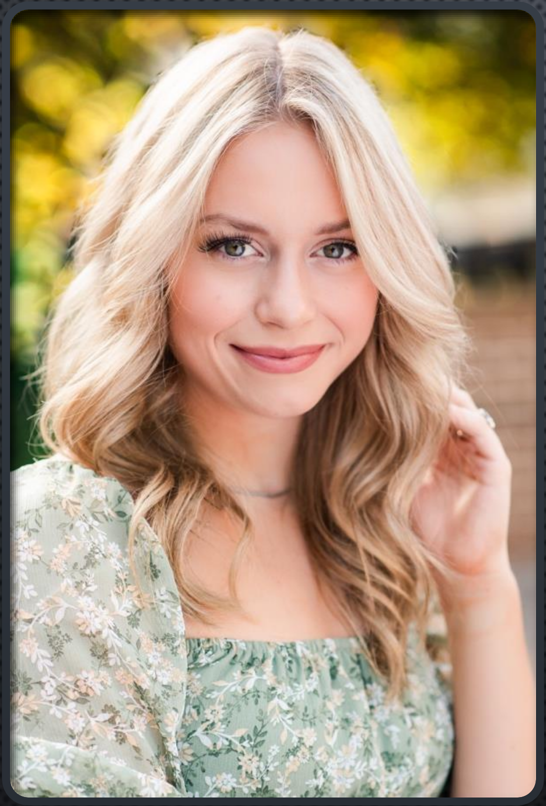
"THE DASHBOARD"

TEJJ

Group 44
Section 002

EMMA LANGFORD

- Scrum-master
- Pursuing bachelors in computer science
- Lead designer of garage door opener
- Co-designer of the physical prototype



Jewels Wolter

- Product Owner
- Lead Designer of the Trip Computer application
- Co-Designer of physical prototype
- Pursuing a bachelors degree in computer science at Auburn

Trey Wendell

- Developer
- Lead designer of GUI and Temperature Control application
- Co-designer of physical prototype
- Pursuing degree in Software Engineering

Jeffrey Carlisle

- Developer
- Lead Designer of Internal Clock App and Weather App
- Co-designer of physical prototype
- Actively pursuing a degree in engineering

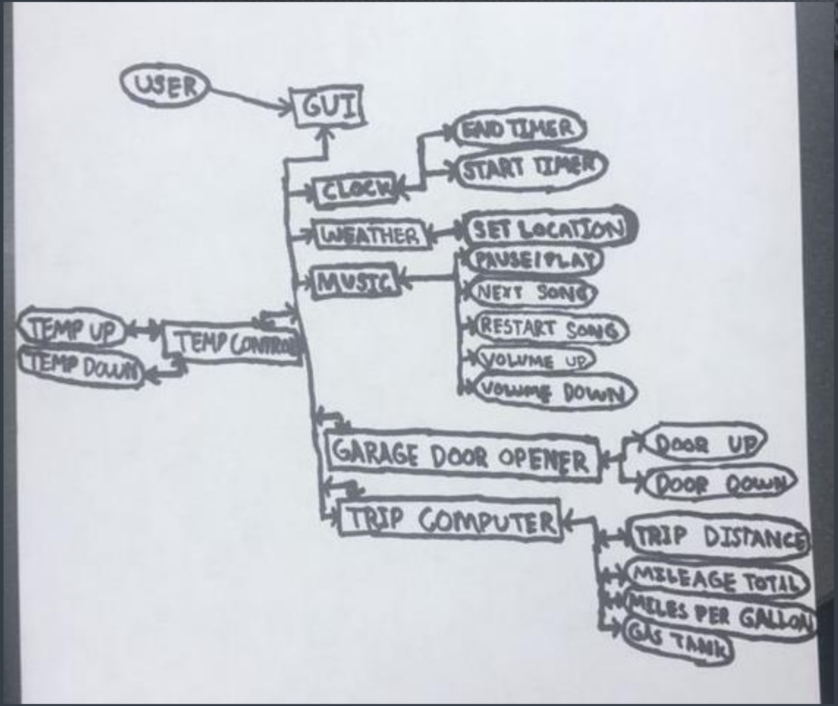
JOSEPH HULL

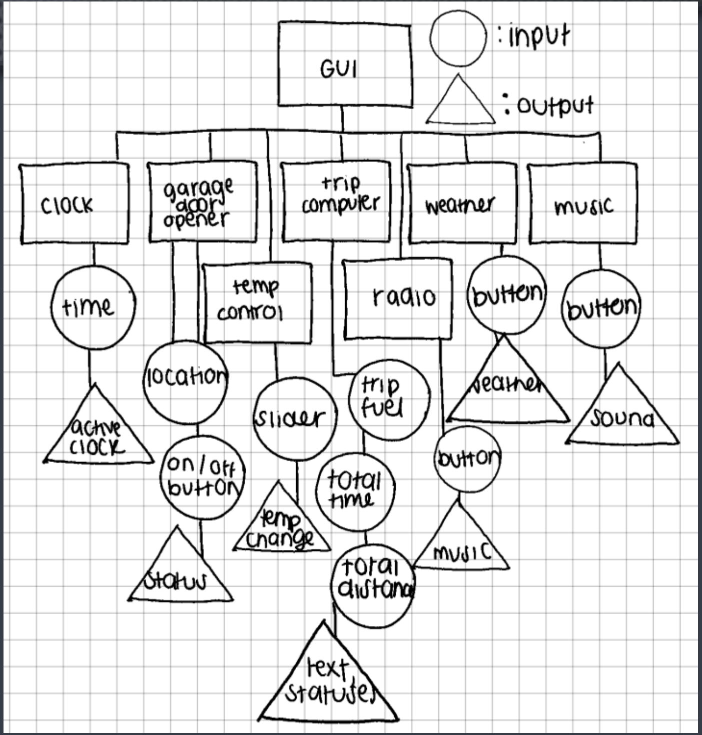
- Developer
- Lead designer on the music player and radio applications
- Co-designer of physical prototype
- Pursuing bachelor's degree in computer science at Auburn

