Design Document
Secret Phrase Game Project (Python)

James Robbie

2022-12-19

James Robbie
CSE 222-101

Table of Contents

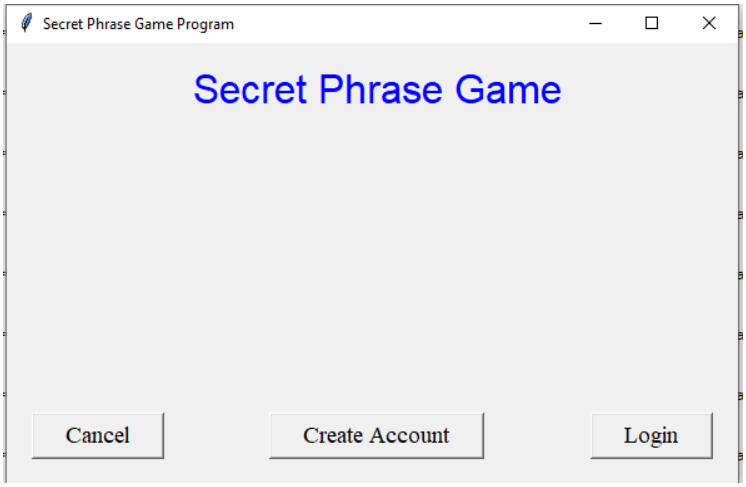
James Robbie
CSE 222-101

Initial Goal

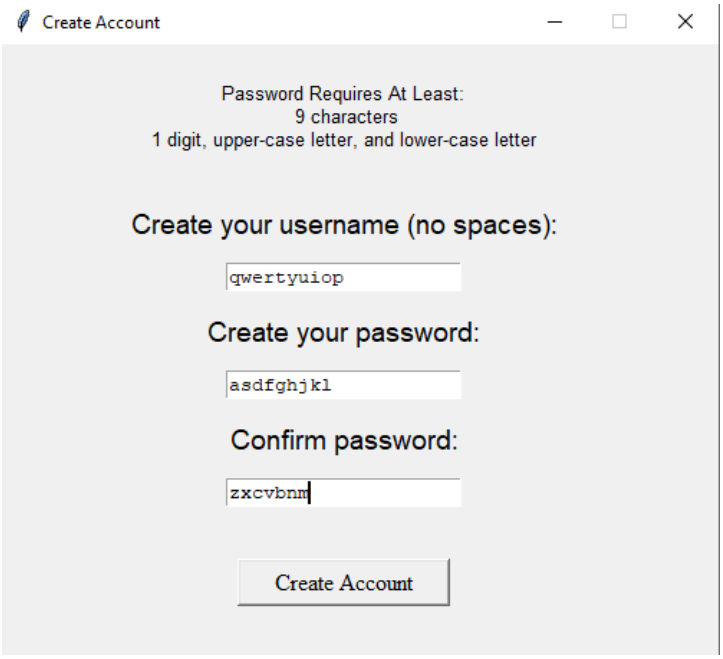
The overall goal of this project is to create a functional Secret Phrase guessing game within Python. For my project, I went for a more Wordle inspired game, where users are to guess a five letter word using clues and within five guesses. There will be menus if the user needs help with the instructions.

Milestone #1 - 9/28/22

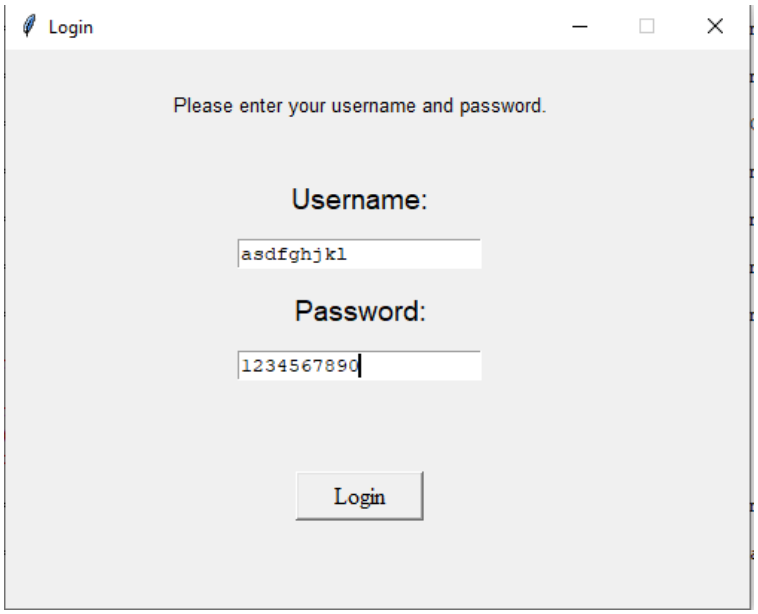
The goal of Milestone #1 is to create the initial window and GUI of the program, as well as create two windows that will function as a Create Account and Login for the user.



Main GUI Window. Picture may be added later on as the game is developed.
Fonts may be changed as well. All buttons are functional.



Create Account Window: Text boxes to be made right-aligned and passwords should be censored. Button is not functional yet.



Login Window: Text boxes to be made right aligned, and password censored as “*****”

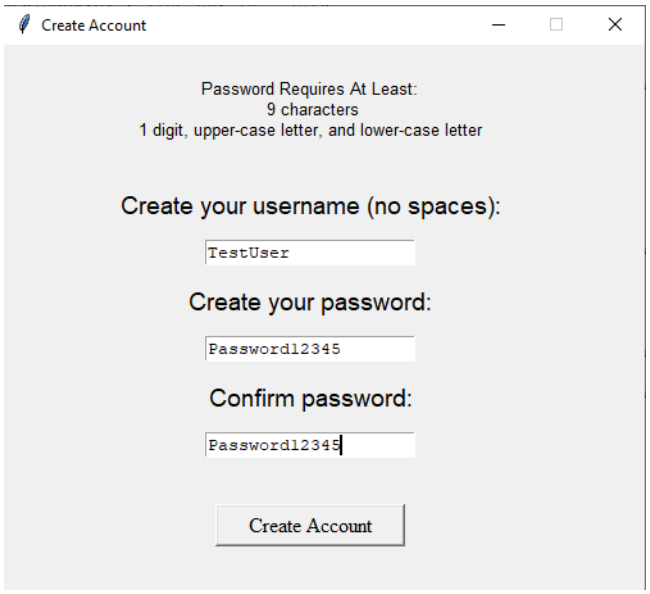
Pseudocode/Checklist:

- Initial GUI opens on program start (**X**)
- GUI has title and picture(possibly) (**X**)
- GUI has Login, Create Account, and Cancel Buttons (**X**)
- Cancel button closes window (**X**)
- Create account button executes createAcct() function (**X**)
- createAcct() function opens new window (**X**)
- Login button executes login() function (**X**)
- login() function opens Login window (**X**)
- CreateAcct window has 3 dialogue boxes (Username, Password, ConfirmPW) (**X**)
- Login window has Username, Password dialogue boxes and Login Button (**X**)

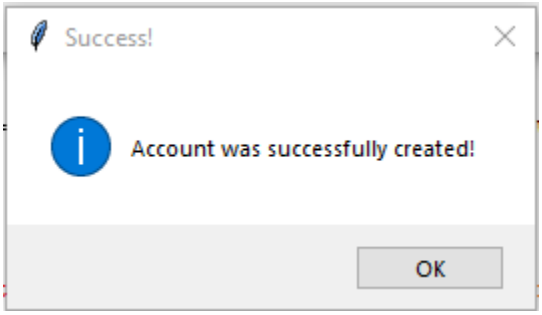
James Robbie
CSE 222-101

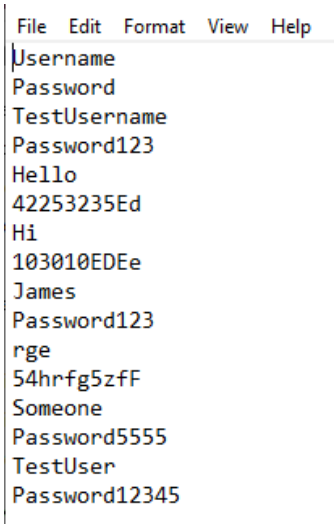
Milestone #2 - 10/12/22

The goal of Milestone 2 is to provide functionality to the Login and Create Account windows, as well as start on the GUI of the main program.

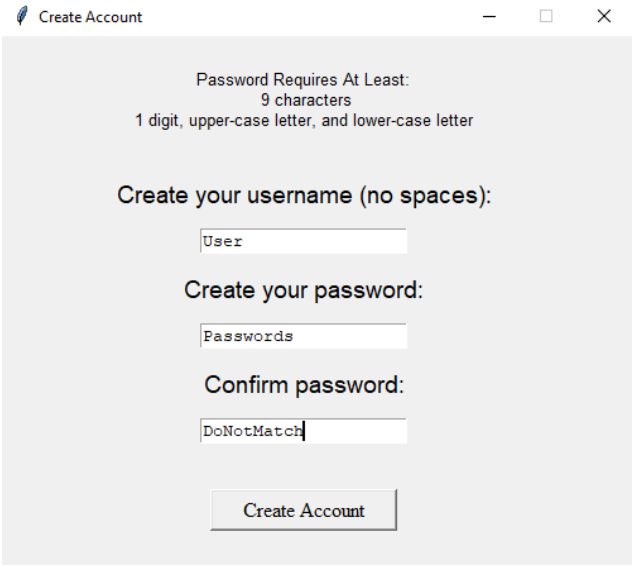


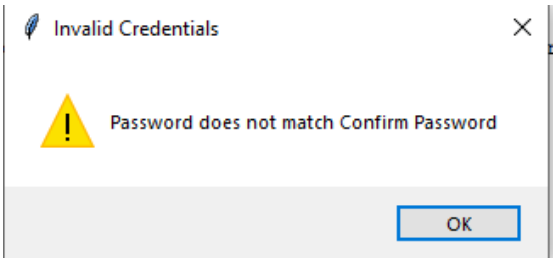
After some troubleshooting, I have figured out that Tkinter Entry boxes would work better than Text boxes, as they can be configured to be right-aligned and censored. I will implement this on a future milestone.



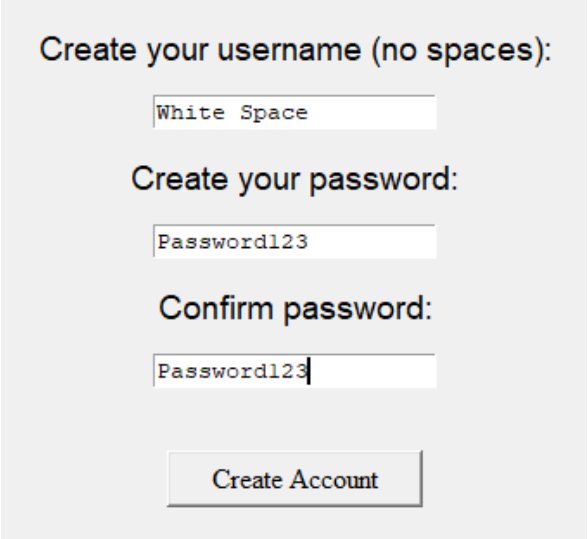


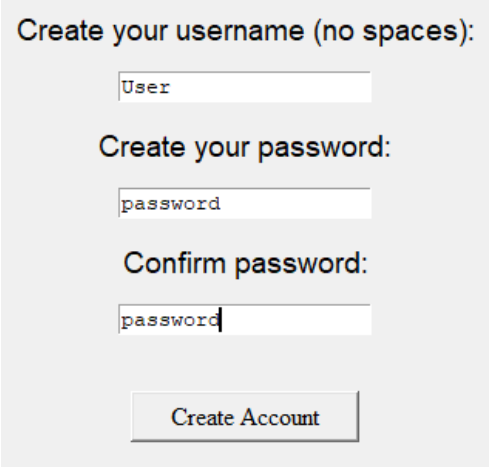
If Username and Password are valid, they are written into a Text File and the program proceeds.

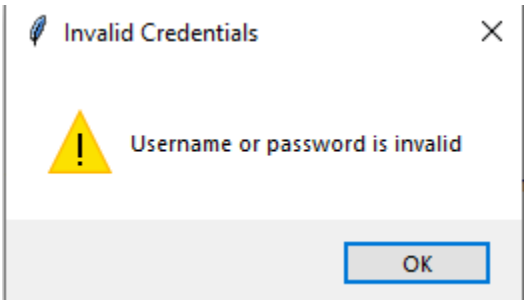




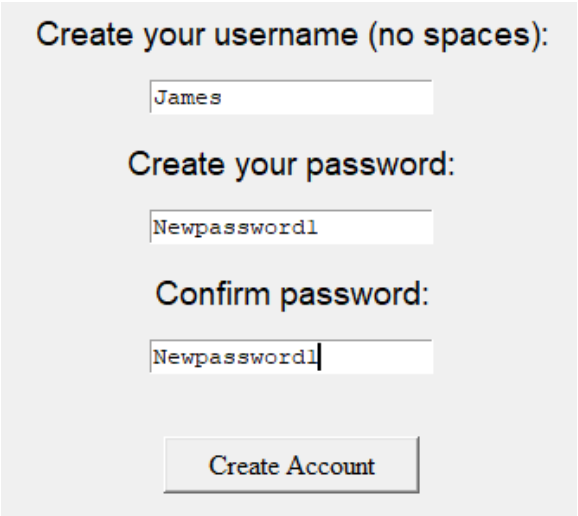
Error comes up if Password and Confirm Password does not match.

