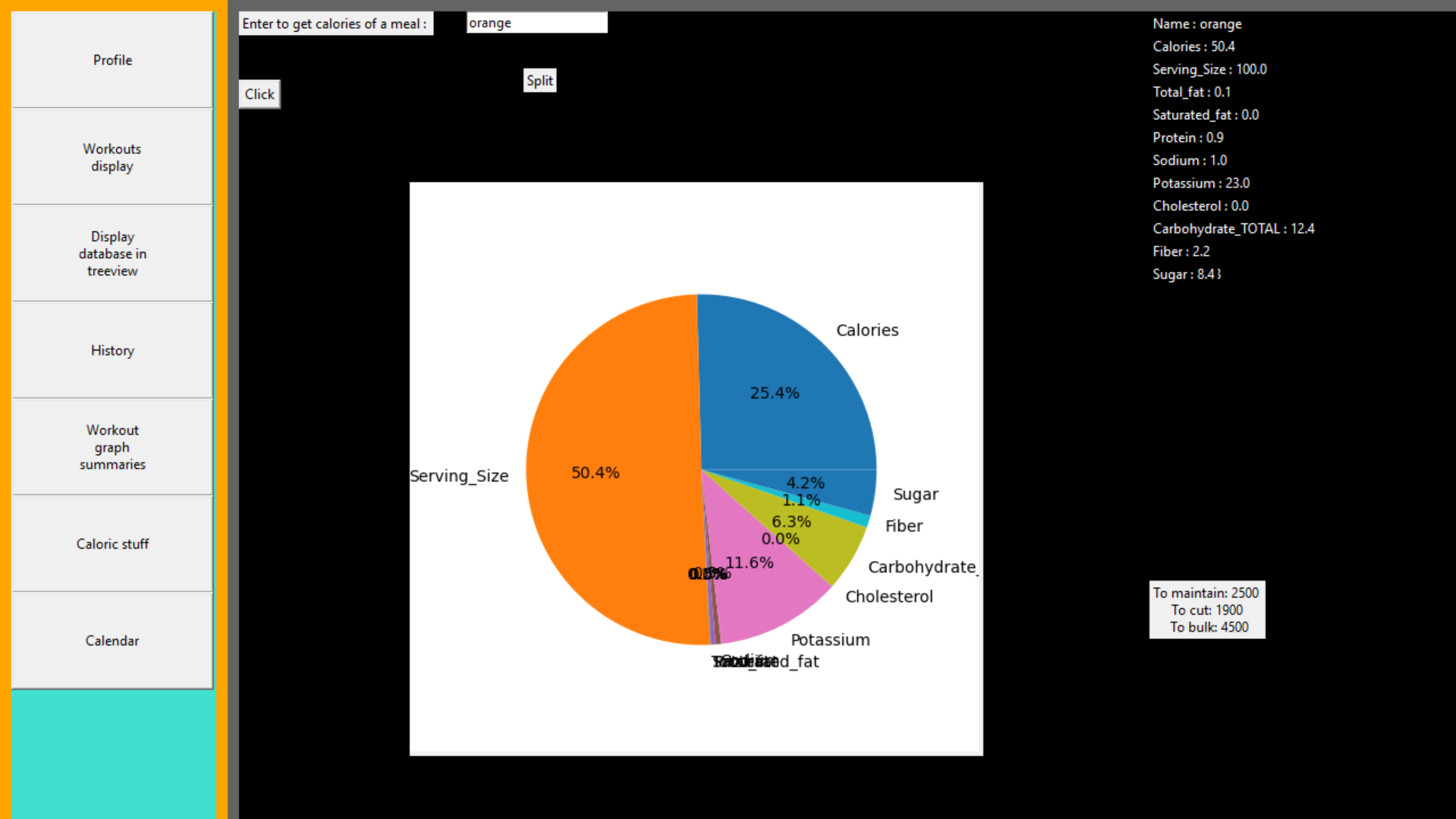
# User Interface Design



This one shows the different workout days possible and a randomly selected exercises generated when pressed. When the exercises are pressed, they show information about it and an image which can be changed and a button which will take them to another tab to update their weights/reps/difficulty etc....



This is where all the updating happens, of all the possible fields that the user is available to update. For the fields that the user can’t update, the widgets are created but just not placed onto the screen. So, in essence those fields are invisible so can’t be edited by the user.



This show calories split and a graph

The UI has been optimised for ease of use of the user. The first tab for example allows for the switch between workout day, specific exercise, change of specific exercise and change of specific exercises picture.

# Hardware specification

A modern dual – core processor or better

4GB of RAM or better

At least 50 MB of free storage space or more

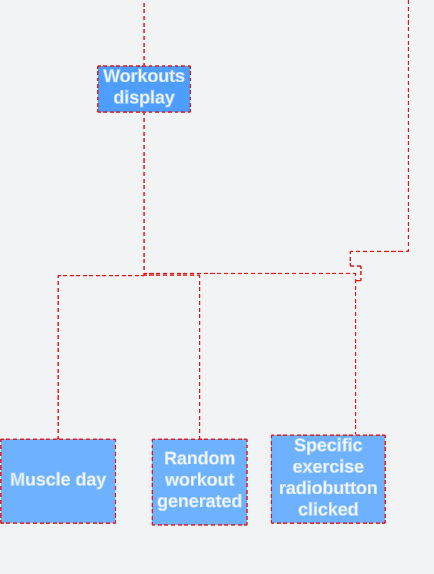
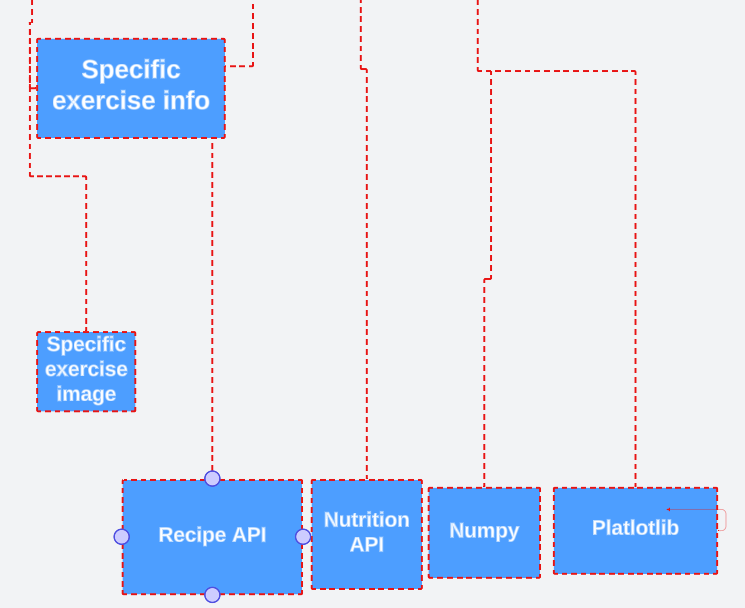
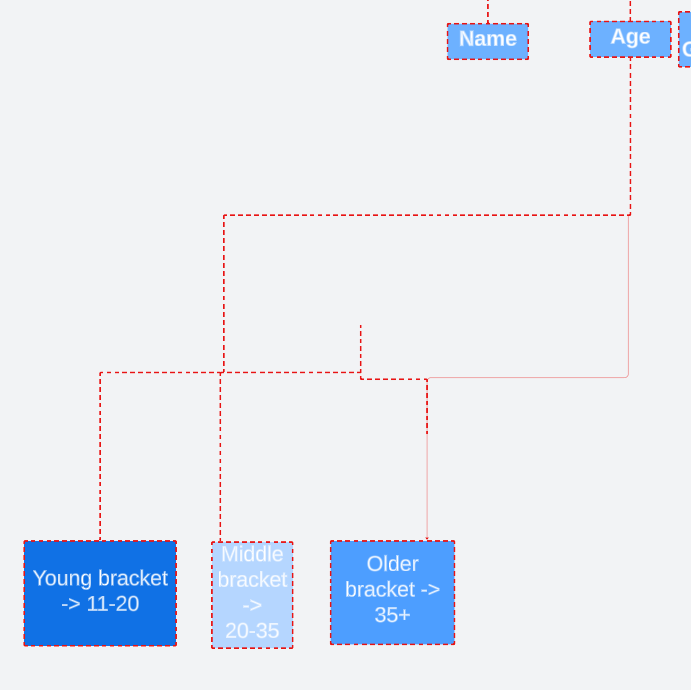
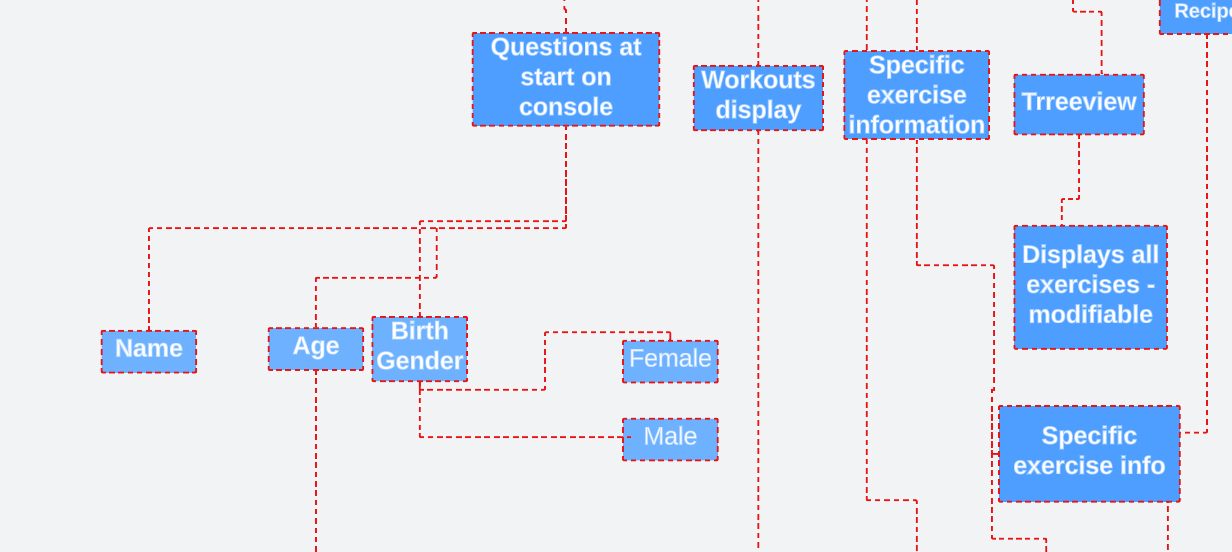
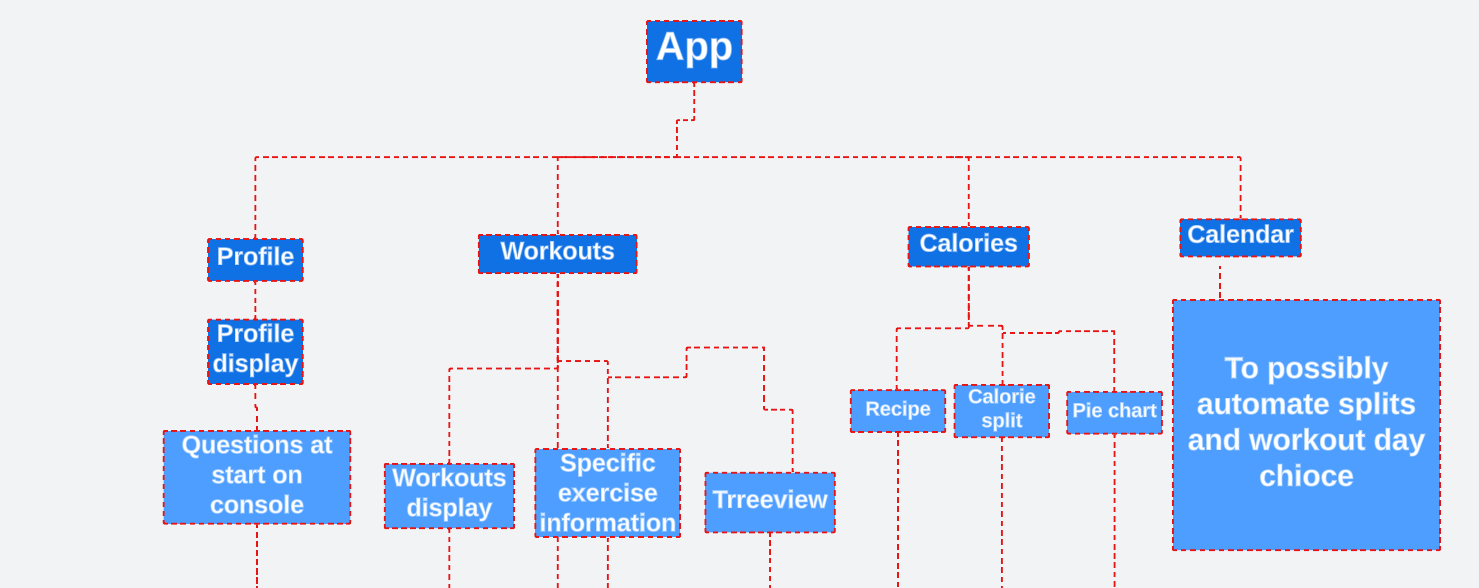
Default integrated graphics (such as integrated GPU or Intel UHD or Iris or baseline AMD Radeon) or better

Extra but not essential specifications:

Intel Core i5 (12th or 13th-gen) or AMD Ryzen 5 (3000 or 5000 series)

512GB or larger NVMe SSD drive

# Program Structure



# Design Data Dictionary

Data dictionary for all data in my program later in design as part of the “normalised database tables”

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data type** | **Characters** | **Status** |
| **Weight** | **Integer** | **2** | **None** |
| **Reps** | **Integer** | **2** | **None** |
| **Name** | **Alphanumeric** | **10-20** | **Not null** |
| **Exercise** | **Text** |  |  |
| **Exercise\_ID** | **Integer** | **1-2** | **Primary Key** |
| **Equipment** | **Alphanumeric Text** | **5-10** | **None** |
| **Difficulty** | **Alphanumeric Text** | **5-15** | **None** |
| **Day** | **Alphanumeric Text** | **7** | **None** |

# Object diagrams and class definitions

1. A function for switching between different frames in my tkinter GUI
2. A function for creating the buttons that switches between the frames. It’s a dynamic function that uses a dictionary to create buttons based on an array of names at the start of the code that defines all the buttons’ names
3. A function called “main” that iterates through a certain folder on local storage looking for a certain file and returns that file path
4. Two functions for scraping images from the internet and saving them to local storage
5. A class to display random auto generated workouts in Tkinter GUI which is dynamic so can be edited based on the user’s requirements
6. A class to create a dynamic Treeview display of all the exercises available that can be modified and changed based on the user’s needs
7. A class for creating GIFs in Tkinter. I use this throughout multiple parts of my code so a separate class for it is a good idea
8. A class for displaying specific exercise information and a check button to mark when the exercise is done which automatically sends the user to a page where they can record their exercise
9. A class for connecting to my google sheets and manipulating it through my tkinter code
10. A class to represent the calendar display
11. A class to represented caloric manipulation, inputting a meal and getting the graph split and food-calorie split. The calculations are accurate upto the point of the API I am using. I might implement the fitness pal database but since it may take too long that may be outside the scope of the project.

# Data Structures

* Stack:
  + History of workouts – first entry put in is the last one to be displayed
* Sqlite3 database
  + To save all the workouts, exercises and be updated
* Dictionary
  + To hold the entry and label widgets in the treeview frame of the GUI
  + To hold the buttons to switch between the frames in the GUI
* Queue
  + The array that holds the information for the profile – the first information is the first one displayed [left-most or uppermost depending on positioning of information on the page]

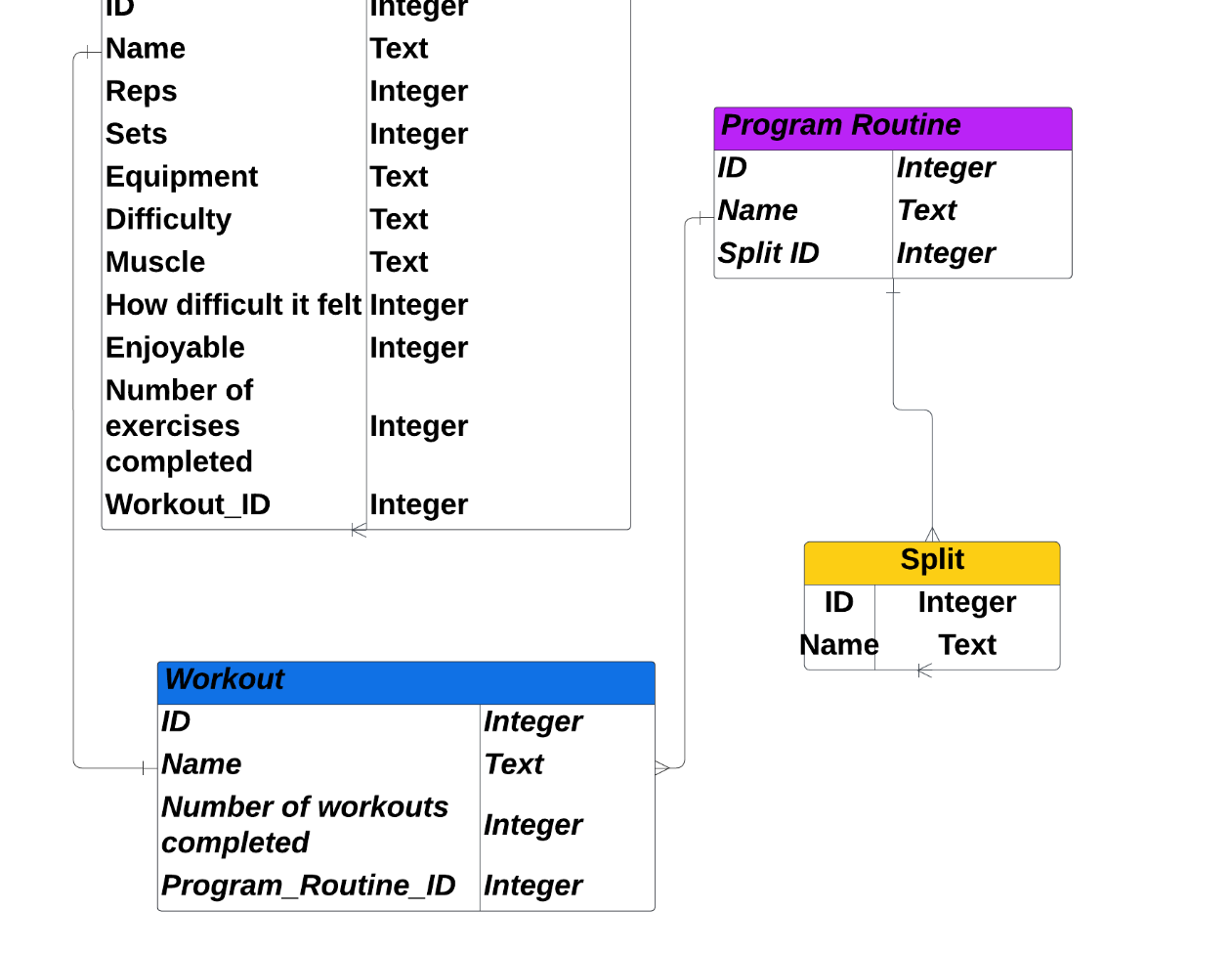
# File Organisation

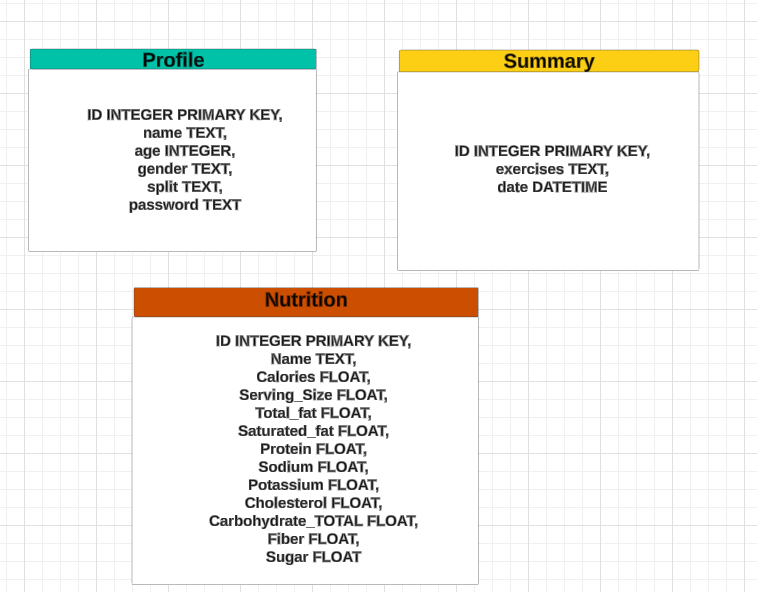
* There is a nea.json file to link my code to my google sheets.
* There is a folder of images and gifs scraped and saved locally to the disk