THE BIG PICTURE

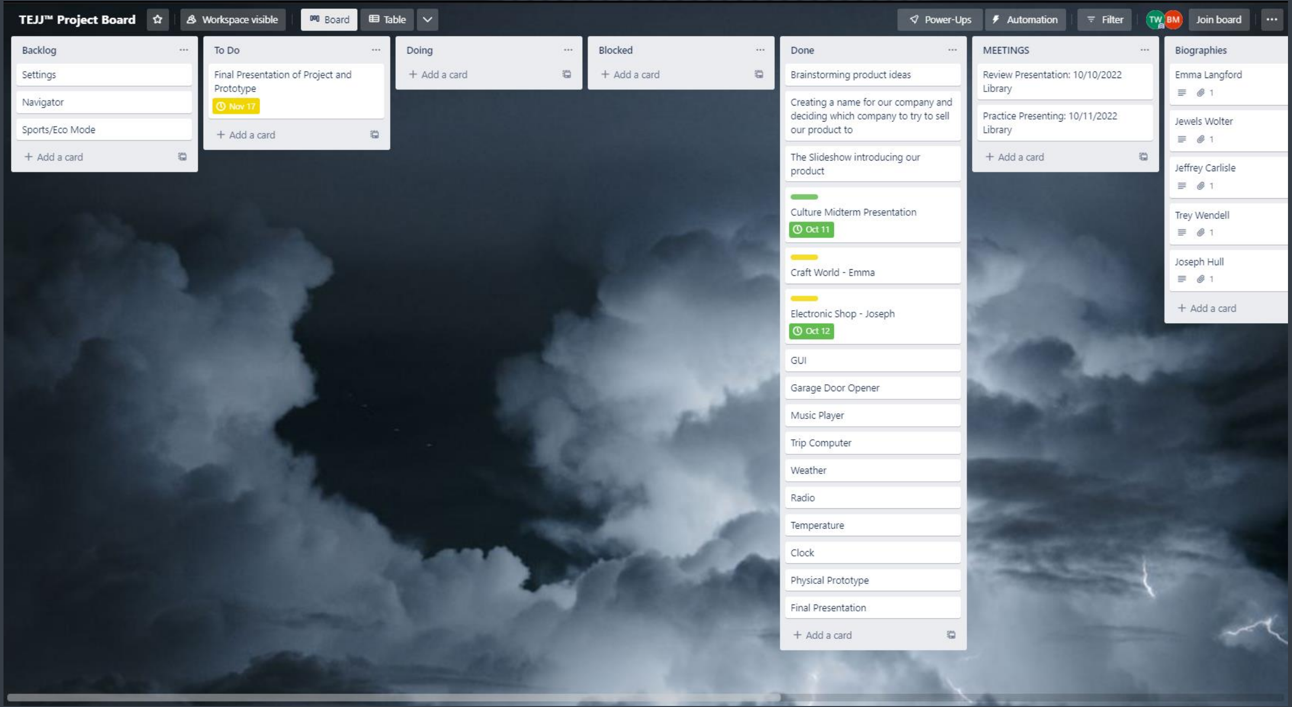
- Simple and reliable
- Assists the driver
- User-friendly and easy to operate
- Minimizes driver distractions

PROCESS


USER
GUI
CLOCK
END TIMER
START TIMER
WEATHER
SET LOCATION
MUSIC
PAUSE/PLAY
NEXT SONG
RESTART SONG
VOLUME UP
VOLUME DOWN
TEMP UP
TEMP DOWN
TEMP CONTROL
GARAGE DOOR OPENER
DOOR UP
DOOR DOWN
TRIP COMPUTER
TRIP DISTANCE
MILEAGE TOTAL
MILES PER GALLON
GAS TANK


GUI
: input
: output
clock
garage door opener
trip computer
weather
music
temp control
radio
time
location
slider
trip fuel
button
button
active clock
on/off button
temp change
total time
button
weather
sound
status
total distance
music
text statuses

TRELLO BOARD



TEJJ™ Project Board

Workspace visible · Board · Table · Power-Ups · Automation · Filter · TW BM · Join board

Backlog
- Settings
- Navigator
- Sports/Eco Mode
- + Add a card

To Do
- Final Presentation of Project and Prototype — Nov 17
- + Add a card

Doing
- + Add a card

Blocked
- + Add a card

Done
- Brainstorming product ideas
- Creating a name for our company and deciding which company to try to sell our product to
- The Slideshow introducing our product
- Culture Midterm Presentation — Oct 11
- Craft World - Emma
- Electronic Shop - Joseph — Oct 12
- GUI
- Garage Door Opener
- Music Player
- Trip Computer
- Weather
- Radio
- Temperature
- Clock
- Physical Prototype
- Final Presentation
- + Add a card

MEETINGS
- Review Presentation: 10/10/2022 Library
- Practice Presenting: 10/11/2022 Library
- + Add a card

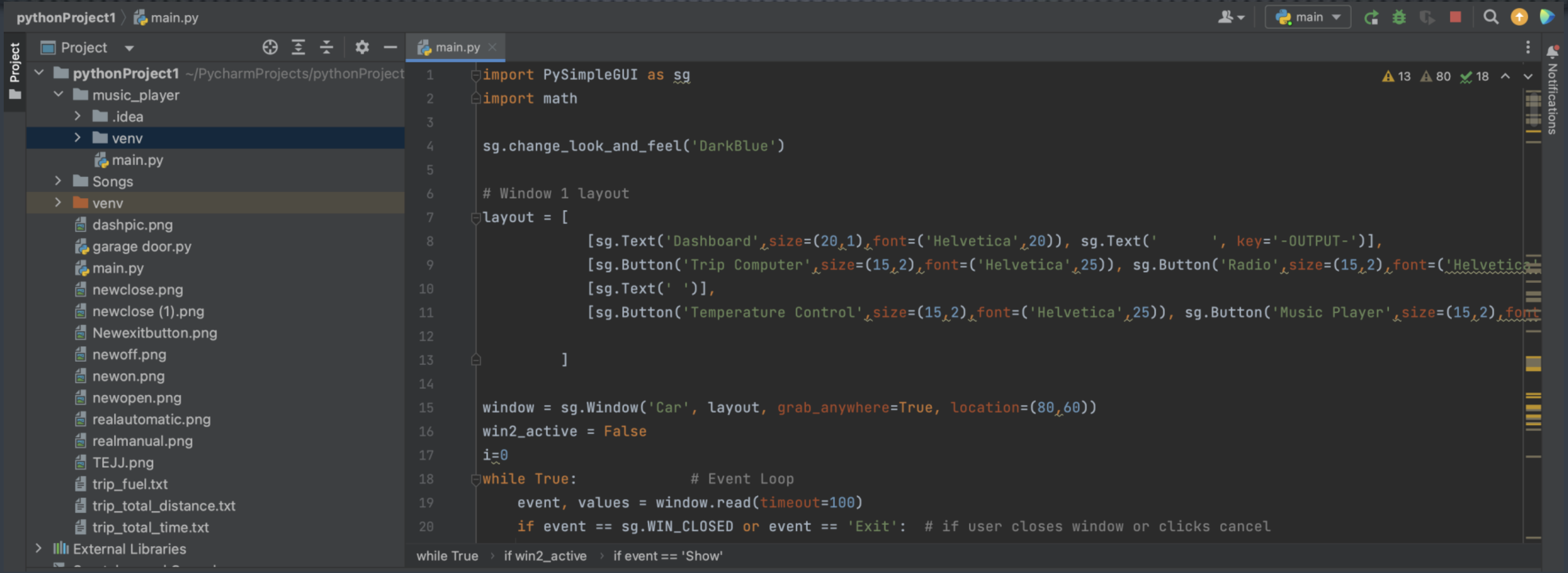
Biographies
- Emma Langford — 1
- Jewels Wolter — 1
- Jeffrey Carlisle — 1
- Trey Wendell — 1
- Joseph Hull — 1
- + Add a card

