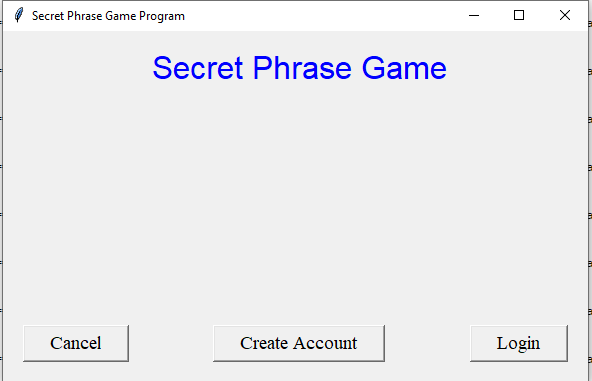
Design Document

Secret Phrase Game Project (Python)

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.

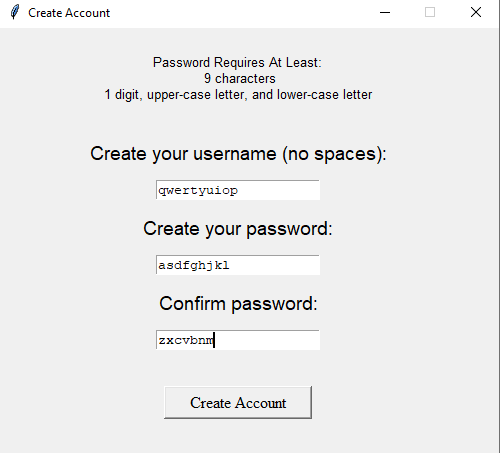
## Milestone #1 - 9/28/22

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.

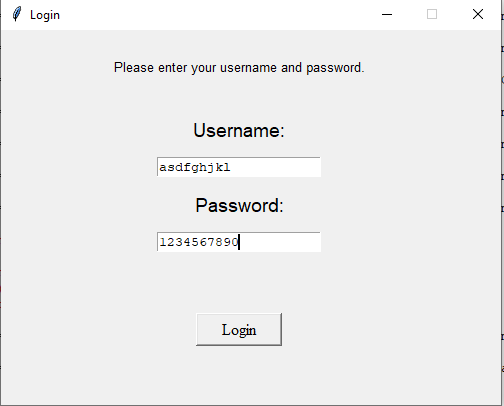


Main GUI Window. Picture may be added later on as the game is developed.

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.



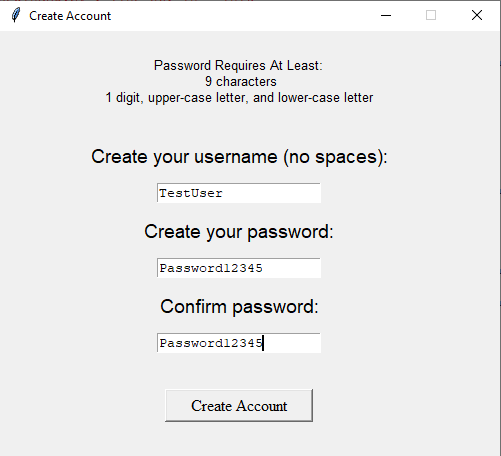
Login Window: Text boxes to be made right aligned, and password censored as “\*\*\*\*\*”

Pseudocode/Checklist:

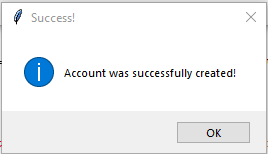
* Initial GUI opens on program start (**X**)
* GUI has title and picture(possibly) (**X**)
* GUI has Login, Create Account, and Cancel Buttons (**X**)
* Cancel button closes window (**X**)
* Create account button executes createAcct() function (**X**)
* createAcct() function opens new window (**X**)
* 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**)

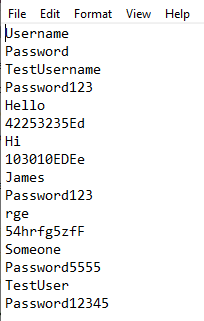
## Milestone #2 - 10/12/22

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.

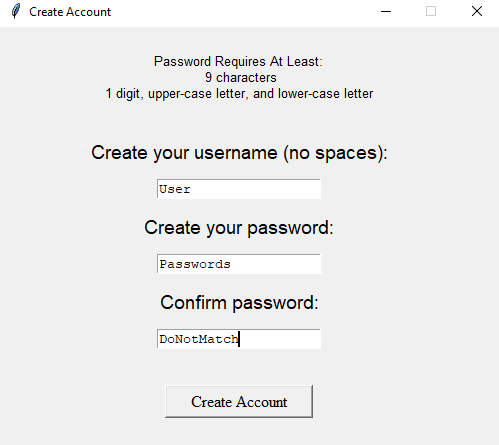


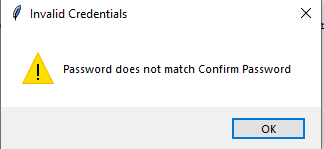
After some troubleshooting, I have figured out that Tkinter Entry boxes would work better than Text boxes, as they can be configured to be right-aligned and censored. I will implement this on a future milestone.



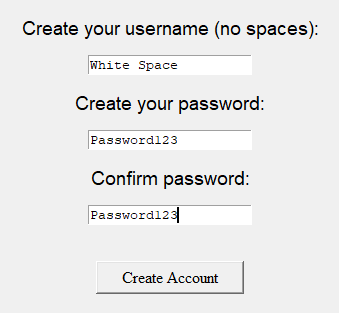


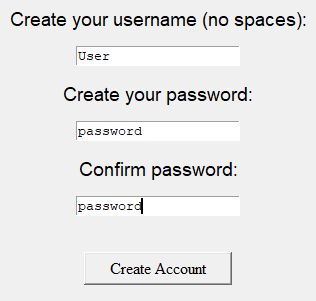
If Username and Password are valid, they are written into a Text File and the program proceeds.