# APIs used

1. Nutrition API
2. Exercise API
3. Calories burned API

# Entity-relationship diagram





# Normalised database tables

|  |  |
| --- | --- |
| **Attribute Name** | **Primary key? Foreign key?** |
| Exercise\_ID | Primary key |
| Name | Normal attribute |
| Reps | Normal attribute |
| Sets | Normal attribute |
| Equipment | Normal attribute |
| Difficulty | Normal attribute |
| Muscle | Normal attribute |
| How difficult it felt | Normal attribute |
| Enjoyable | Normal attribute |
| Number\_of\_exercises\_completed | Normal attribute |
| Workout\_ID | Foreign key |
| Workout\_ID | Primary key |
| Name | Normal attribute |
| Number\_of\_workouts\_completed | Normal attribute |
| Program\_Routine\_ID | Foreign key |
| Program\_Routine\_ID | Primary key |
| Name | Normal attribute |
| Split\_ID | Foreign key |
| Split\_ID | Primary key |
| Name | Normal attribute |

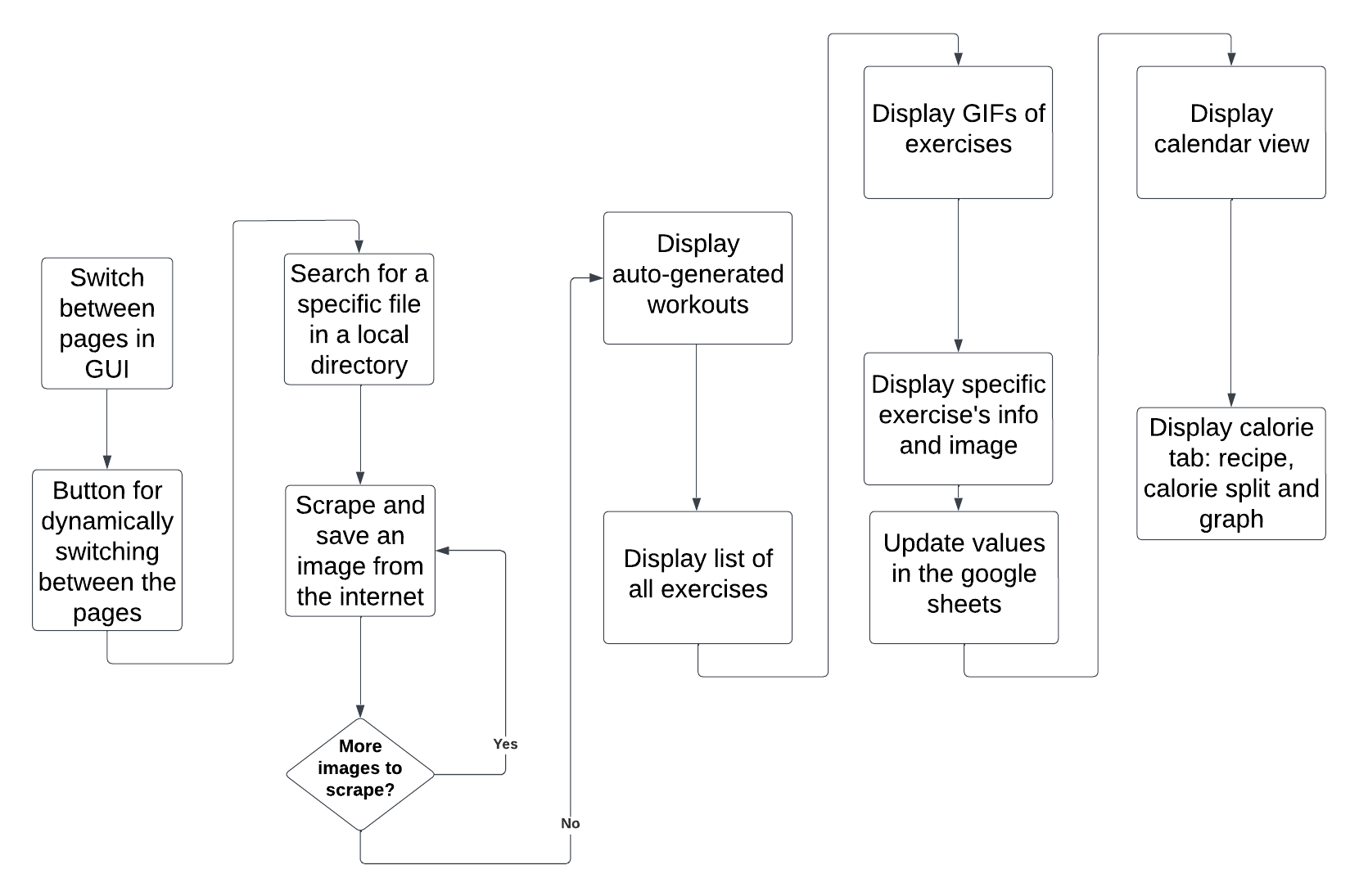
|  |  |
| --- | --- |
| **Attribute Name** | **Sample data** |
| Exercise\_ID | 1 |
| Name | Pullups |
| Reps | 10 |
| Sets | 3 |
| Equipment | None |
| Difficulty | 5 |
| Muscle | Back |
| How difficult it felt | 6 |
| Enjoyable | 7 |
| Number\_of\_exercises\_completed | 3 |
| Workout\_ID | 3 |
| Workout\_ID | 5 |
| Name | Glutes |
| Number\_of\_workouts\_completed | 7 |
| Program\_Routine\_ID | 4 |
| Program\_Routine\_ID | 2 |
| Name | Legs |
| Split\_ID | 8 |
| Split\_ID | 9 |
| Name | 5-day split |

|  |  |
| --- | --- |
| **Attribute Name** | **Data type [Sqlite3]** |
| Exercise\_ID | INTEGER |
| Name | TEXT |
| Reps | INTEGER |
| Sets | INTEGER |
| Equipment | TEXT |
| Difficulty | INTEGER |
| Muscle | TEXT |
| How difficult it felt | INTEGER |
| Enjoyable | INTEGER |
| Number\_of\_exercises\_completed | INTEGER |
| Workout\_ID | INTEGER |
| Workout\_ID | INTEGER |
| Name | TEXT |
| Number\_of\_workouts\_completed | INTEGER |
| Program\_Routine\_ID | INTEGER |
| Program\_Routine\_ID | INTEGER |
| Name | TEXT |
| Split\_ID | INTEGER |
| Split\_ID | INTEGER |
| Name | TEXT |

|  |  |
| --- | --- |
| **Attribute Name** | **Data type [Python]** |
| Exercise\_ID | Integer |
| Name | String |
| Reps | Integer |
| Sets | Integer |
| Equipment | String |
| Difficulty | Integer |
| Muscle | String |
| How difficult it felt | Integer |
| Enjoyable | Integer |
| Number\_of\_exercises\_completed | Integer |
| Workout\_ID | Integer |
| Workout\_ID | Integer |
| Name | String |
| Number\_of\_workouts\_completed | Integer |
| Program\_Routine\_ID | Integer |
| Program\_Routine\_ID | Integer |
| Name | String |
| Split\_ID | Integer |
| Split\_ID | Integer |
| Name | String |

|  |  |
| --- | --- |
| **Attribute Name** | **Size or range** |
| Exercise\_ID | Max 4 letters |
| Name | Max 30 characters |
| Reps | Max 4 letters |
| Sets | Max 4 letters |
| Equipment | Max 30 characters |
| Difficulty | Max 4 letters |
| Muscle | Max 30 characters |
| How difficult it felt | Max 4 letters |
| Enjoyable | Max 4 letters |
| Number\_of\_exercises\_completed | Max 4 letters |
| Workout\_ID | Max 4 letters |
| Workout\_ID | Max 4 letters |
| Name | Max 30 characters |
| Number\_of\_workouts\_completed | Max 4 letters |
| Program\_Routine\_ID | Max 4 letters |
| Program\_Routine\_ID | Max 4 letters |
| Name | Max 30 characters |
| Split\_ID | Max 4 letters |
| Split\_ID | Max 4 letters |
| Name | Max 30 characters |

# Algorithms



This flowchart is the representation of the approximate solution to the dynamic workout objectives

# Top-down description of modules

import sqlite3

**I need sqlite3 to create and utilise a database**

import os

**I need os to manipulate files on my local storage and write programs that can adapt and carry out processing on any user’s local storage**

import string

import re

**I need re and string to carry out generalised string character recognition**

import babel

from babel import numbers

import math

import numpy as np

**I need babel, numpy and math for more complex maths operation like resizing image arrays**

import tkinter as tk

**I need tkinter to create a GUI**

from tkinter import ttk,filedialog as fd,messagebox

**I need filedialog and messagebox to allow user to access local storage folders through the code**

from PIL import Image,ImageTk, Image, ImageSequence

**I need PIL to make Images and GIFs in my GUI**

import requests

**I need requests to utilise my API information and to carry out internet scraping**

from bs4 import BeautifulSoup

from urllib.parse import urljoin

import urllib.request

import threading

import gspread

**I need urllib.parse, urllib.request, threading and gspread and bs4 to carry out internet scraping**

import random

from collections import Counter

**I need random to generate random numbers and Counter to carry out string frequency occurrence**

import time

import imageio.v2 as imageio

**I need imageio.v2 to manipulate images**

from datetime import datetime

**I need datetime to extract and manipulate dates**

from oauth2client.service\_account import ServiceAccountCredentials

from googleapiclient.http import MediaIoBaseDownload

from googleapiclient.discovery import build

**I need these to access and update my google sheets**

import tkcalendar

from tkcalendar import \*

**I need tkcalendar to create and use a calendar in my GUI**

import subprocess

**I need subprocess to dynamically install modules**

import ast

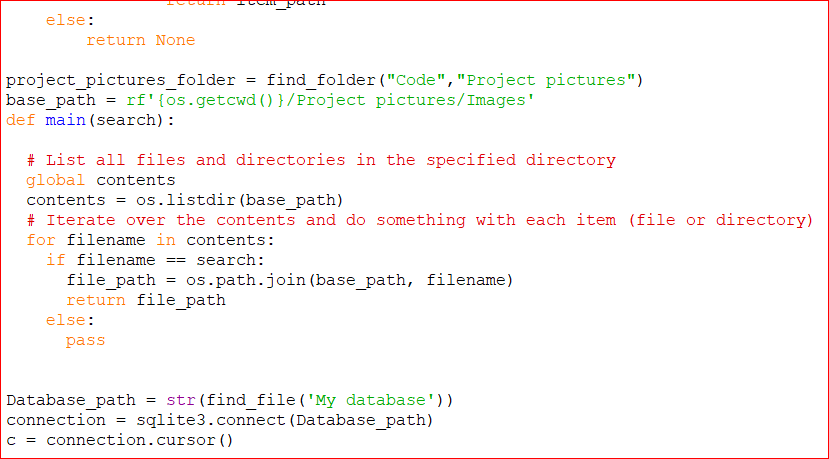
**I need ast to produce the literal array from a string that hold an array**

# Detailed test data

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute**  **Name** | **Valid data example(s)** | **Invalid data example(s)** | **Boundary data example(s)** |
| Exercise\_ID | 3 | -6, True, “Hello”,17.5 | 0 |
| Name | “Lateral pulldown” | 17, True | "True” or “False” |
| Reps | 3 | -6, True, “Hello”,17.5 | 0 |
| Sets | 3 | -6, True,  “Hello”,17.5 | 0 |
| Equipment | Barbell | True,17, -5,17.8 | "True” or “False” |
| Difficulty | Beginner | True,17, -5,17.8 | "True” or “False” |
| Muscle | Glutes | True,17, -5,17.8 | "True” or “False” |
| How difficult it felt | 3 | -6, True, “Hello”,17.5 | 0 |
| Enjoyable | 3 | -6, True, “Hello”,17.5 | 0 |
| Number\_of\_exercises\_completed | 3 | -6, True, “Hello”,17.5 | 0 |
| Workout\_ID | 3 | -6, True, “Hello”,17.5 | 0 |
| Workout\_ID | 3 | -6, True, “Hello”,17.5 | 0 |
| Name |  |  |  |
| Number\_of\_workouts\_completed | 3 | -6, True, “Hello”,17.5 | 0 |
| Program\_Routine\_ID | 3 | -6, True,  “Hello”,17.5 | 0 |
| Program\_Routine\_ID | 3 | -6, True, “Hello”,17.5 | 0 |
| Name | “Lateral pulldown” | 17, True | "True” or “False” |
| Split\_ID | 3 | -6, True, “Hello”,17.5 | 0 |
| Split\_ID | 3 | -6, True, “Hello”,17.5 | 0 |
| Name | “Lateral pulldown” | 17, True | "True” or “False” |

# TECHNICAL SOLUTION:

## Basics file



These 2 screenshots above are the “basic.py” file. They serve 3 functions:

*To import the necessary imports that are needed throughout the code and are also not GUI imports*

*A function that returns the file path for a specific folder on any user's computer*

*These takes 2 inputs:*

*Name of the parent folder name*

*Name of target folder name*

*It uses the method in the OS module called “.getcwd(~)”. This method returns the beginning file path of the user's current working directory enabling the file path to adapt to any computer*

*It then joins the parent folder name to the getcwd () file path to create the file path to the parent folder name on any computer.*

*It then checks if this file path exists and whether this file path is a folder. Then it iterates through all the files and folders in that path until it finds the one that matches the required folder name. Next it joins this folder name to the file path to the parent folder name and returns this file path. It also checks whether there is a valid parent folder name input – if there isn’t it simply uses the getcwd () file path as the parent folder file path*

*A function that returns the file path for a specific file on any user's computer*

*This one take 1 input -> Name of target file name*

*It uses the method in the os module called “. getcwd ()”. This method returns the beginning file path of the user's local working directory enabling the file path to adapt to any computer*

*It then joins the parent folder name to the getcwd () file path to create the file path to the parent folder name on any computer.*

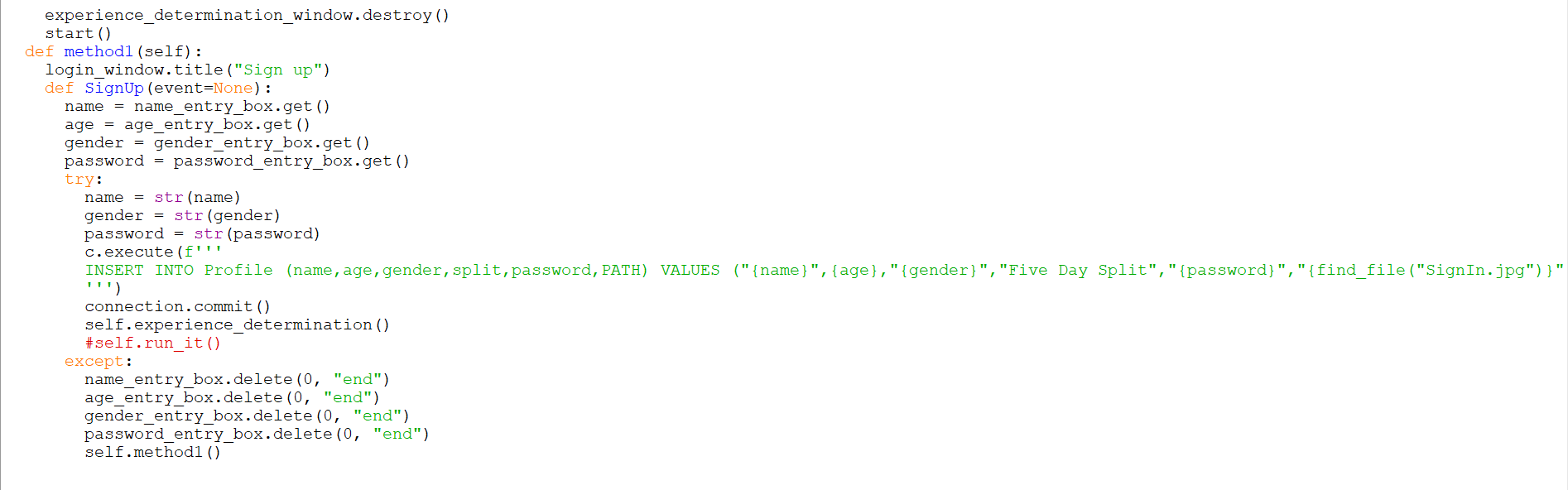
*It then checks if this parent folder file path exists and whether this file path is a folder, then it iterates through all the files and folders in that path until it finds the one that matches the required file name. Lastly it joins this file name to the file path to the parent folder name and returns this file path*

*It uses the find\_file function to get the file path to the database file and then initialises the connection to the database*

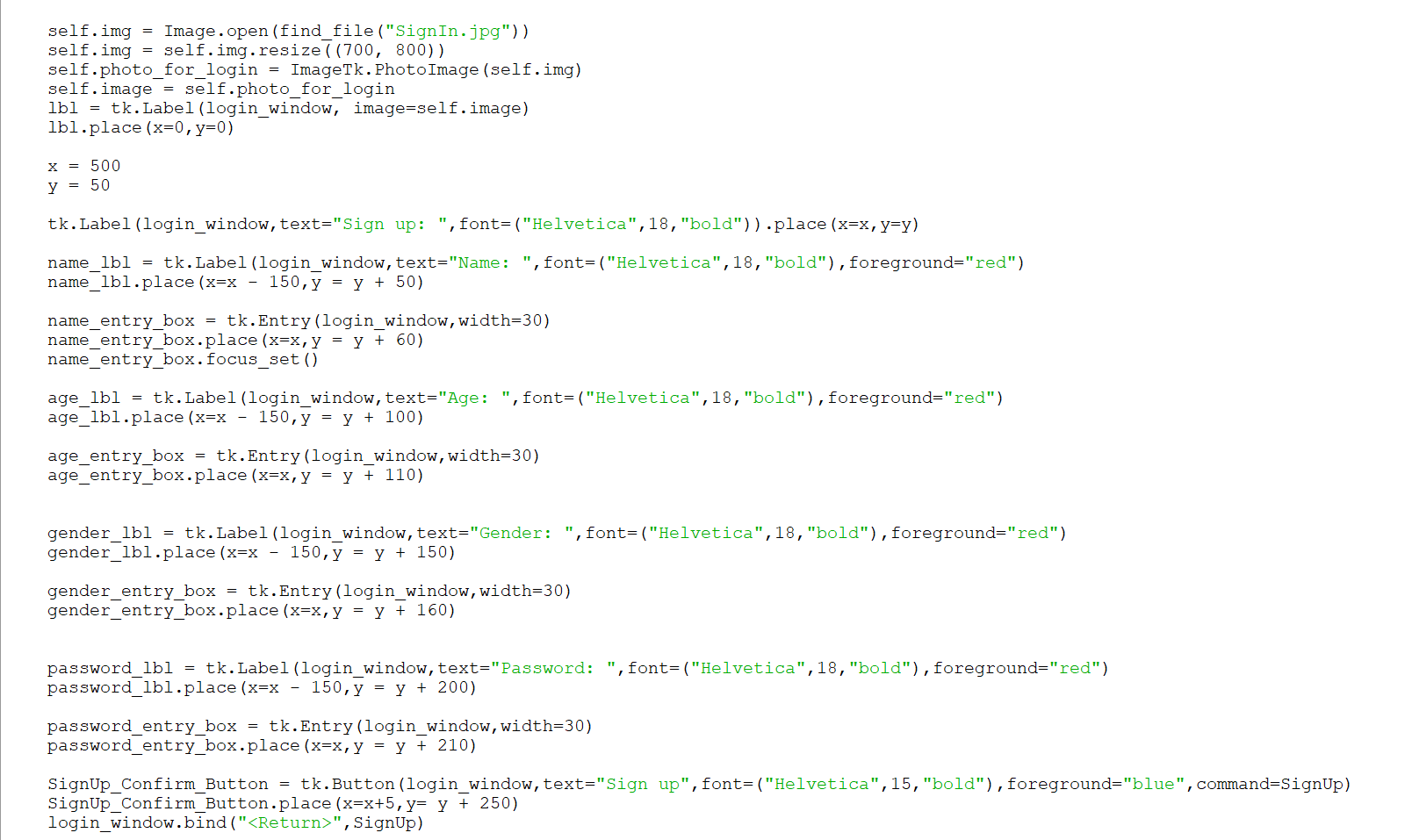
## Login system:











**This is my login and sign-up system.**

*Firstly, it checks whether there are any entries in the database*

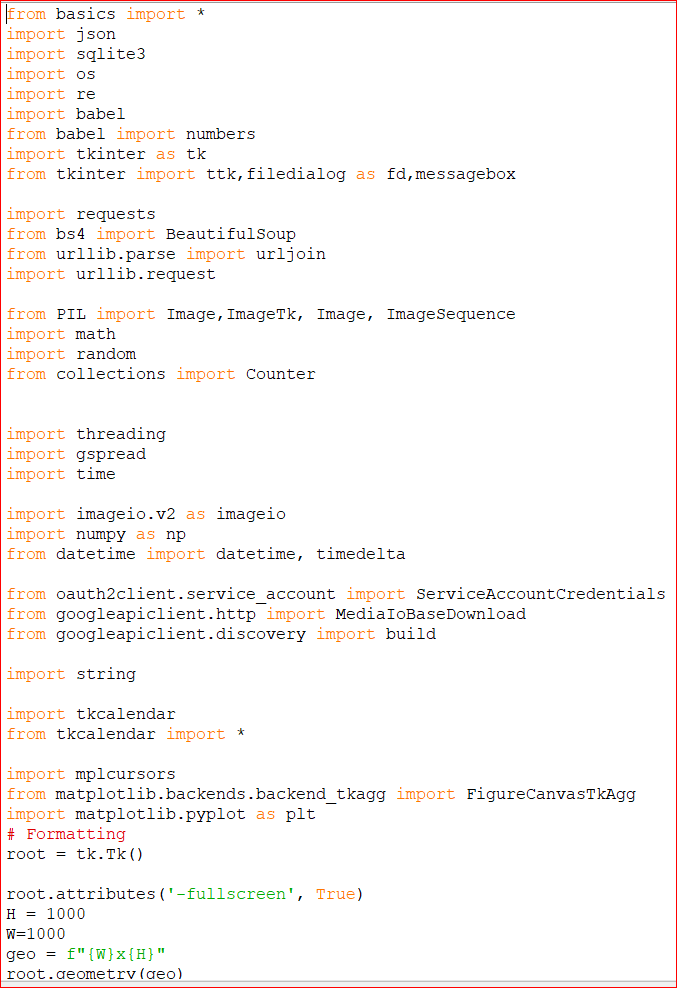
*If there are entries, then it goes to the login part by entering 2nd method*