INTERFACE AND DESIGN

PySimpleGUI

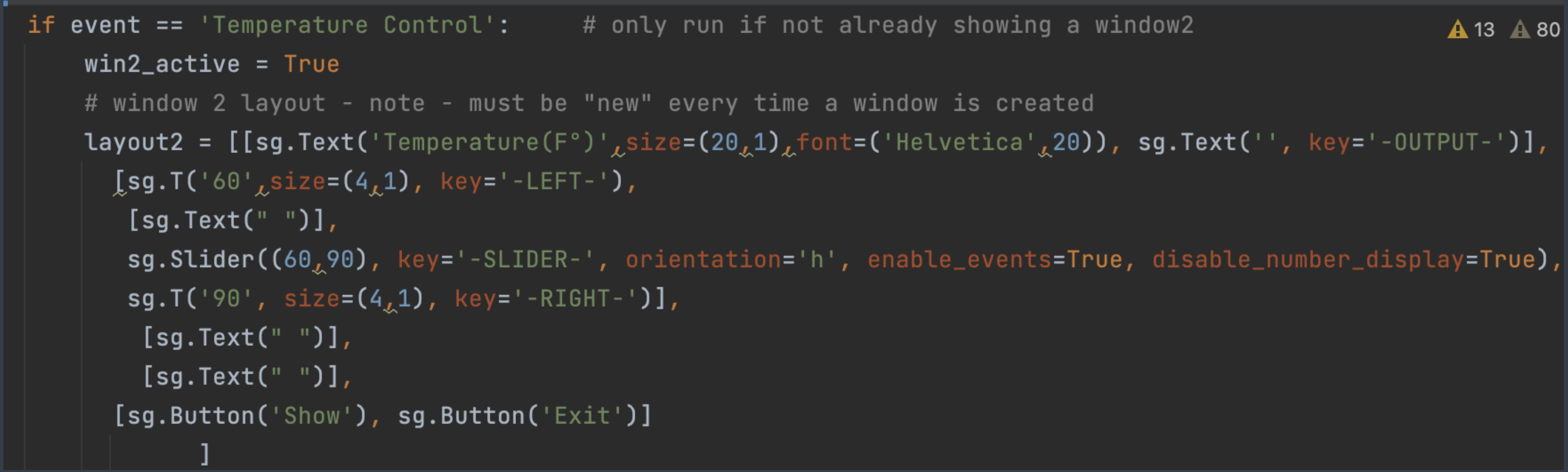
- Based on a Graphical User Interface(GUI)
- Built in Python using PySimpleGUI
- Contains multiple applications

GUI DESIGN (CODE)



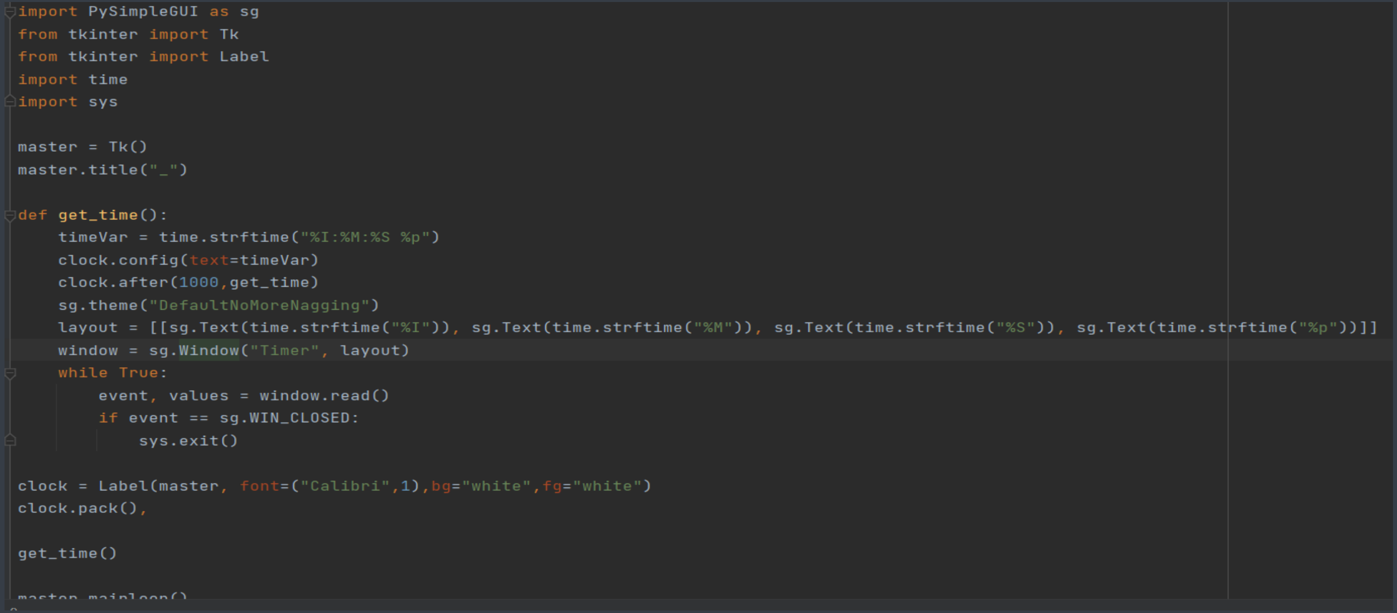
```
import PySimpleGUI as sg
import math

sg.change_look_and_feel('DarkBlue')

# Window 1 layout
layout = [
            [sg.Text('Dashboard',size=(20,1),font=('Helvetica',20)), sg.Text('      ', key='-OUTPUT-')],
            [sg.Button('Trip Computer',size=(15,2),font=('Helvetica',25)), sg.Button('Radio',size=(15,2),font=('Helvetica
            [sg.Text(' ')],
            [sg.Button('Temperature Control',size=(15,2),font=('Helvetica',25)), sg.Button('Music Player',size=(15,2),font

         ]

window = sg.Window('Car', layout, grab_anywhere=True, location=(80,60))
win2_active = False
i=0
while True:             # Event Loop
    event, values = window.read(timeout=100)
    if event == sg.WIN_CLOSED or event == 'Exit':  # if user closes window or clicks cancel
```