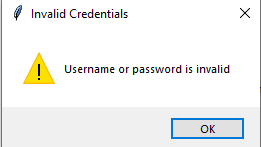




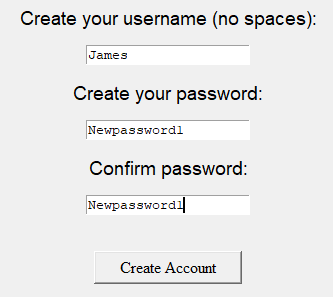
Error comes up if Password and Confirm Password does not match.

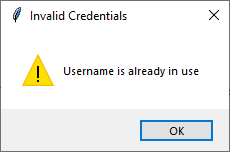




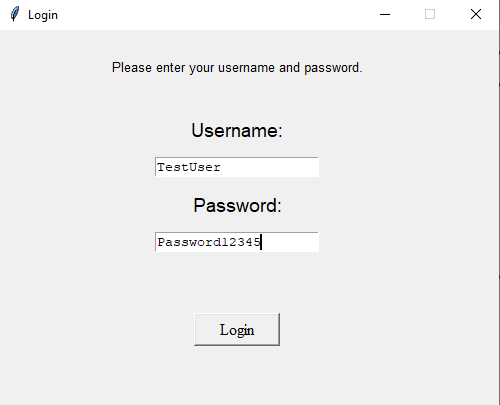


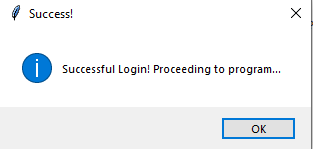
Error will be given if Username and Password requirements are not met.



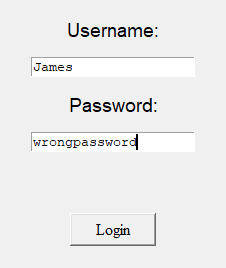


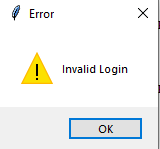
Error will be given if Username is already in system