*If there aren’t entries, then it goes to the sign-up part by entering 1st method*

*The algorithm* ***recurses*** *based on valid or invalid inputs*

*Then it enters the experience level determination algorithm. It is a very simple algorithm that determines the user’s experience level based on arbitrary value ranges based on the users answers to some questions*

## Constants file





*This imports all required modules for all the GUIs functionality [including the imports from the basics file]*

*This initialises the dynamic addition of the tabs in the GUI*

*This stores the tab number like “tab1” in an array of strings and the corresponding colours in another array and the names of the tabs in another array*

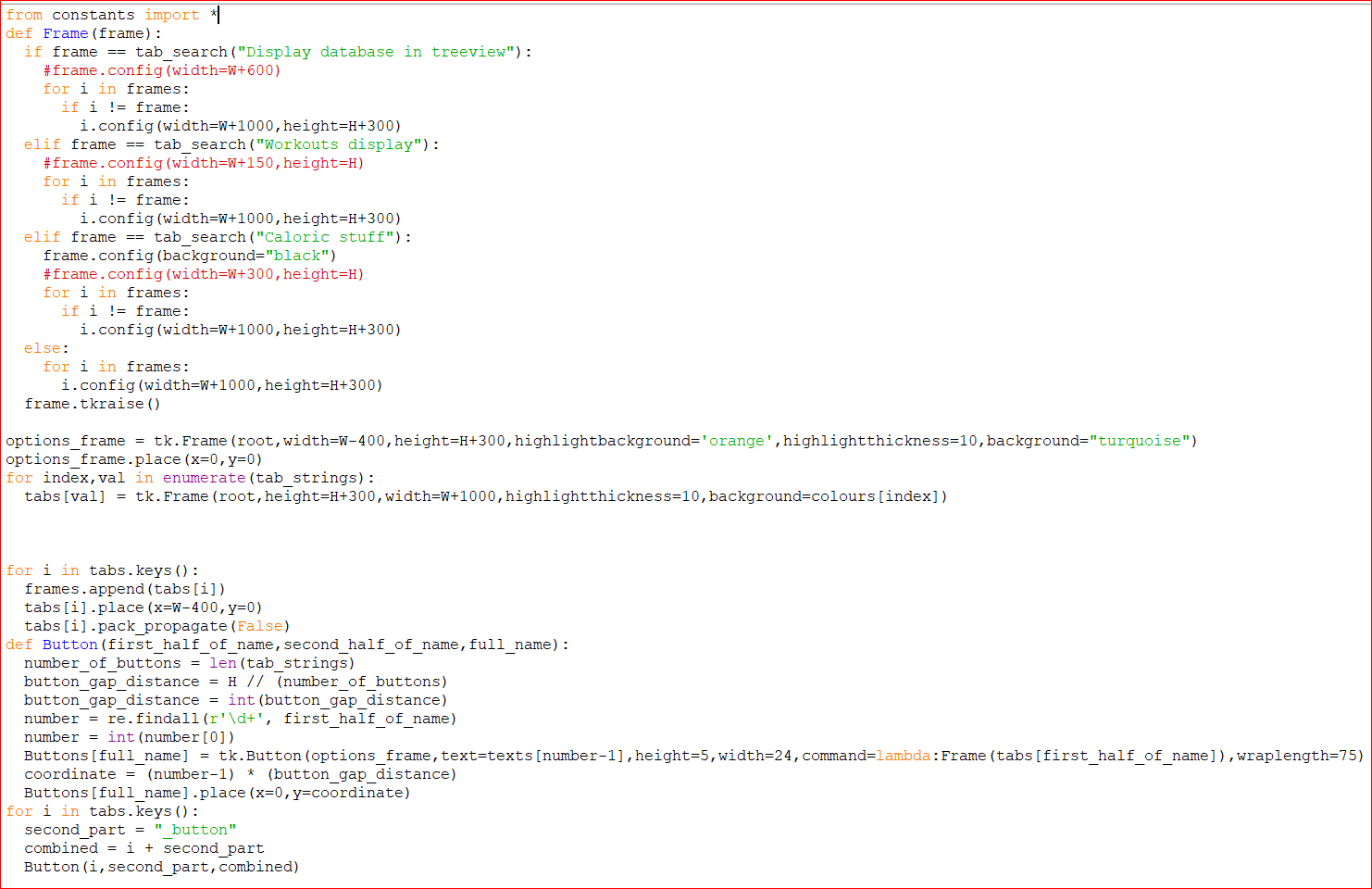
*The tab\_search function if an extremely useful function that will return the tab number based on the name of the tab entered so tabs can be easily added and removed, and their colours easily altered.*

*The main function can iterate through the pictures that are stored in a folder and return the file path of any one of them needed based on the inputted name*

*There are separate files for constants and basics because this file initialises all the GUI functionality required, and “basics” has only the bar imports and modules required throughout the code.*

*This is so that the functions that don’t require GUI initialisation and imports can run more efficiently. As seen, the constants file imports the basics file meaning the constants file contains ALL the necessary imports and functions required.*

## GUI basics file



*This file handles the appearance of the basics of the GUI*

*It creates the frame where the buttons to switch between tabs sit*

*The next line loops through the tab\_strings array and dynamically adds to the empty dictionary tabs.*

*It assigns each string in the tab\_strings array to a Frame in tkinter and adds it to the tabs dictionary*

*It uses the enumerate function to return both the value and index at each point in the tab\_strings so that the colours array can be iterated through simultaneously as well*

*Then by iterating through the tab names, the tk.Frames are appended to the frames array and the frames are all placed in the same place one on top of another. The “pack. Propagate (False)” means that the tab is stopped from auto changing its dimensions based on the widgets inside it*

*After that it iterates through the tab names again and a name is formed for each frame to represent the button for that frame. For example, the frame “Workouts display” would have the name “Workouts display\_button”. The “\_button” string is added to the name of the key*

*Then the Button function is called with 3 parameters given:*

*The name of the tab/ the first part of the button name*

*The second part of the button name i.e... “\_button”*

*The full combined name of them added together*

*First the place of each button is calculated by dividing up the height of the option frame by the number of buttons, which is represented by the length of the tab\_strings array, or texts array or colours array. It doesn’t really matter which one because they all have the same length – because the colours and tab\_strings arrays are dynamically created from the text array*

*The re.findall(r’\d+’,first\_half\_of\_name) finds all the numbers in the string. It returns it as a string in an array so the line number = int (number [0]) manipulates it into a number*

*The next line assigns the full button name string to a Button in tkinter and adds it to the Buttons empty dictionary*

*The Buttons are placed within the options\_frame window and their texts are taken from the pre-defined texts array in the constants file. “Number –1” is used to make sure the right element is accessed because indexes in python start from 0.*

*The coordinates increase because of the number value so that the buttons are placed neatly one beneath each other*

*The coordinates are calculated and kept in the coordinates variable which is then used in the y axis of the place () geometry manager.*

## Profile



*The self.name value is obtained by searching the database*

*The x variable stores the file path of the profile picture of the user, which is queried from the database*

*A button to change the profile picture is created*

*The names variable stores the column names*

*The info variable stores the user’s information: name, age, gender and split*

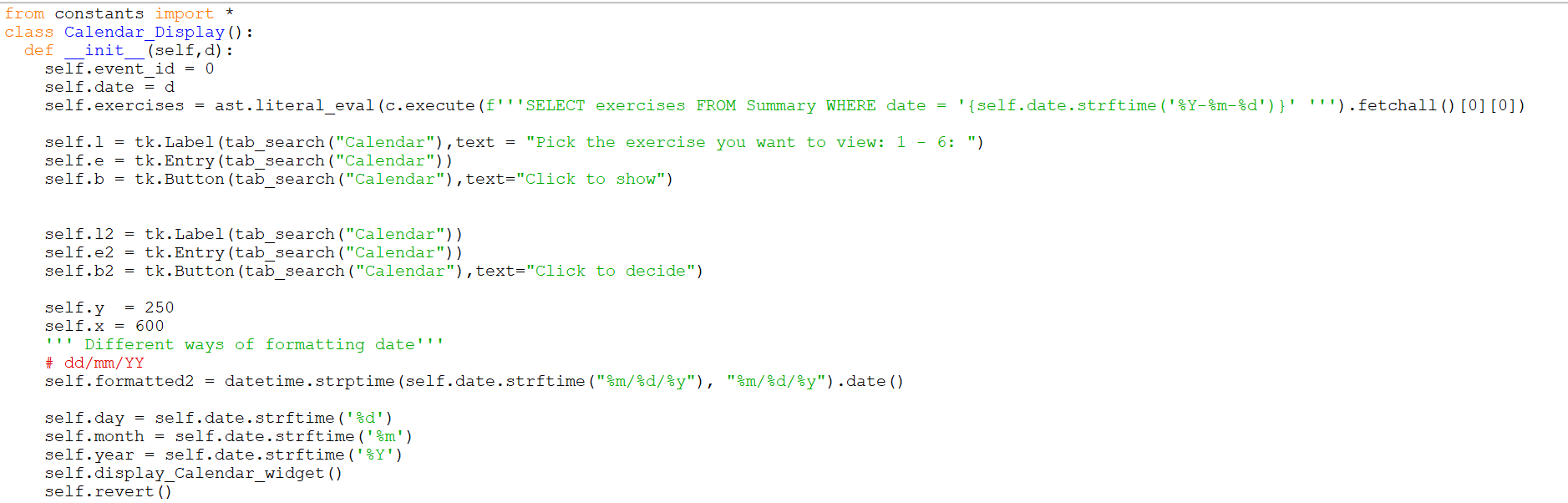
*The user's information is iterated through and displayed on the Profile tab*

*The dialog method uses filedialog to open a dialog box in tkinter so the user can choose the file they want from their computer as a profile picture. The dialog box only shows jpeg and gif files so compatibility with tkinter is already dealt with since the user cannot pick files that aren’t compatible with tkinter.*

*When a valid file it picked and open is pressed the change method is run.*

*This renders the selected picture to the profile tab and updates the file path in the database so that next time the user loads the application, there chosen profile picture is displayed.*

## Calendar



*This part picture initialises all the attributes of the calendar class:*

*The exercises attribute finds the absolute list from a string that holds a list*

*The labels, entries and buttons for the choice of summary are initialised here*

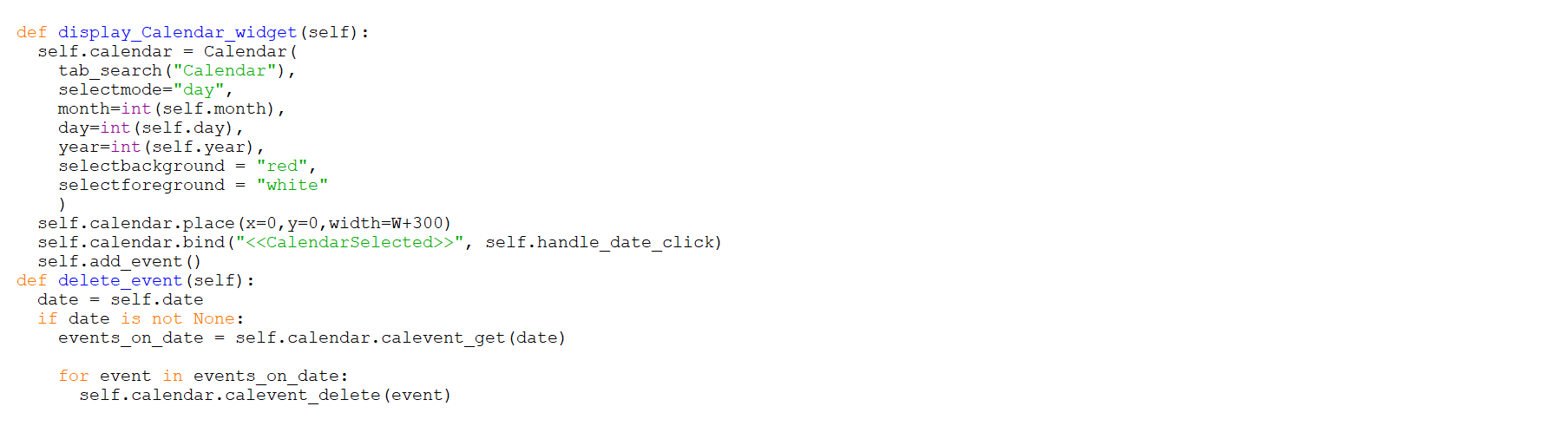
*The positioning of the labels, entries and buttons are initialised here in the self.x*

*and self.y attributes*

*The self. formatted2 attribute manipulates the datetime input into something*

*useable in the code*

*Three methods are instantiated here: Get,add\_event and revert*



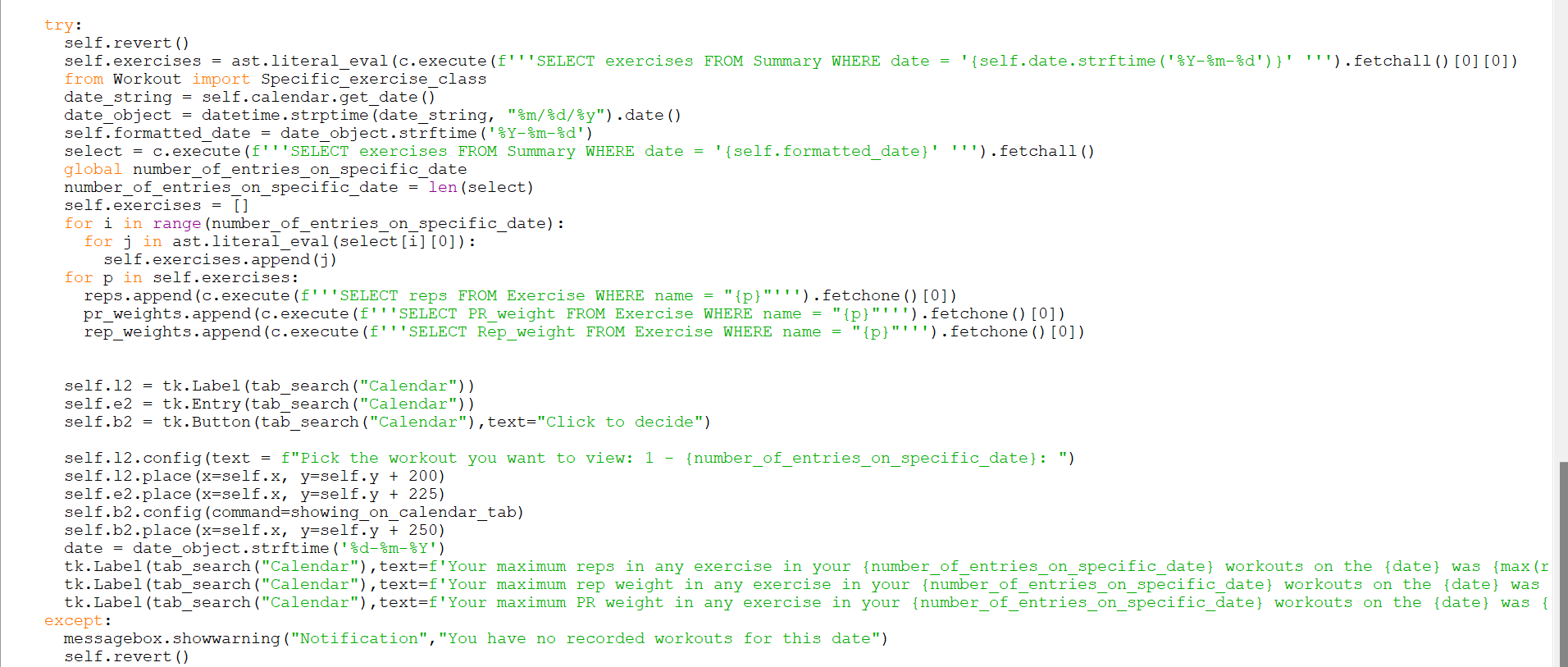
*The display\_Calendar\_widget creates and places the tkcalendar widget down*

*In the calendar tab, hence the name of the method. It also binds the choice*

*of any event on the tkcalendar widget to the function self.handle\_date\_click*

*The delete\_event method removes all events on a given date*





*This method handles what happens when any event on the tkcalendar is*

*Pressed. First the entry boxes are cleared, and labels destroyed to make*

*sure, only new data is considered.*

*The date that is being clicked on is extracted and formatted to a form that*

*can be used.*

*It is changed into a date object from a datetime object*

*It is changed from month [2 digit]/day [2 digit] /year [2 digit] to a year [4 digit]/months [2 digit]/ day [2 digit] format*

*Then all exercises with this date is searched for in the database*

*The number of records found is saved to variable “number\_of\_entries\_on\_specific\_date”*

*Then the l2, e2 and b2 are re-created, configured and placed*

*Certain values are entered by the user in the GUI, then:*

*In the showing\_on\_calendar\_tab function:*

*Whichever workout the user chose affects how the exercise attribute is formatted. If the user chose the seconds workout, then choice would become 1 and the second set of exercise results become exercises*

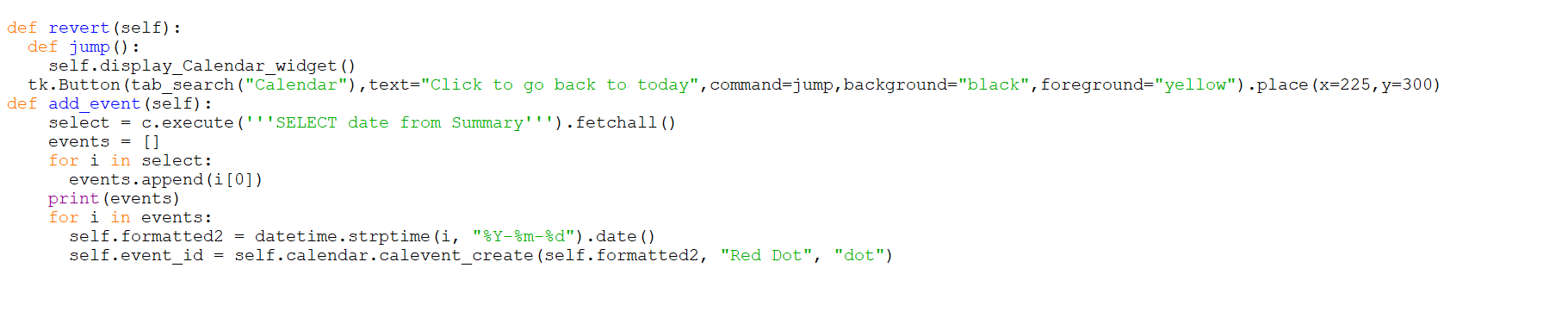
*Then l, e, and b are re-created, configured and placed and link to the summary\_part1 function*

*In the summary\_part1 function:*

*The value of the entry box decides which exercise is shown. The index is one lower than the entered value to keep index integrity [indexes start from 0]*

*The show function from the Workout file [discussed after this] shows the specific exercise*

*The calendar widget is re-created, and events re-added, and all the label, entries and buttons re-created ready for the next event click*



*The revert function re-creates the calendar and changes it back to the current day*

*The add\_event method searches the database for all available dates and loops through them and adds them to the calendar as events*

## Dynamic workout display



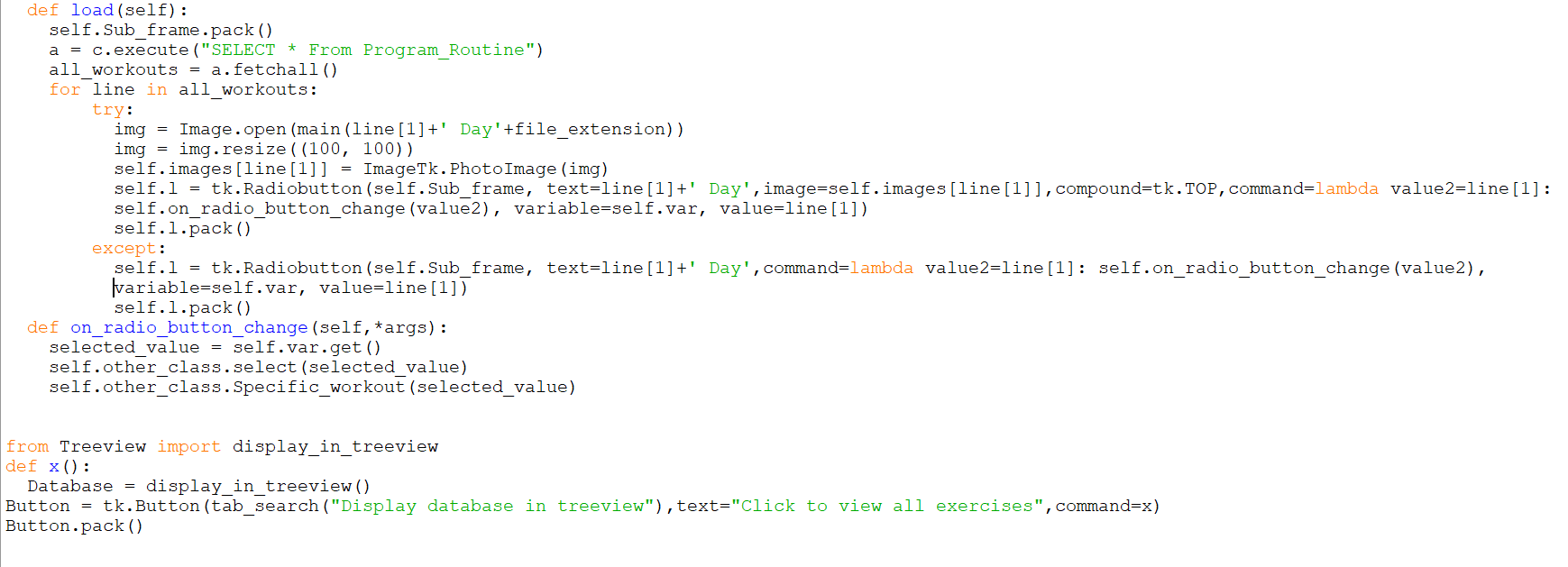
*This initialises all the attributes needed for this class. It also instantiates the*

