zCalendar - Terminal Application

Developed by Zakary Sutherland

Deciding on What I Wanted to Build

Going into this assignment, I had three different ideas:

A calendar app.

A text RPG.

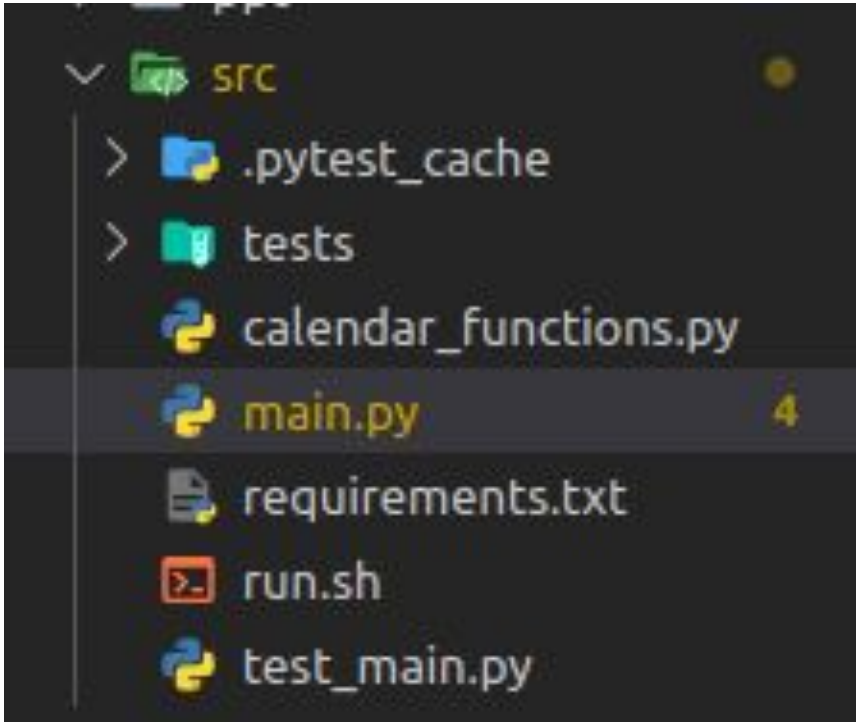
Or a Rust Raid Calculator.

After careful consideration, I decided on a calendar app, as I felt this made the most sense given the time to complete the project.

Knowing what I had to, I began planning the app.

Application Features

File Structure



The code for my application is stored in a src/ folder. The main files include main.py, and calendar_functions.py. Test cases that are used to check functionality using monkeypatch are stored in test_main.py, using files inside of the tests/ folder. The required libraries for use are contained in the requirements.txt file. The run.sh script is used to ensure that the venv is correctly initialised, and everything is installed correctly.

Main Menu

The main menu acts as the main interface for the application, providing the user with the opportunity to provide input based on what task they would like to complete.

The main menu contains 6 different options, allowing for input of the integers 1 - 6.

- Add activity
- Delete activity
- View activities in a certain month
- Receive an input analysing how busy a month is
- Get the current date
- End app

Add Activity to Date

One of the main features of the application is the ability to add ‘activities’ to specific days, like someone would on a normal calendar.

Repeat activities can be stored on multiple dates, and multiple activities can be stored on a single date.

This function asks the user to input the activity name, the month that the activity will take place, and the day that the activity will take place.

Delete Activity from Date

One of the main features of the application is the ability to remove ‘activities’ from specific days, much easier than a regular calendar.

Activities are deleted one at a time.

This function asks the user to input the activity name, and then removes the activity from the calendar.

View Activities from Month

Another feature of the application is the ability to view all of the activities that have been stored in a certain month.

The activities are presented in a list.

The program asks for a user to input a month, and then runs through the list to look for lowercase versions of the month that was input by the user.

Month Output

Another feature of the application is the ability to receive an output based on how many activities are in a certain month.

The output is displayed as a string, and is dependent on if the month has >= 0 activities, >= 2 activities, or >= 4 activities.

The program asks for a user to input the string, and then uses that to display the output. The outputs are:

4 = “[month] is a really busy month!

2 = “[month] is a relatively busy month!”

0 = “[month] is a really quiet month!”

Get Date

The last feature included in the app is the ability to print the current date.

The user simply types ‘6’ at the main menu, and will be presented with the current date.