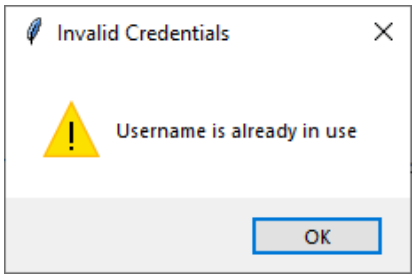




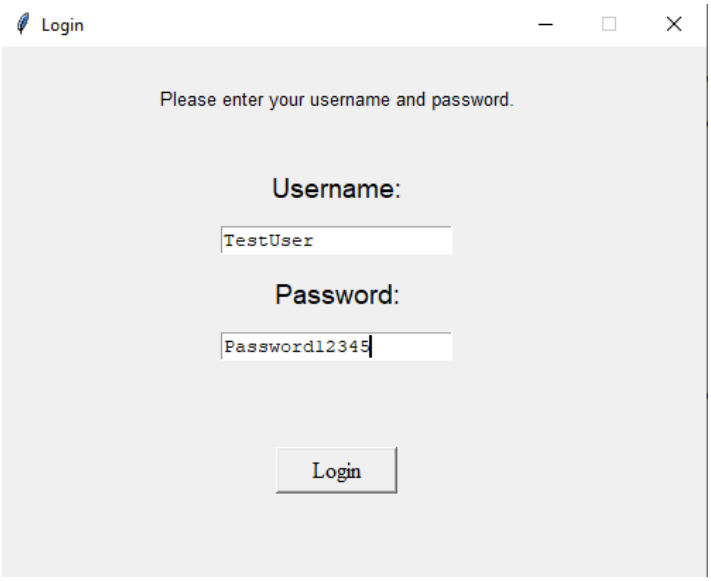


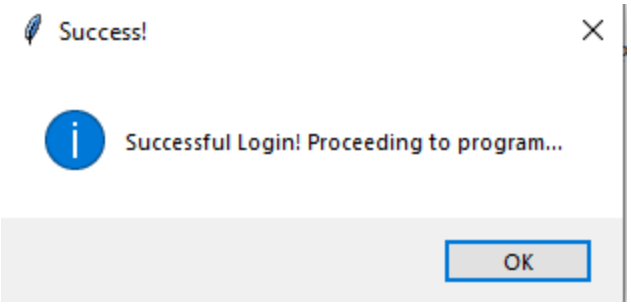
Error will be given if Username and Password requirements are not met.



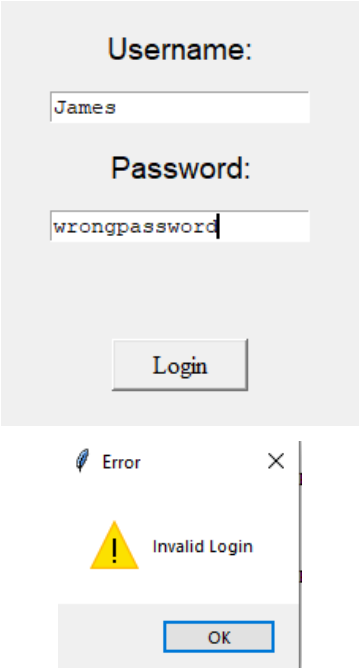


Error will be given if Username is already in system

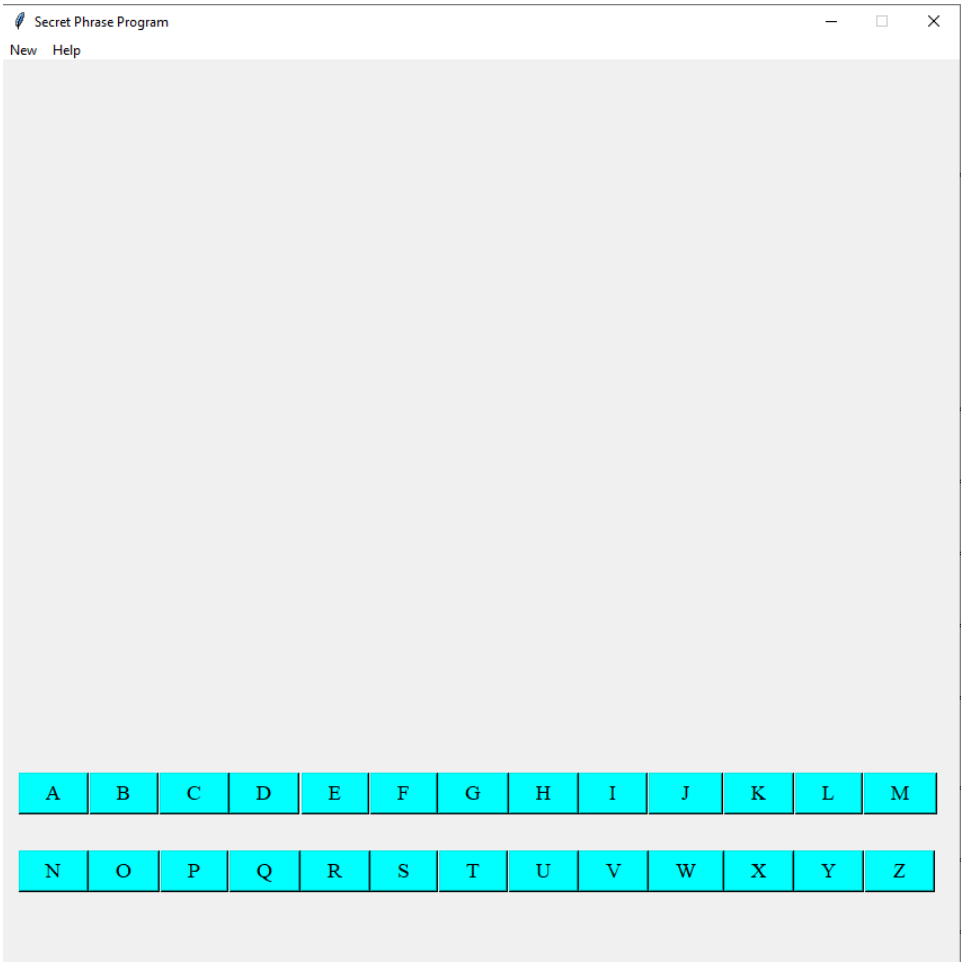




Login window is now functional



Gives Error if Login is incorrect

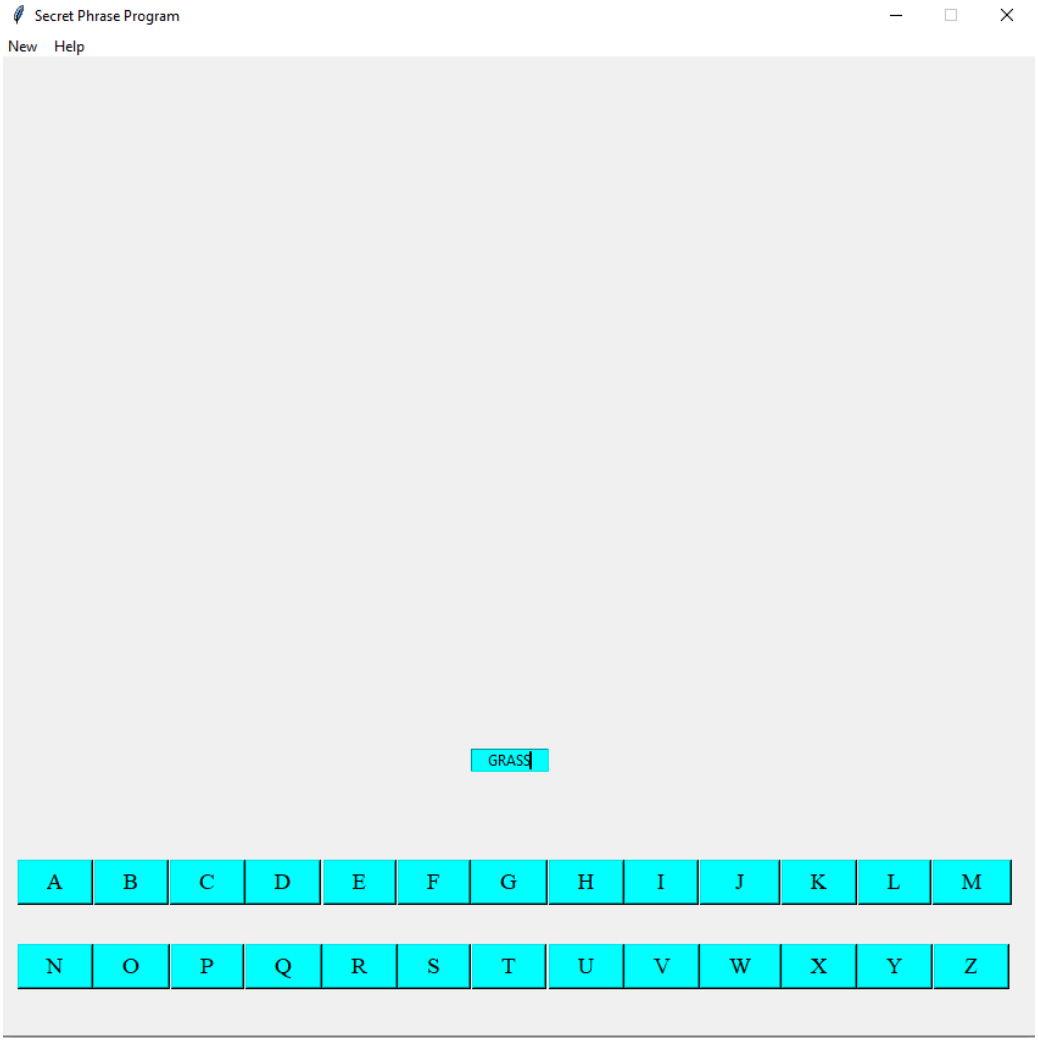


Main Program GUI. Keyboard Buttons. Menu created at the top left as well for options.