*“load” method and assigns another class to an attribute in this class, so that the other class can be called from this class linking them.*

*The line self.var.trace('w', self.on\_radio\_button\_change) in the code sets up a trace on the Tkinter variable self.var.*

*In Tkinter, a trace is a way to associate the callback function with changes to my Tkinter variable.*

*Here, the trace is set on write operations ('w'), meaning the callback function self.on\_radio\_button\_change will be called whenever the value of self.var is changed.*



*This code displays all the different workout days as a radiobutton with Images*

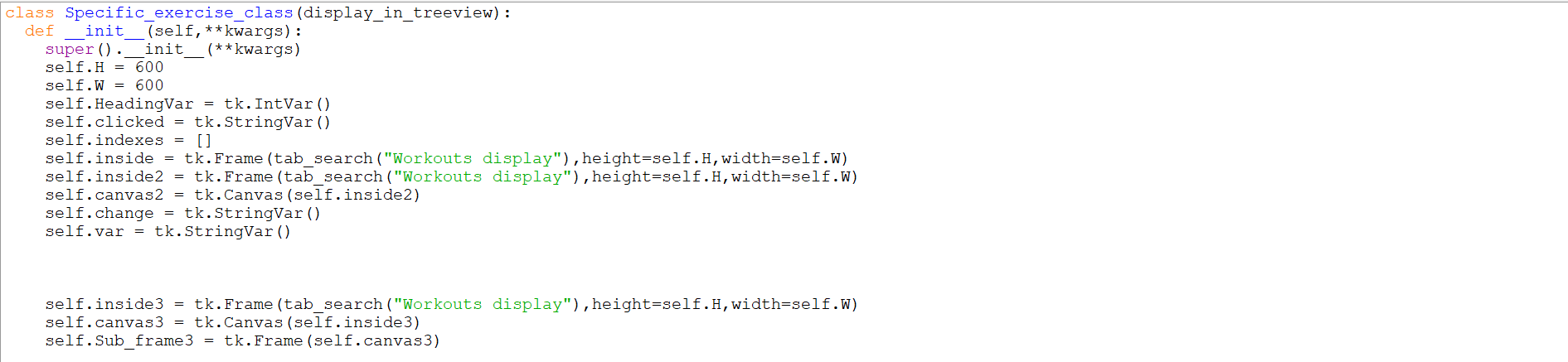
*When the radiobutton is clicked the function self.on\_radio\_button\_change is*

*Called with the parameter passed as the name of the workout day for example “Leg” for “Leg Day”.*

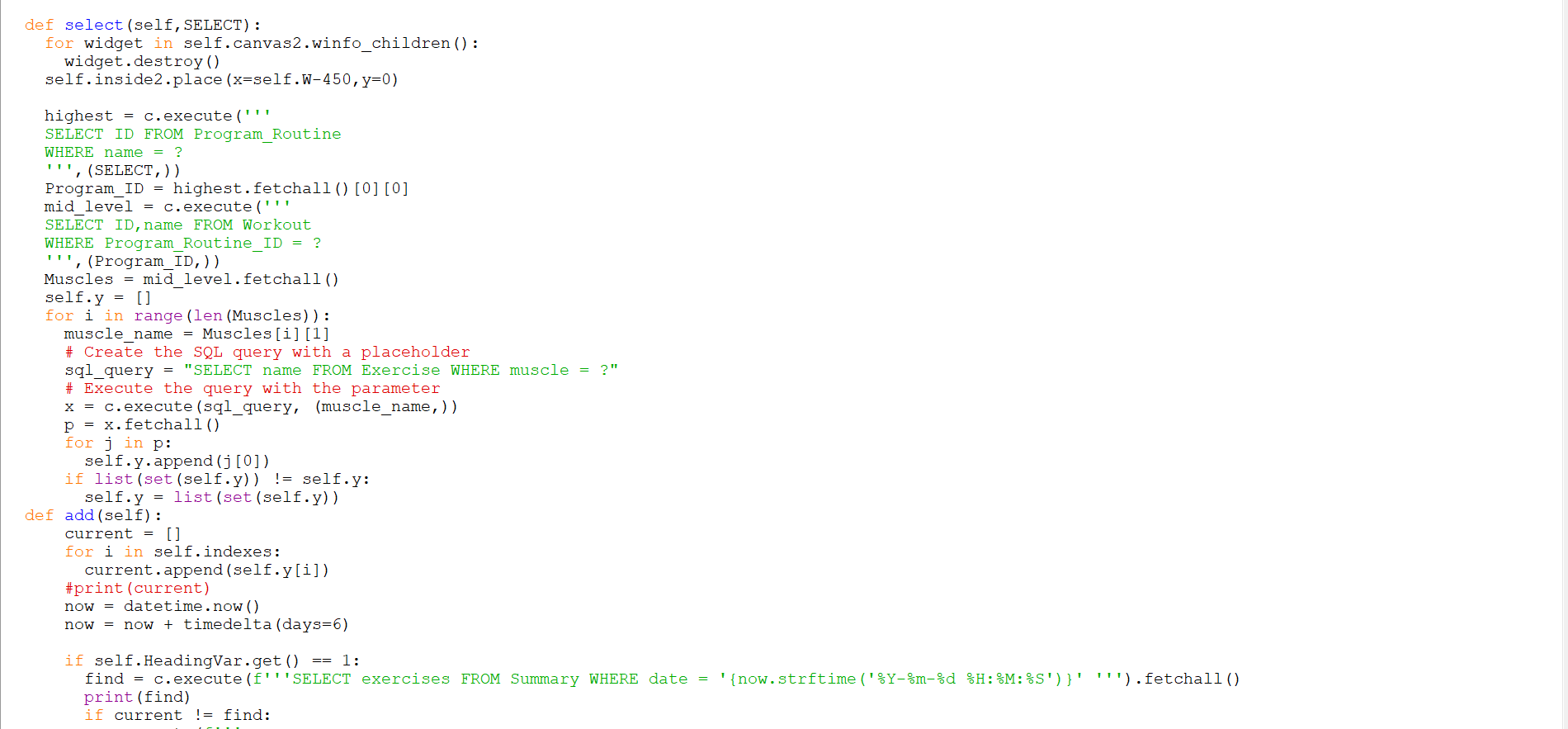
*The self.on\_radio\_button\_change function accepts all arguments. The value of the selected\_value variable is gotten by finding the current value of the radiobutton and entering it as a parameter into specific functions of the Specific\_exercise\_class*

*The last 5 lines initialise the treeview class which is in a separate file and*

*explained later.*



*This code initialises all the attributes of the Specific exercise class*



*The first task this method completes is it clears all the widget inside the*

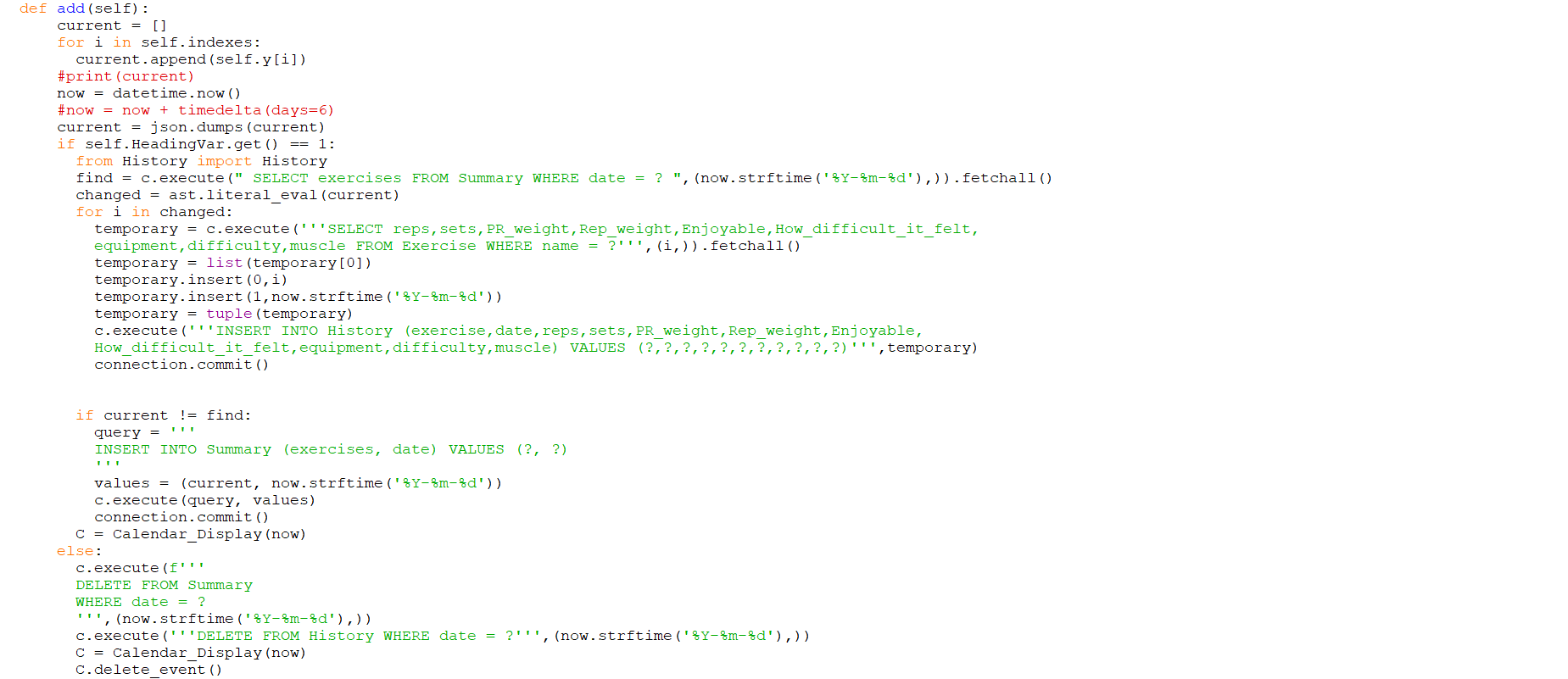
*“canvas2” window which is contained in the first tab*

*It then searches for the ID of the workout day in the database and saves it to the variable “highest” which represents the highest level in the hierarchy of the database*

*Then it searches for the ID and name of all the workouts in that workout day and saves it to the variable “Muscles”*

*Then every exercise within each of those workouts is iterated through and put in the self.y array*

*Then the self.y array is checked to make sure there are no repeats by using the set () method of python*



*Then the add method is started: Firstly, it appends all the exercises with indexes in self. Indexes to current.*

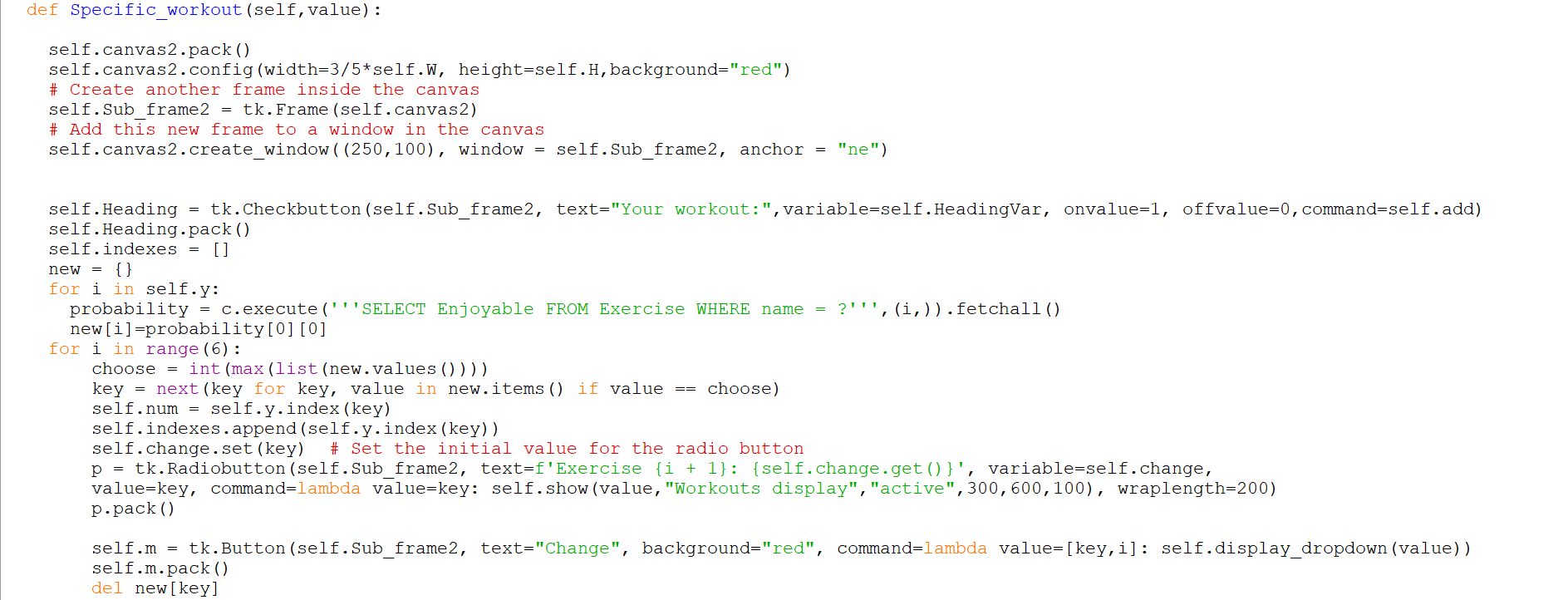
*Then it sets a certain time through “datetime.now()” [and the commented part afterwards forwards it by 6 days – that's there because it was a straightforward way, I found to obtain other dates while maintaining the same datetime format].*

*Then the current array is formatted so that it can correctly be passed as a parameter to the SQL query because SQL doesn't except arrays as records so it must to change to JSON form*

*The next part checks whether the “Your Workout” checkbutton has been pressed.*

*If it has then it appends the exercises to the database and calendar if they aren’t already there and then showing the calendar.*

*If the “Your Workout” button is deselected, then it deletes those exercises from the database and the calendar. Then shows the calendar*



*It configures all the canvas and frames and windows necessary for the*

*specific workout on the same tab*

*It creates a Checkbutton to be clicked when the workout is done*

*It randomly generates 6 exercises to do for that workout from the list of*

*exercises in self.y*

*These exercises are all displayed as radio buttons with corresponding images,*

*and all linked to the function self.show ()*

*There are buttons beneath each exercise that allow the exercise to be changed through the function display dropdown [discussed below]*



*The display dropdown shows all the possible exercises that can be replaced*

*with that exercise in a dropdown menu and allows the user to replace that*

*exercise with whatever one they want.*

*It then re-creates all the different radio buttons and buttons with the newly replaced exercise(s)*





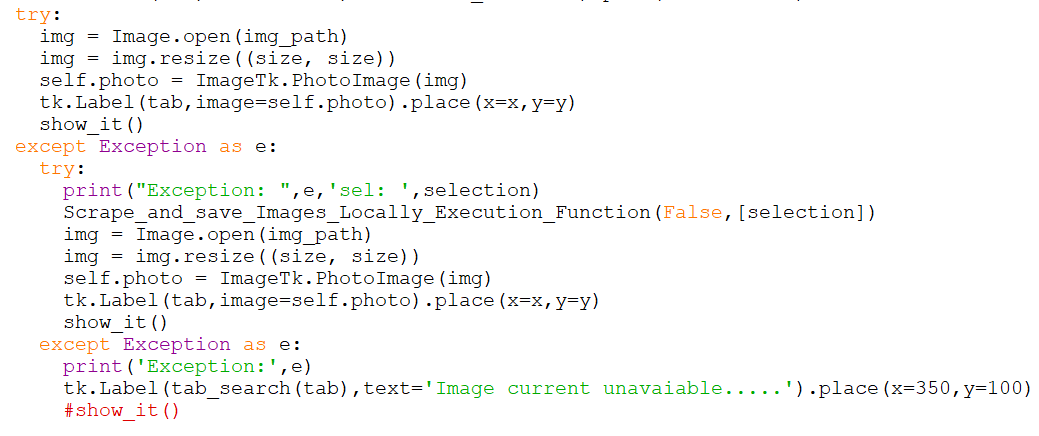
*The show function displays an image for a specific exercise and all its relevant information.*

*It takes 5 parameter that allow the exercise to be decided, the placement of the image and text related to the exercise to be decided, to allow the button to confirm exercise finished to be created or not and change the size of the image*

*For example, in the calendar summary the done button is blocked from being created*

*It uses the main () function to obtain the image path*

*If the done button IS created, then the attributes of treeview are inherited and the user is sent to the treeview page to update their progress*



This is the part that opens the image and renders it in the GUI.

## Caloric section



*Initialises everything*

*Instantiates “Base” method*

*Creates a label called self.lbl*

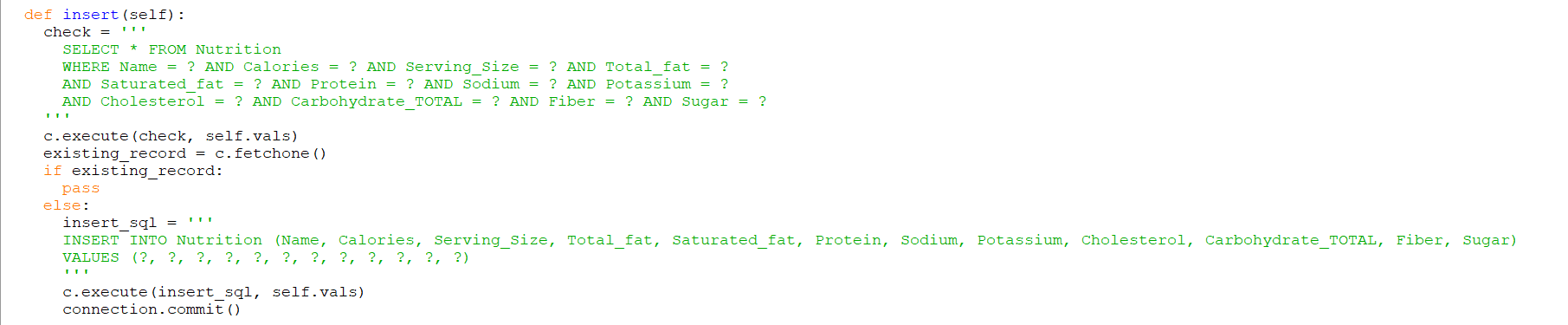
*Creates a method that clears the Nutrition table*

*Creates a method that displays all the information in the nutrition table*

*Creates a method that converts pounds to kg*

*Create a method that tells the calories in a certain food using an API and saves it to self.keys and self.vals*

*This one isn’t saved to a database yet so requires internet connection to work, which*  *may be a limitation that I will address in due course*



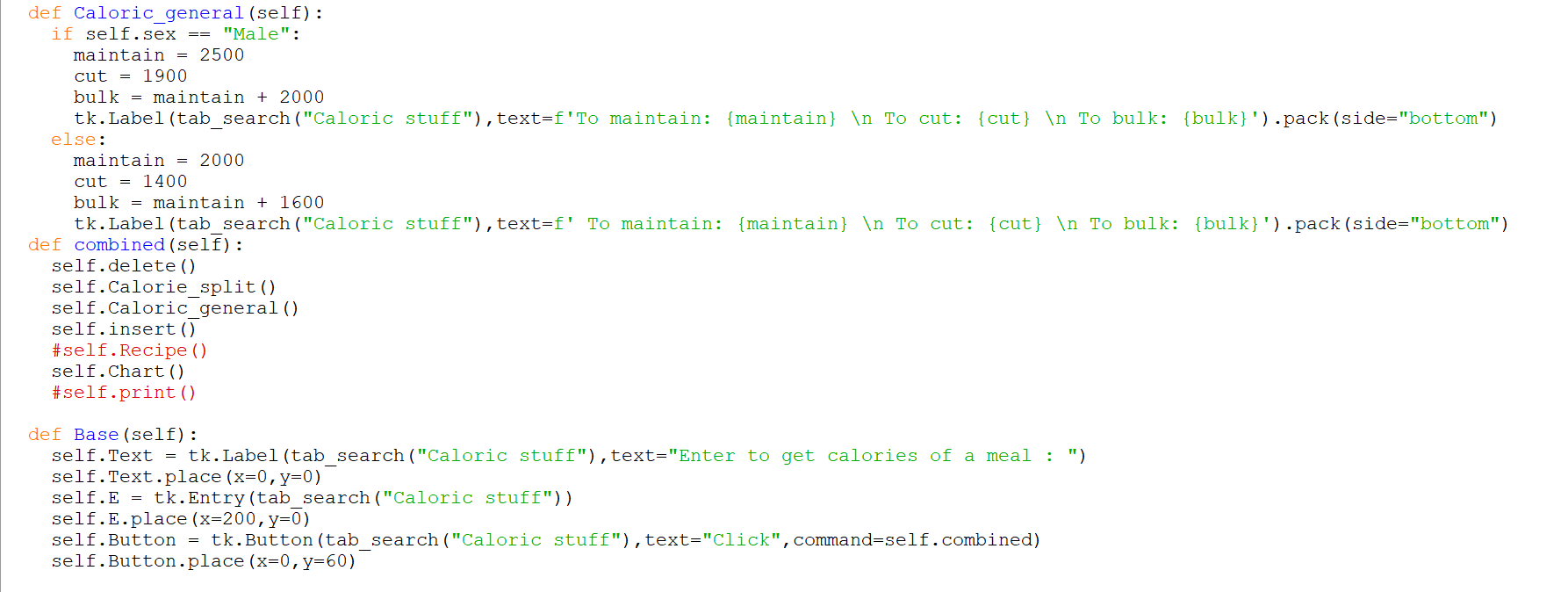
*Inserts nutrition split information into database if not already there.*