Code Overview

Initial Setup

```
file = "calendar.csv"

try:
    ffile = open(file, "r")

    ffile.close()

except FileNotFoundError as file_not_found:
    print(f"{fg('red')}calendar.csv not found, initialising set up...")
    while True:
        try:
            year = int(input("Input the calendar year: "))
            break
        except ValueError as value_error:
            print("Incorrect value, please type a year.")
    print("Success!")
    input("Press enter to continue...")
    print("For a guide on how to use the application, please visit https://github.com/zakaryjs/T1A3-Terminal_Application")
    input(f"Press enter to continue...{attr('reset')}")
    ffile = open(file, "w")
    ffile.write(f"Year: {year}\n")
    ffile.write("Activity_Title, Month, Day\n")
    ffile.close()
```

The initial set up checks to see if the calendar.csv file already exists. If not, the setup process is initiated. This asks the user to input a year for the calendar, and then provides a series of instructions before writing the calendar year and standard format to the file.

Initial Setup in Action

```
calendar.csv not found, initialising set up...
Input the calendar year: 2023
Success!
Press enter to continue...
For a guide on how to use the application, please visit https://github.com/zakaryjs/T1A3-Terminal_Application
Press enter to continue...
```

calendar.csv is not found

User is asked to input date and check github repo for instructions.

Main Menu

```python
def main_menu():
    print("MAIN MENU:")
    print("1. Input 1 to add an activity to the calendar")
    print("2. Input 2 to remove an activity from the calendar")
    print("3. Input 3 to view activities stored in a certain month")
    print("4. Input 4 to see how busy a certain month is")
    print("5. Input 5 to get today's date")
    print("6. Input 6 to end the application")
    user_input = input("What would you like to do? ")
    return user_input

decision = ""

while decision != "6":
    decision = main_menu()

    if (decision == "1"):
        add_calendar(file)
    elif (decision == "2"):
        delete_calendar(file)
    elif (decision == "3"):
        view_calendar(file)
    elif (decision == "4"):
        measure_calendar(file)
    elif (decision == "5"):
        get_date()
    elif (decision == "6"):
        pass
    else:
        print("Invalid input. Please return an input between 1 and 5.")
        input("Press enter to continue...")
```

The first part of this section of code defines the main_menu() function, printing the text that the user will see after the initial set up.

A user input variable is then defined, which allows the user to input the number of their choice.

The decision variable is dictated by what is input for user_input, and when 6 is input, the program ends.

Main Menu in Action

User is presented with menu interface.

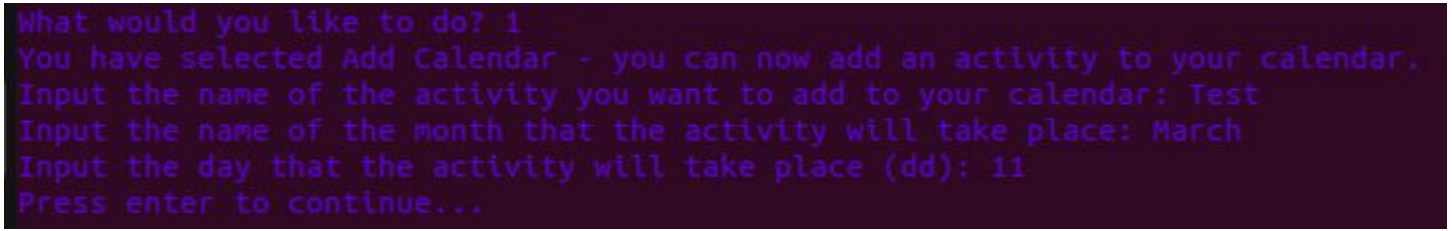
User is presented with opportunity to provide input based on what they would like to do.

Add Calendar

```
def add_calendar(calendar_file):
    month_list = ["january", "february", "march", "april", "may", "june", "july", "august", "september", "october", "november", "december"]
    activity_title = ""
    activity_day = ""
    activity_month = ""
    print("You have selected Add Calendar - you can now add an activity to your calendar.")
    while activity_title == "":
        activity_title = input("Input the name of the activity you want to add to your calendar: ")
    while activity_month == "":
        activity_month = input("Input the name of the month that the activity will take place: ")
        while activity_month.lower() not in month_list:
            activity_month = input("Input the name of the month that the activity will take place: ")
    while activity_day == "":
        try:
            activity_day = int(input("Input the day that the activity will take place (dd): "))
        except ValueError as add_cal_value_error:
            print("Incorrect value. Please type a numerical value.")
    with open(calendar_file, "a") as file:
        write_to_csv = csv.writer(file)
        write_to_csv.writerow([activity_title, activity_month, activity_day])
    input("Press enter to continue...")
```

This part of the code defines the add_calendar function. First, a list is created which contains all of the calendar months. All variables are set to blank, as to be used in the while loops to force users to provide a correct input. After the information has been input, the information is then written to the calendar.csv file.