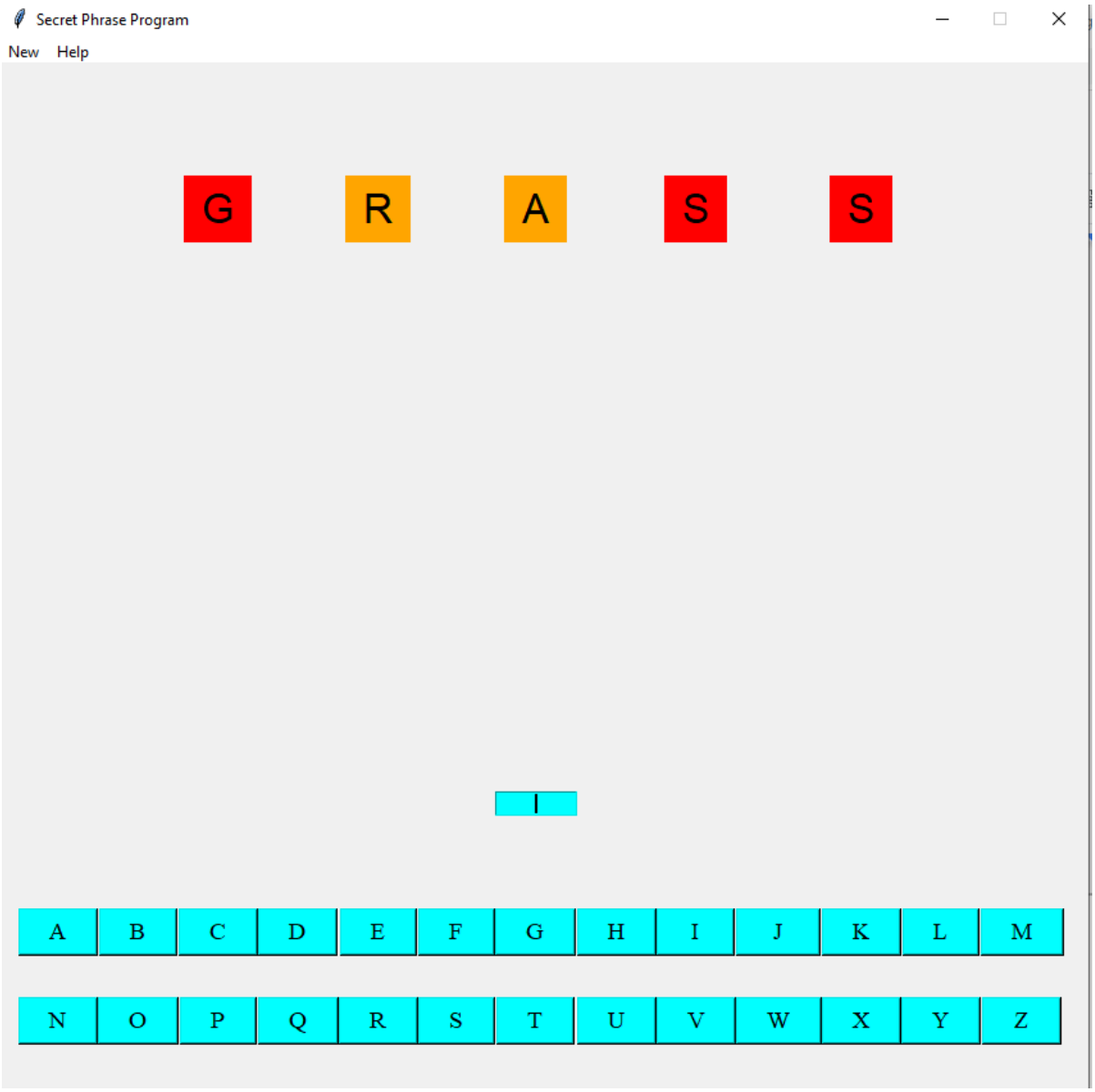
James Robbie
CSE 222-101

Milestone #3 - 11/2/22

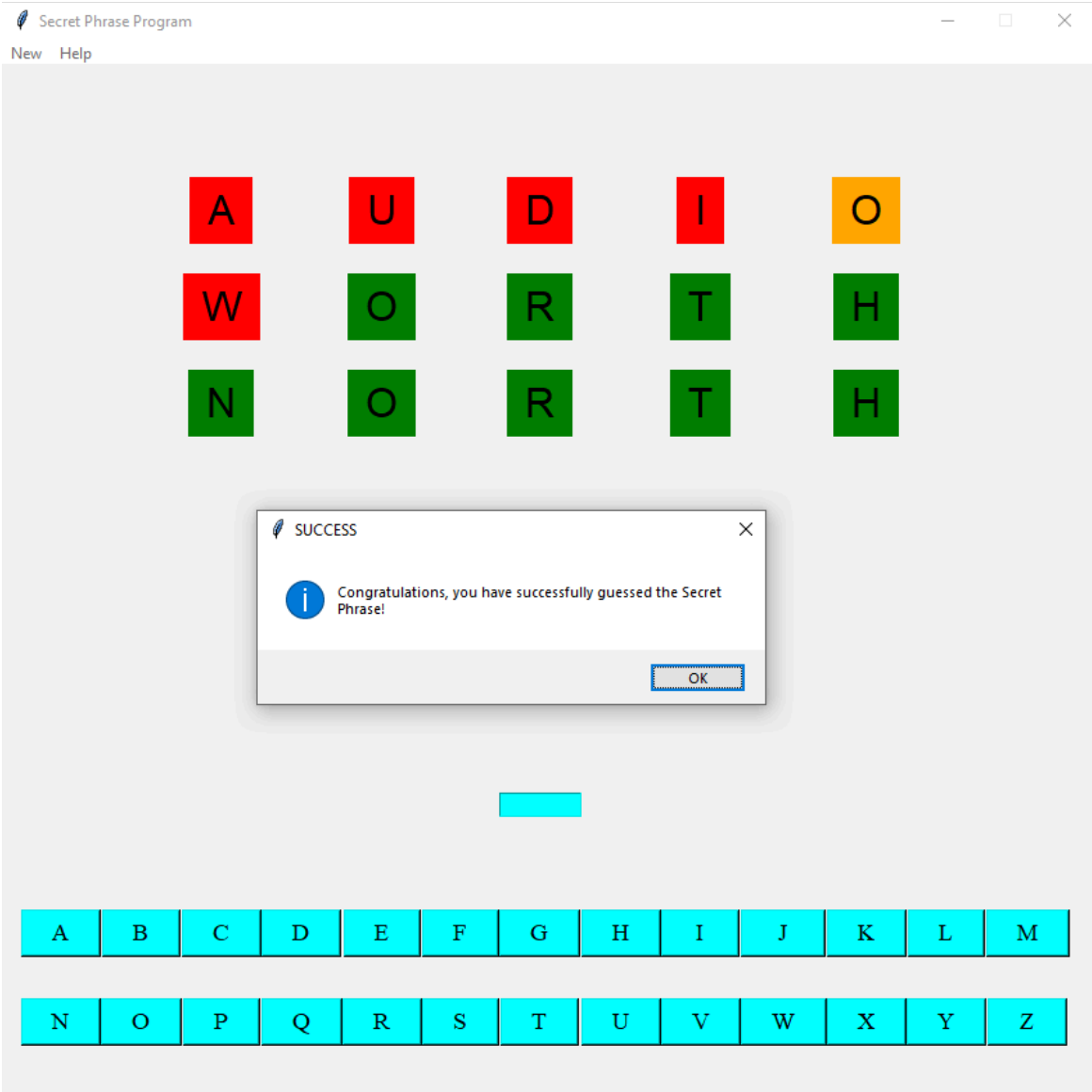
The goal of Milestone 3 is to provide semi-function to the overall program. In this Milestone I achieved the base functionality of the program, and will make quality-of-life tweaks to the program in future Milestones. The game itself is complete.



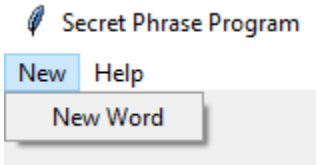
New Main Window, there is now an entry bar for the user to enter guesses. The buttons and typing both work.

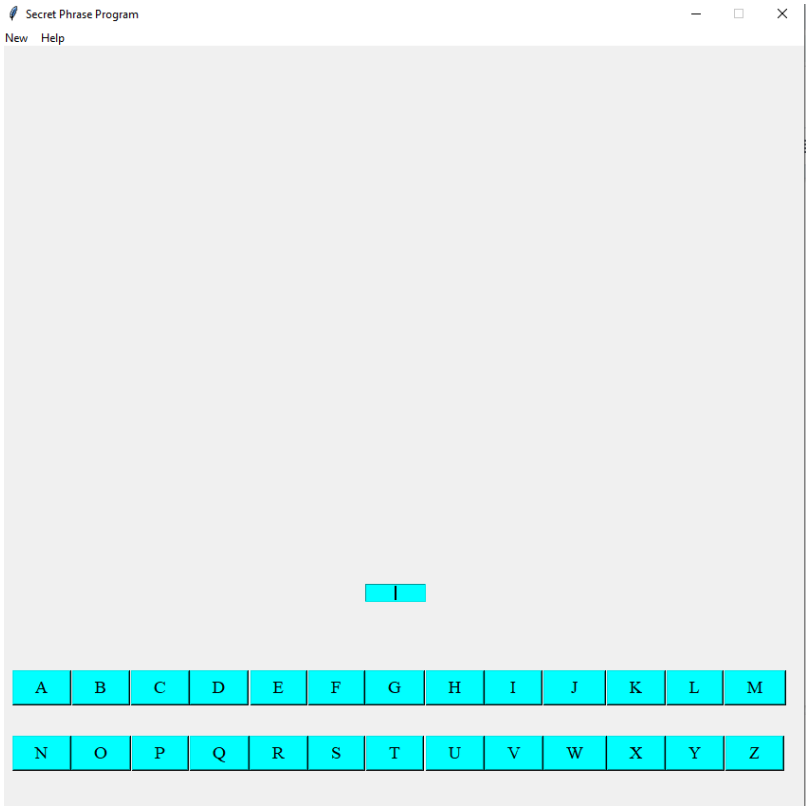


When the user presses ENTER, the guess is processed onto the window. It provides clues to what the word is that they are trying to guess. Red means the letter is not in the word. Orange means the letter is in the word but not in the right position. Green means the letter is in the word and in the right position.

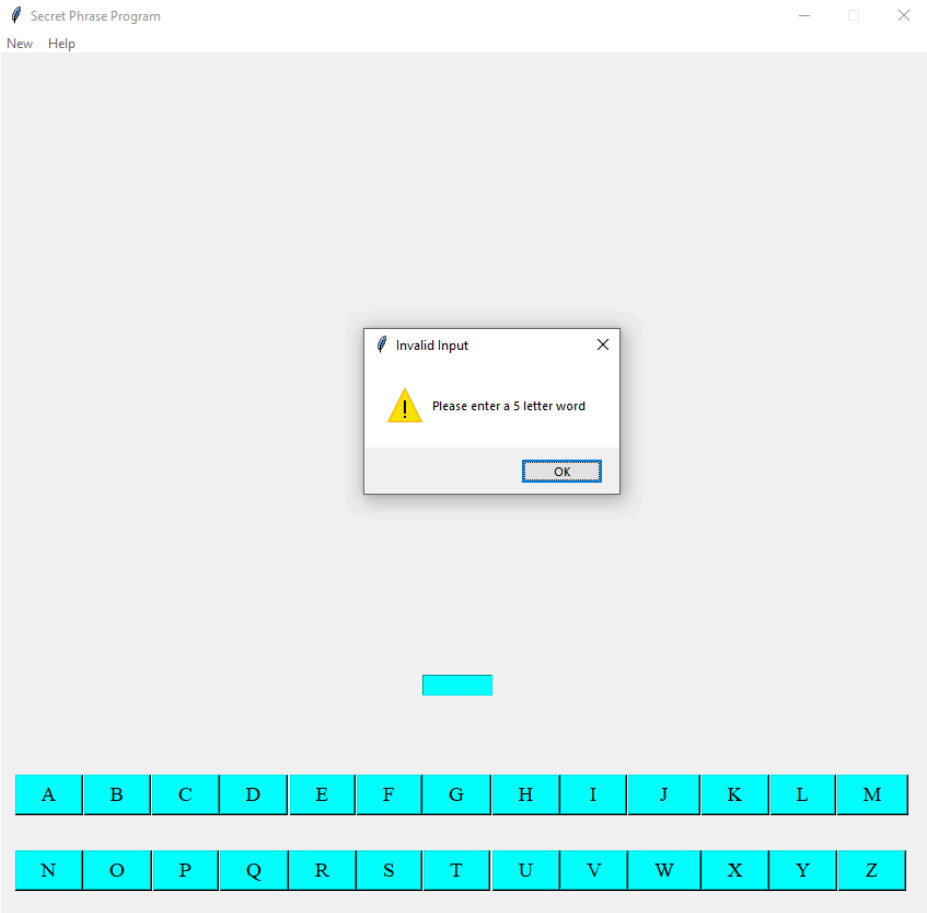


The program can determine when you have guessed the secret phrase.

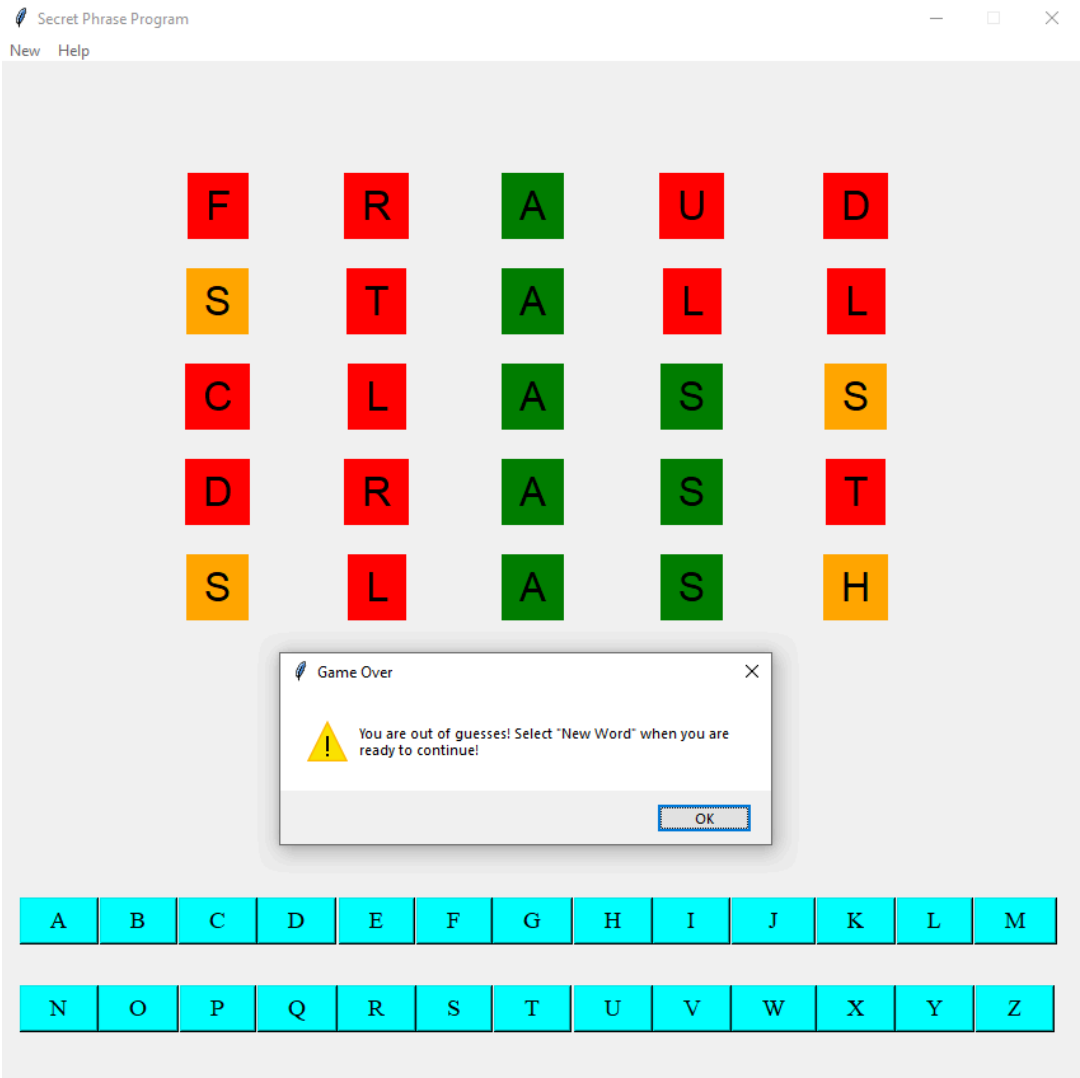




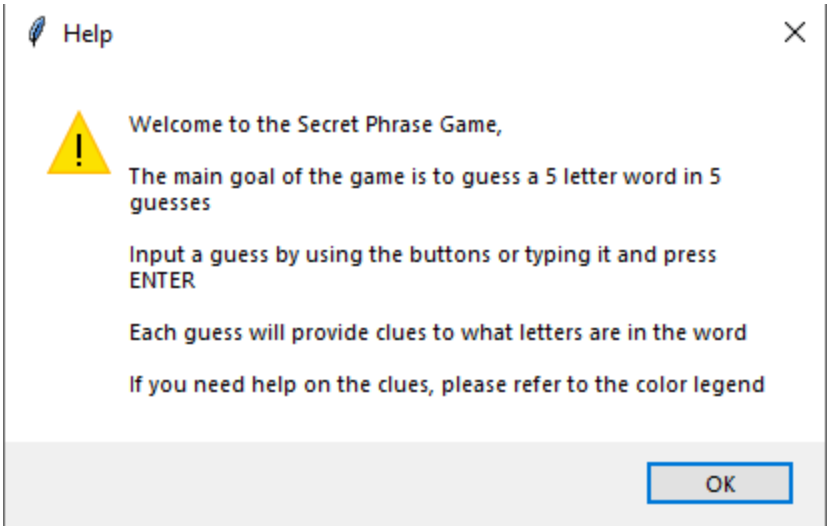
Users can reset the game by selecting New > New Word