TEMPERATURE CONTROL CODE



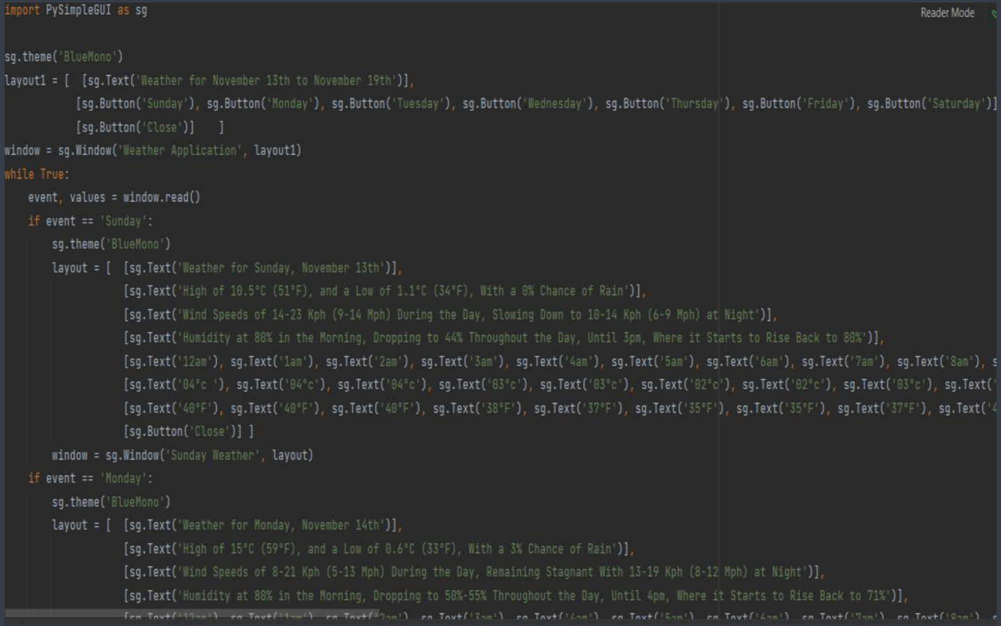
```
if event == 'Temperature Control':      # only run if not already showing a window2
    win2_active = True
    # window 2 layout - note - must be "new" every time a window is created
    layout2 = [[sg.Text('Temperature(F°)',size=(20,1),font=('Helvetica',20)), sg.Text('', key='-OUTPUT-')],
      [sg.T('60',size=(4,1), key='-LEFT-'),
       [sg.Text(" ")],
       sg.Slider((60,90), key='-SLIDER-', orientation='h', enable_events=True, disable_number_display=True),
       sg.T('90', size=(4,1), key='-RIGHT-')],
        [sg.Text(" ")],
        [sg.Text(" ")],
      [sg.Button('Show'), sg.Button('Exit')]
            ]
```

INTERNAL CLOCK CODE

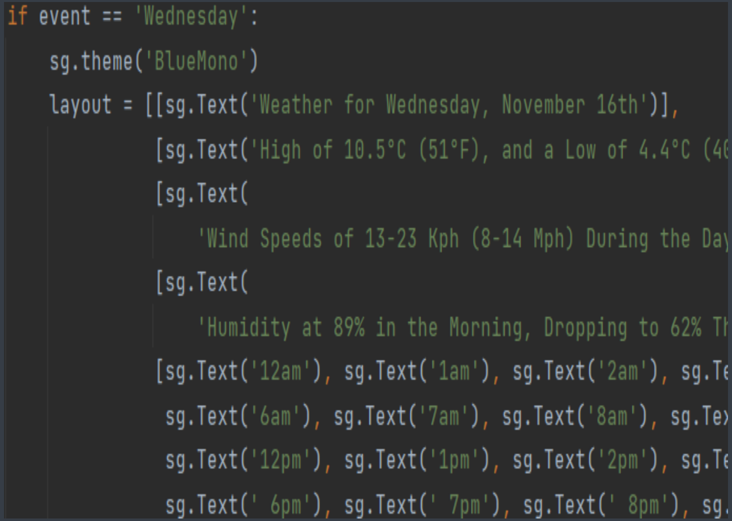


```python
import PySimpleGUI as sg
from tkinter import Tk
from tkinter import Label
import time
import sys

master = Tk()
master.title("_")

def get_time():
    timeVar = time.strftime("%I:%M:%S %p")
    clock.config(text=timeVar)
    clock.after(1000,get_time)
    sg.theme("DefaultNoMoreNagging")
    layout = [[sg.Text(time.strftime("%I")), sg.Text(time.strftime("%M")), sg.Text(time.strftime("%S")), sg.Text(time.strftime("%p"))]]
    window = sg.Window("Timer", layout)
    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED:
            sys.exit()

clock = Label(master, font=("Calibri",1),bg="white",fg="white")
clock.pack(),

get_time()

master.mainloop()
```