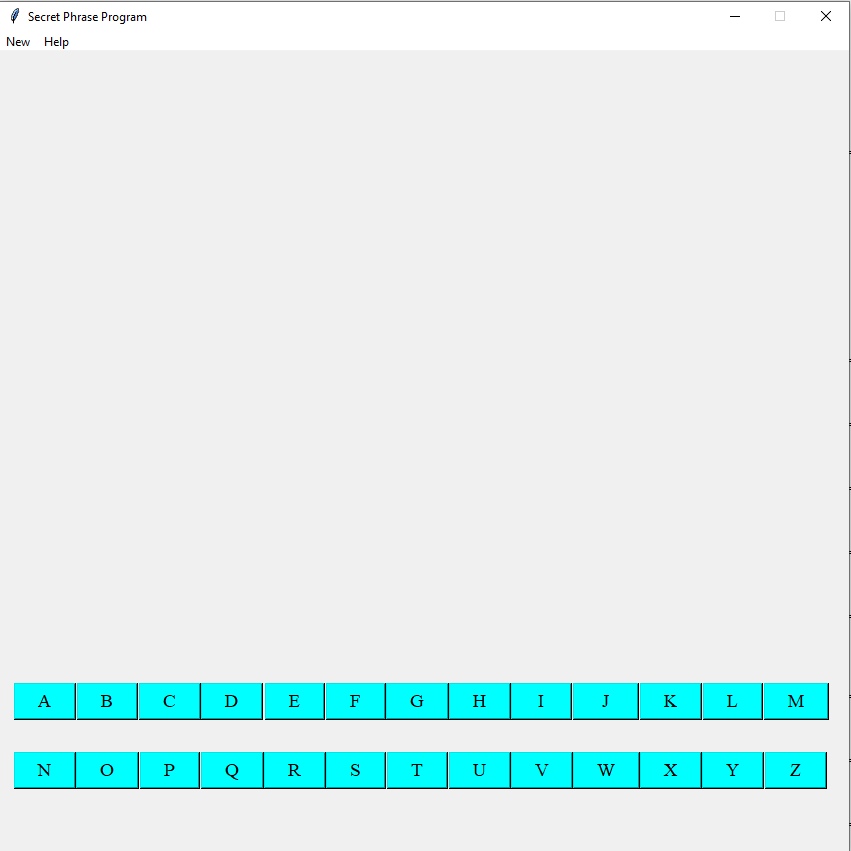


Login window is now functional





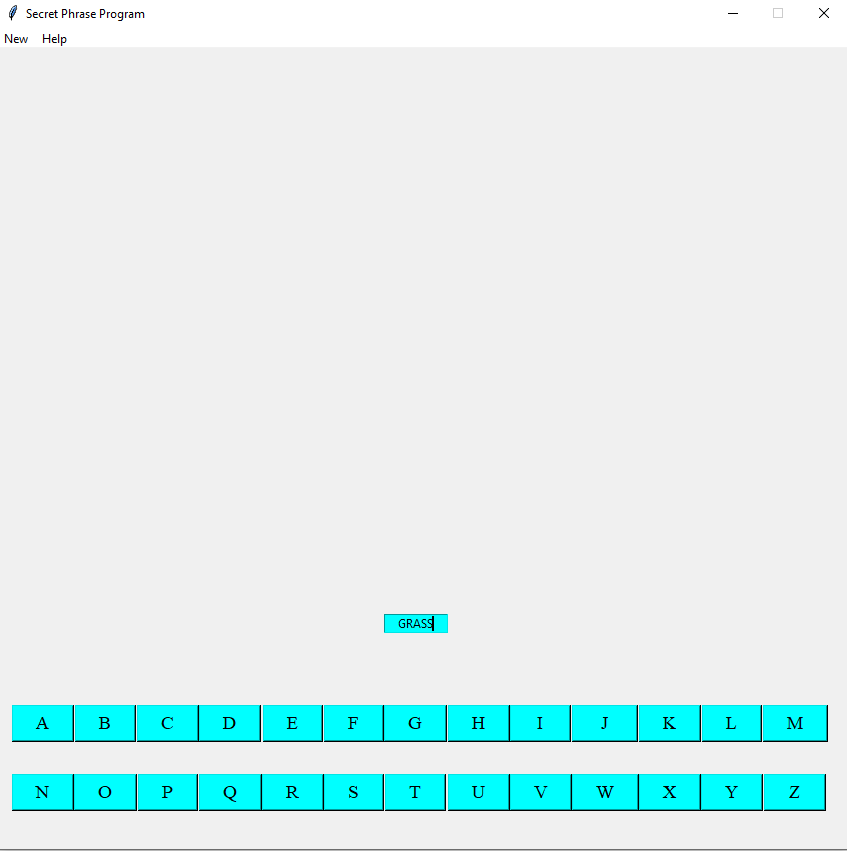
Gives Error if Login is incorrect



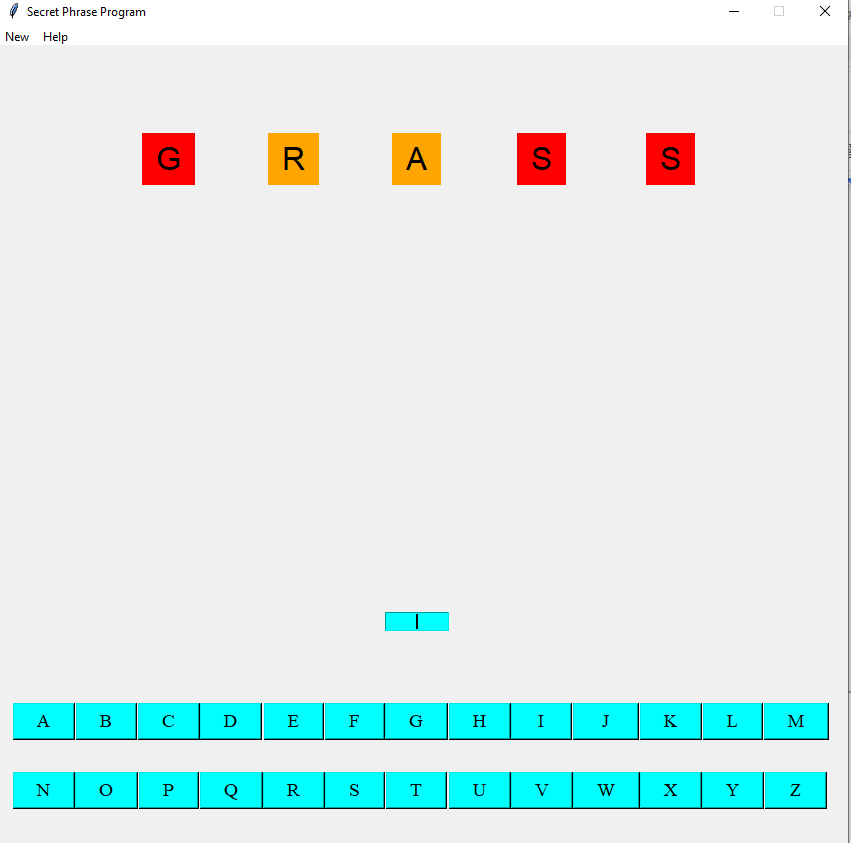
Main Program GUI. Keyboard Buttons. Menu created at the top left as well for options.

## Milestone #3 - 11/2/22

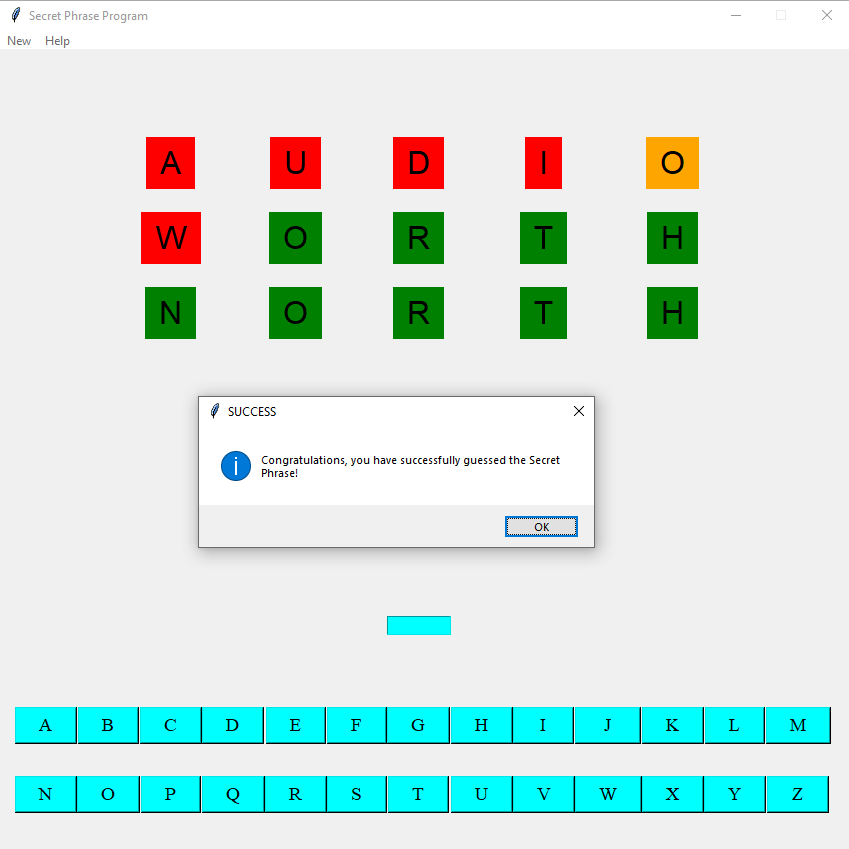
The goal of Milestone 3 is to provide semi-function to the overall program. In this Milestone I achieved the base functionality of the program, and will make quality-of-life tweaks to the program in future Milestones. The game itself is complete.



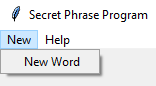
New Main Window, there is now an entry bar for the user to enter guesses. The buttons and typing both work.