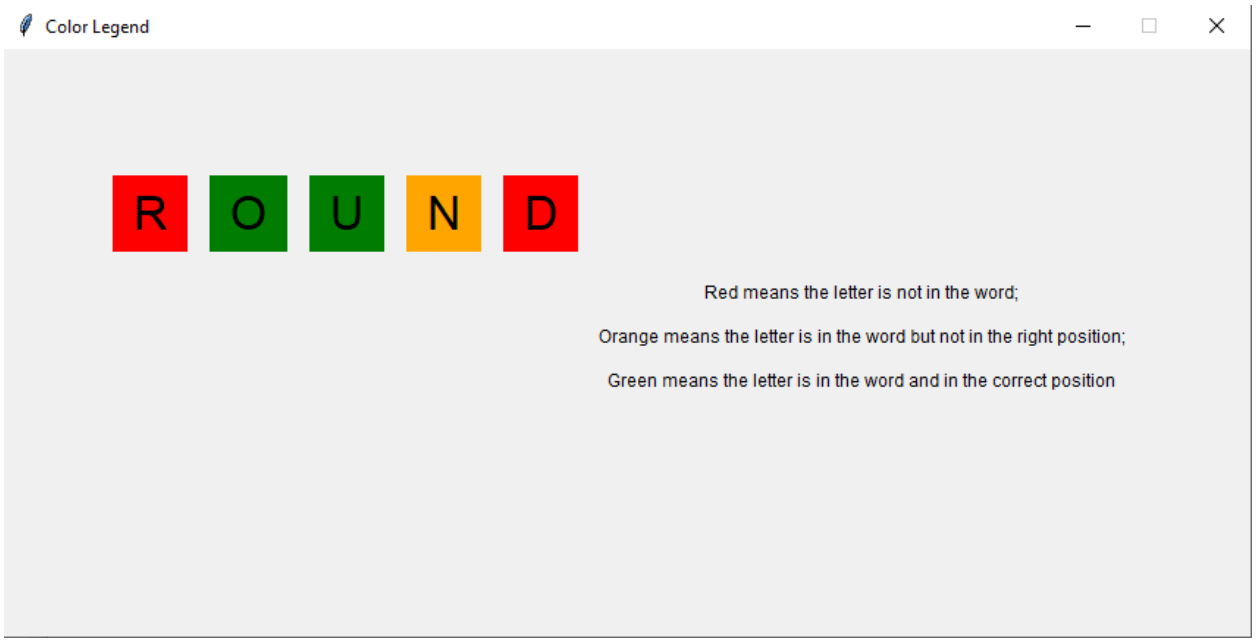


If the user tries to enter a word that is not 5 letters, this error pops up.



The game locks after the user's 5 guesses are up





The Help Menu is included if needed.

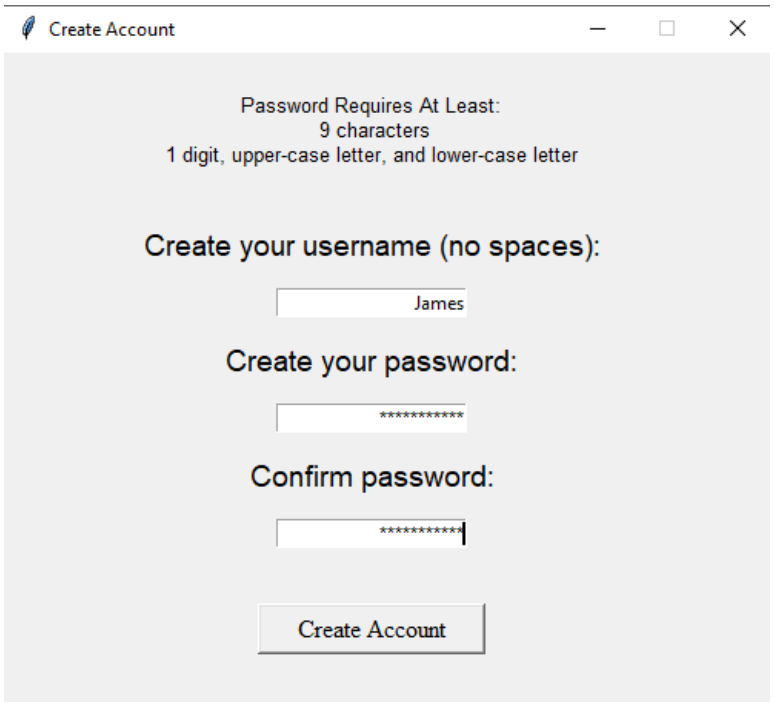
To Do:

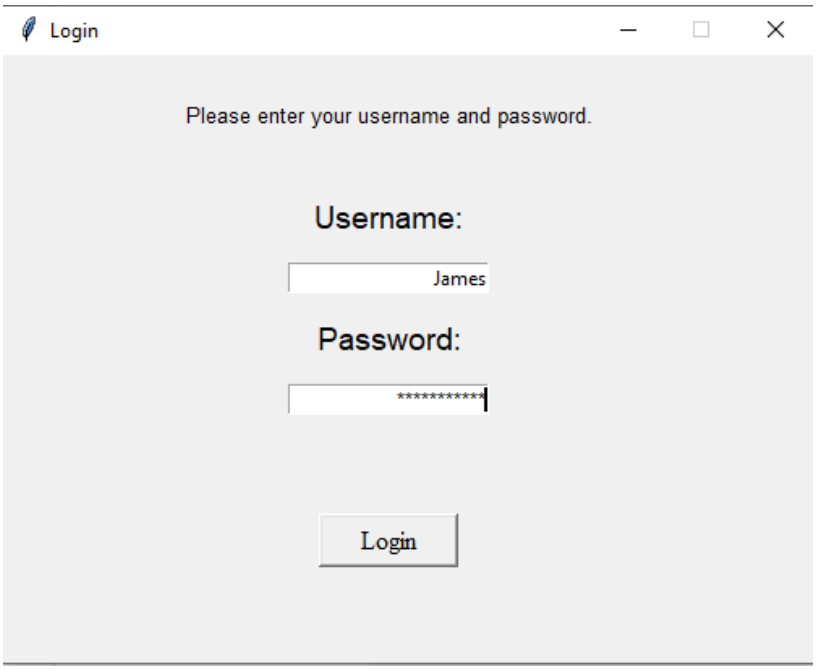
- Make incorrect letters lock the buttons after being guessed
- Provide a solution for words with 2 of the same letters

James Robbie
CSE 222-101

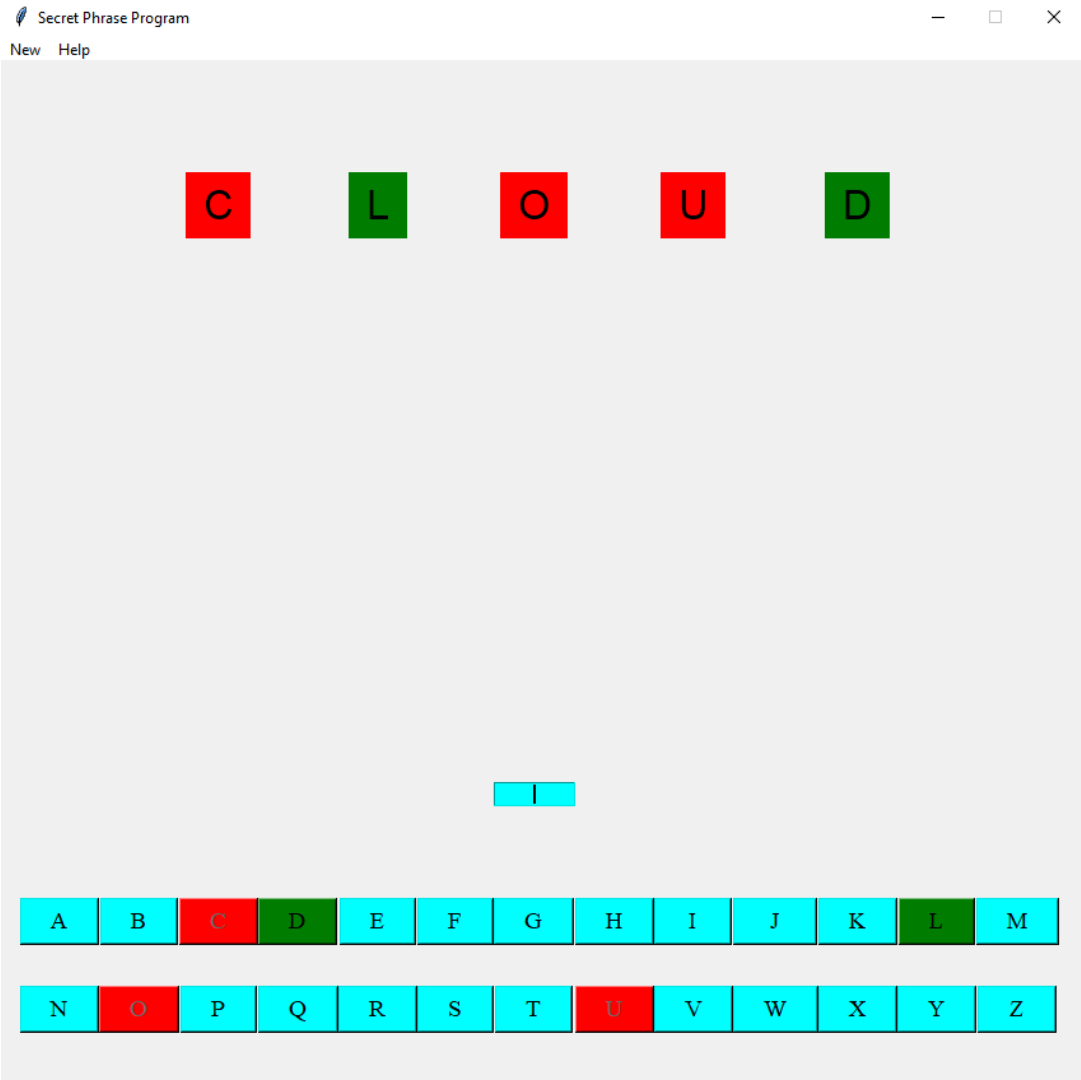
Milestone #4 - 11/16/22

The goal of Milestone #4 is to give further functionality and changes to the buttons and user interface. The layout of the previous windows have been touched up as well.

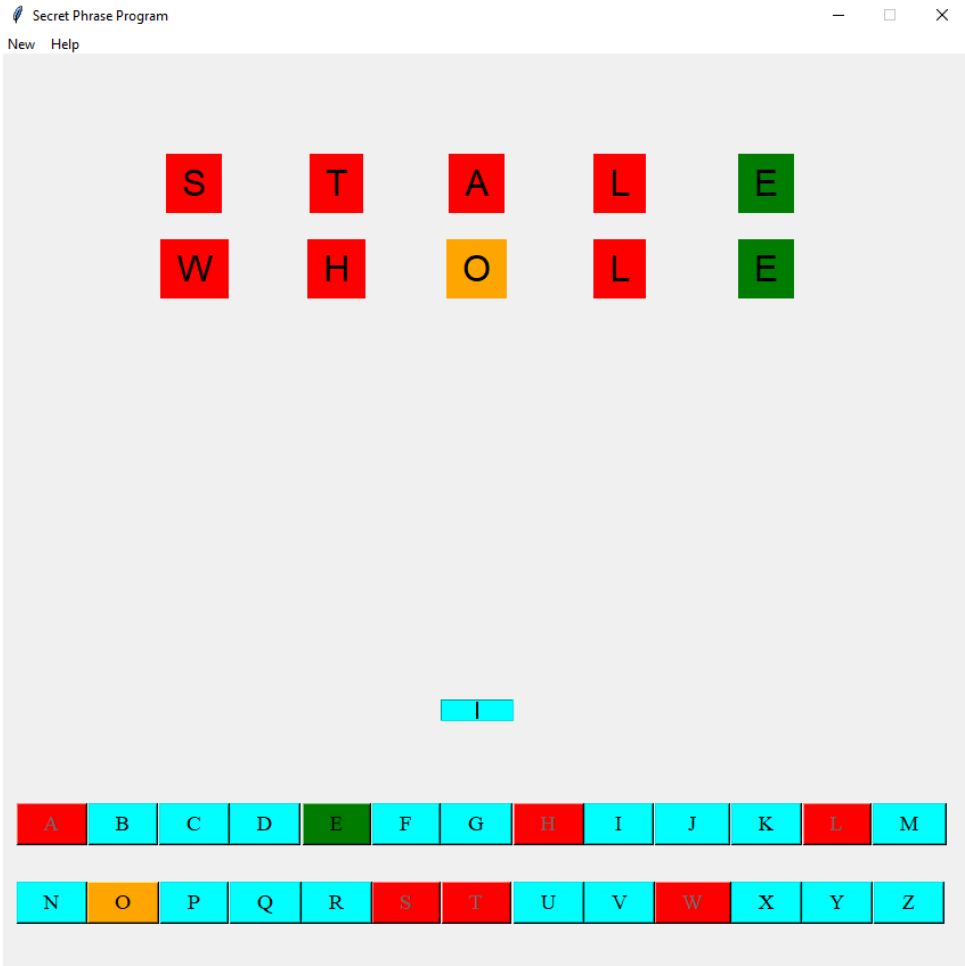




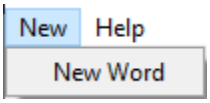
tk.Text boxes are changed to tk.Entry boxes, allowing right alignment as well as password censoring.