Add Calendar in Action



User selects 1 - add calendar.

User inputs activity name - Test.

User inputs activity month - March.

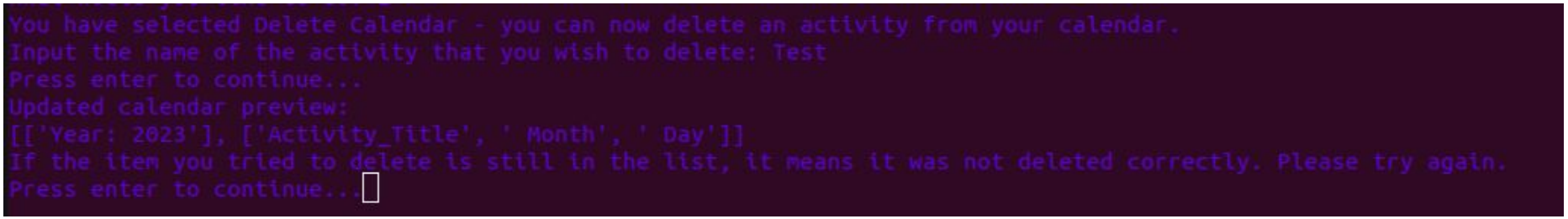
User inputs activity day - 11.

Delete Calendar

```
def delete_calendar(calendar_file):
    print("You have selected Delete Calendar - you can now delete an activity from your calendar.")
    delete_activity = input("Input the name of the activity that you wish to delete: ")
    calendar_list = []
    with open(calendar_file, "r") as file:
        read_csv = csv.reader(file)
        for row in read_csv:
            if(delete_activity != row[0]):
                calendar_list.append(row)
    input("Press enter to continue...")
    with open(calendar_file, "w") as file:
        write_to_csv = csv.writer(file)
        write_to_csv.writerows(calendar_list)
    print("Updated calendar preview:")
    print(calendar_list)
    print("If the item you tried to delete is still in the list, it means it was not deleted correctly. Please try again.")
    input("Press enter to continue...")
```

This part of the code defines the delete_calendar function. The user is first asked to input the name of the activity that they would like to delete. A list is then created in which all activities in the calendar.csv file are stored, before a for loop iterates through the different items, adding activities that don’t share a name with the user input back to the list. An updated list is then written to the calendar.csv file.

Delete Calendar in Action



User selects 2 - delete calendar.

User provides name of activity they would like to delete.

User is provided with list of all contents in CSV.

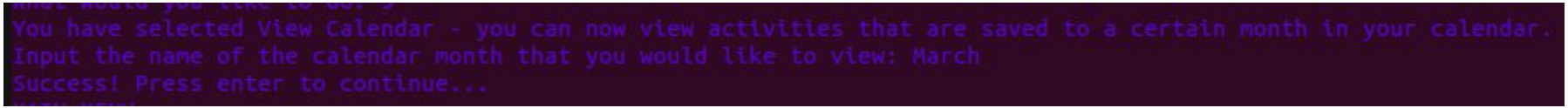
User is provided with information about why their item may still be in the list.

View Calendar

```
def view_calendar(calendar_file):
    month_list = ["january", "february", "march", "april", "may", "june", "july", "august", "september", "october", "november", "
    view_month = ""
    print("You have selected View Calendar - you can now view activities that are saved to a certain month in your calendar.")
    while view_month == "":
        view_month = input("Input the name of the calendar month that you would like to view: ")
        while view_month.lower() not in month_list:
            view_month = input("Input the name of the calendar month that you would like to view: ")
    view_calendar_list = []
    with open(calendar_file, "r") as file:
        read_csv = csv.reader(file)
        for row in read_csv:
            view_calendar_list.append(row)
    # print(view_calendar_list[2])
    for i in view_calendar_list[2:]:
        month = i[1]
        if month.lower() == view_month.lower():
            print(i)

    input("Success! Press enter to continue...")
```

This part of the code defines the view_calendar function. First, a list is created which contains all of the calendar months. The view month variable is set to blank, as to be used in the while loops to force users to provide a correct input. The bottom for loop iterates through all of the activities and if any match the user input, they are printed in a list.