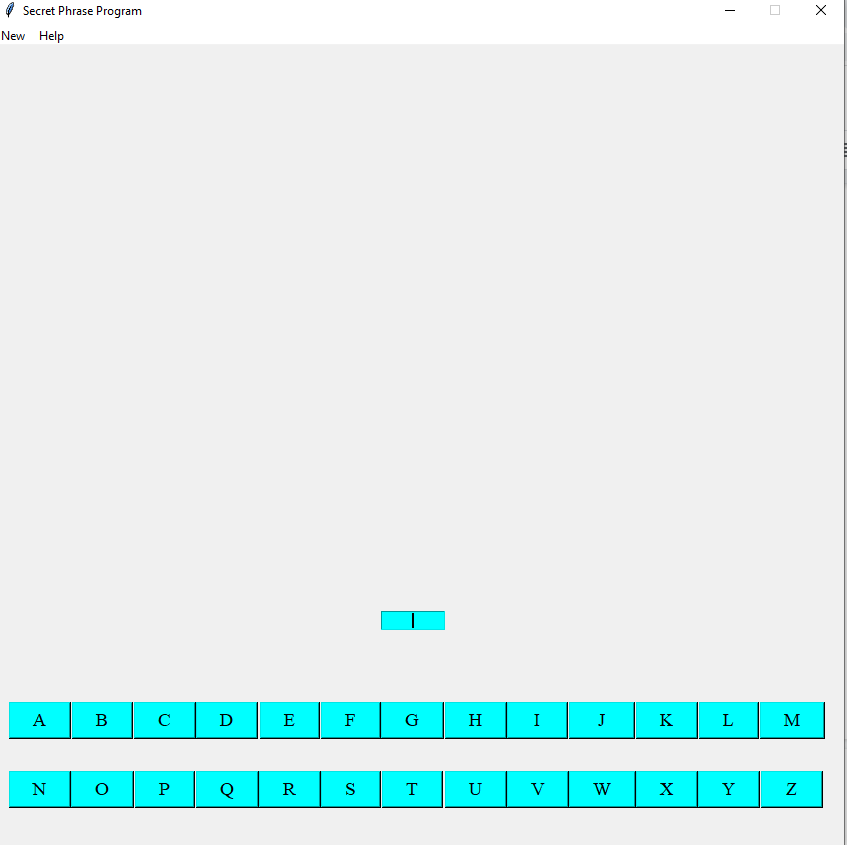


When the user presses ENTER, the guess is processed onto the window. It provides clues to what the word is that they are trying to guess. Red means the letter is not in the word. Orange means the letter is in the word but not in the right position. Green means the letter is in the word and in the right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

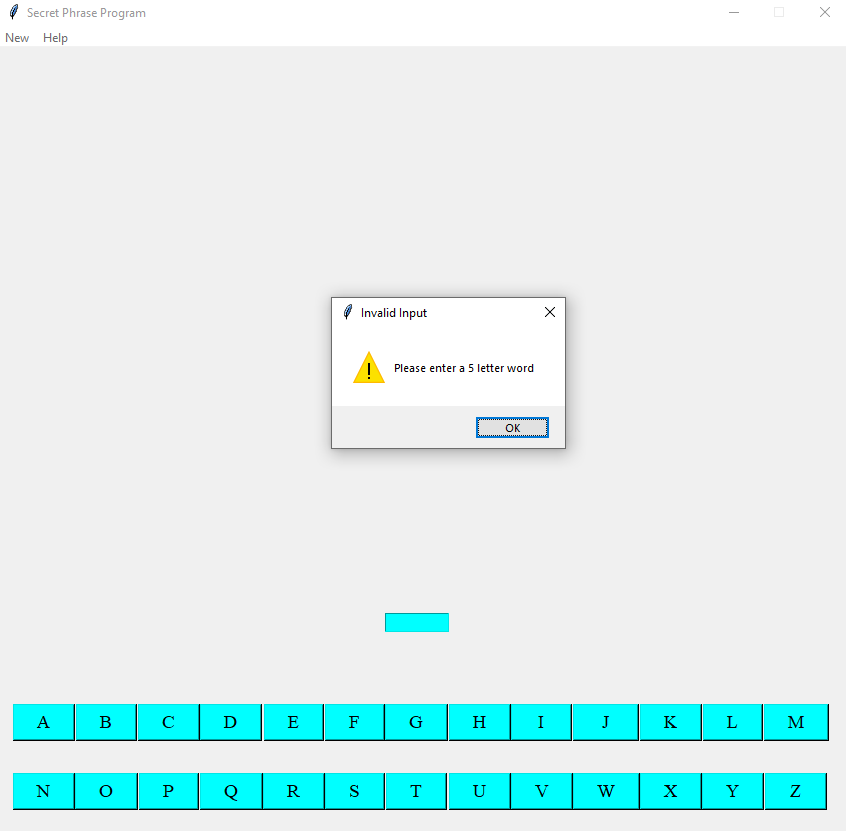


The program can determine when you have guessed the secret phrase.