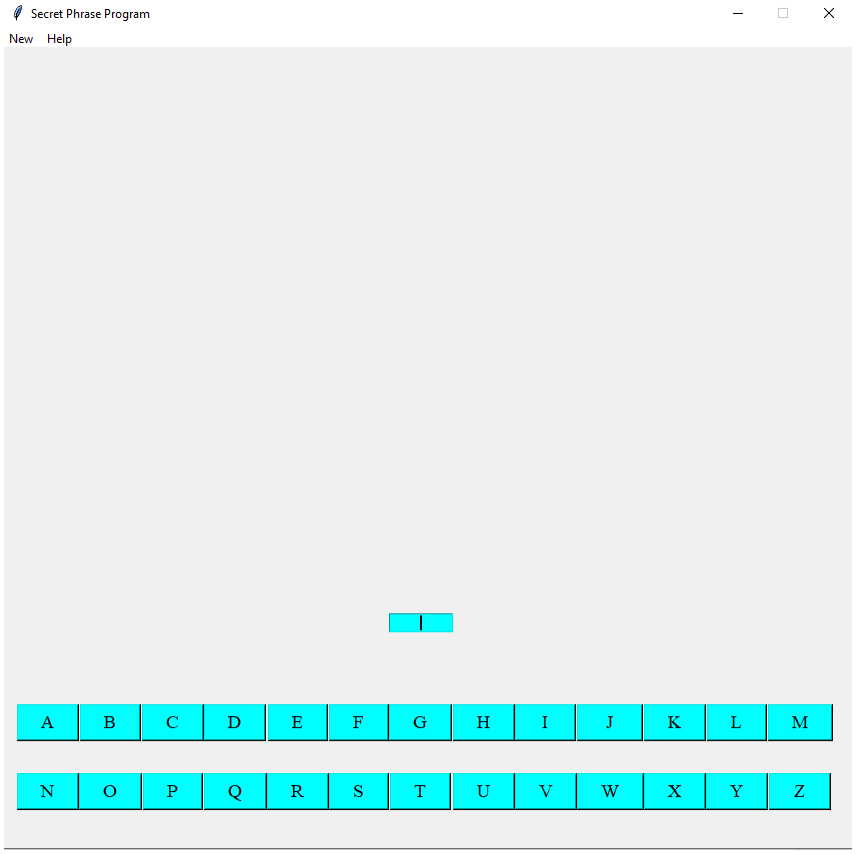


Letters that are not in the word have their buttons disabled. Button color is also displayed respective to the visual ui.



Further demonstration of the button ui



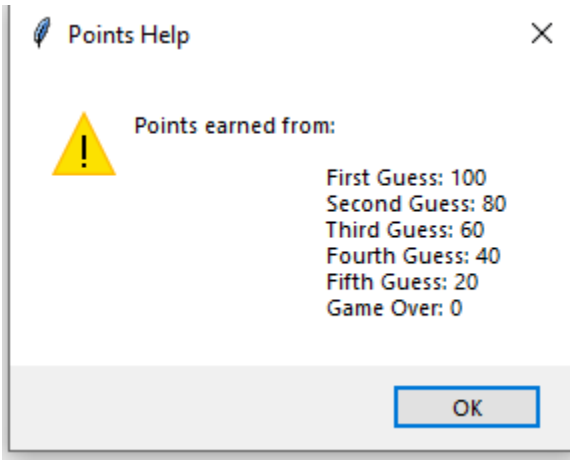


Picking a new word resets the buttons

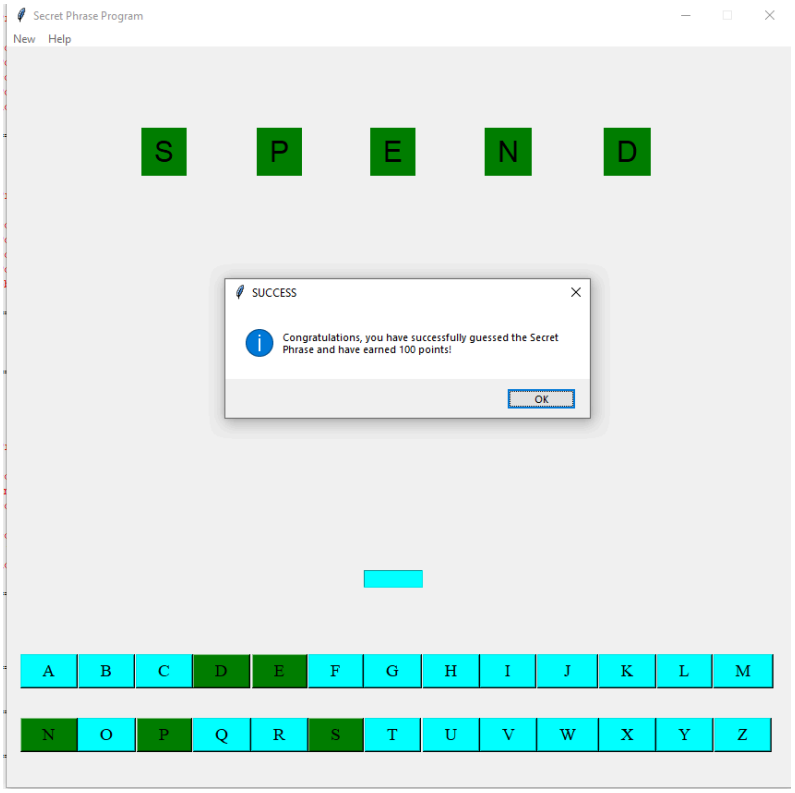
James Robbie
CSE 222-101

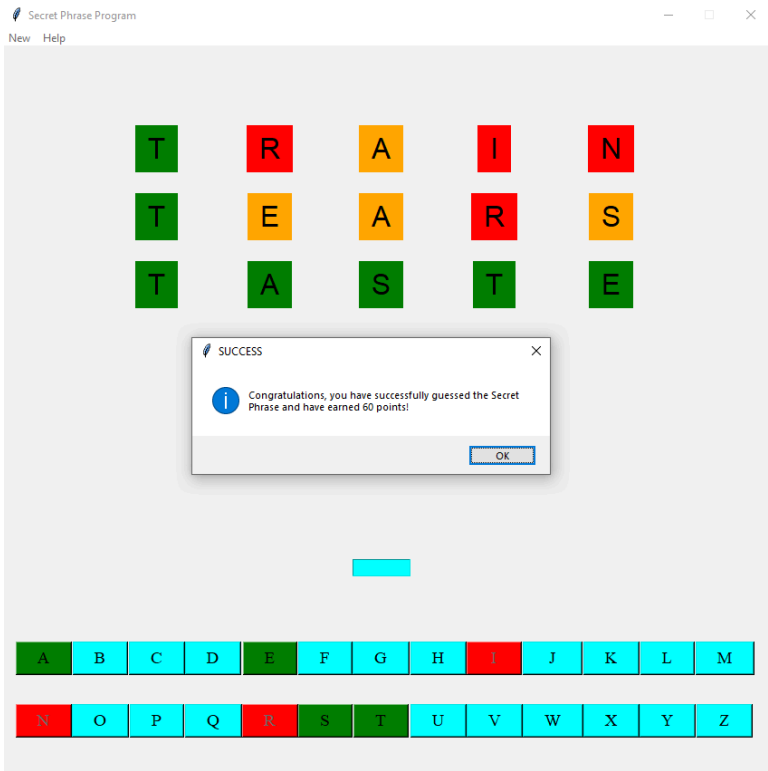
Milestone #5 - 11/30/22

The goal of Milestone #5 is to develop a point system to accommodate the game. The game already has a game display graphic from previous milestones.

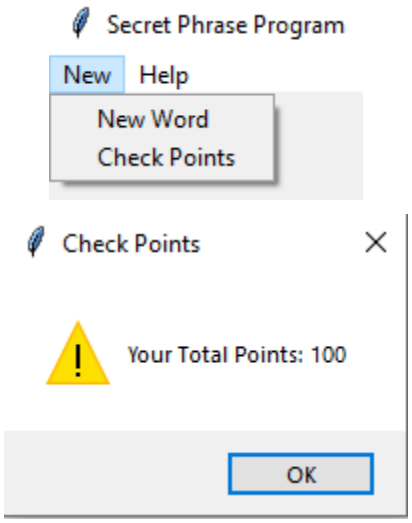


User can click Help > Points Help to learn about the point system





The success window now displays how many points you earned after every round.

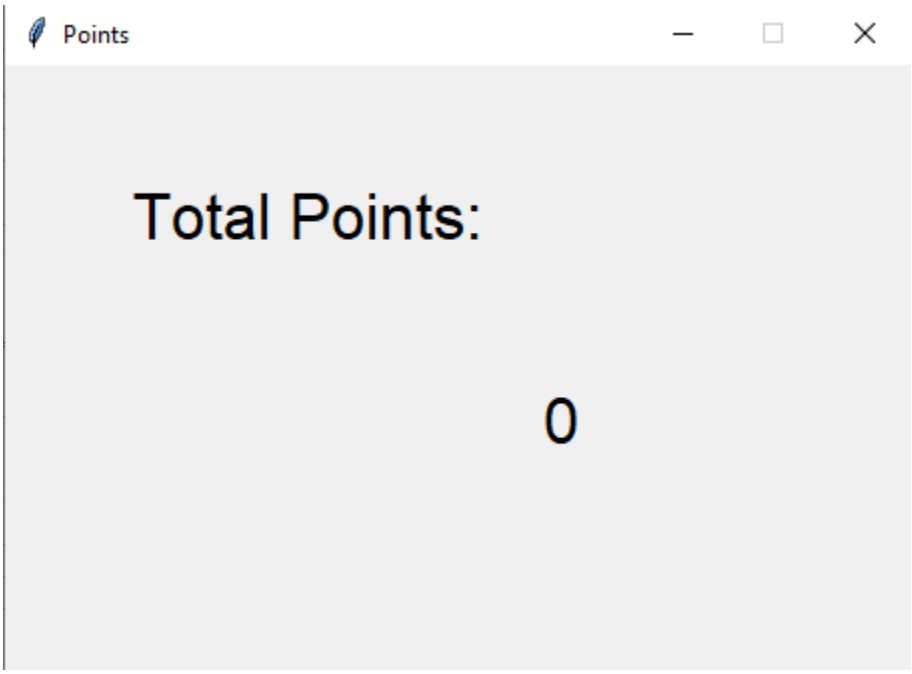


This menu is used to check how many points you have. (Temporary, it will be converted into a window in the next milestone)

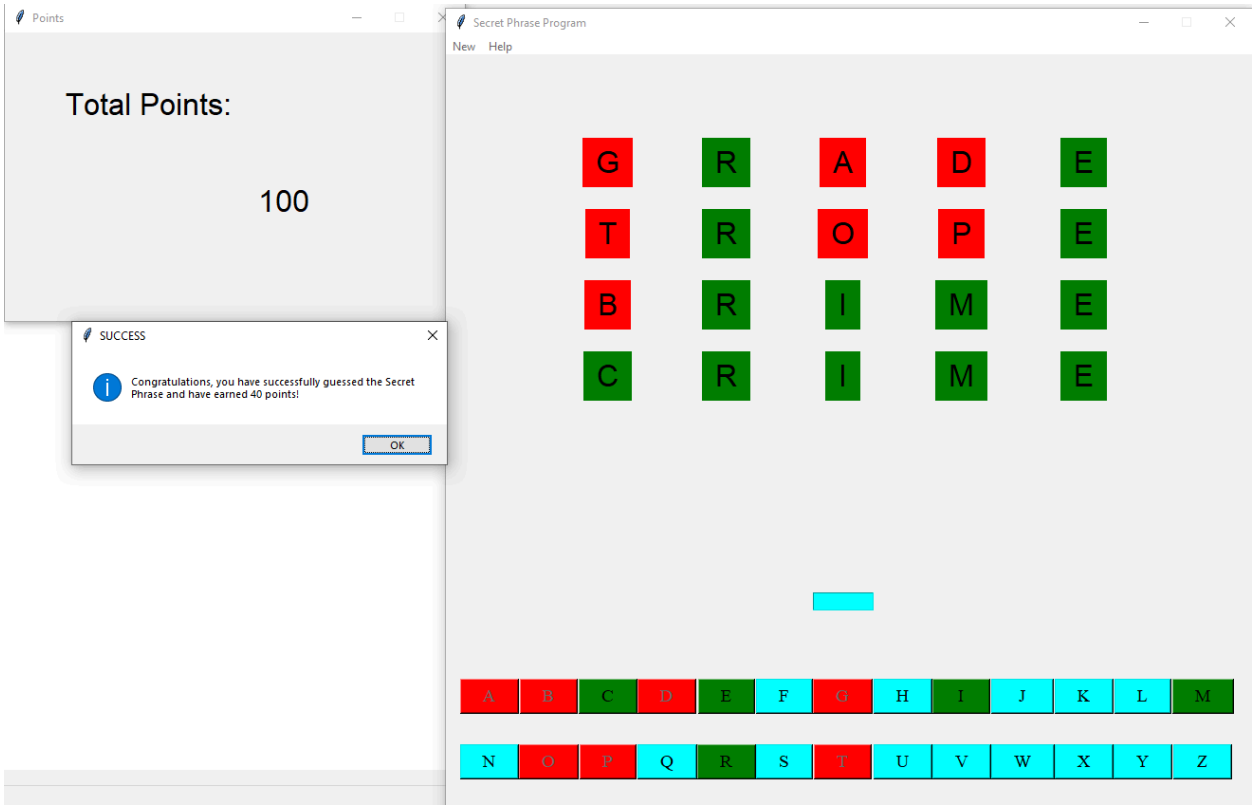
James Robbie
CSE 222-101

Milestone #6 - 12/19/22

The goal of the sixth and final milestone of this project is to create a more user-friendly display of the points they have earned from playing the game. That, and making a few final tweaks to the program.