WEATHER CODE

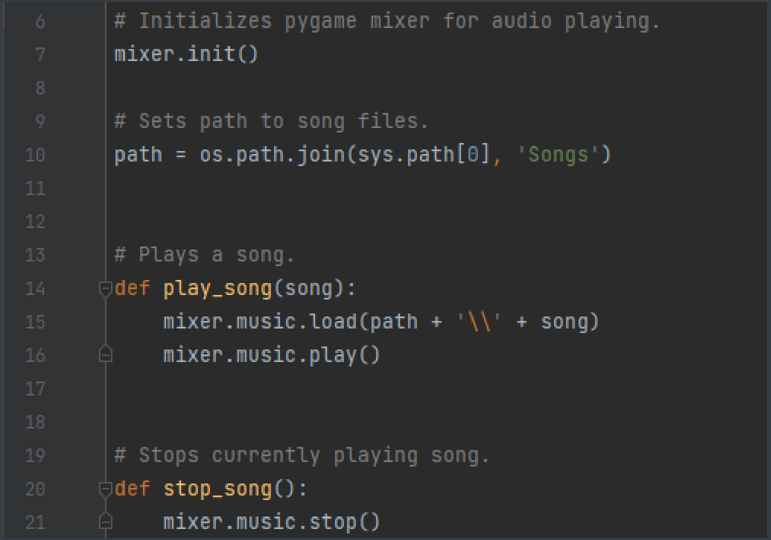


```
import PySimpleGUI as sg


sg.theme('BlueMono')
layout1 = [  [sg.Text('Weather for November 13th to November 19th')],
           [sg.Button('Sunday'), sg.Button('Monday'), sg.Button('Tuesday'), sg.Button('Wednesday'), sg.Button('Thursday'), sg.Button('Friday'), sg.Button('Saturday')]
           [sg.Button('Close')]    ]
window = sg.Window('Weather Application', layout1)
while True:
    event, values = window.read()
    if event == 'Sunday':
        sg.theme('BlueMono')
        layout = [  [sg.Text('Weather for Sunday, November 13th')],
                    [sg.Text('High of 10.5°C (51°F), and a Low of 1.1°C (34°F), With a 0% Chance of Rain')],
                    [sg.Text('Wind Speeds of 14-23 Kph (9-14 Mph) During the Day, Slowing Down to 10-14 Kph (6-9 Mph) at Night')],
                    [sg.Text('Humidity at 80% in the Morning, Dropping to 44% Throughout the Day, Until 3pm, Where it Starts to Rise Back to 80%')],
                    [sg.Text('12am'), sg.Text('1am'), sg.Text('2am'), sg.Text('3am'), sg.Text('4am'), sg.Text('5am'), sg.Text('6am'), sg.Text('7am'), sg.Text('8am'), s
                    [sg.Text('04°c '), sg.Text('04°c'), sg.Text('04°c'), sg.Text('03°c'), sg.Text('03°c'), sg.Text('02°c'), sg.Text('02°c'), sg.Text('03°c'), sg.Text('
                    [sg.Text('40°F'), sg.Text('40°F'), sg.Text('40°F'), sg.Text('38°F'), sg.Text('37°F'), sg.Text('35°F'), sg.Text('35°F'), sg.Text('37°F'), sg.Text('4
                    [sg.Button('Close')] ]
        window = sg.Window('Sunday Weather', layout)
    if event == 'Monday':
        sg.theme('BlueMono')
        layout = [  [sg.Text('Weather for Monday, November 14th')],
                    [sg.Text('High of 15°C (59°F), and a Low of 0.6°C (33°F), With a 3% Chance of Rain')],
                    [sg.Text('Wind Speeds of 8-21 Kph (5-13 Mph) During the Day, Remaining Stagnant With 13-19 Kph (8-12 Mph) at Night')],
                    [sg.Text('Humidity at 80% in the Morning, Dropping to 50%-55% Throughout the Day, Until 4pm, Where it Starts to Rise Back to 71%')],
                    [sg.Text('12am'), sg.Text('1am'), sg.Text('2am'), sg.Text('3am'), sg.Text('4am'), sg.Text('5am'), sg.Text('6am'), sg.Text('7am'), sg.Text('8am'), s
```

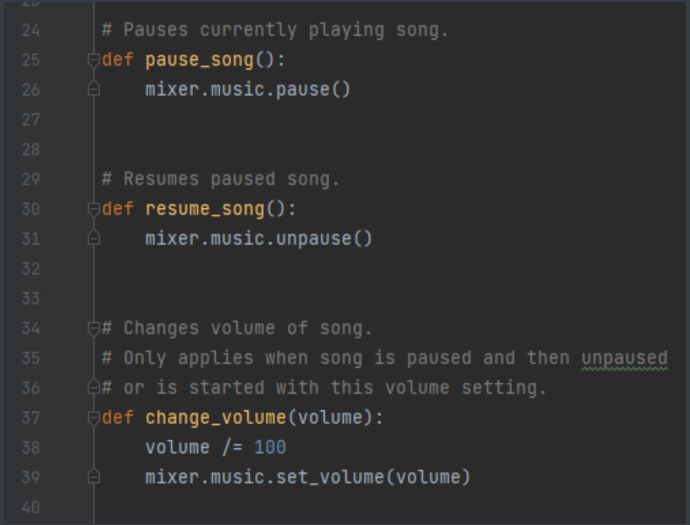


```
if event == 'Wednesday':
    sg.theme('BlueMono')
    layout = [[sg.Text('Weather for Wednesday, November 16th')],
              [sg.Text('High of 10.5°C (51°F), and a Low of 4.4°C (4
              [sg.Text(
                  'Wind Speeds of 13-23 Kph (8-14 Mph) During the Day
              [sg.Text(
                  'Humidity at 89% in the Morning, Dropping to 62% Th
              [sg.Text('12am'), sg.Text('1am'), sg.Text('2am'), sg.Te
               sg.Text('6am'), sg.Text('7am'), sg.Text('8am'), sg.Tex
               sg.Text('12pm'), sg.Text('1pm'), sg.Text('2pm'), sg.Te
               sg.Text(' 6pm'), sg.Text(' 7pm'), sg.Text(' 8pm'), sg.
```

MUSIC PLAYER CODE

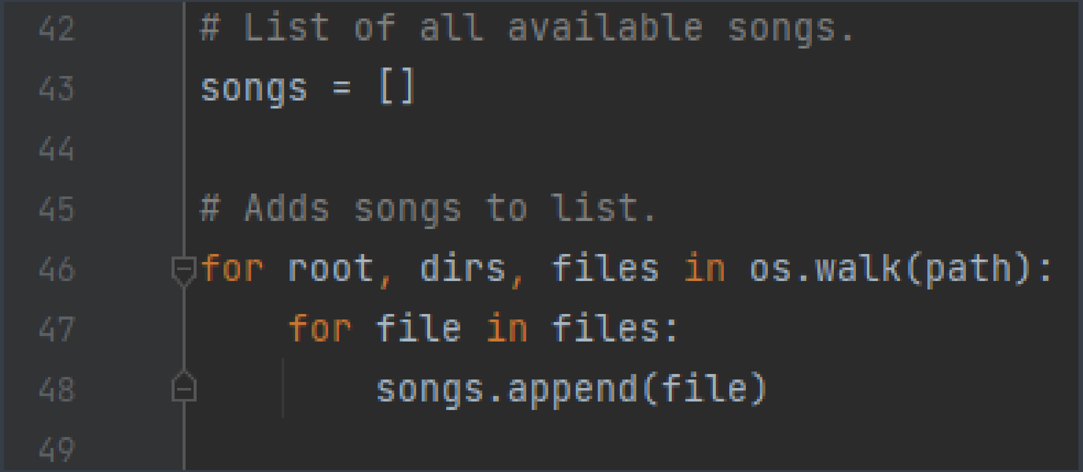


```python
# Initializes pygame mixer for audio playing.
mixer.init()

# Sets path to song files.
path = os.path.join(sys.path[0], 'Songs')


# Plays a song.
def play_song(song):
    mixer.music.load(path + '\\' + song)
    mixer.music.play()


# Stops currently playing song.
def stop_song():
    mixer.music.stop()
```