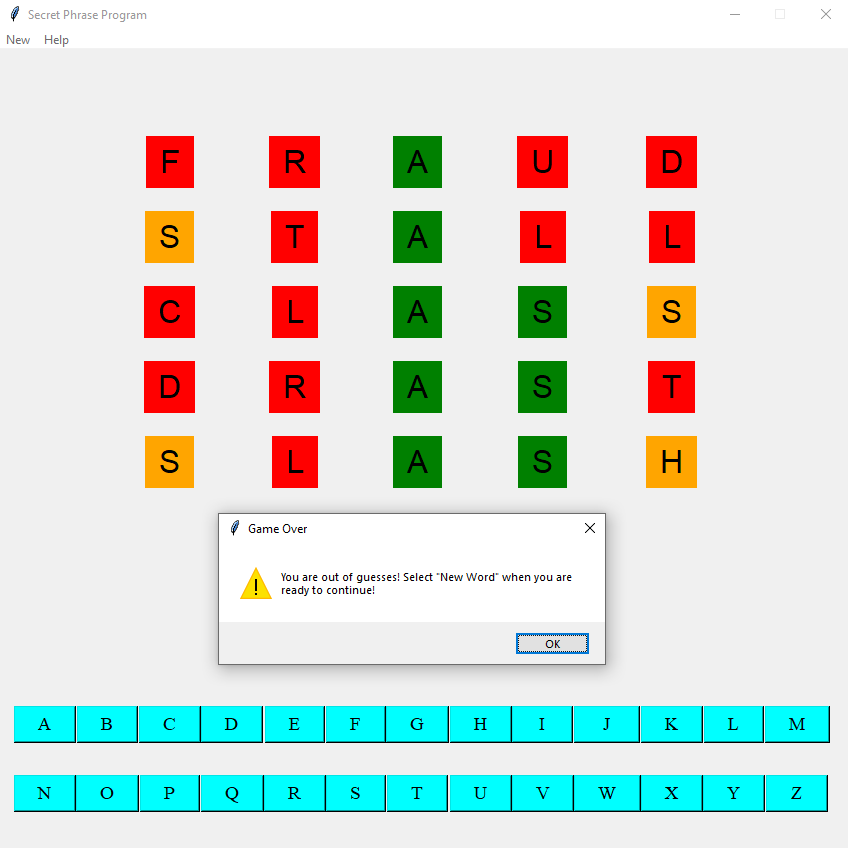




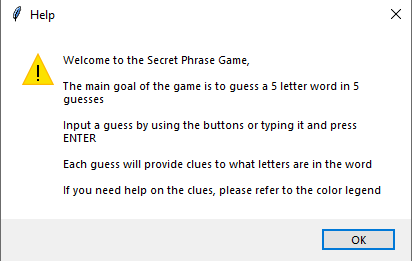
Users can reset the game by selecting New > New Word

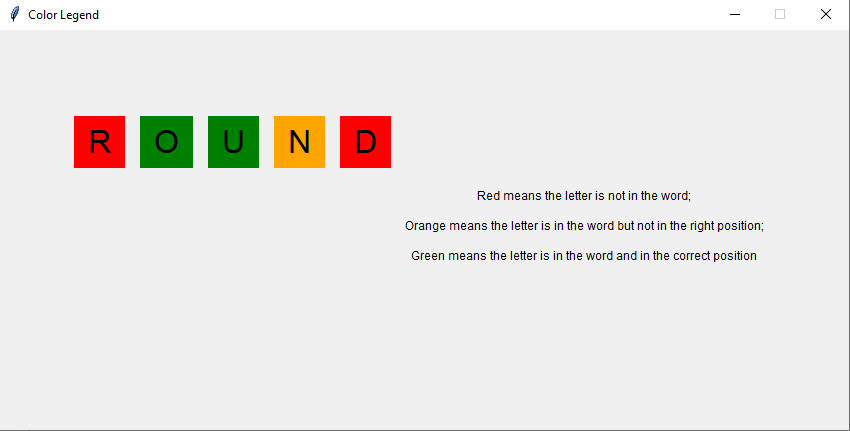


If the user tries to enter a word that is not 5 letters, this error pops up.



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





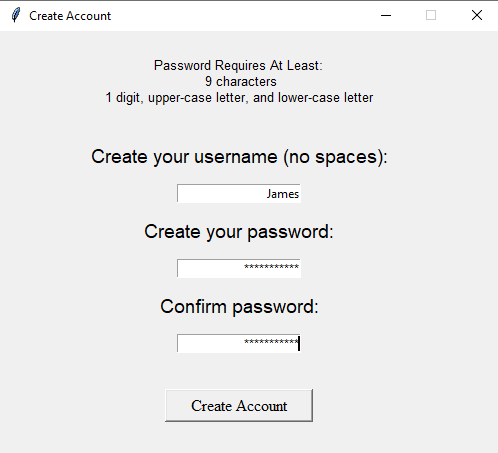
The Help Menu is included if needed.

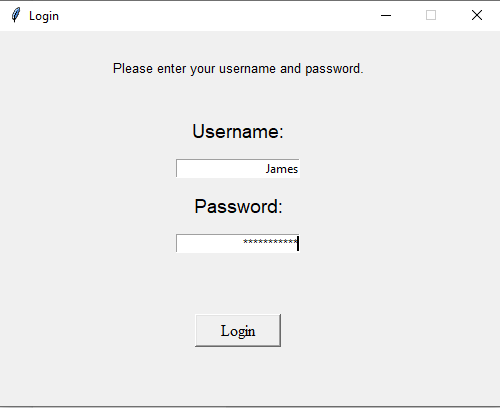
To Do:

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

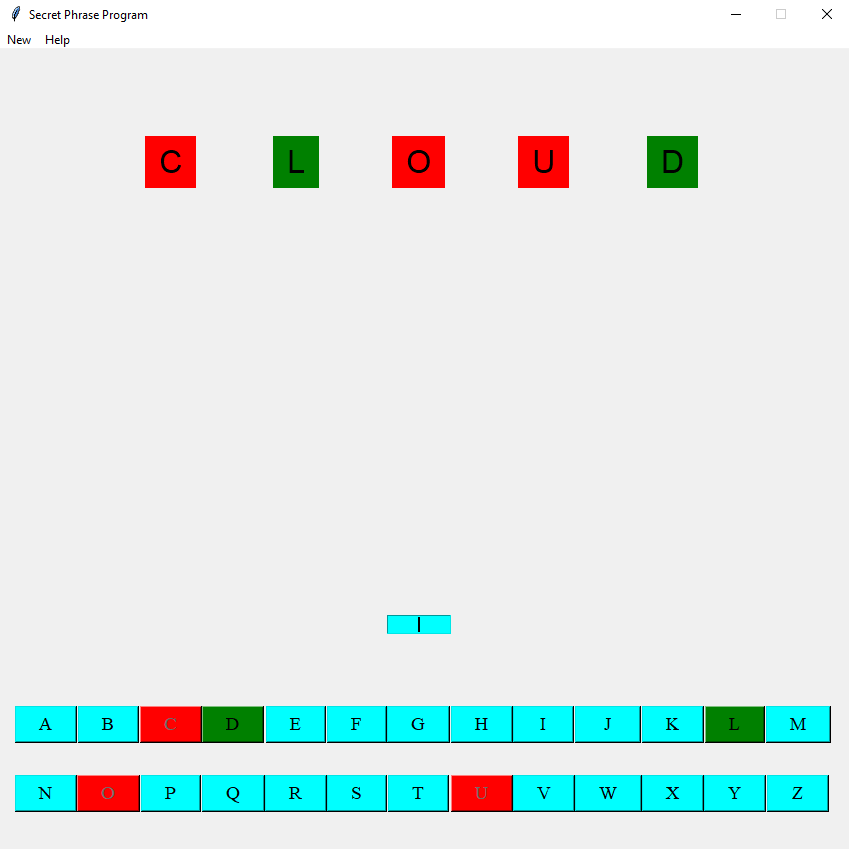
## Milestone #4 - 11/16/22

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.