*Perhaps as a start towards using less internet connection this the data could be queried whether the data is already in the database before using the API.*

*The logic at the start of this method could perhaps be used in the “Calorie\_split” method*



*This creates a pie chart of the nutrition split*



The “Caloric\_general” method calculates the number of calories required for a bulk, maintaining of weight or cut

The “Combined” method sequentially instantiates all the necessary methods of the class

The “Base” method places all the widgets on the GUI window and binds the self. Button to the “combined” method

## Treeview class



*Initialises all necessary values*

*Saves column names of Exercise table into an array*



*The “**Update\_ID” very simply just re-writes all the ID values in the Treeview sequentially*

*The use of it becomes clearer later in the class*

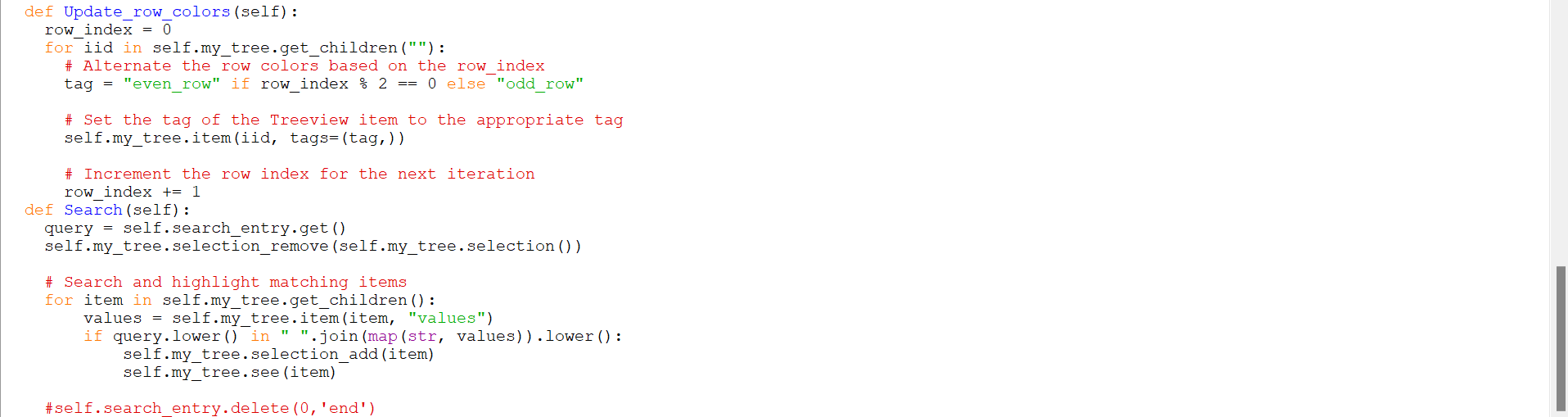




*This creates the Treeview*

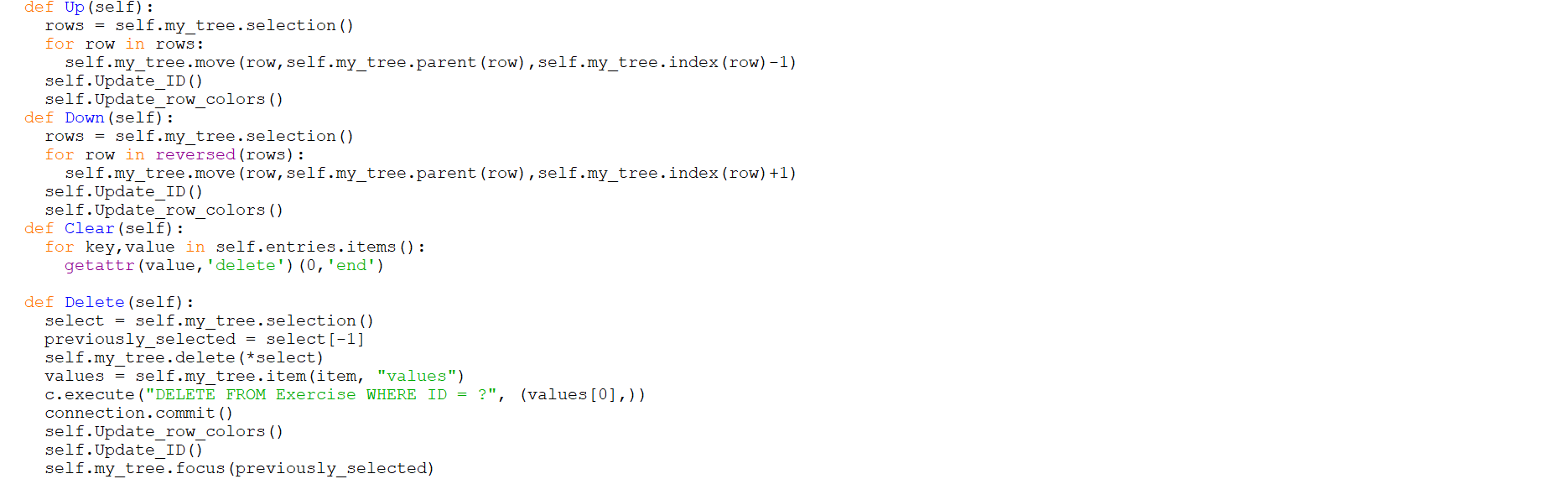
*It loops through the column names and assigns them as headers*

*It loops through the exercises and each exercises different characters are saved to a repeatedly emptying tuple list called “v”*



*The “Update\_row\_colours” loops through all the Treeview’s “children” which just means all the records and assigns them alternating tags again to keep the striped colour scheme consistent*

*The “Search” method searches for a specific exercise record. It lowercases the query and finds all the exercises that contain the query so if the exercise name is incomplete when searched for the exercise is still able to be found*



This adds functionality to the treeview

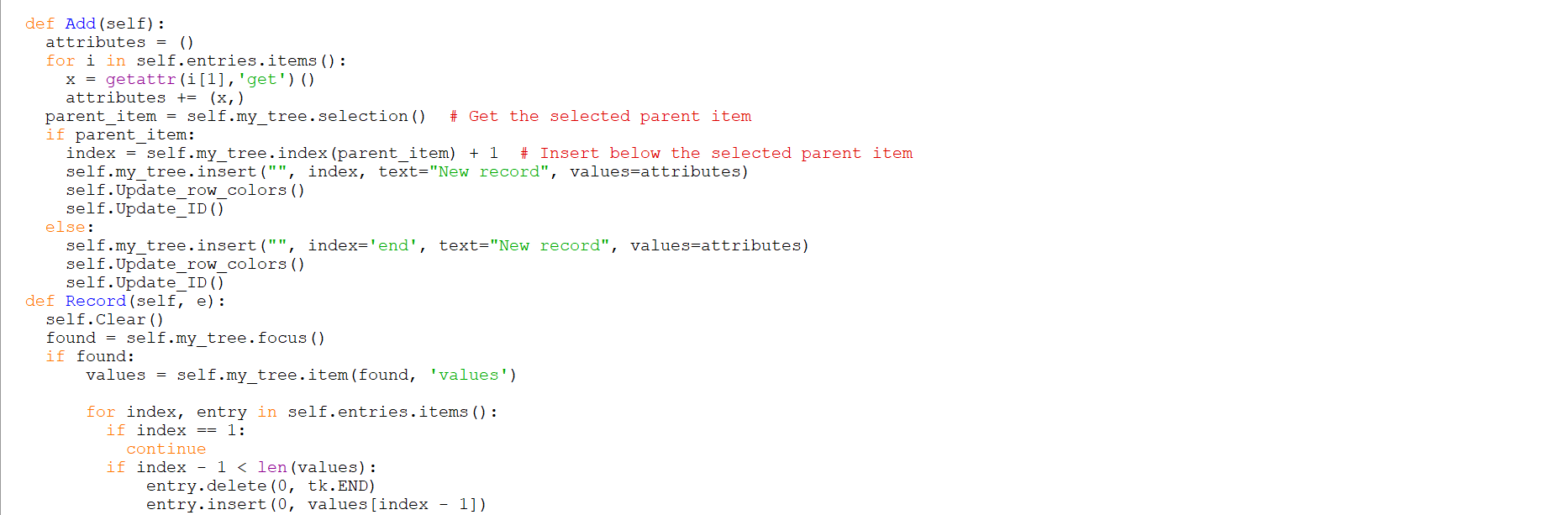
The user can move records up or down

The user can clear the entry boxes

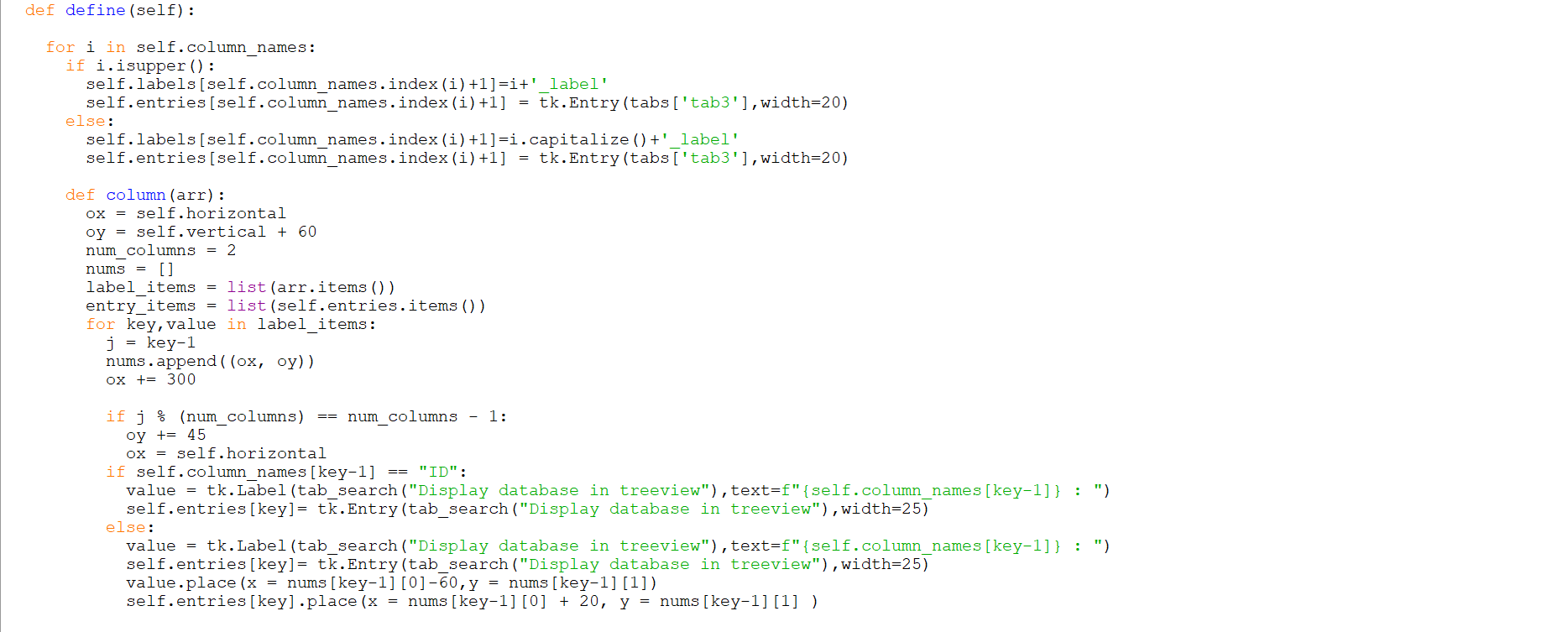
The user can delete records



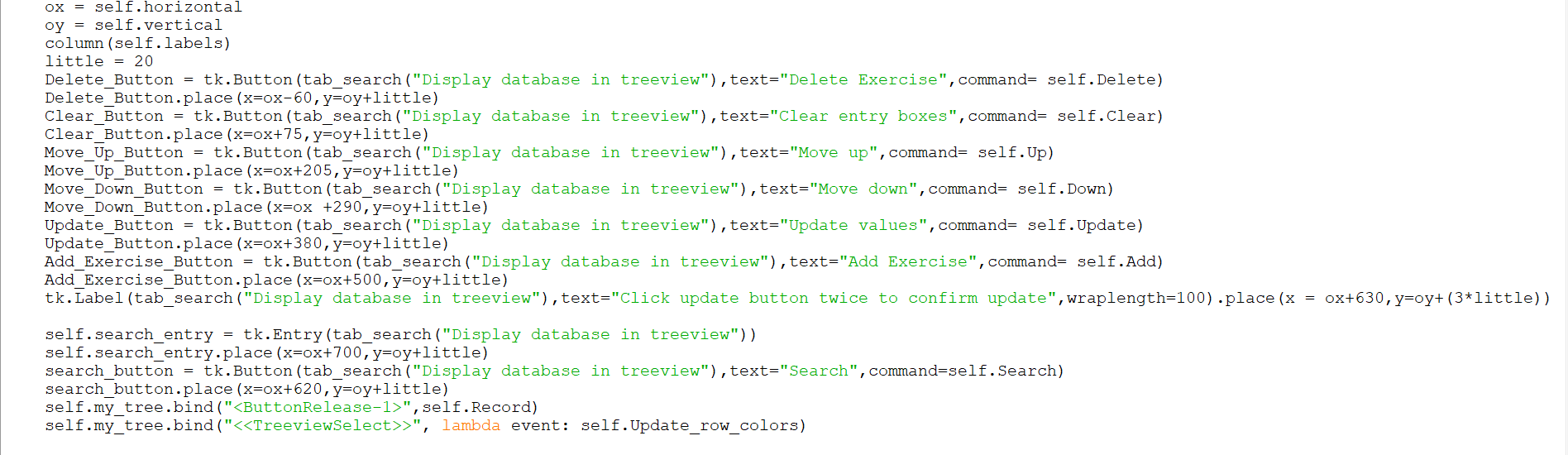
*This one updates the database and the Treeview when the user changes a certain characteristic about a record*



*The “Add” method allows a user to add an exercise. If they don’t select where they want the exercise to be added it simply adds at the bottom of the Treeview. If the user selects a certain record, it inserts the new record underneath that one*



*This dynamically places the entry boxes under the Treeview to allow the user to edit records*



*This places all the functionality widgets directly beneath the Treeview and binds the selecting of a record to self. Record method.*

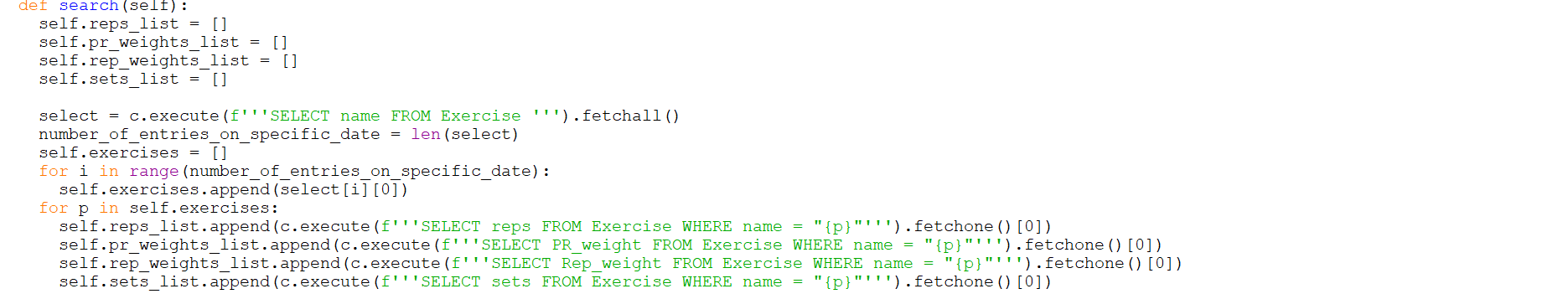
## Graphing



*The constructor initialises everything*

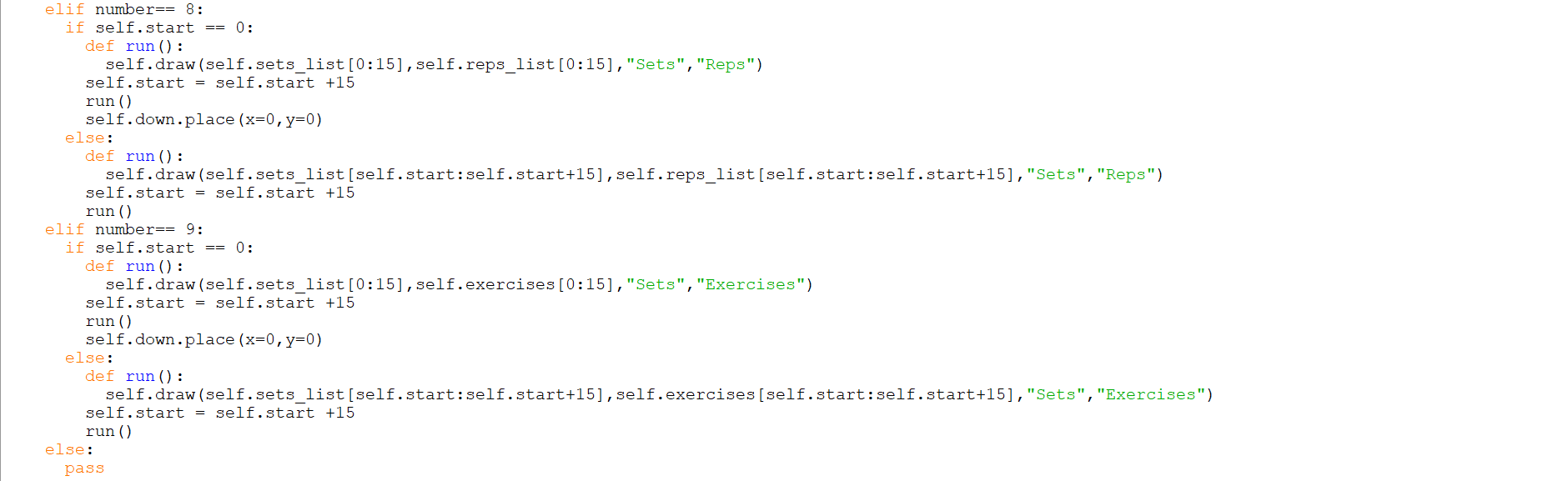
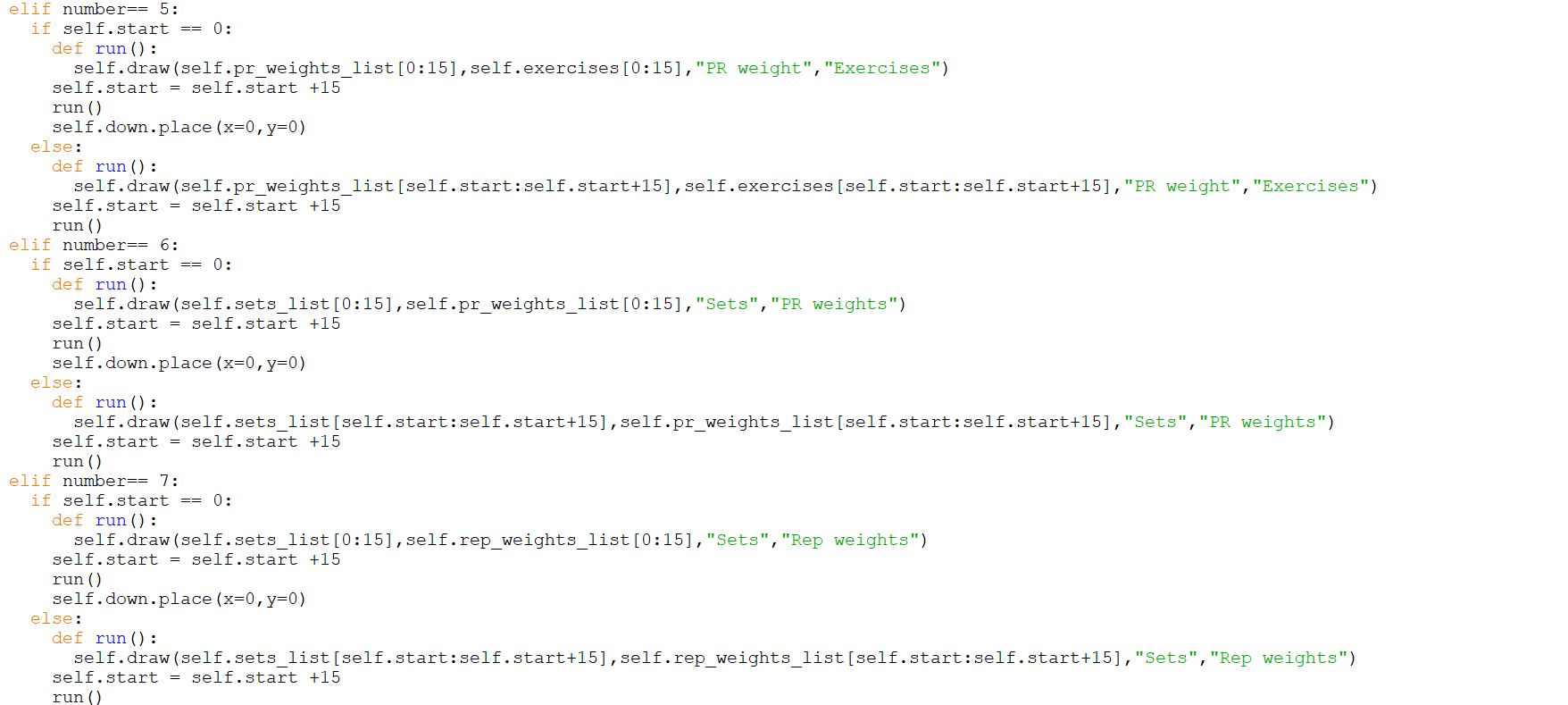
*The draw method draws a horizontal bar chart*