Points Window was implemented



Points window keeps track of how many points the user has earned

Program Files

 # SPMain.py

```
# Main for Secret Phrase Game Program

import CreateAcctGUI
import CreateLoginGUI
import tkinter as tk

# Main
def main():

    gui = rootGUI() # creates initial gui window

# CreateAcct Window Function
def createAcct():
    print ("Creating CreateAcct Window")

    createAccountGUI = CreateAcctGUI.createAcctGUI()

# Login Window Function
def login():
    print ("Creating Login Window")

    loginGUI = CreateLoginGUI.createLoginGUI()

# Initial GUI Class
class rootGUI:
    def __init__ (self):

        print('Creating RootGUI Window') # Announces creation of window

        self.main_win = tk.Tk() # create the main window
        self.main_win.title("Secret Phrase Game Program") # title bar label
        self.main_win.minsize(width=500,height=250) # window size
        self.main_win.resizable(height = False, width = False) # locks window's width and height

        # Declare column and row sizes
        self.main_win.columnconfigure(0, minsize = 0)
        self.main_win.columnconfigure(1, minsize = 0)
        self.main_win.columnconfigure(2, minsize = 0)
        self.main_win.rowconfigure(0, minsize = 75)
        self.main_win.rowconfigure(1, minsize = 200)
```

```
        self.main_win.rowconfigure(2, minsize = 75)

        # Create Label
        self.heading_label = tk.Label(text='Secret Phrase Game', # label
                                font=("Helvetica Bold", 24), fg="blue") # font, size, and color of label

        self.heading_label.grid(row=0, column=1) # label location


        # Create Cancel Button
        self.cancel_button = tk.Button(text=' Cancel ', command = self.main_win.destroy,\
                                font=("Times", 14)) # declare font
        self.cancel_button.grid(row=2,column=0, padx=20, pady=0, ipadx=15) # declare grid


        # Create CreateAcct Button
        self.createacct_button = tk.Button(text=' Create Account ', command =
lambda:createAcctPress(self),\
                                font=("Times", 14))
        self.createacct_button.grid(row=2,column=1, padx=20, pady=0, ipadx=15) # declare grid

        # Create Login Button
        self.login_button = tk.Button(text=' Login ', command = lambda:loginPress(self),\
                                font=("Times", 14))
        self.login_button.grid(row=2,column=2, padx=20, pady=0, ipadx=15)

        # Executes when Create Account Button is pressed
        def createAcctPress(self):
            self.main_win.destroy()

            createAcct()

        # Executes when Login Button is pressed
        def loginPress(self):
            self.main_win.destroy()

            login()

       # photo = tk.PhotoImage(file = "thought_bubble.jpg")
       # self.labelGIF = tk.Label(image = photo)
       # self.labelGIF.image = photo
       # self.labelGIF.grid(row=1, column=1) # photo location

        tk.mainloop() # the main loop
```

```
main()
```

 # CreateAcctGUI.py

```
# File for Create Account Window
import tkinter as tk
import tkinter.messagebox
import CreateAcctCommands
import SecretPhraseWindow


class createAcctGUI:
    def __init__ (self):

        self.main_win = tk.Tk() # create the main window
        self.main_win.title("Create Account") # title bar label
        self.main_win.minsize(width=500,height=375) # window size
        self.main_win.resizable(height = False, width = False) # locks window's width and height


        # Declare column and row sizes
        self.main_win.columnconfigure(0, minsize = 88)
        self.main_win.columnconfigure(1, minsize = 0)
        self.main_win.columnconfigure(2, minsize = 0)
        self.main_win.rowconfigure(0, minsize = 100)
        self.main_win.rowconfigure(1, minsize = 50)
        self.main_win.rowconfigure(2, minsize = 25)
        self.main_win.rowconfigure(3, minsize = 50)
        self.main_win.rowconfigure(4, minsize = 25)
        self.main_win.rowconfigure(5, minsize = 50)
        self.main_win.rowconfigure(6, minsize = 25)
        self.main_win.rowconfigure(7, minsize = 100)

        # Create Password Requirements Label (ROW 0)
        self.heading_label = tk.Label(text='Password Requires At Least:\n 9 characters\n1 digit,
upper-case letter, and lower-case letter', # label
                                font=("Helvetica Bold", 10), fg="black") # font, size, and color of label

        self.heading_label.grid(row=0, column=1) # label location

        # Create Username Label (ROW 1)
        self.heading_label = tk.Label(text='Create your username (no spaces):', # label
                                font=("Helvetica Bold", 14), fg="black") # font, size, and color of label
```

```
        self.heading_label.grid(row=1, column=1) # label location


        # Create Username TextBox (ROW 2)
        self.username_input = tk.Entry(justify = 'right',
                                       width = 20)
        self.username_input.grid(row=2,column=1, padx=0, pady=0, ipadx=0) # declare grid

        # Create Password Label (ROW 3)
        self.heading_label = tk.Label(text='Create your password:', # label
                                      font=("Helvetica Bold", 14), fg="black") # font, size, and color of label

        self.heading_label.grid(row=3, column=1) # label location

        # Create Password TextBox (ROW 4)
        self.password_input = tk.Entry(justify = 'right', show="*",
                                       width = 20)
        self.password_input.grid(row=4,column=1, padx=0, pady=0, ipadx=0) # declare grid

        # Create Confirm Password Label (ROW 5)
        self.heading_label = tk.Label(text='Confirm password:', # label
                                      font=("Helvetica Bold", 14), fg="black") # font, size, and color of label

        self.heading_label.grid(row=5, column=1) # label location

        # Create Confirm Password TextBox (ROW 6)
        self.confirm_password_input = tk.Entry(justify = 'right', show="*",
                                       width = 20)
        self.confirm_password_input.grid(row=6,column=1, padx=0, pady=0, ipadx=0) # declare
grid

        # Create CreateAcct Button
        self.login_button = tk.Button(text=' Create Account ', command =
lambda:createAcctPress(self),\
                                      font=("Times", 12))
        self.login_button.grid(row=7,column=1, padx=20, pady=0, ipadx=15)

        # Create Account Button function
        def createAcctPress(self):

            # Writes credentials to string
            username = self.username_input.get()
            password = self.password_input.get()
```

```
        confirm_password = self.confirm_password_input.get()

        # Tests if Password matches confirm password
        if (password == confirm_password):

            isUnique = CreateAcctCommands.createAcctIsUnique(username) # Calls function to
test if Username is unique
            isValid = CreateAcctCommands.createAcctIsValid(username, password) # Calls
function to test if Username and Password are valid

            if (isUnique == True and isValid == True):
                CreateAcctCommands.createAcctFile(username, password) # Calls function to
Write Username and Password to File

                tk.messagebox.showinfo("Success!", 'Account was successfully created!') #
Success dialogue box

                self.main_win.destroy() # Destroys window

                SecretPhraseWindow.secretPhraseWindow() # Creates Secret Phrase Game

            elif (isUnique == False):

                # Deletes input from Text Boxes
                self.username_input.delete(0, tk.END)
                self.password_input.delete(0, tk.END)
                self.confirm_password_input.delete(0, tk.END)

                tk.messagebox.showwarning("Invalid Credentials", 'Username is already in use') #
Error dialogue box for Non Unique User

            elif (isValid == False):

                # Deletes input from Text Boxes
                self.username_input.delete(0, tk.END)
                self.password_input.delete(0, tk.END)
                self.confirm_password_input.delete(0, tk.END)

                tk.messagebox.showwarning("Invalid Credentials", 'Username or password is
invalid') # Error dialogue box for Invalid Credentials

        else:

            # Deletes input from Text Boxes
```

```
            self.username_input.delete(0, tk.END)
            self.password_input.delete(0, tk.END)
            self.confirm_password_input.delete(0, tk.END)

            tk.messagebox.showwarning("Invalid Credentials", 'Password does not match
Confirm Password') # Error dialogue box

        self.main_win.bind('<Return>', (lambda event:createAcctPress(self)))

        tk.mainloop() # the main loop

# test function
# gui = createAcctGUI()
```

 # CreateLoginGUI.py

```
# File for Login Window
import tkinter as tk
import LoginCommands
import SecretPhraseWindow

class createLoginGUI:
    def __init__ (self):

        self.main_win = tk.Tk() # create the main window
        self.main_win.title("Login") # title bar label
        self.main_win.minsize(width=500,height=325) # window size
        self.main_win.resizable(height = False, width = False) # locks window's width and height

        # Declare column and row sizes
        self.main_win.columnconfigure(0, minsize = 110)
        self.main_win.columnconfigure(1, minsize = 0)
        self.main_win.columnconfigure(2, minsize = 0)
        self.main_win.rowconfigure(0, minsize = 75)
        self.main_win.rowconfigure(1, minsize = 50)
        self.main_win.rowconfigure(2, minsize = 25)
        self.main_win.rowconfigure(3, minsize = 50)
        self.main_win.rowconfigure(4, minsize = 25)
        self.main_win.rowconfigure(5, minsize = 150)

        # Create Enter Username/Password Label (ROW 0)
        self.heading_label = tk.Label(text='Please enter your username and password.', # label
                                      font=("Helvetica Bold", 10), fg="black") # font, size, and color of label
```

```
        self.heading_label.grid(row=0, column=1) # label location

        # Create Username Label (ROW 1)
        self.heading_label = tk.Label(text='Username:', anchor = 'center', # label
                                font=("Helvetica Bold", 14), fg="black") # font, size, and color of label

        self.heading_label.grid(row=1, column=1) # label location

        # Create Username TextBox (ROW 2)
        self.username_input = tk.Entry(justify = 'right',
                                width = 20)

        self.username_input.grid(row=2,column=1, padx=0, pady=0, ipadx=0) # declare grid

        # Create Password Label (ROW 3)
        self.heading_label = tk.Label(text='Password:', # label
                                font=("Helvetica Bold", 14), fg="black") # font, size, and color of label

        self.heading_label.grid(row=3, column=1) # label location

        # Create Password TextBox (ROW 4)
        self.password_input = tk.Entry(justify = 'right', show="*",
                                width = 20)

        #self.password_input.config(show='*')
        self.password_input.grid(row=4,column=1, padx=0, pady=0, ipadx=0) # declare grid

        # Create Login Button (ROW 5)
        self.login_button = tk.Button(text=' Login ', command = lambda:loginPress(self),\
                                  font=("Times", 12))
        self.login_button.grid(row=5,column=1, padx=0, pady=0, ipadx=15)

        # Create Login Button function
        def loginPress(self):

            username = self.username_input.get()
            password = self.password_input.get()

            validLogin = LoginCommands.loginIsValid(username, password)

            if (validLogin == True):

                tk.messagebox.showinfo("Success!", 'Successful Login! Proceeding to program...') #
Success dialogue box
```

```
            self.main_win.destroy() # destroy Login Window

            SecretPhraseWindow.secretPhraseWindow() # Creates Secret Phrase Game

        else:
            # Deletes input from Text Boxes
            self.username_input.delete(0, tk.END)
            self.password_input.delete(0, tk.END)

            tk.messagebox.showwarning("Error", 'Invalid Login') # Error dialogue box for Non
Unique User

    #self.main_win.bind('<Return>', self.loginPress(self))

    self.main_win.bind('<Return>', (lambda event:loginPress(self)))

    tk.mainloop() # the main loop

# test function
#gui = createLoginGUI()
```