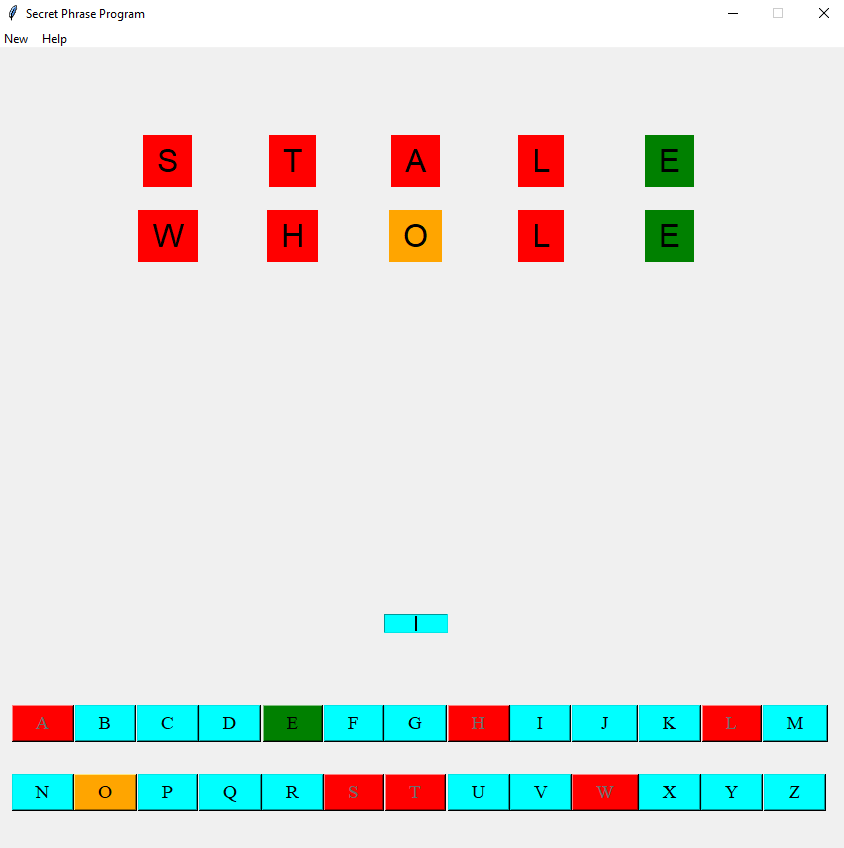




tk.Text boxes are changed to tk.Entry boxes, allowing right alignment as well as password censoring.

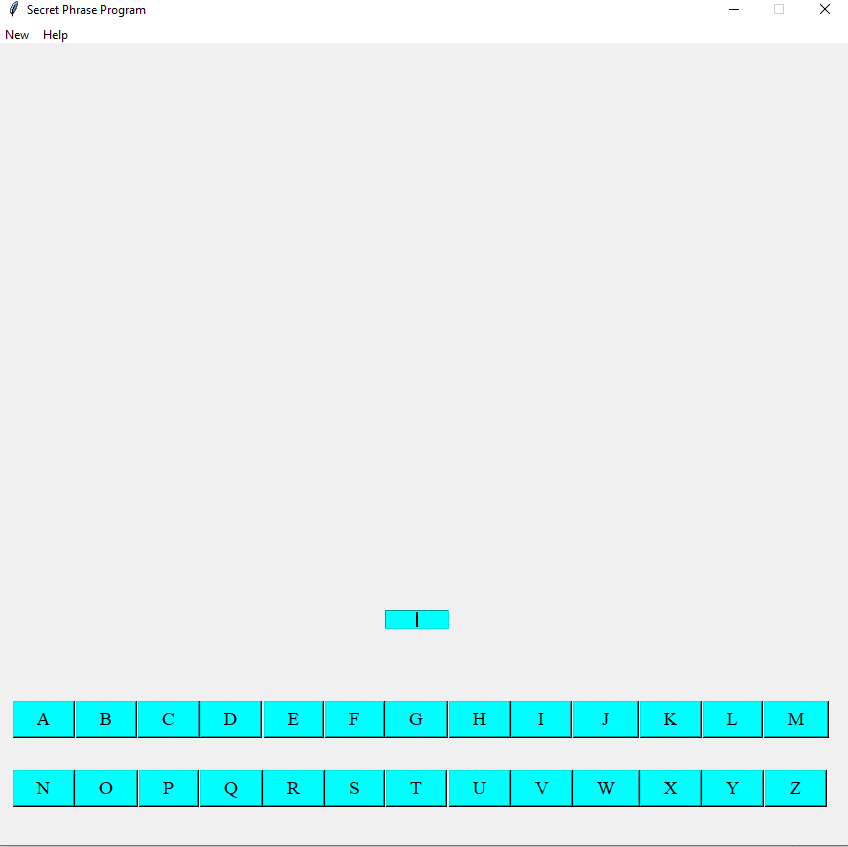


Letters that are not in the word have their buttons disabled. Button color is also displayed respective to the visual ui.