*It takes 4 inputs, the values its plotting, the categories it’s plotting, the name of the x axis and the name of the y axis. [In this case the y axis acts as the conventional x axis as is known by most due to it being a horizontal bar graph.]*

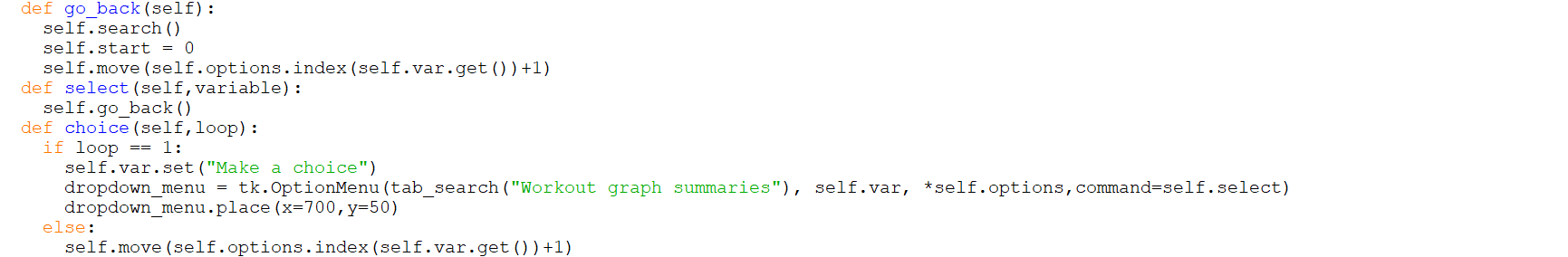


*This method saves the reps, weight and sets of each exercise to a local list*





*This creates 9 viable options of graphs that can be created with a down button that when pressed moves down to show the following data, simulating the scrolling function on websites*

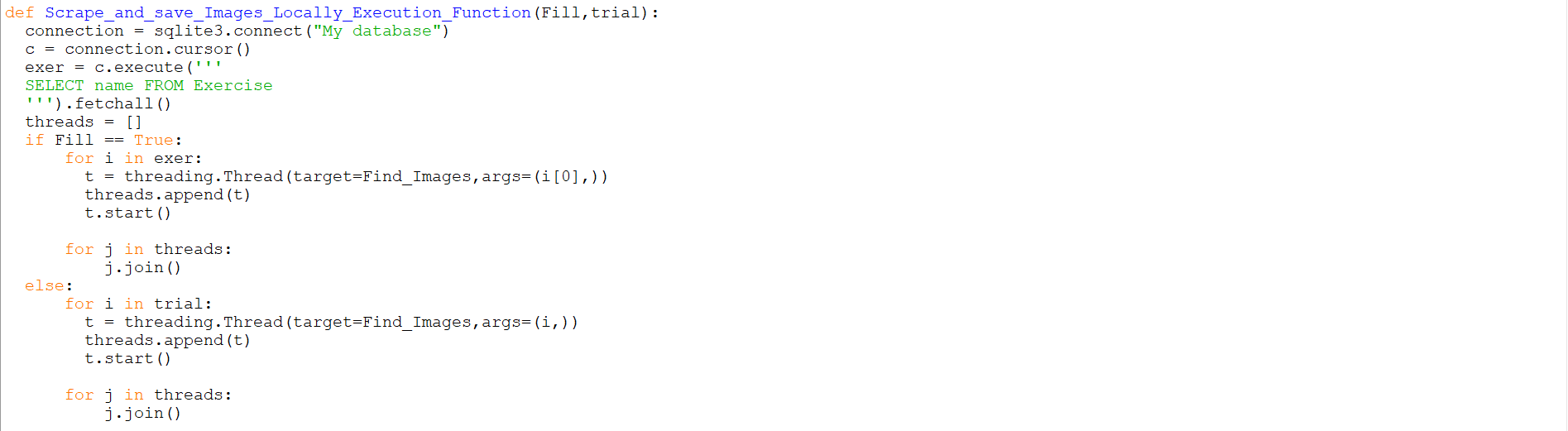


*The ”go\_back” method does the whole information searching again and that creates the graph based on the option the user picked (explained further below)*

*The choice method says that if the loop value is 1 then it creates a dropdown menu of the 9 possible graphs for the user to pick*

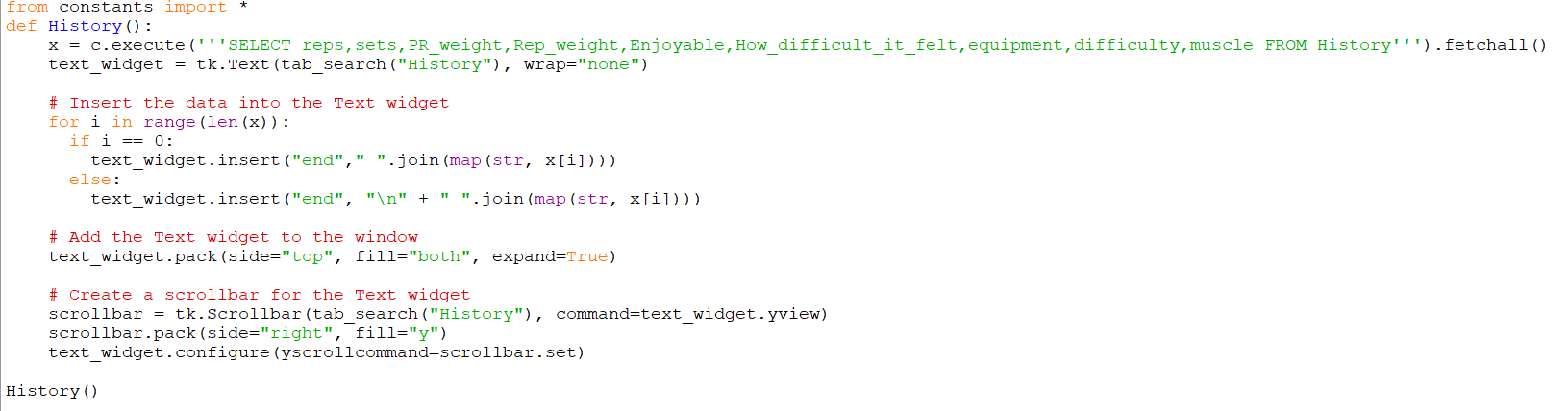
*If the choice isn’t 1 then it simply runs the move method again which is simply the effect of pressing the down button*

## Search for Images file

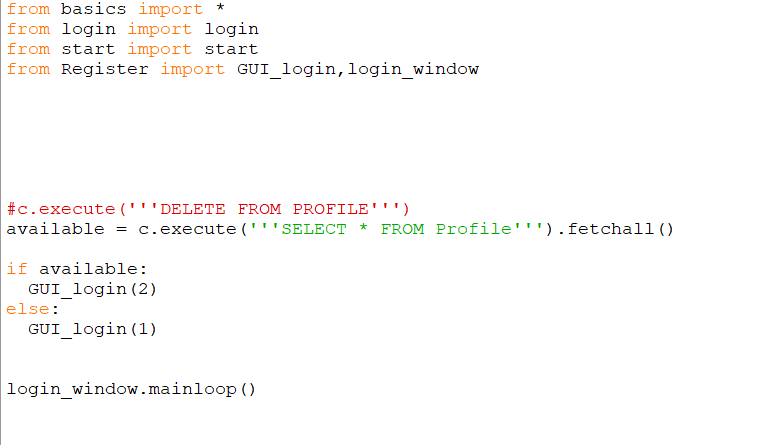


*This uses threading to scrape the internet for images by dynamically modifying the URL of the searched image and dynamically downloading and saving the image to a specified directory*

## History file



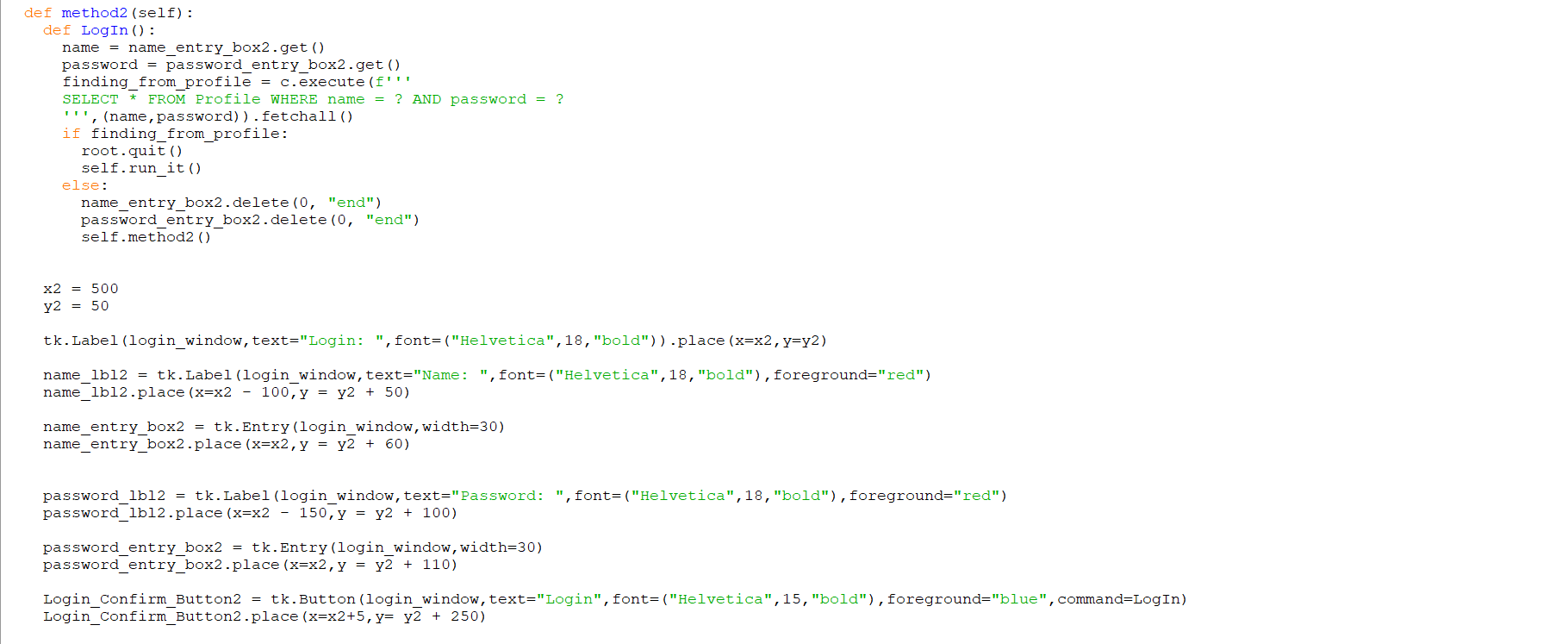
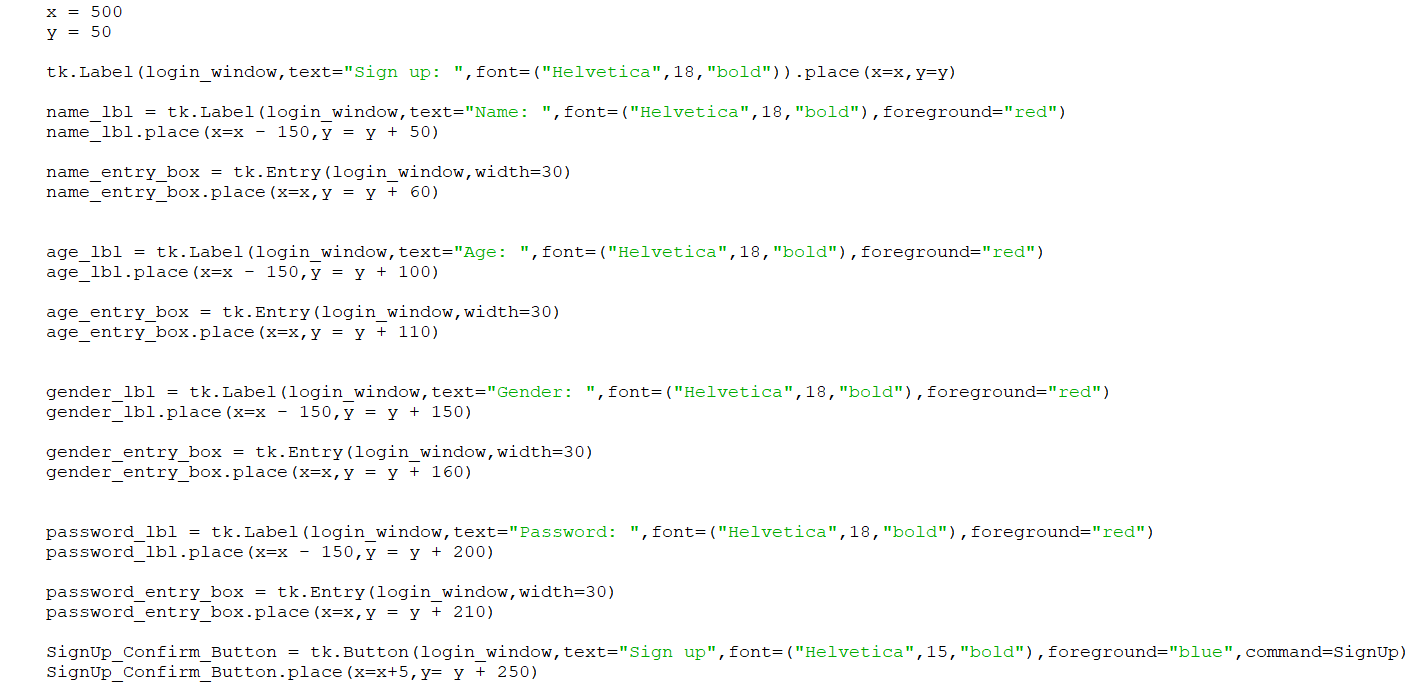
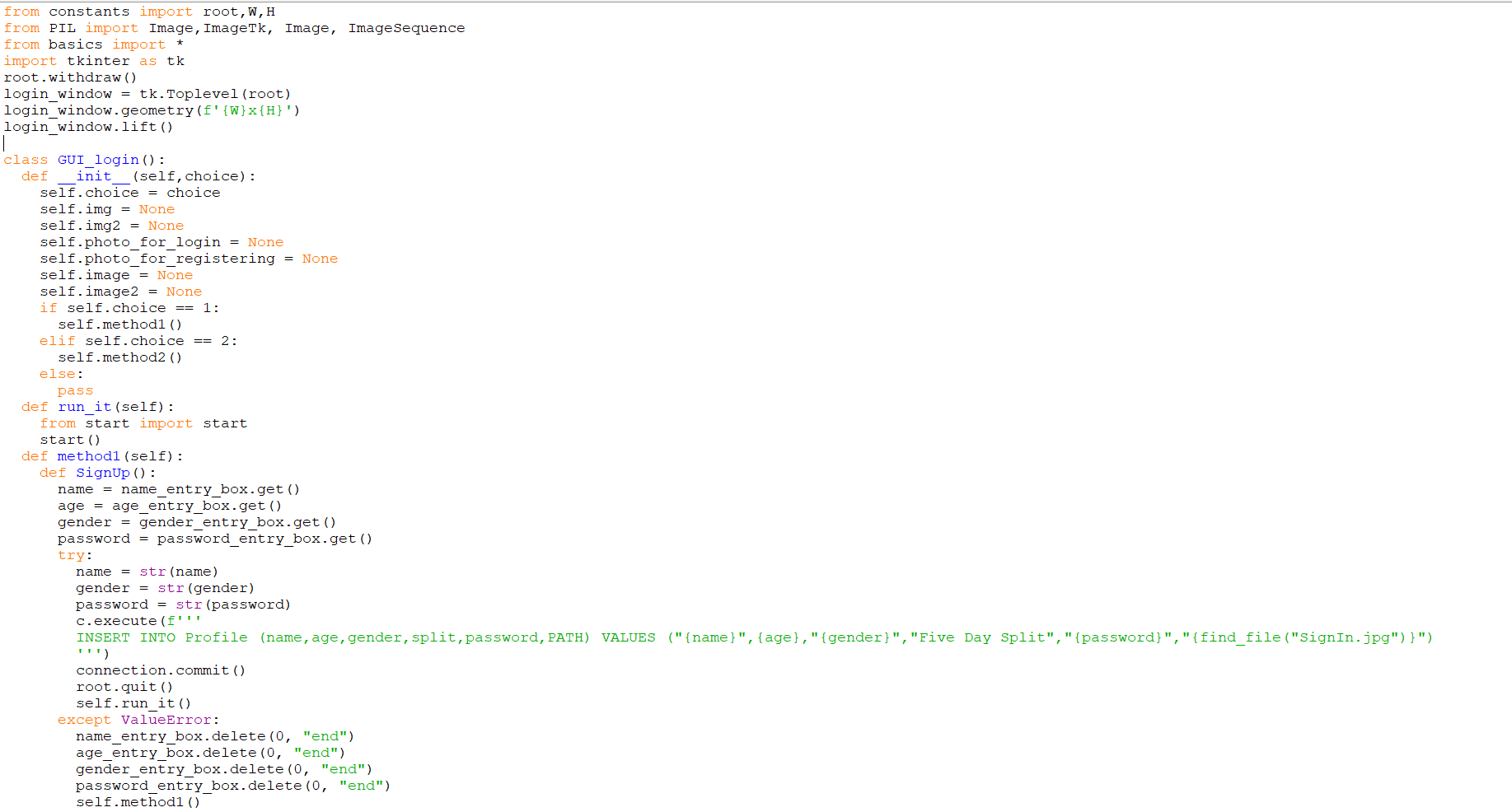
## Main file



*This file simply runs the application*

*Without this the application wouldn’t work*

## GUI Login file



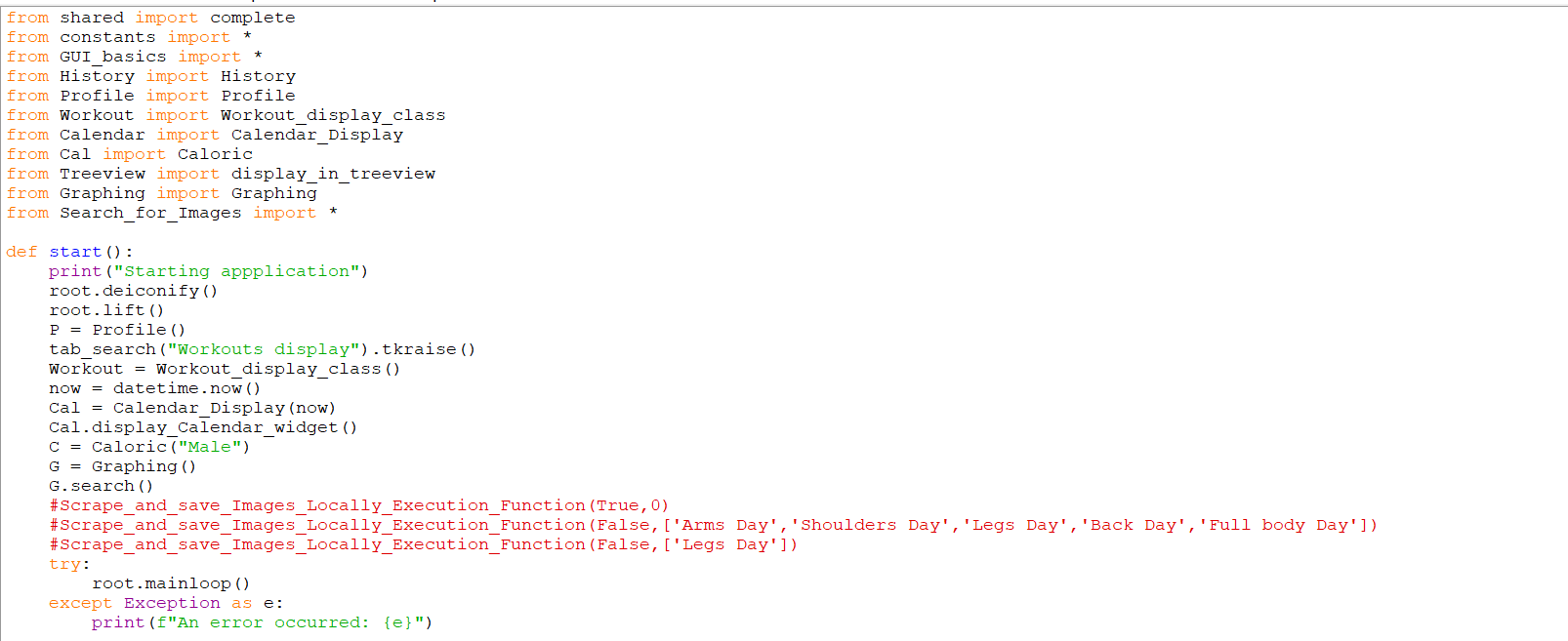
*This is simply the logic for the placement of the Signup and Login system within a separate window from the main application that opens first.*

