Main.py

from tkinter import \*

import canvas

root = Tk()

root.geometry("1617x1296")

root.title("Mine Sweeper")

#Start-up funtions

canvas.title\_canvas\_build(root)

canvas.title\_canvas.create\_text(808,100,text="Mine Sweeper", font=("Arial",50))

root.mainloop()

Canvas.py

from tkinter import \*

import sweeper\_screen

#Creates Canvas for background Image

def title\_canvas\_build(root):

    #Globals need for title\_canvas and title\_backgroud so root.mainloop can read and display info

    global title\_canvas, title\_background

    #Canvas creation

    title\_canvas = Canvas(root, width=1617, height=1296)

    title\_canvas.place(x=0,y=0, relwidth=1,relheight=1)

    #Assign image to var

    title\_background = PhotoImage(file="Assets/background.png")

    #Add image to canvas

    title\_canvas.create\_image(0,0, image=title\_background, anchor="nw",)

    build\_layout(root)

def build\_miner(root):

    #Function defines the number of tiles then calls the builder functions to add the tiles to the screen

    min = 0

    max = 15

    rows = []

    global buttonsx

    #Look at file: Sweeper\_Screen.py for functions

    rows = sweeper\_screen.build\_mines(min,max,rows)

    rows = sweeper\_screen.build\_number(min,max,rows)

    buttonsx = sweeper\_screen.build\_button(min,max,rows,root)

def start\_click(start\_button,exit\_button,root):

    #Function clears the screen so that the builder function can add the tiles to the screen

    start\_button.destroy()

    exit\_button.destroy()

    #settings\_button.destroy()

    #Function at line 17 in canvas.py

    build\_miner(root)

def run\_quit(root):

    #Function will close program

    root.quit()

def build\_layout(root):

    #Called from main.py will create title screen

    title\_canvas.create\_text(808,100,text="Mine Sweeper", font=("Arial",50))

    #settings\_button = Button(root, text="Settings", font=("Arial", 50), padx=100)

    #settings\_button.place(x=600,y=550)

    exit\_button = Button(root, text="Exit",font=("Arial",50),command= lambda: run\_quit(root),padx=100)

    exit\_button.place(x=650,y=700)

    start\_button = Button(root, text= "Play", font=("Arial",50), command= lambda: start\_click(start\_button,exit\_button,root), padx = 100)

    start\_button.place(x=650,y=400)

Sweeper\_Screen.py

import random

from tkinter import \*

import canvas

import time

def build\_button(min,max,rows,root):

    #Main function to build tiles

    buttonsx = []

    buttonsy = []

    frame = Frame(root)

    #For loops creates all the tiles as buttons with command linking to button\_press

    for x in range(min,max):

        for y in range(min,max):

            temp = ("Button" + str(x) + str(y))

            buttonsy.append(temp)

            (buttonsy[y]) = Button(frame,text=(rows[x])[y],command=lambda data = [x,y]: button\_press(data,buttonsx),padx=5)

            (buttonsy[y]).grid(row= x,column=y)

            (buttonsy[y]).config(foreground="white", activeforeground="white", background="white")

        buttonsx.append(buttonsy)

        buttonsy = []

    frame.place(relx=0.5,rely=0.5, anchor=CENTER)

    #Declaring globals for time, score, and flag

    global reset\_button

    global back\_button

    global score

    global cscore

    global etime

    global ctime

    global flag

    global check\_time

    global flag\_button

    check\_time = True

    flag = False

    cscore = 0

    ctime = 000

    # Adds all other buttons to the main game page

    reset\_button = Button(root,text="Reset Game",padx=5,pady=5,command=lambda data =[frame,root]: game\_reset(data))

    reset\_button.place(relx=0.5,rely=0.75, anchor=CENTER)

    reset\_button.config(state=DISABLED)

    back\_button = Button(root,text="Back",padx=5,pady=5,command=lambda data =[frame,root]:back\_click(data))

    back\_button.place(relx=0.75,rely=0.75, anchor=CENTER)

    score = Label(root,text="Score:   " + str(cscore))

    etime = Label(root,text="Time:   " + str(int(ctime)))

    score.place(relx=0.25,rely=0.25, anchor=CENTER)

    etime.place(relx=0.75,rely=0.25, anchor=CENTER)

    flag\_button = Button(root, text="Flag", command=setchange)

    flag\_button.place(relx=0.25,rely=0.75, anchor=CENTER)

    return buttonsx

def setchange():