Further demonstration of the button ui

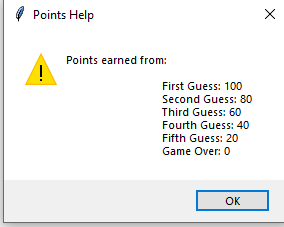




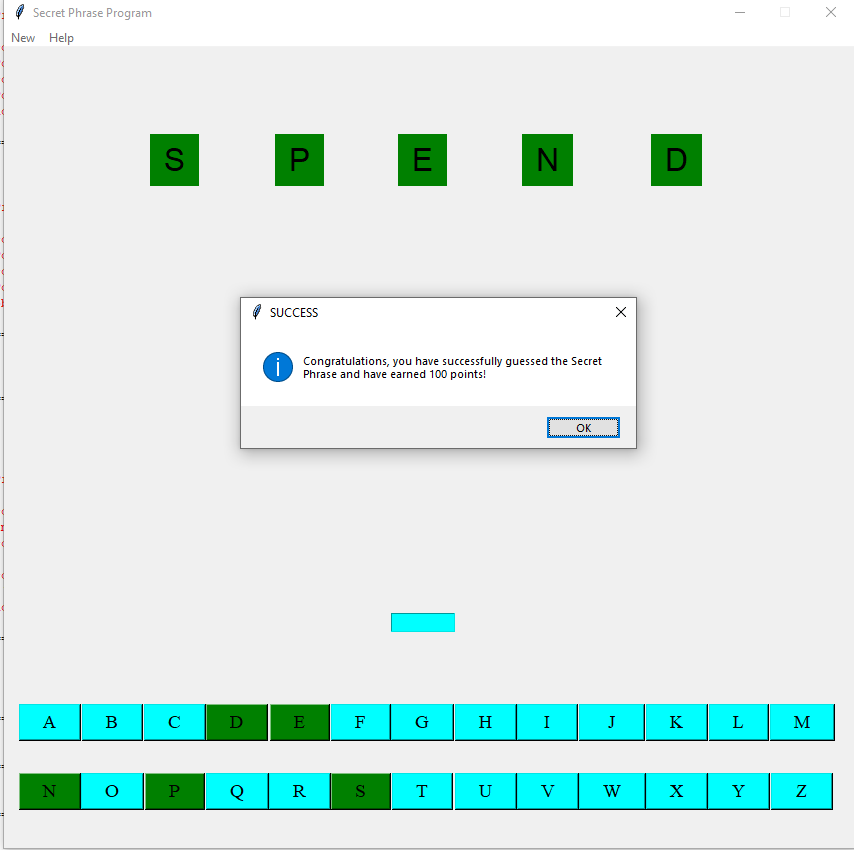
Picking a new word resets the buttons

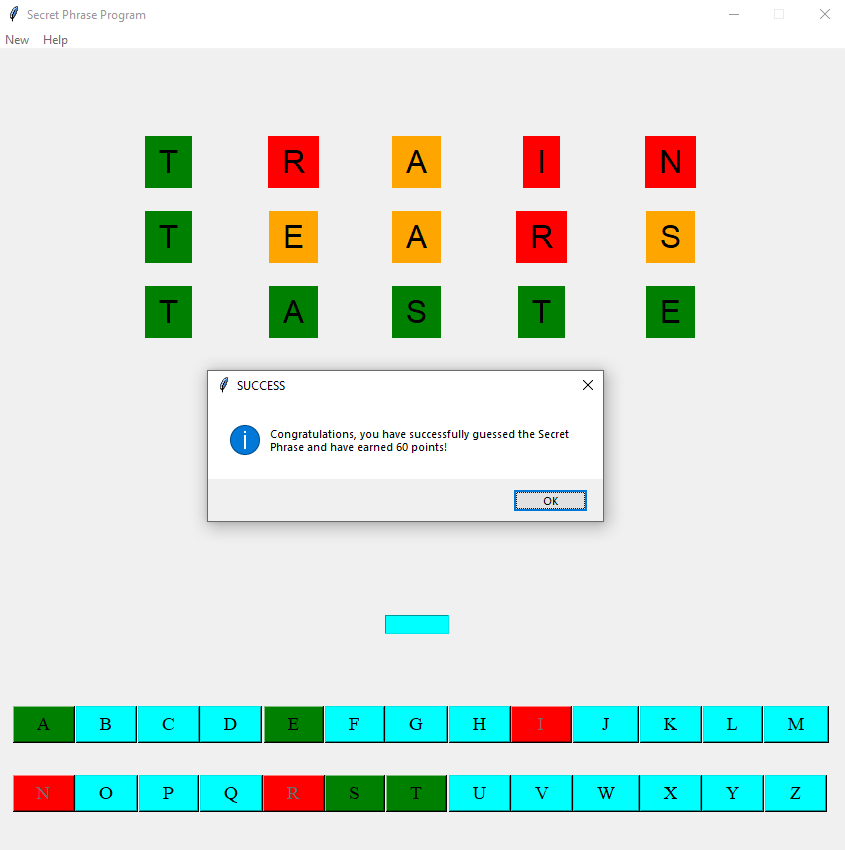
## Milestone #5 - 11/30/22

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.

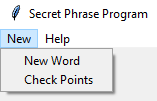


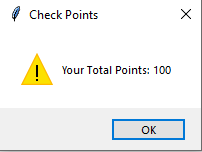
User can click Help > Points Help to learn about the point system