*It also inserts the user’s credentials into a database when signing up and queries from the database to validate the user’s credentials when logging in.*

*The window is destroyed when the sign up or login is successful*

*It also hides the main application window, so it appears like only the login window / signup window has opened*

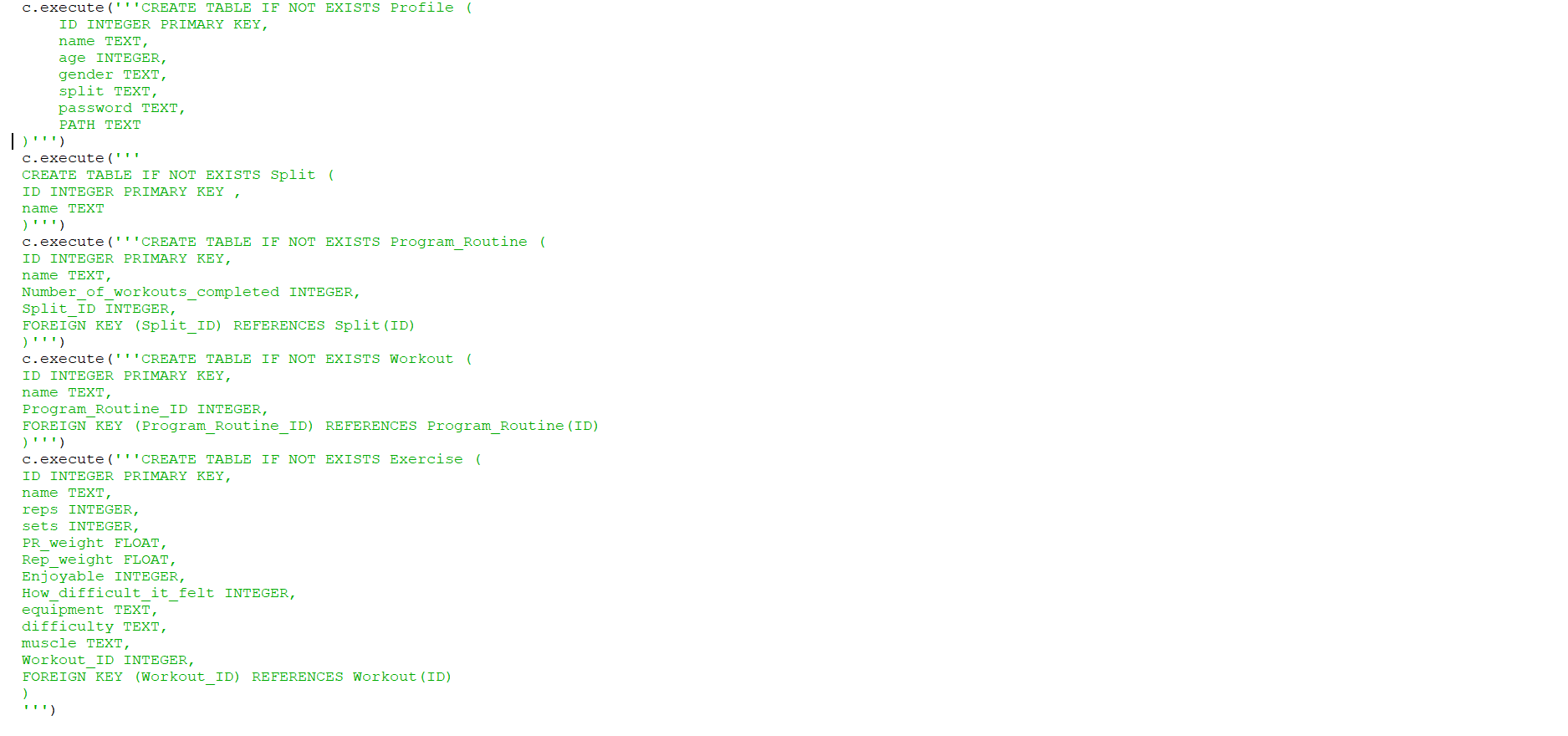
## Start file



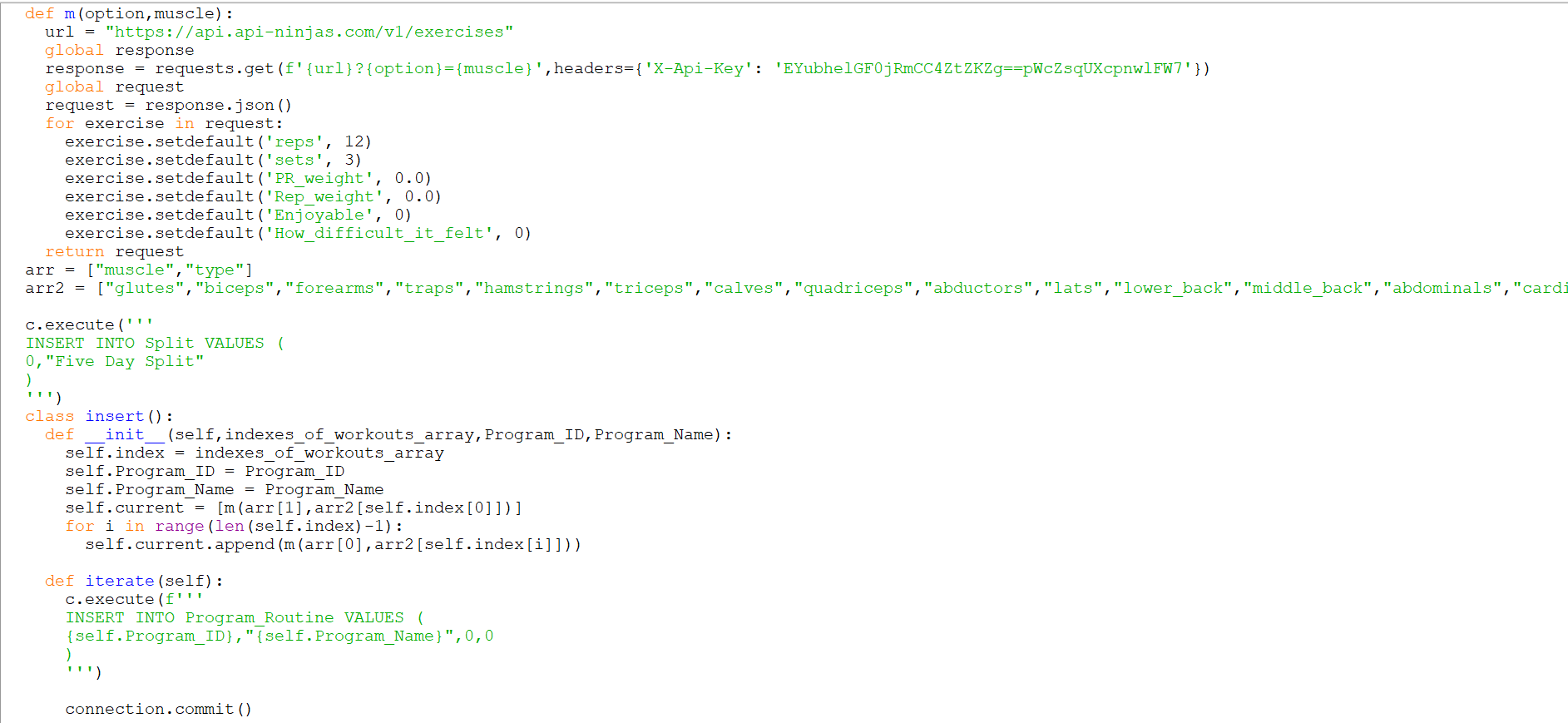
At this point, the login / signup window has been destroyed. The main application is then showed again through the “root.deiconify” command

This file simply initialises and imports all the other necessary files in the right order so that the program can run correctly

## Database Creation file



The two screenshots above represent the creation of a database called “My database” and the creation of the tables within this database and what values each record in each table in this database will hold.

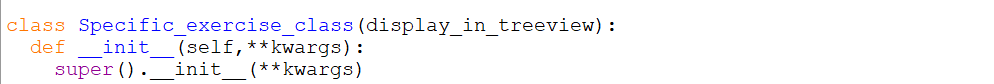


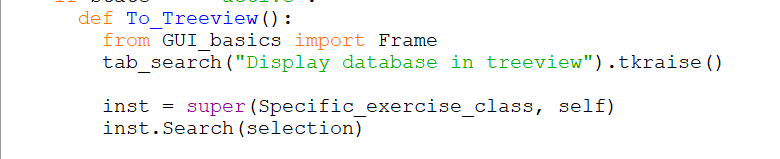
This dynamically populates the database with data from the Workout API sorting the data into the 4 tables specified above - [Split, Program\_Routine, Workout, Exercise]

## Implementation techniques:

### Inheritance

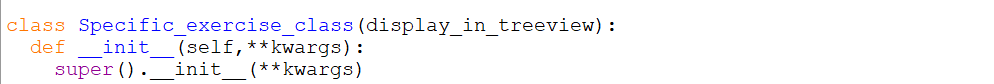




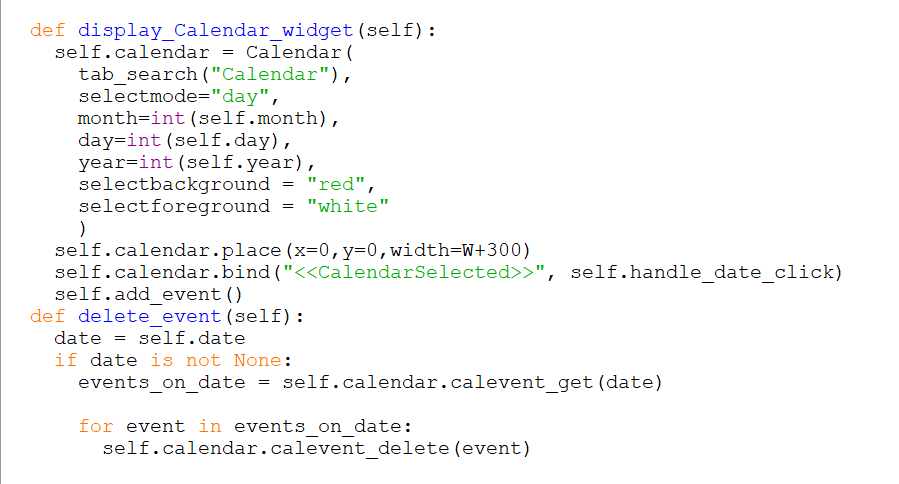
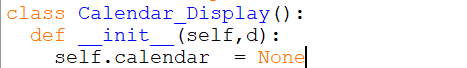


Very clearly the specific exercise class is inheriting from the treeview class. It inherits all the parameters and functions of the treeview class. An instance of that inherited class is created in the To\_Treeview method - {refer above for full code context}

### Polymorphism

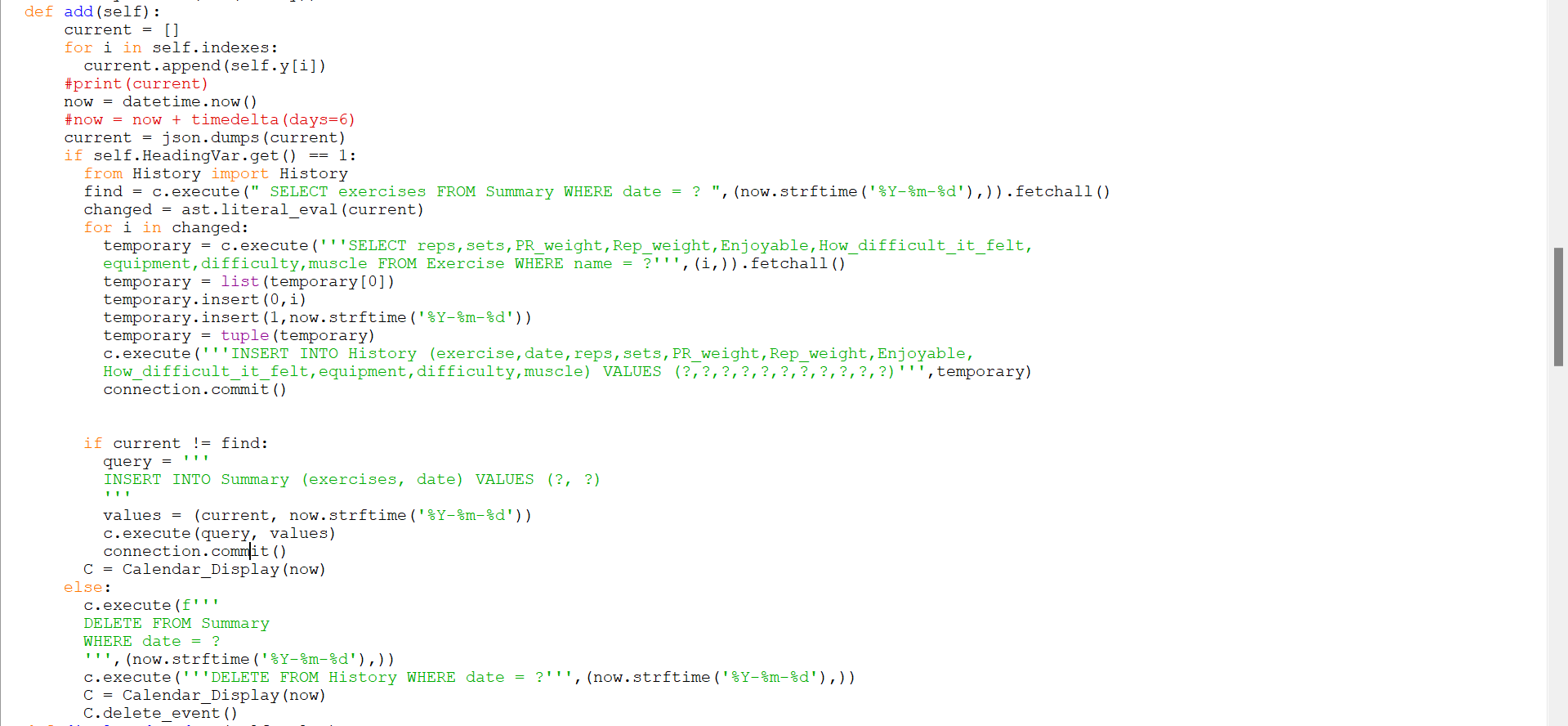
Since the specific exercise class accepts any number of arguments through the” \*\*kwargs” parameter this is an example of compile time polymorphism because there can be a different number of arguments to treeview each time an instance is created.

### Composition



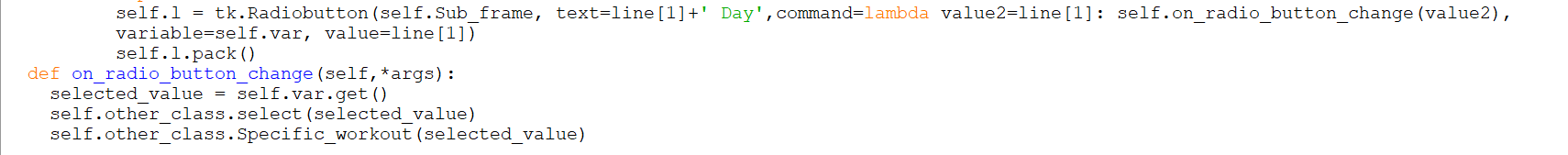






All these screenshots show composition as the workout class calls events of the calendar class within itself





This also shows composition as the workout display class specifies an instance of the specific exercise class and then calls two methods of that class when the radiobutton is pressed, giving it it’s dynamicity.

These are composition because the instances of classes called don’t make sense outside the class and if they were to be destroyed the other instances of the classes would be affected

# TESTING

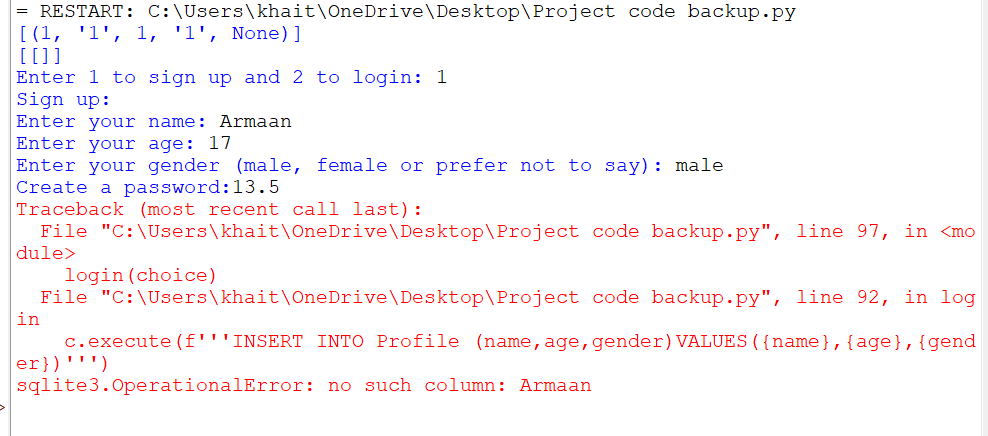
## Mistakes encountered while coding



*Since the Specific\_exercise class inherited from the Workout\_display class, self.other\_class stores the instantiation of the other class through the Workout\_display class.*

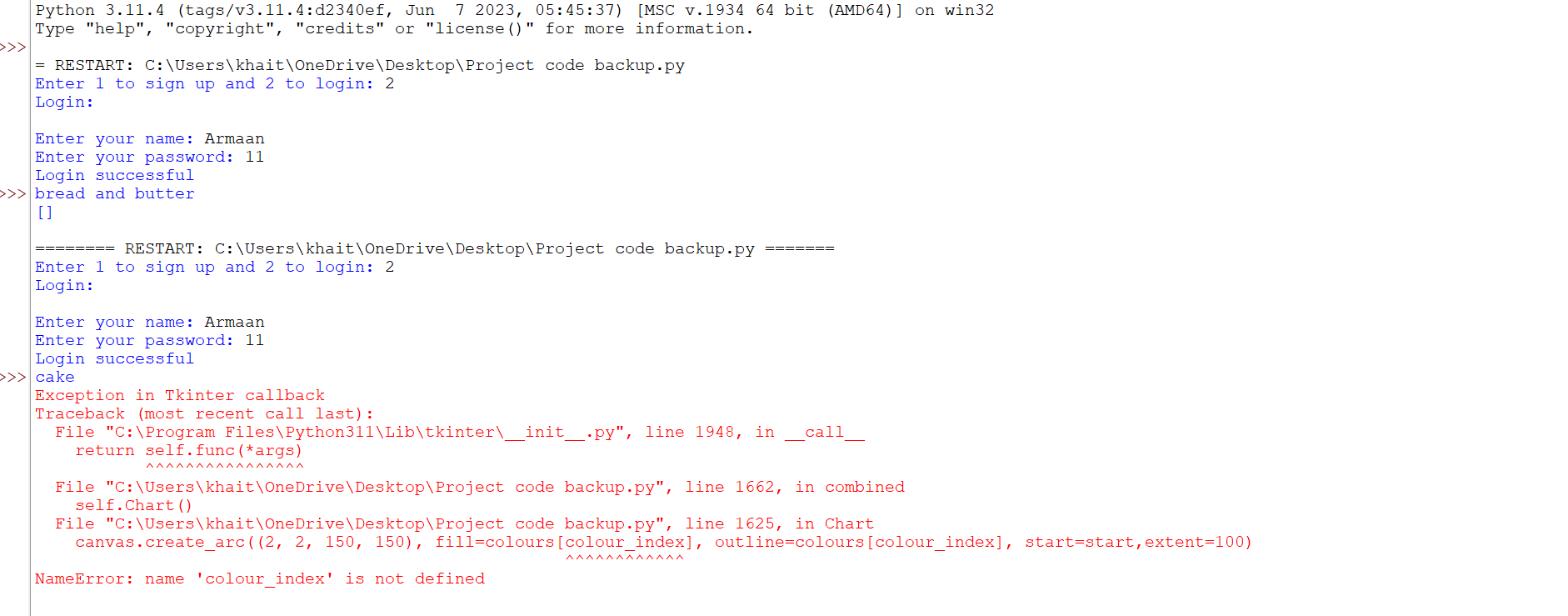
*If “on\_button\_click” doesn’t implement the calling of the other class properly then this error was thrown.*

*This error frequently came up as I modified my Workout file in several ways affecting the “on\_button\_click” method*

This is a formatting error.