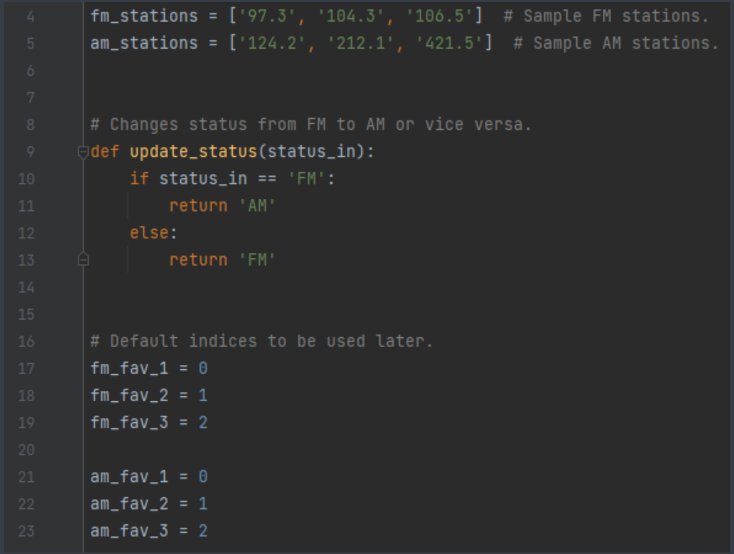
```python
# Pauses currently playing song.
def pause_song():
    mixer.music.pause()


# Resumes paused song.
def resume_song():
    mixer.music.unpause()


# Changes volume of song.
# Only applies when song is paused and then unpaused
# or is started with this volume setting.
def change_volume(volume):
    volume /= 100
    mixer.music.set_volume(volume)

```

MUSIC PLAYER CODE

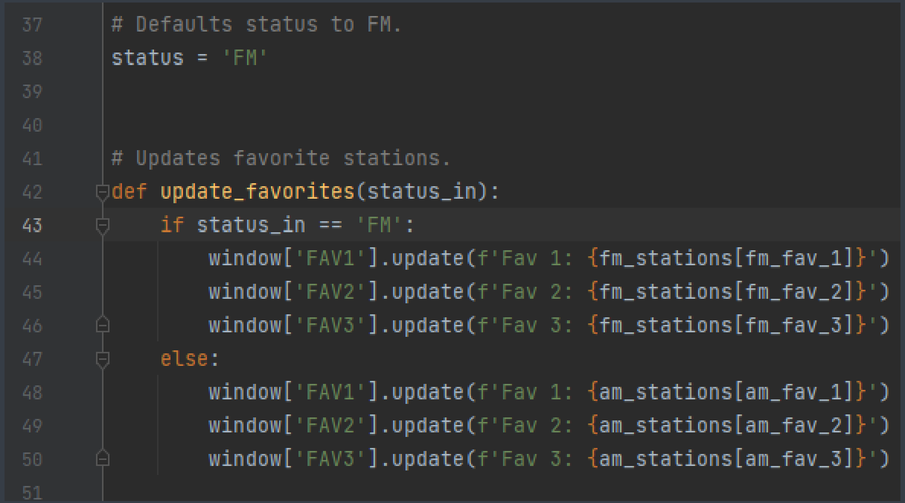


```python
# List of all available songs.
songs = []

# Adds songs to list.
for root, dirs, files in os.walk(path):
    for file in files:
        songs.append(file)

```

RADIO CODE

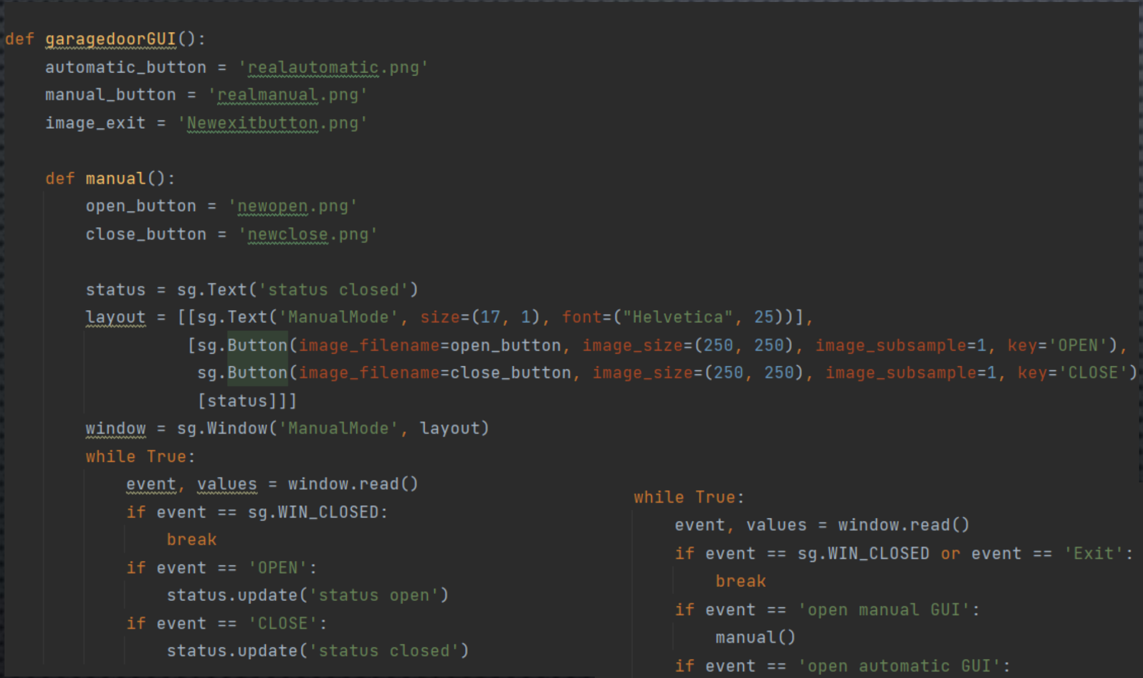


```python
fm_stations = ['97.3', '104.3', '106.5']  # Sample FM stations.
am_stations = ['124.2', '212.1', '421.5']  # Sample AM stations.


# Changes status from FM to AM or vice versa.
def update_status(status_in):
    if status_in == 'FM':
        return 'AM'
    else:
        return 'FM'


# Default indices to be used later.
fm_fav_1 = 0
fm_fav_2 = 1
fm_fav_3 = 2

am_fav_1 = 0
am_fav_2 = 1
am_fav_3 = 2
```