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

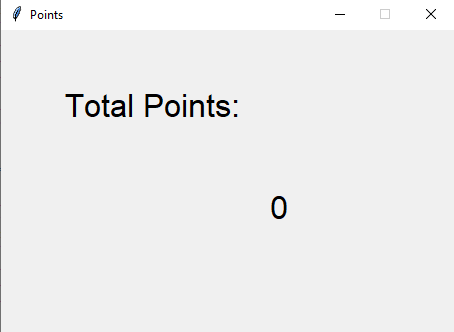




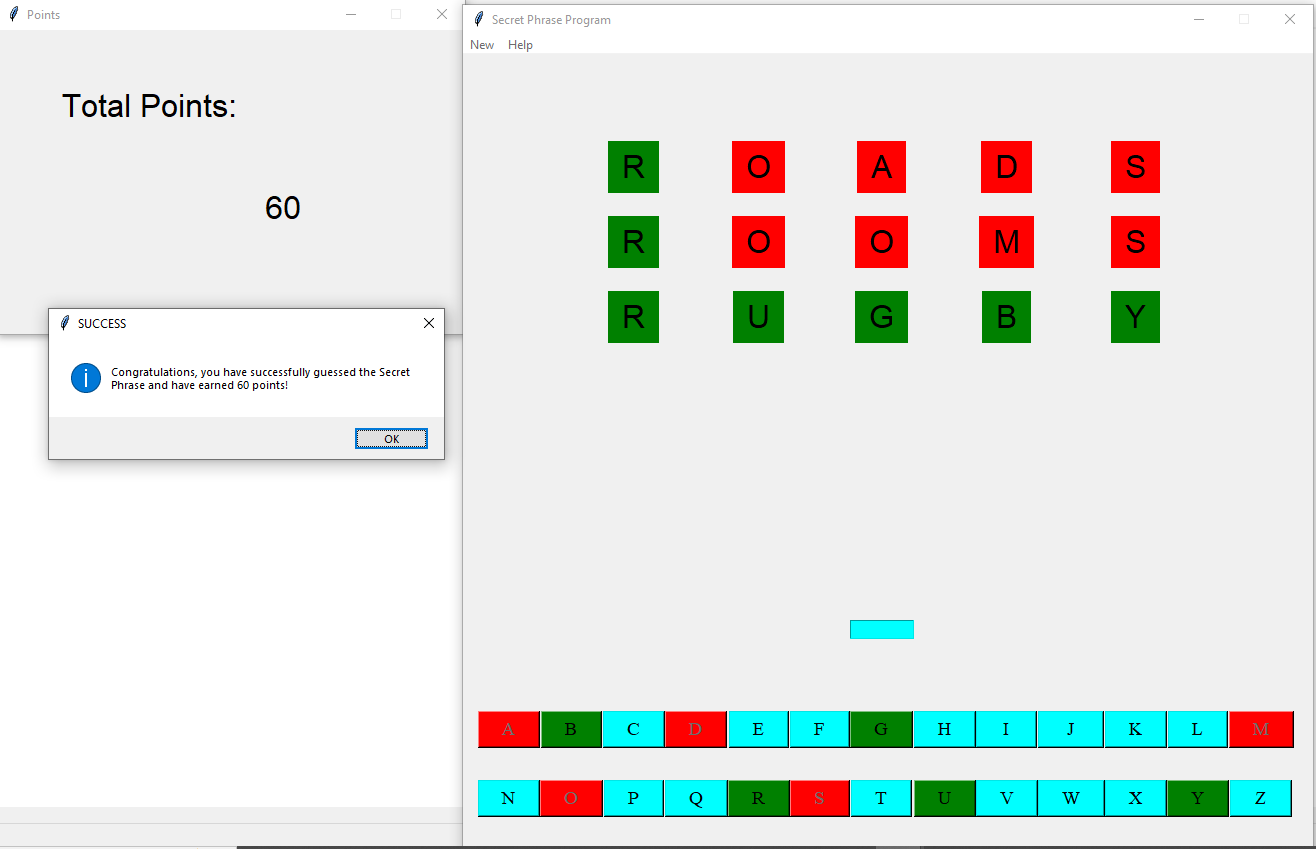
This menu is used to check how many points you have. (Temporary, it will be converted into a window in the next milestone)

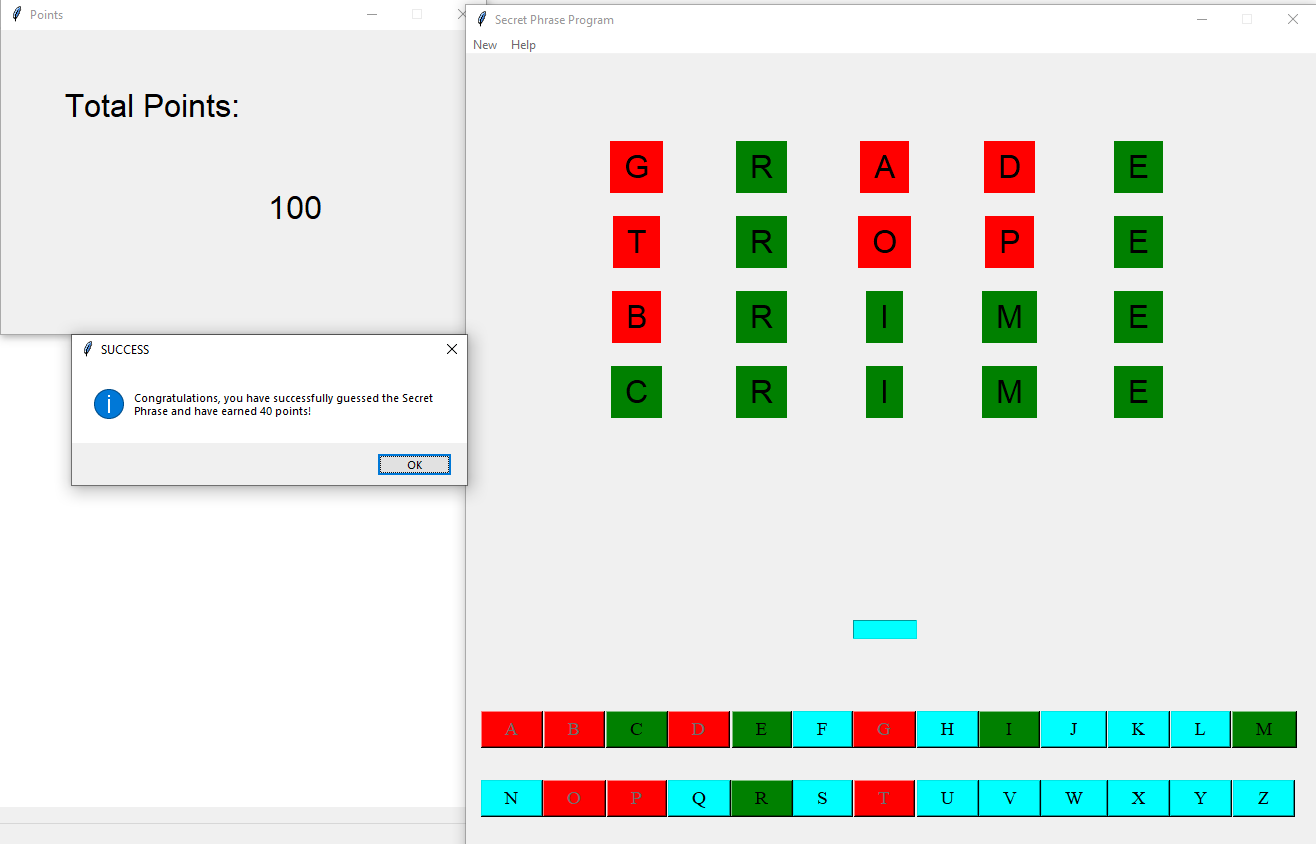
## Milestone #6 - 12/19/22

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

**# 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**

**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