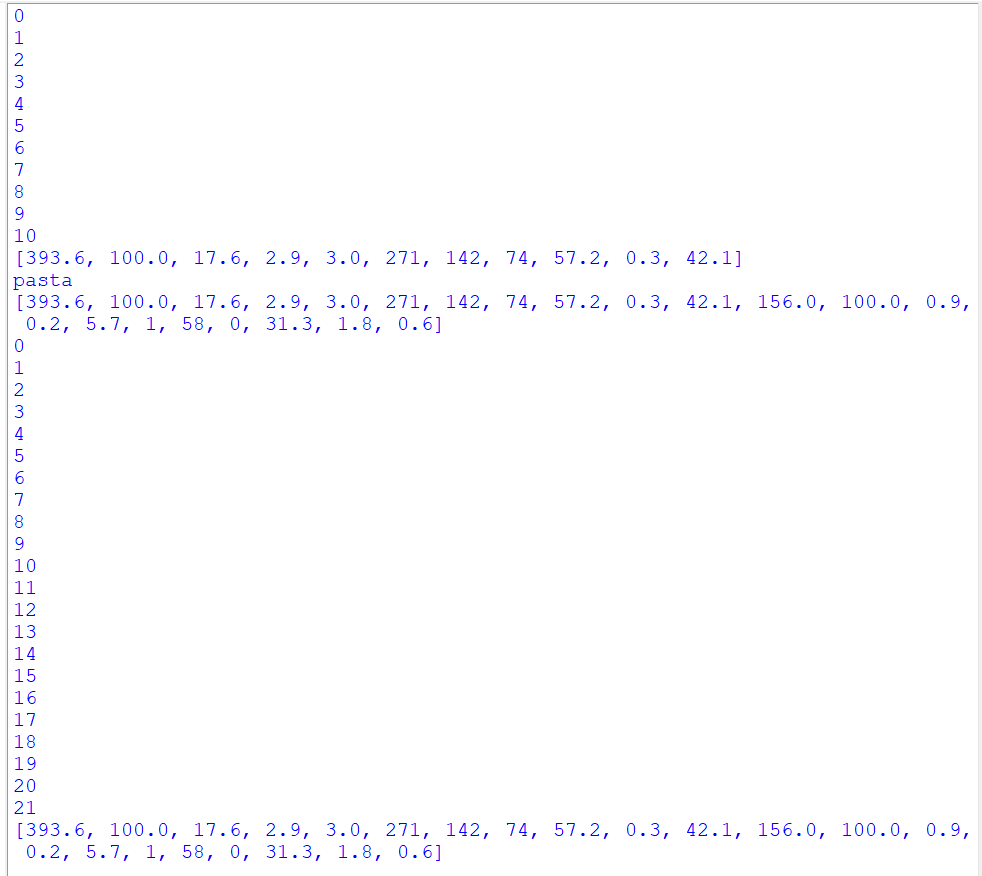
Since in this instance I didn’t use SQL parameters and was using string formatting instead I forgot to add quotes around the value

Therefore, the code assumed it was a column name

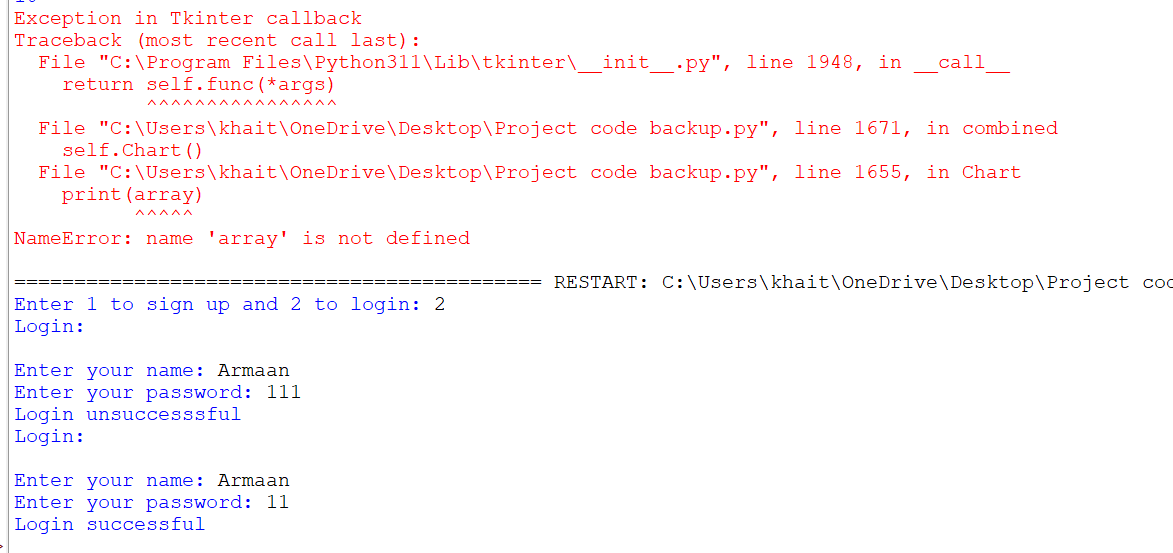


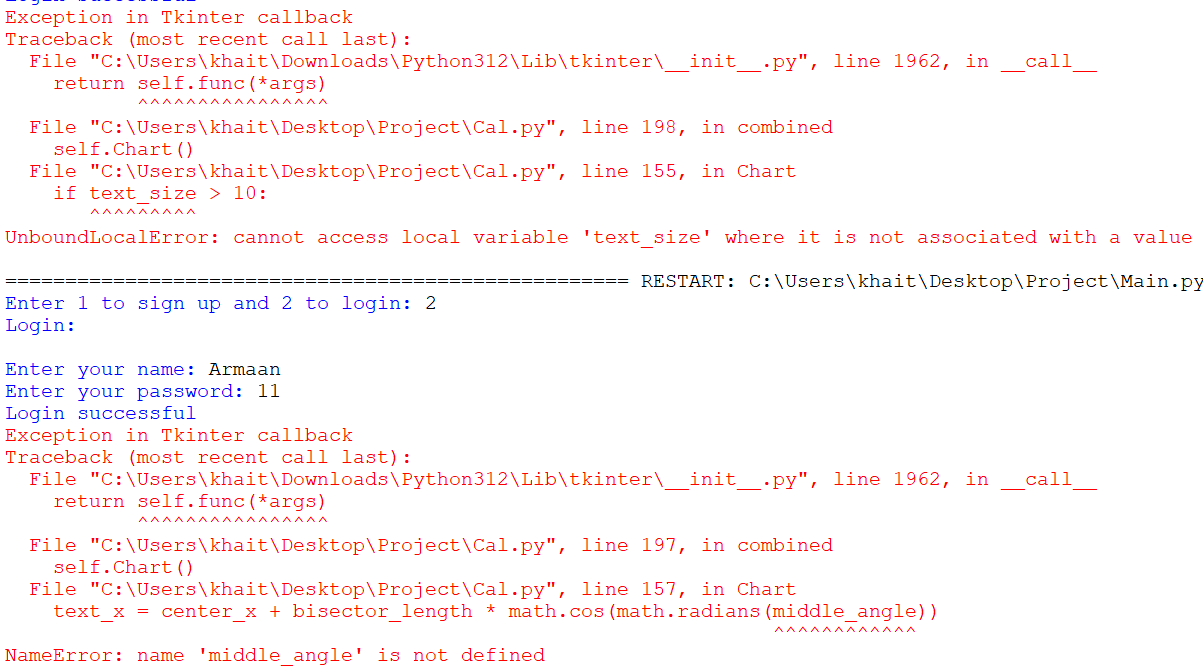
I tried to iteratively loop through the colours array in the constants file but forgot to initialise the colour\_index variable outside the for loop

The loop was dynamically iterating through something else which is why I needed to define a separate value to iterate through the colour\_index simultaneously



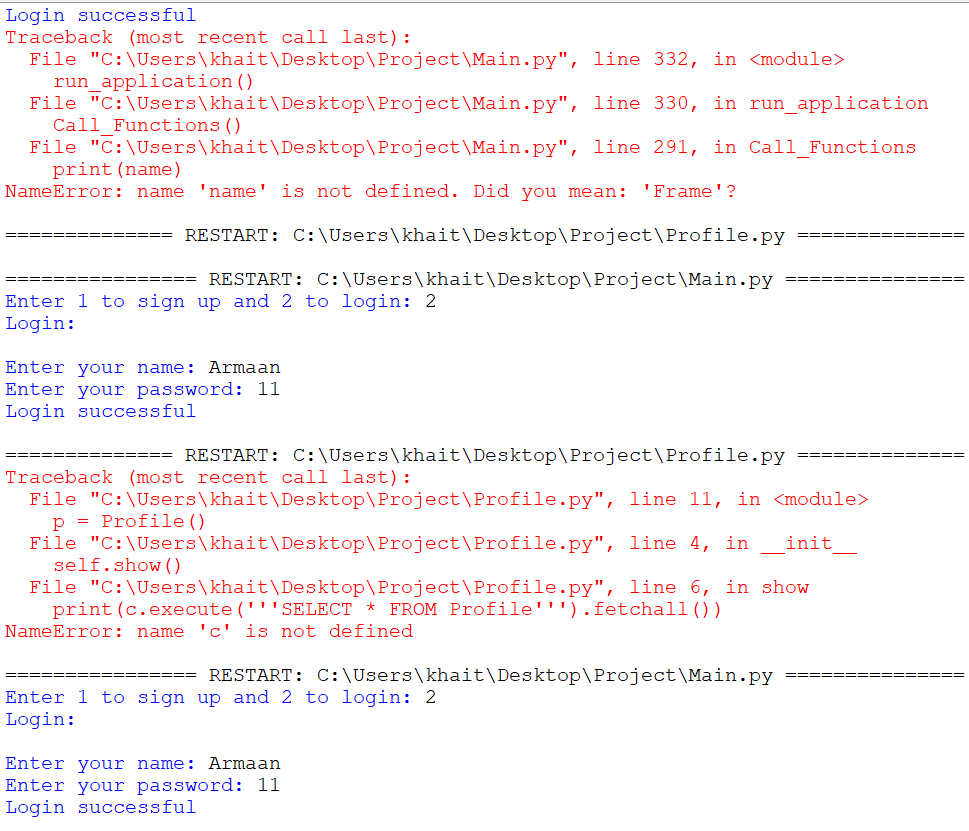
This was a series of print commands I used to check the database wasn’t saving the same data repeatedly in the table





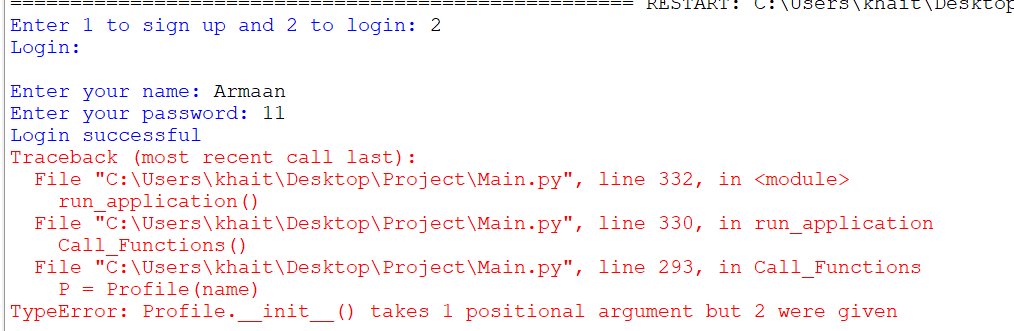
Moving my code around I erased the calculation for middle\_angle accidentally so this error was thrown.

It was an easy mistake to fix



This is due to my import statements. Since all my definitions for cursor connection to the database was kept in one file for faster running speeds, I forgot to import the file into the Profile file

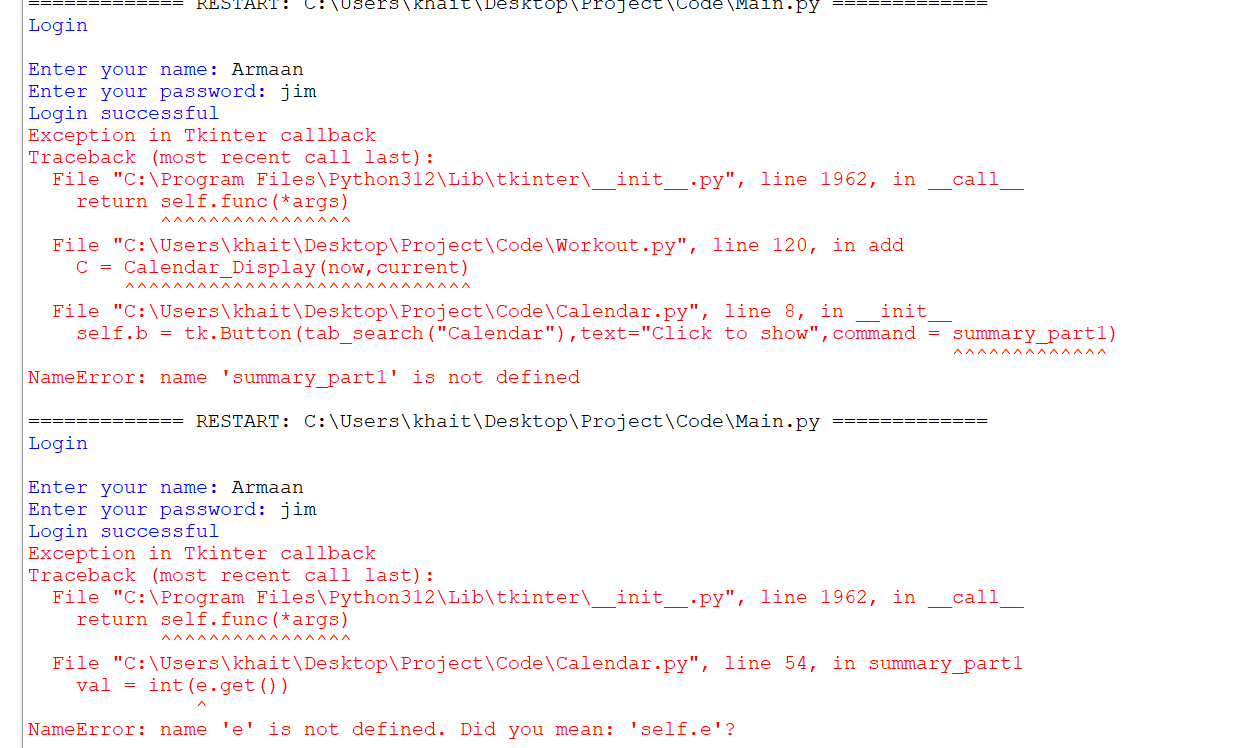
Again, a quite easy mistake to fix



This is due to the syntax of the constructor in classes.

Since my Profile part is made using a class, I forgot to add name as a parameter in the constructor so therefore my input of name while creating an instance of a class threw an error

Quite a simple mistake to fix



This was because I had changed my entry widget variable into an attribute of the Calendar class, so I forgot to replace “e” with “self.e”.

Remarkably simple mistake to fix

## Video link to testing of application

<https://www.youtube.com/watch?v=QxX9rjtEwSM>

[](https://www.youtube.com/watch?v=QxX9rjtEwSM)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part being tested | Description | Expected result | Pass/Fail | Data type |
| Signup |  |  |  |  |
| Details | Signup with correct details | Closes window and opens experience determination window | Pass | Valid data |
| Details | Signup with incorrect details | Empties entry boxes and doesn’t close window. Keeps repeating till user enters valid details. [Only the age must be a number because the other fields can be whatever the user feels like because they are strings, so won’t throws an error] | Pass | Invalid data |
| Login |  |  |  |  |
| Details | Login with correct details | Closes window and opens main application window | Pass | Valid data |
| Details | Login with incorrect details | Empties entry boxes and doesn’t close window. Keeps repeating till user enters valid details. | Pass | Invalid data |
| Workouts display + Specific exercise |  |  |  |  |
| Image rendering | Image is unavailable | If an image is unavailable or something goes wrong with it then the user is told to choose another image from their local storage | Pass | Invalid data |
| Image rendering | Image is available | Image is available and displays next to it when the button is clicked | Pass | Valid data |
| Calendar |  |  |  |  |
| Display calendar summary | No recorded data to display | If day is clicked and there is no data on it to display nothing will happen if the user clicks on it, no matter how many times they press. A notification will pop up telling them that they is no recorded workouts on this day | Pass | Invalid data |
| Display calendar summary | Data is available | A box will pop up asking the user which workout on that day they want to view | Pass | Valid data |
| Display calendar summary | Workout number outside range | If the user picks a workout number that is outside the range like if it’s 1-5 and the user picks 7 then a notification will pop up saying they should pick again | Pass | Invalid data |
| Display calendar summary | Workout number inside range | If the user picks a workouts number inside the range a box will pop up asking the user which exercise in that workout they want to view | Pass | Valid data |
| Display calendar summary | Exercise number outside range | If the user picks an exercise number that is outside the range like if it’s 1-6 and the user picks 9 then a notification will pop up saying they should pick again | Pass | Invalid data |
| Display calendar summary | Exercise number inside range | If the user picks a workouts number inside the range a the image and information of that exercise will appear on the bottom left of the calendar tab | Pass | Valid data |
| Graph summaries |  |  |  |  |
| Show graph | Pick a graph | The user doesn’t have the opportunity to enter invalid data here since it’s a drop down so no invalid data ever occurs. When the user picks the graph, they would like the graph renders in the middle of the tab | Pass | Valid data |
| Show graph | Down | When the down button is pressed then the graph displays the next 15 data points | Pass | Valid data |
| Show graph | Down | If there is no more data to move on to then the graph, simply reloads | Pass | Valid data |
| Show graph | Reset | If the user wants to go back to the first 15 data points, then the reset button does that | Pass | Valid data |
| Calorie |  |  |  |  |
| Calorie split and graph | Information from API | If data not already in database, then a notification is shown that says that internet connection is required, and the information is obtained from the API and split, and graph rendered, and information saved to the database | Pass | Valid data |
| Calorie split and graph | Information from database | If data already in database, then then split and graph rendered using information already in the database | Pass | Valid data |
| History |  |  |  |  |
| Showing history of workouts | Click button pressed | Shows the history of all workouts recorded | Pass | Valid data |
| Profile |  |  |  |  |
| Change profile picture | Change profile picture | When the button is pressed a pop-up screen shows the users filtered local storage allowing them to pick a file to change to their profile picture | Pass | Valid data |
| Can’t find profile picture | Can’t find profile picture | If the picture is moved or another computer is used the program will reopen the pop-up window and ask the user to re select their file at its new location | Pass | Invalid data |
| Display in treeview |  |  |  |  |
| Update | Update value | If you select a record and update the values in the entry boxes, then press update it updates these values in the database and treeview | Pass | Valid data |
| Search | Search for a record | If the record exists then the treeview will jump to that record and highlight it | Pass | Valid data |
| Search | Search for a record | If the record doesn’t exists it will do nothing | Pass | Invalid data |
| Clear | Clear boxes | Clears all the entry boxes | Pass | Valid data |
| Add | Add exercise | Adds a new exercise allowing the user full dynamic control over their workouts | Pass | Valid data |
| Delete | Delete exercise | Deletes selected exercise allowing the user full dynamic control over their workouts | Pass | Valid data |

# Evaluation

Legend:

Object achieved

Object not achieved

|  |  |
| --- | --- |
| Objective | Met/ Not met |
| A login page will load when the app is started. This will include a sign-in page and a registration page. |  |
| The registration page with collect name, age, gender, password while the sign in page will allow the user to enter their username and password to access the system |  |
| There will be a set of questions to decide the experience level based on the user’s answer to sthe questions |  |
| A page will load showing the user’s profile of experience, age, gender, name with some colour and a picture if they want, else a default will be provided. |  |
| The user interface will load the profile of the user within a few seconds / immediately of logging in or signing up. All other functionality of the application will also respond within a few seconds / immediately of clicking on any feature of the application. |  |
| There will be a separate page for 6 different pre-made workouts. |  |
| I could possibly build functionality of being able to change the number of pre-made workouts. |  |
| Each workout will have editable and removable options and a dropdown menu of all the workouts available. |  |
| There will also be a choice to add exercises if not available. |  |
| A dynamic workout saved database and workout history will be in place. The database will update instantly as soon as the user implements any of the update functionalities |  |
| There will be a separate section that appears when a certain workout is picked showing the exercises to be completed with the weight, reps, sets and equipment, muscle and difficulty level all there. When a certain exercise is pressed there will be an image and the relevant information will be rendered there |  |
| There will be a checkbox beside each one and a master checkbox for every workout that will increase the number of workouts completed when clicked. |  |
| There will be a button to view a graph showing the progress of workouts info: reps against set or reps against weight etc... |  |
| There will be a separate tab for calories. |  |
| There will be an entry box to either search up the meal in a databank of meals |  |
| Or manually enter the calories. |  |
| There will be a circle showing how many calories have been eaten and are left. |  |
| There will be a clickable button to show a pie chart of the split of calories into different food group categories with different colours representing each category. |  |
| There will be a history of nutrition showing the meals each day. |  |

* *Considering my limited timeframe, knowledge and resources I think the numer of objectives I have completed are reasonable (15/19 objectives)*
* *I chose to make my NEA a computer app because it was what I was most familiar with and already had knowledge at my disposal that I could work on. In retrospect a mobile app may have been more practical, but a computer app is still equally as useful for recording and tracking diet and workout.*