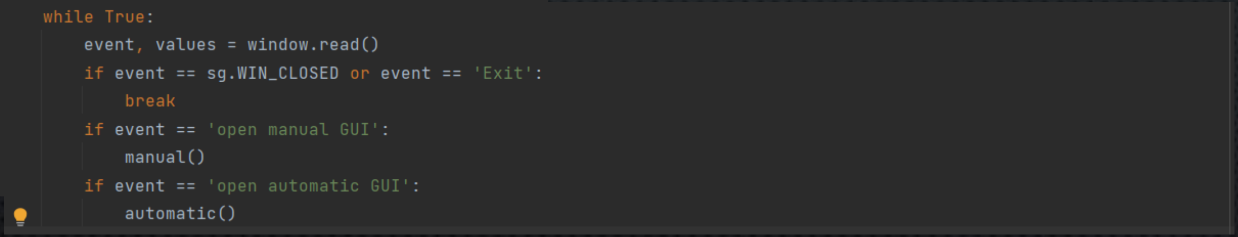


```python
# Defaults status to FM.
status = 'FM'


# Updates favorite stations.
def update_favorites(status_in):
    if status_in == 'FM':
        window['FAV1'].update(f'Fav 1: {fm_stations[fm_fav_1]}')
        window['FAV2'].update(f'Fav 2: {fm_stations[fm_fav_2]}')
        window['FAV3'].update(f'Fav 3: {fm_stations[fm_fav_3]}')
    else:
        window['FAV1'].update(f'Fav 1: {am_stations[am_fav_1]}')
        window['FAV2'].update(f'Fav 2: {am_stations[am_fav_2]}')
        window['FAV3'].update(f'Fav 3: {am_stations[am_fav_3]}')

```

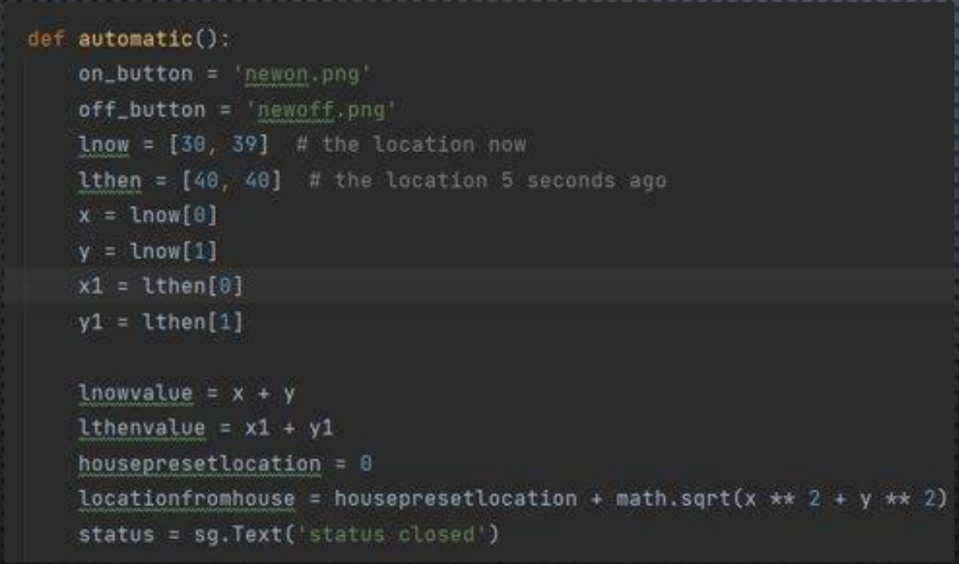
GARAGE DOOR CODE



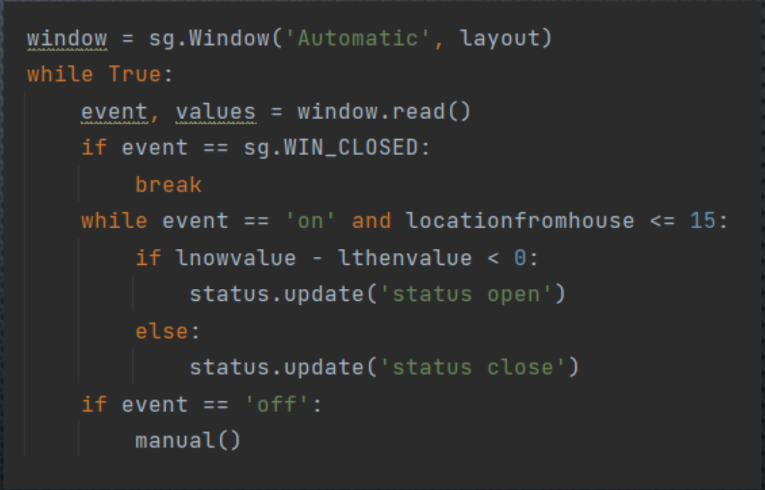
```python
def garagedoorGUI():
    automatic_button = 'realautomatic.png'
    manual_button = 'realmanual.png'
    image_exit = 'Newexitbutton.png'

    def manual():
        open_button = 'newopen.png'
        close_button = 'newclose.png'

        status = sg.Text('status closed')
        layout = [[sg.Text('ManualMode', size=(17, 1), font=("Helvetica", 25))],
                  [sg.Button(image_filename=open_button, image_size=(250, 250), image_subsample=1, key='OPEN'),
                   sg.Button(image_filename=close_button, image_size=(250, 250), image_subsample=1, key='CLOSE')],
                   [status]]]
        window = sg.Window('ManualMode', layout)
        while True:
            event, values = window.read()
            if event == sg.WIN_CLOSED:
                break
            if event == 'OPEN':
                status.update('status open')
            if event == 'CLOSE':
                status.update('status closed')
```



```python
while True:
    event, values = window.read()
    if event == sg.WIN_CLOSED or event == 'Exit':
        break
    if event == 'open manual GUI':
        manual()
    if event == 'open automatic GUI':
        automatic()
```