 # CreateAcctCommands.py

```
def createAcctIsUnique(user):

    try:
        isUnique = True

        credentials_file = open('credentials.txt', 'r')

        # Declare Line Counting Variable
        lineCount = 0

        # Read every line in file
        for line_of_text in credentials_file:

            lineNoSpace = line_of_text.strip() # remove White Space

            if (lineCount % 2 == 0): # Checks if Line is a Username Line (Even)
                if (lineNoSpace == user): # Checks if Username is already on file
                    isUnique = False # Returns false is Username is already used

            lineCount += 1 # Keeps count of Lines
```

```
        credentials_file.close # Closes file

        return isUnique

    except IOError:
        print('No File exists.')

def createAcctIsValid(user, pswd):

    isValid = False

    # Declare Username Validators
    userHasWhiteSpace = False

    # Declare Password Validators
    pswdHasWhiteSpace = False
    pswdHasNineChars = False
    pswdHasDigit = False
    pswdHasUpper = False
    pswdHasLower = False

    # Tests if username has White Space
    for temp in user:
        if (temp.isspace()):
            userHasWhiteSpace = True

    # Count for Password Validation
    pswdCount = 0

    # Tests if Password is Valid (9 chars, 1 Upper, 1 Lower, 1 Digit)
    for temp in pswd:

        pswdCount += 1

        if (temp.isdigit()): # checks for digit in password
            pswdHasDigit = True

        if (temp.isupper()): # checks for upper case letter in password
            pswdHasUpper = True

        if (temp.islower()): # checks for lower case letter in password
            pswdHasLower = True
```

```
        if (temp.isspace()): # checks for space in password
            pswdHasWhiteSpace = True

        if (pswdCount >= 9): # checks if password is 9 chars long
            pswdHasNineChars = True

    # If every Validation is Passed, IsValid is returned as True
    if (userHasWhiteSpace == False and pswdHasWhiteSpace == False and pswdHasNineChars
== True and pswdHasDigit == True and pswdHasUpper == True and pswdHasLower == True):
        isValid = True
    else:
        isValid = False

    return isValid

def createAcctFile(user, pswd):

    try:
        credentials_file = open('credentials.txt', 'a')

        credentials_file.write(user)
        credentials_file.write("\n")
        credentials_file.write(pswd)
        credentials_file.write("\n")

        credentials_file.close

    except IOError:
        print('No File exists.')

#int = createAcctIsUnique("Username")
#print(int)
#int = createAcctIsValid("Userna888me", "Pas888sword")
#print(int)
#createAcctFile("Username", "Password")
```

 # LoginCommands.py

```
def loginIsValid(user, pswd):

    try:
        isValid = False

        credentials_file = open('credentials.txt', 'r') # Open file
```

```
        # Declare line counting variable
        lineCount = 0
        usernameFound = False

        for line_of_text in credentials_file:

            lineNoSpace = line_of_text.strip() #Removes white space

            if (lineCount % 2 == 0): #checks if line is a username line (odd #)
                if (lineNoSpace == user): # checks if line is equal to entered username
                    usernameFound = True
                    passwordLine = lineCount + 1 # writes the password line as the line after found
username

            if (usernameFound == True and lineCount == passwordLine): # Checks if password
matches username
                if (lineNoSpace == pswd):
                    isValid = True

            lineCount += 1 # Update line count

        credentials_file.close # Close file

        return isValid


    except IOError:
        print('No file exists.')

#loginValid = loginIsValid("Username", "Password")
#print(loginValid)
```

#SecretPhraseWindow.py

```
import tkinter as tk
import tkinter.messagebox
import SecretPhraseFunctions

class secretPhraseWindow:
    def __init__ (self):

        # MAIN WINDOW
        self.main_win = tk.Tk() # create the main window
```

```
self.main_win.title("Secret Phrase Program") # title bar label
self.main_win.minsize(width=850,height=800) # window size
self.main_win.resizable(height = False, width = False) # locks window's width and height

# Finds the starting random word
self.randomWord = SecretPhraseFunctions.randomWord()

# Create points variable
self.totalPoints = 0

# Row Configuration
for c in range(14):
    self.main_win.columnconfigure(c, minsize = 15)

for r in range(7):
    self.main_win.rowconfigure(r, minsize = 75)

self.main_win.rowconfigure(7, minsize = 100)
self.main_win.rowconfigure(8, minsize = 100)
self.main_win.rowconfigure(9, minsize = 5)

# Create New Menu Bar
self.menu_bar = tk.Menu()

# Create New Menu
newmenu = tk.Menu(self.menu_bar, tearoff=0)
newmenu.add_command(label="New Word", command = lambda:newWord(self))
newmenu.add_command(label="Check Points", command = lambda:checkPoints(self))
self.menu_bar.add_cascade(label="New", menu=newmenu)

# Create Help Menu
helpmenu = tk.Menu(self.menu_bar, tearoff=0)
helpmenu.add_command(label="Color Legend", command = lambda:colorLegend(self))
helpmenu.add_command(label="Help", command = lambda:help(self))
helpmenu.add_command(label="Points Help", command = lambda:pointsHelp(self))
self.menu_bar.add_cascade(label="Help", menu=helpmenu)

self.main_win.config(menu=self.menu_bar)

# Create Entry Box
entry_word = ""

self.entry_box = tk.Entry(width=10, bg="cyan", justify="center", textvariable=entry_word)
self.entry_box.grid(row=7, column=7, pady=20)
```

```
# A Button
self.a_button = tk.Button(text=' A ', bg = "cyan", command = lambda:insertLetter(self, 'A'),\
                          font=("Times", 14))
self.a_button.grid(row=8,column=1, pady=0, ipadx=12)

# B Button
self.b_button = tk.Button(text=' B ', bg = "cyan", command = lambda:insertLetter(self, 'B'),\
                          font=("Times", 14))
self.b_button.grid(row=8,column=2, pady=0, ipadx=12)

# C Button
self.c_button = tk.Button(text=' C ', bg = "cyan", command = lambda:insertLetter(self, 'C'),\
                          font=("Times", 14))
self.c_button.grid(row=8,column=3, pady=0, ipadx=12)

# D Button
self.d_button = tk.Button(text=' D ', bg = "cyan", command = lambda:insertLetter(self, 'D'),\
                          font=("Times", 14))
self.d_button.grid(row=8,column=4, pady=0, ipadx=12)

# E Button
self.e_button = tk.Button(text=' E ', bg = "cyan", command = lambda:insertLetter(self, 'E'),\
                          font=("Times", 14))
self.e_button.grid(row=8,column=5, pady=0, ipadx=12)

# F Button
self.f_button = tk.Button(text=' F ', bg = "cyan", command = lambda:insertLetter(self, 'F'),\
                          font=("Times", 14))
self.f_button.grid(row=8,column=6, pady=0, ipadx=12)

# G Button
self.g_button = tk.Button(text=' G ', bg = "cyan", command = lambda:insertLetter(self, 'G'),\
                          font=("Times", 14))
self.g_button.grid(row=8,column=7, pady=0, ipadx=12)

# H Button
self.h_button = tk.Button(text=' H ', bg = "cyan", command = lambda:insertLetter(self, 'H'),\
                          font=("Times", 14))
self.h_button.grid(row=8,column=8, pady=0, ipadx=12)

# I Button
self.i_button = tk.Button(text=' I ', bg = "cyan", command = lambda:insertLetter(self, 'I'),\
                          font=("Times", 14))
```

```
self.i_button.grid(row=8,column=9, pady=0, ipadx=15)

# J Button
self.j_button = tk.Button(text=' J ', bg = "cyan", command = lambda:insertLetter(self, 'J'),\
                          font=("Times", 14))
self.j_button.grid(row=8,column=10, pady=0, ipadx=17)

# K Button
self.k_button = tk.Button(text=' K ', bg = "cyan", command = lambda:insertLetter(self, 'K'),\
                          font=("Times", 14))
self.k_button.grid(row=8,column=11, pady=0, ipadx=12)

# L Button
self.l_button = tk.Button(text=' L ', bg = "cyan", command = lambda:insertLetter(self, 'L'),\
                          font=("Times", 14))
self.l_button.grid(row=8,column=12, pady=0, ipadx=12)

# M Button
self.m_button = tk.Button(text=' M ', bg = "cyan", command = lambda:insertLetter(self, 'M'),\
                          font=("Times", 14))
self.m_button.grid(row=8,column=13, pady=0, ipadx=12)

# N Button
self.n_button = tk.Button(text=' N ', bg = "cyan", command = lambda:insertLetter(self, 'N'),\
                          font=("Times", 14))
self.n_button.grid(row=9,column=1, pady=0, ipadx=12)

# O Button
self.o_button = tk.Button(text=' O ', bg = "cyan", command = lambda:insertLetter(self, 'O'),\
                          font=("Times", 14))
self.o_button.grid(row=9,column=2, pady=0, ipadx=12)

# P Button
self.p_button = tk.Button(text=' P ', bg = "cyan", command = lambda:insertLetter(self, 'P'),\
                          font=("Times", 14))
self.p_button.grid(row=9,column=3, pady=0, ipadx=12)

# Q Button
self.q_button = tk.Button(text=' Q ', bg = "cyan", command = lambda:insertLetter(self, 'Q'),\
                          font=("Times", 14))
self.q_button.grid(row=9,column=4, pady=0, ipadx=12)

# R Button
self.r_button = tk.Button(text=' R ', bg = "cyan", command = lambda:insertLetter(self, 'R'),\
```

```
                font=("Times", 14))
self.r_button.grid(row=9,column=5, pady=0, ipadx=12)

# S Button
self.s_button = tk.Button(text=' S ', bg = "cyan", command = lambda:insertLetter(self, 'S'),\
                font=("Times", 14))
self.s_button.grid(row=9,column=6, pady=0, ipadx=12)

# T Button
self.t_button = tk.Button(text=' T ', bg = "cyan", command = lambda:insertLetter(self, 'T'),\
                font=("Times", 14))
self.t_button.grid(row=9,column=7, pady=0, ipadx=12)

# U Button
self.u_button = tk.Button(text=' U ', bg = "cyan", command = lambda:insertLetter(self, 'U'),\
                font=("Times", 14))
self.u_button.grid(row=9,column=8, pady=0, ipadx=12)

# V Button
self.v_button = tk.Button(text=' V ', bg = "cyan", command = lambda:insertLetter(self, 'V'),\
                font=("Times", 14))
self.v_button.grid(row=9,column=9, pady=0, ipadx=12)

# W Button
self.w_button = tk.Button(text=' W ', bg = "cyan", command = lambda:insertLetter(self, 'W'),\
                font=("Times", 14))
self.w_button.grid(row=9,column=10, pady=0, ipadx=12)

# X Button
self.x_button = tk.Button(text=' X ', bg = "cyan", command = lambda:insertLetter(self, 'X'),\
                font=("Times", 14))
self.x_button.grid(row=9,column=11, pady=0, ipadx=12)

# Y Button
self.y_button = tk.Button(text=' Y ', bg = "cyan", command = lambda:insertLetter(self, 'Y'),\
                font=("Times", 14))
self.y_button.grid(row=9,column=12, pady=0, ipadx=12)

# Z Button
self.z_button = tk.Button(text=' Z ', bg = "cyan", command = lambda:insertLetter(self, 'Z'),\
                font=("Times", 14))
self.z_button.grid(row=9,column=13, pady=0, ipadx=13)

# POINTS WINDOW
```

```
        self.points_win = tk.Tk() # create the main window
        self.points_win.title("Points") # title bar label
        self.points_win.minsize(width=300,height=300) # window size
        self.points_win.resizable(height = False, width = False) # locks window's width and height

        # Create rows and Columns
        for r in range(6):
            self.points_win.rowconfigure(r, minsize = 50)

        for c in range(6):
            self.points_win.columnconfigure(c, minsize = 50)

        # Create label for the total points
        self.total_points_label = tk.Label(self.points_win, text = "Total Points:", padx=12, pady=6,
font=("Helvetica", 24))

        # Attach to grid
        self.total_points_label.grid(row=1, column=1)

        # Convert total points to string
        self.strTotalPoints = str(self.totalPoints)

        # Create the display of the total points
        self.total_points = tk.Label(self.points_win, text = self.strTotalPoints, padx=12, pady=6,
font=("Helvetica Bold", 24))

        # Attach to grid
        self.total_points.grid(row=3, column=2)

        # Function for button presses
        def insertLetter(self, letter):

            self.entry_box.insert(100, letter)

        # Function for when the user enters input
        def entryEnter(self):

            user_entry = self.entry_box.get() # grabs the user's input

            user_entry = user_entry.upper() # converts users guess to uppercase

            self.entry_box.delete(0, 100) # deletes the input in the entry box

            # If the user has guesses left
```

```
        if (self.row_num < 6):

            # If the entry is 5 chars
            if (len(user_entry) == 5):
                checkWord(self, user_entry)

            else :
                tk.messagebox.showwarning("Invalid Input", 'Please enter a 5 letter word') # Tells
user their input is invalid

        else:
            tk.messagebox.showwarning("Game Over", 'You are out of guesses! Select "New
Word" when you are ready to continue!') # Error dialogue box for Non Unique User

    # Bind the Enter Key to the Window
    self.main_win.bind('<Return>', (lambda event:entryEnter(self)))

    self.row_num = 1

    # Checks if Guess matches the Word and displays it on the Window
    def checkWord(self, entry):

        # Call Letter Count and Column Num Variables
        letter_count = 0
        column_num = 3

        # Checks each letter in the entry
        for letter in entry:

            # Call background color variable
            background = ""

            # Correct Letter : Green Background
            if (letter == self.randomWord[letter_count]):
                background = "green"

                # Changes the Button Background to Green
                changeButton(self, letter, 'normal', background, 'false')

            # Letter is in word but not in right place : Orange Background
            elif (letter == self.randomWord[0] or letter == self.randomWord[1] or letter ==
self.randomWord[2] or letter == self.randomWord[3] or letter == self.randomWord[4]) :
                background = "orange"
```

```
                # Changes the Button Background to Orange
                changeButton(self, letter, 'normal', background, 'false')

            # Letter is not in word : Red Background
            else :
                background = "red"

                changeButton(self, letter, 'disabled', background, 'false')

            # Creates the Label for the letter
            self.word_char = tk.Label(text = letter, bg = background, padx=12, pady=6,
font=("Helvetica Bold", 24))

            self.word_char.grid(row=self.row_num, column=column_num)

            # Adds iteration to the Letter and Column num
            letter_count += 1
            column_num += 2

        if (self.randomWord == entry):

            # Points algorithm (Start out with 100 points and for each guess deduct 20
            guess_deduction = 20 * (self.row_num - 1)

            round_points = 100 - guess_deduction

            # Add the round points to total points
            self.totalPoints += round_points

            # Add the points to the points window
            strTotalPoints = str(self.totalPoints)
            self.total_points.config(text=strTotalPoints)

            # Convert round points to string for message box
            strRoundPoints = str(round_points)

            # Success message box
            tk.messagebox.showinfo("SUCCESS", 'Congratulations, you have successfully
guessed the Secret Phrase and have earned ' + strRoundPoints + ' points!')

        elif (self.row_num > 4):
            tk.messagebox.showwarning("Game Over", 'You are out of guesses! Select "New
Word" when you are ready to continue!') # Error dialogue box for Non Unique User
```

```
    self.row_num += 1

# For the New Word Menu
def newWord(self):

    # Creates a new random word
    self.randomWord = SecretPhraseFunctions.randomWord()

    # Resets the row number for the Guess Labels
    self.row_num = 1

    # Delets all labels (guesses)
    for widget in self.main_win.winfo_children():
        if isinstance(widget, tk.Label):
            widget.destroy()

    # Resets all buttons
    changeButton(self, 'A', 'normal', 'cyan', 'true')

# For the New Word Menu, Check points
def checkPoints(self):

    # Converts total points to string for message box
    strTotalPoints = str(self.totalPoints)

    tk.messagebox.showwarning("Check Points", 'Your Total Points: ' + strTotalPoints)


# Changes the button color or disables/enables it
def changeButton(self, letter, state, bg, reset):

    if (letter == "A" or reset == 'true'):
        self.a_button['state'] = state
        self.a_button['bg'] = bg

    if (letter == "B" or reset == 'true'):
        self.b_button['state'] = state
        self.b_button['bg'] = bg

    if (letter == "C" or reset == 'true'):
        self.c_button['state'] = state
        self.c_button['bg'] = bg

    if (letter == "D" or reset == 'true'):
```

```
    self.d_button['state'] = state
    self.d_button['bg'] = bg

if (letter == "E" or reset == 'true'):
    self.e_button['state'] = state
    self.e_button['bg'] = bg

if (letter == "F" or reset == 'true'):
    self.f_button['state'] = state
    self.f_button['bg'] = bg

if (letter == "G" or reset == 'true'):
    self.g_button['state'] = state
    self.g_button['bg'] = bg

if (letter == "H" or reset == 'true'):
    self.h_button['state'] = state
    self.h_button['bg'] = bg

if (letter == "I" or reset == 'true'):
    self.i_button['state'] = state
    self.i_button['bg'] = bg

if (letter == "J" or reset == 'true'):
    self.j_button['state'] = state
    self.j_button['bg'] = bg

if (letter == "K" or reset == 'true'):
    self.k_button['state'] = state
    self.k_button['bg'] = bg

if (letter == "L" or reset == 'true'):
    self.l_button['state'] = state
    self.l_button['bg'] = bg

if (letter == "M" or reset == 'true'):
    self.m_button['state'] = state
    self.m_button['bg'] = bg

if (letter == "N" or reset == 'true'):
    self.n_button['state'] = state
    self.n_button['bg'] = bg

if (letter == "O" or reset == 'true'):
```

```
    self.o_button['state'] = state
    self.o_button['bg'] = bg

if (letter == "P" or reset == 'true'):
    self.p_button['state'] = state
    self.p_button['bg'] = bg

if (letter == "Q" or reset == 'true'):
    self.q_button['state'] = state
    self.q_button['bg'] = bg

if (letter == "R" or reset == 'true'):
    self.r_button['state'] = state
    self.r_button['bg'] = bg

if (letter == "S" or reset == 'true'):
    self.s_button['state'] = state
    self.s_button['bg'] = bg

if (letter == "T" or reset == 'true'):
    self.t_button['state'] = state
    self.t_button['bg'] = bg

if (letter == "U" or reset == 'true'):
    self.u_button['state'] = state
    self.u_button['bg'] = bg

if (letter == "V" or reset == 'true'):
    self.v_button['state'] = state
    self.v_button['bg'] = bg

if (letter == "W" or reset == 'true'):
    self.w_button['state'] = state
    self.w_button['bg'] = bg

if (letter == "X" or reset == 'true'):
    self.x_button['state'] = state
    self.x_button['bg'] = bg

if (letter == "Y" or reset == 'true'):
    self.y_button['state'] = state
    self.y_button['bg'] = bg

if (letter == "Z" or reset == 'true'):
```

```
            self.z_button['state'] = state
            self.z_button['bg'] = bg


        # For the help selection
        def help(self):

            tk.messagebox.showwarning("Help", 'Welcome to the Secret Phrase Game, \n\nThe
main goal of the game is to guess a 5 letter word in 5 guesses\n\nInput a guess by using the
buttons or typing it and press ENTER\n\nEach guess will provide clues to what letters are in the
word \n\nIf you need help on the clues, please refer to the color legend') # Error dialogue box for
Non Unique User

        # For the point help selection
        def pointsHelp(self):

            tk.messagebox.showwarning("Points Help", 'Points earned from:\n\n\t\tFirst Guess: 100
\n\t\tSecond Guess: 80 \n\t\tThird Guess: 60 \n\t\tFourth Guess: 40 \n\t\tFifth Guess: 20
\n\t\tGame Over: 0')

        # For the Color Legend Selection
        def colorLegend(self):

            SecretPhraseFunctions.colorLegendWindow()

        tk.mainloop()

#secretPhraseWindow()
```