    #Allows the user to activate flag mode

    global flag

    if (flag):

        flag = False

    else:

        flag = True

def back\_click(data):

    #Basically destroys everything from main game page and calls the tile building function

    frame = data[0]

    root = data[1]

    frame.destroy()

    reset\_button.destroy()

    back\_button.destroy()

    flag\_button.destroy()

    score.destroy()

    etime.destroy()

    canvas.build\_layout(root)

def game\_reset(data):

    #Called when game is over, allows player to play again

    frame = data[0]

    root = data[1]

    frame.destroy()

    #Destroys the tile frame then rebuilds it with a new set of mines

    canvas.build\_miner(root)

def button\_press(data,buttonsx):

    #Called when tile is pressed

    # Checks if flag mode is active

    if (flag  == False):

        #Disables tile then checks what number is associated with it

        (buttonsx[ data[0]])[data[1]].config(state=DISABLED)

        print((buttonsx[ data[0]])[data[1]].cget('text'))

        mine\_logic(((buttonsx[ data[0]])[data[1]].cget('text')),data,buttonsx)

    else:

        #Blacks out the tile / flagging

        (buttonsx[ data[0]])[data[1]].config(foreground="black", background="black")

def mine\_logic(value,data, buttonsx):

    #Function called from button press to check what the value the tile is and update score and time

    global score

    global cscore

    global etime

    global ctime

    match value:

        case 9:

            #The user has struck a mine, the game is over.

            #((buttonsx[ data[0]])[data[1]]).config(foreground="red", background="red")

            end\_game(buttonsx)

        case 0:

            #If tile number is 0

            ((buttonsx[ data[0]])[data[1]]).config(foreground="blue", background="blue")

            cscore = cscore + 100

            ctime = ctime + 1

            score.config(text="Score:  " + str(cscore))

            etime.config(text="Time:   " + str(ctime))

            score.place()

            etime.place()

        case \_:

           #If tile number is any warning number

           ((buttonsx[ data[0]])[data[1]]).config(foreground="yellow", background="yellow")

           cscore = cscore + 100

           ctime = ctime + 1

           score.config(text="Score:  " + str(cscore))

           etime.config(text="Time:   " + str(ctime))

           score.place()

           etime.place()

def end\_game(buttonsx):

    #Called when player clicks tile number with a 9

    min = 0

    max = len(buttonsx)

    #Disables all tiles and changes color to blue

    for x in range(min,max):

        for y in range(min,max):

            buttonsx[x][y].config(background="Blue",activeforeground="Red")

            buttonsx[x][y].config(state=DISABLED)

    reset\_button.config(state=NORMAL)

def clear\_sea():

    return

def warning():

    return

def build\_mines(min,max,rows):

    #Function creates 2D array and fills  in where mines are

    list1=[0,1,0,0,0,0,0,0]

    column = []

    for i in range(min,max):

        for j in range(min,max):

            num = random.choice(list1)

            if num == 1:

                column.append(9)

            else:

                column.append(0)

        rows.append(column)

        column = []

    return rows

def build\_number(min,max,rows):

    #Looks very confusing but basically adds all warning numbers so all adjacent tiles around a mine have numbers

    #Basically goes through 2D array, checks if value is a 0 or 9

        # If the location in the 2D  array has a 9 then the script will add a 1 to all adjacent tiles

        #However it checks before adding a 1 if the adjacent tile is a 9 then it will not add a 1 as it is mine

    for i in range(min,max):

        for j in range(min,max):

            if (rows[i])[j] == 9:

                if i > 0 and i < len(rows):

                    if (rows[(i-1)])[j-1] != 9:

                        (rows[(i-1)])[j-1]+= 1

                    if (rows[i-1])[j] != 9:

                        (rows[i-1])[j]+= 1

                    if j < (max -1):

                        if (rows[i-1])[j+1] != 9:

                            (rows[i-1])[j+1]+= 1

                        if i < (max-1):

                            if (rows[i+1])[j+1] != 9:

                                (rows[i+1])[j+1]+= 1

                    if i < (max-1):

                        if (rows[(i+1)])[j-1] != 9:

                            (rows[(i+1)])[j-1]+= 1

                        if (rows[(i+1)])[j] != 9:

                            (rows[(i+1)])[j]+= 1

                if j > 0 and j < max-1:

                    if (rows[(i)])[j-1] != 9:

                        (rows[(i)])[j-1]+= 1

                    if (rows[(i)])[j+1] != 9:

                        (rows[(i)])[j+1]+= 1

    return rows