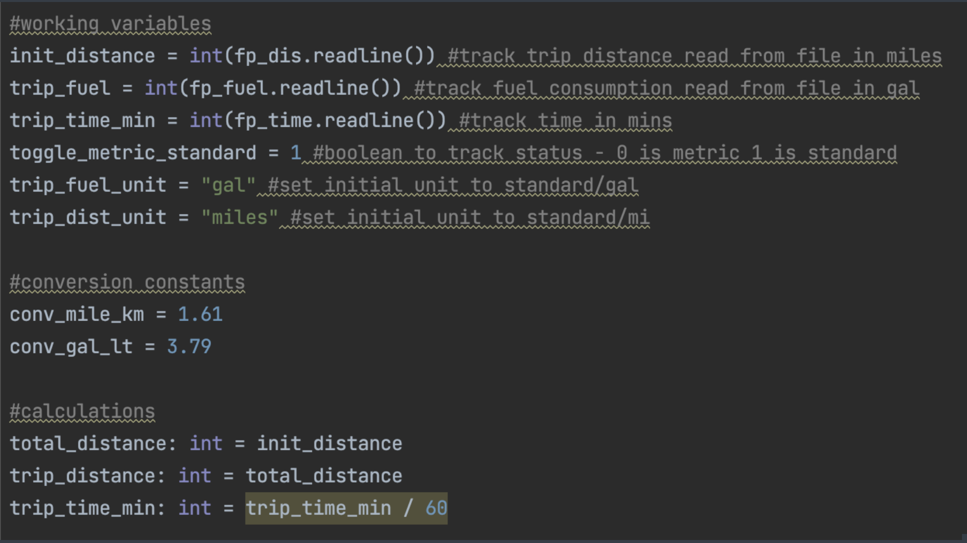
```python
def automatic():
    on_button = 'newon.png'
    off_button = 'newoff.png'
    lnow = [30, 39]  # the location now
    lthen = [40, 40]  # the location 5 seconds ago
    x = lnow[0]
    y = lnow[1]
    x1 = lthen[0]
    y1 = lthen[1]

    lnowvalue = x + y
    lthenvalue = x1 + y1
    housepresetlocation = 0
    locationfromhouse = housepresetlocation + math.sqrt(x ** 2 + y ** 2)
    status = sg.Text('status closed')
```



```python
window = sg.Window('Automatic', layout)
while True:
    event, values = window.read()
    if event == sg.WIN_CLOSED:
        break
    while event == 'on' and locationfromhouse <= 15:
        if lnowvalue - lthenvalue < 0:
            status.update('status open')
        else:
            status.update('status close')
    if event == 'off':
        manual()
```

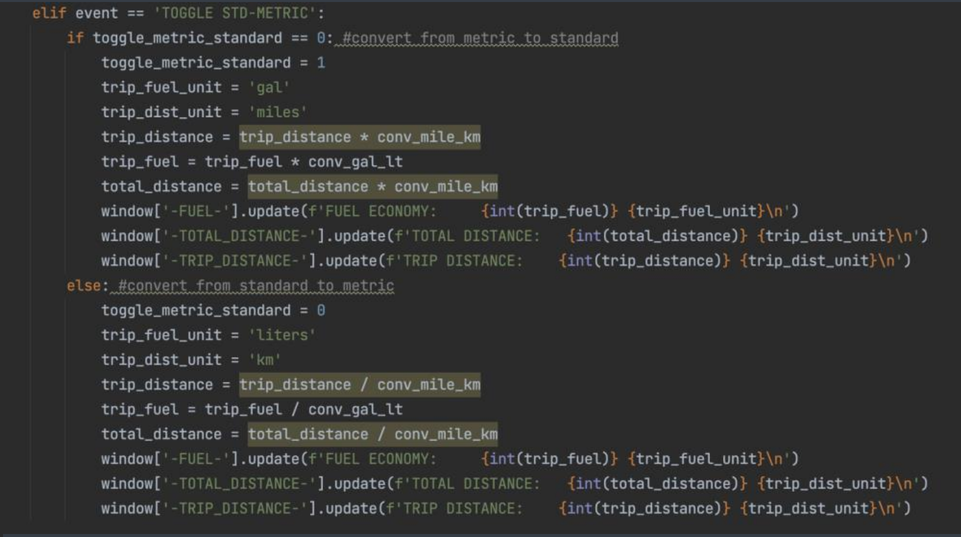
TRIP COMPUTER CODE

VARIABLES AND CALCULATIONS:



```python
#working variables
init_distance = int(fp_dis.readline()) #track trip distance read from file in miles
trip_fuel = int(fp_fuel.readline()) #track fuel consumption read from file in gal
trip_time_min = int(fp_time.readline()) #track time in mins
toggle_metric_standard = 1 #boolean to track status - 0 is metric 1 is standard
trip_fuel_unit = "gal" #set initial unit to standard/gal
trip_dist_unit = "miles" #set initial unit to standard/mi

#conversion constants
conv_mile_km = 1.61
conv_gal_lt = 3.79

#calculations
total_distance: int = init_distance
trip_distance: int = total_distance
trip_time_min: int = trip_time_min / 60
```

TOGGLE STANDARD-METRIC:



```python
elif event == 'TOGGLE STD-METRIC':
    if toggle_metric_standard == 0: #convert from metric to standard
        toggle_metric_standard = 1
        trip_fuel_unit = 'gal'
        trip_dist_unit = 'miles'
        trip_distance = trip_distance * conv_mile_km
        trip_fuel = trip_fuel * conv_gal_lt
        total_distance = total_distance * conv_mile_km
        window['-FUEL-'].update(f'FUEL ECONOMY:     {int(trip_fuel)} {trip_fuel_unit}\n')
        window['-TOTAL_DISTANCE-'].update(f'TOTAL DISTANCE:   {int(total_distance)} {trip_dist_unit}\n')
        window['-TRIP_DISTANCE-'].update(f'TRIP DISTANCE:    {int(trip_distance)} {trip_dist_unit}\n')
    else: #convert from standard to metric
        toggle_metric_standard = 0
        trip_fuel_unit = 'liters'
        trip_dist_unit = 'km'
        trip_distance = trip_distance / conv_mile_km
        trip_fuel = trip_fuel / conv_gal_lt
        total_distance = total_distance / conv_mile_km
        window['-FUEL-'].update(f'FUEL ECONOMY:     {int(trip_fuel)} {trip_fuel_unit}\n')
        window['-TOTAL_DISTANCE-'].update(f'TOTAL DISTANCE:   {int(total_distance)} {trip_dist_unit}\n')
        window['-TRIP_DISTANCE-'].update(f'TRIP DISTANCE:    {int(trip_distance)} {trip_dist_unit}\n')
```