 # SecretPhraseFunctions

```
import random
import tkinter as tk

def randomWord():

    try:
        word_file = open('wordlist.txt', 'r') #open file

        # Gets the amount of Words in File
        wordFileSize = lineCounter() - 1

        # Returns random number
        randomWordLine = random.randint (0, wordFileSize)
```

```
        # Line Count Variable
        lineCount = 0

        # Random Word Variable
        randomWord = ''

        # Read file
        for line_of_text in word_file:

            # Gets the word on the random number's line
            if (lineCount == randomWordLine):
                randomWord = line_of_text.strip()

            lineCount += 1


        word_file.close #close file

        return randomWord


    except IOError:
        print('No file exists.')

def lineCounter():

    try:
        word_file = open('wordlist.txt', 'r') #open file

        # Line Count Variable
        lineCount = 0

        # Counts the number of lines
        for line_of_text in word_file:
            lineCount += 1

        word_file.close #close file

        return lineCount


    except IOError:
        print('No file exists.')
```

```
class colorLegendWindow():
    def __init__ (self):

        # Standard window configuration
        self.color_win = tk.Tk()
        self.color_win.title("Color Legend")
        self.color_win.minsize(width=850, height=400)
        self.color_win.resizable(height = False, width = False)

        # Create rows and Columns
        for r in range(3):
            self.color_win.rowconfigure(r, minsize = 75)

        for c in range(14):
            self.color_win.columnconfigure(c, minsize = 15)

        # Displays the example word
        self.word_char = tk.Label(self.color_win, text = "R", bg = "red", padx=12, pady=6,
font=("Helvetica Bold", 24))

        self.word_char.grid(row=1, column=5)

        self.word_char = tk.Label(self.color_win, text = "O", bg = "green", padx=12, pady=6,
font=("Helvetica Bold", 24))

        self.word_char.grid(row=1, column=7)

        self.word_char = tk.Label(self.color_win, text = "U", bg = "green", padx=12, pady=6,
font=("Helvetica Bold", 24))

        self.word_char.grid(row=1, column=9)

        self.word_char = tk.Label(self.color_win, text = "N", bg = "orange", padx=12, pady=6,
font=("Helvetica Bold", 24))

        self.word_char.grid(row=1, column=11)

        self.word_char = tk.Label(self.color_win, text = "D", bg = "red", padx=12, pady=6,
font=("Helvetica Bold", 24))

        self.word_char.grid(row=1, column=13)

        # Create the explanation label
```

```
    self.info = tk.Label(self.color_win, text = "Red means the letter is not in the
word;\n\nOrange means the letter is in the word but not in the right position;\n\nGreen means
the letter is in the word and in the correct position", padx=12, pady=6, font=("Helvetica Bold", 9))

    self.info.grid(row=2, column=14)


    tk.mainloop()
```

 # END OF PROGRAM

Word List

wordlist.txt (the phrases that the program reads from)

```
ABOVE
ABUSE
ACUTE
ADMIT
ADOPT
ADULT
AGENT
AGREE
ALIVE
ALLOW
ALONE
ALTER
ANGER
ANGRY
APPLE
APPLY
ARGUE
ARISE
AVOID
AWARD
AWARE
AWFUL
BASIC
BASIS
BEACH
BEGIN
BIRTH
BLACK
BLAME
```

BLIND
BLOCK
BLOOD
BOARD
BRAIN
BRAVE
BREAD
BREAK
BREAK
BRIEF
BRING
BROAD
BROWN
BUILD
BURST
BUYER
CARRY
CATCH
CAUSE
CAUSE
CHAIN
CHAIR
CHEAP
CHECK
CHEST
CHIEF
CHIEF
CHILD
CHINA
CIVIL
CLAIM
CLAIM
CLASS
CLEAN
CLEAN
CLEAR
CLEAR
CLIMB
CLOCK
CLOSE
CLOSE
COACH
COAST
COUNT

COURT
COVER
COVER
CRAZY
CREAM
CRIME
CROSS
CROSS
CROWD
CROWN
CYCLE
DAILY
DANCE
DEATH
DEPTH
DIRTY
DOUBT
DOUBT
DRAFT
DRAMA
DREAM
DRESS
DRINK
DRIVE
EARLY
EARTH
EMPTY
ENEMY
ENJOY
ENTER
ENTRY
EQUAL
ERROR
EVENT
EXACT
EXIST
EXTRA
FAINT
FAITH
FALSE
FAULT
FIELD
FIFTH
FIGHT

FIGHT
FINAL
FINAL
FIRST
FLOOR
FOCUS
FOCUS
FORCE
FORCE
FRAME
FRANK
FRESH
FRONT
FRONT
FRUIT
FUNNY
GIANT
GLASS
GRAND
GRANT
GRASS
GREAT
GREEN
GREEN
GROSS
GROUP
GUESS
GUIDE
HAPPY
HARSH
HEART
HEAVY
HORSE
HOTEL
HOUSE
HUMAN
IDEAL
IMAGE
IMPLY
INDEX
INNER
INPUT
ISSUE
JAPAN

JOINT
JONES
JUDGE
KNIFE
LARGE
LAUGH
LAURA
LAYER
LEARN
LEAVE
LEGAL
LETS
LEVEL
LIGHT
LIMIT
LOCAL
LOOSE
LUCKY
LUNCH
MAGIC
MAJOR
MARCH
MARRY
MATCH
MATCH
METAL
MINOR
MODEL
MONEY
MONTH
MORAL
MOTOR
MOUTH
MUSIC
NASTY
NAVAL
NERDY
NIGHT
NOISE
NORTH
NOVEL
NURSE
OCCUR
OFFER

ORDER
OTHER
OUTER
OWNER
PANEL
PAPER
PARTY
PEACE
PHASE
PHONE
PIECE
PILOT
PITCH
PLACE
PLAIN
PLANE
PLANT
PLATE
POINT
POUND
POWER
PRESS
PRICE
PRIDE
PRIME
PRIOR
PRIZE
PROOF
PROUD
PROVE
QUACK
QUEEN
QUICK
QUIET
RADIO
RAISE
RANGE
RAPID
RATIO
REACH
READY
REFER
RELAX
REPLY

RIGHT
RIVER
ROMAN
ROUGH
ROUND
ROUTE
ROYAL
RUGBY
RURAL
SCALE
SCENE
SCOPE
SCORE
SENSE
SERVE
SHALL
SHAPE
SHARE
SHARE
SHARP
SHEEP
SHEER
SHEET
SHIFT
SHIRT
SHOCK
SHOOT
SHORT
SIGHT
SILLY
SIXTH
SKILL
SLEEP
SLEEP
SMALL
SMART
SMILE
SMITH
SMOKE
SOLID
SOLVE
SORRY
SOUND
SOUND

SOUTH
SPACE
SPARE
SPEAK
SPEED
SPEND
SPITE
SPLIT
SPORT
SQUAD
STAFF
STAGE
STAND
START
START
STATE
STATE
STEAM
STEEL
STEEP
STICK
STILL
STOCK
STONE
STORE
STUDY
STUDY
STUFF
STYLE
SUGAR
SUPER
SWEET
TABLE
TASTE
TEACH
TERRY
THANK
THEME
THICK
THING
THINK
THIRD
THROW
TIGHT

TITLE
TOTAL
TOTAL
TOUCH
TOUCH
TOUGH
TOWER
TRACK
TRADE
TRAIN
TRAIN
TREAT
TREND
TRIAL
TRUST
TRUTH
UNCLE
UNION
UNITY
UPPER
UPSET
URBAN
USUAL
VAGUE
VALID
VALUE
VIDEO
VISIT
VITAL
VOICE
VOICE
WASTE
WASTE
WATCH
WATER
WHILE
WHITE
WHOLE
WOMAN
WORLD
WORRY
WOULD
WRITE
WRONG

YOUNG
YOUTH