View Calendar in Action



User selects 3 - view calendar.

User inputs the month that they would like to see the activities for.

If activities were in the month, they would be listed;

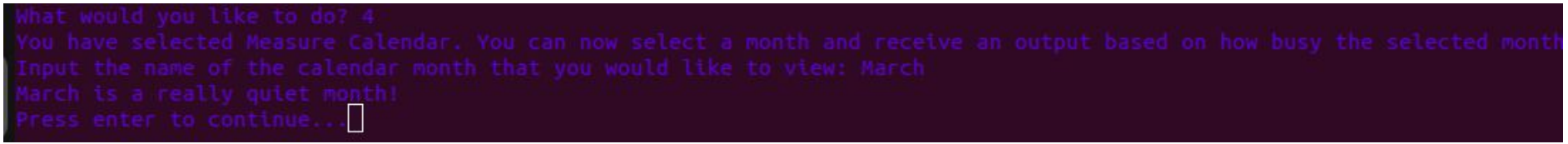
Since there are no activities in March, none are listed.

Measure Calendar

```
def measure_calendar(calendar_file):
    month_list = ["january", "february", "march", "april", "may", "june", "july", "august", "september", "october", "november", "december"]
    view_month = ""
    score = 0
    print("You have selected Measure Calendar. You can now select a month and receive an output based on how busy the selected month is.")
    while view_month == "":
        view_month = input("Input the name of the calendar month that you would like to view: ")
        while view_month.lower() not in month_list:
            view_month = input("Input the name of the calendar month that you would like to view: ")
    view_calendar_list = []
    with open(calendar_file, "r") as file:
        read_csv = csv.reader(file)
        for row in read_csv:
            view_calendar_list.append(row)
    for i in view_calendar_list[2:]:
        month = i[1]
        if month.lower() == view_month.lower():
            score += 1

    if score >= 4:
        print(f"{view_month} is a really busy month!")
        input("Press enter to continue...")

    elif score >= 2:
        print(f"{view_month} is a relatively busy month!")
        input("Press enter to continue...")

    elif score >= 0:
        print(f"{view_month} is a really quiet month!")
        input("Press enter to continue...")
```

This part of the code defines the measure calendar function. First, a list is created which contains all of the calendar months. The view month variable is set to blank, as to be used in the while loops to force users to provide a correct input. A score variable is defined, to be used as a count in the for loop. For every activity that mentions the user input, 1 is added to the score. If else statements are then used to determine which output will be printed, as to determine how busy the month is.

Measure Calendar in Action



User selects 4 - measure calendar.

User inputs the month that they would like to be provided information regarding how busy the month is.

User receives output based on how many activities are in the month.

Get Date

```
def get_date():
    print("You have selected Get Date. Today's date will now be printed.")
    todays_date = date.today()
    input("Press enter to continue...")
    print(f"Today's date is {todays_date}.")
    input("Press enter to continue...")
```

This part of the code defines the get_date function. This function uses the datetime module in order to receive the current day’s date, before being printed to the user.

Get Date in Action

User selects 5 - get date.

User receives an output of today’s date.

Running the Program

run.sh

```
#!/bin/bash

python3 -m venv calendar-venv
source calendar-venv/bin/activate
pip install -r requirements.txt
clear
python3 main.py
```

The run.sh script file is the easiest way to run the application.

Simply right click on the app, and select ‘run as program’.

Otherwise, open the src/ directory in the terminal and type `sh run.sh`.

Manual Install

If the script is not working for whatever reason, simply open the src/ directory inside a terminal window and type the following commands:

```
python3 -m venv calendar-venv
source calendar-venv/bin/activate
pip install -r requirements.txt
clear
python3 main.py
```

Looking Back

An Overview

I found the for loops to be the biggest challenge. When I first tried to teach myself Python, it is at loops that I became confused and gave up. With perseverance however, and thanks to this project, I am now much more confident in my abilities.

This project, like the last appeared to be really intimidating. However, once using Trello to break down the project it became much more manageable.

I really enjoyed the project - it was a fun challenge, and I already have many more ideas for awesome Python applications.