CYB 220-T01

Programming Project I

Professor Grech

Mikayla Hubbard

February 22, 2025

Table of Contents

Introduction

The python programming project I have created is a Choose Your Own Adventure. This is based on the program I began creating in Exercise 4. The end product looks very similar, except that I used tkinter to render the GUI instead of easyGUI, and I moved the creation of the Choose Your Own Adventure (CYOA) layout and functionality into its own class.

The program consists of 4-5 files:

1. The CYOA class file

   This sets up the tkinter frames and has methods to create different views and functionalities.

2. The CYOA instance file

   This creates and instance of the CYOA class, linking together different functions which use the class's methods to show different views. This is where the story is linked together.

3. The refences file

   A simple file with a list of reference links

4. The Azure folder and related files

   These are included in the project folder in order to implement the Azure styling.

5. A text write-out file (optional/not created until program runs)

A text file is created when the program calls a certain method in the class and this file is where the story it written out to.

Below I shall walk through the code in each python file:

CYOAclass.py

```
from tkinter import filedialog, ttk
import tkinter.font as tkFont
```

^ imports, I used tkinter.ttk after some trial and error. Ttk ended up being used because it worked with the style theme I implemented. Filedialog was imported to allow a user to create a file to save the write-out to. tkFont was used for styling the text font.

```
class CYOA:
    def __init__(self, root):

        # Theme:
        # https://github.com/rdbende/Azure-ttk-theme
        root.tk.call('source', 'Azure/azure.tcl')
        root.tk.call('set_theme', 'dark')

        # styles:
        self.s = ttk.Style()
        self.headerFont = tkFont.Font(size=18, weight="bold", family="Times")
        self.bodyFont = tkFont.Font(size=16, weight="normal", family="Times")
        # self.btnFont = tkFont.Font(size=14, weight="normal")
        #styling the button, this had to go after the implimentation of the theme
        self.s.configure('TButton', font=('Times', 14))


        # set the size of the window
        root.geometry("800x400")
        # set the window title
        root.title("Choose Your Own Adventure!")
        # label for the page title
        self.title = ttk.Label(root, font=self.headerFont)
        self.title.pack(pady=10)
```

^ this is the first part of the __init__.

__init__ takes the parameter root, which only works if the root you are adding is tk.Tk() (this is done in the CYOAinstance file, where ‘root = tk.Tk()’ and then it creates an instance of the class with the parameter ‘root’

The theme section is where I am implementing the ttk Theme ‘Azure’. Linked is the GitHub page where I downloaded the neccissary files for it. I followed the instructions of the GitHub page to write the neccissary code to implement it and set it to the dark theme.

The styles section creates styles for the header, body text, and button. My main goal was to be able to change the font and to make the text size bigger because it was hard to read. However, I had difficulty getting the button styles to work. In order do accomplish this. I used ttk.Style() and configured the button with that.

The next section of code sets up the tkinter window. It sets the size of the window and the title of the window. Under this, we create a title label which is placed at the top of the window. We

will use self.title.config(text=) to change the title based on the current page in the story/program.

```
# frame for the story
self.storyFrame = ttk.Frame(root)
# add columns to the frame, there are two columns of equal weight
self.storyFrame.columnconfigure(0, weight=2)
self.storyFrame.columnconfigure(1, weight=2)
# create 2 labels to hold 2 potential paragraphs
self.storyIntro = ttk.Label(self.storyFrame, wraplength=500, font=self.bodyFont, justify="left")
self.story = ttk.Label(self.storyFrame, wraplength=500, font=self.bodyFont, justify="left")
# put the story in the columns, starting in column 0 and spanning both columns so that it's centered
self.storyIntro.grid(row=0, column=0, columnspan=2, sticky='wens',padx=70, pady=5)
self.story.grid(row=1, column=0, columnspan=2, sticky='wens',padx=70, pady=20)

self.storyFrame.pack()


# make a button frame, similarly to the story from
self.buttonFrame = ttk.Frame(root)
self.buttonFrame.columnconfigure(0, weight=1)
self.buttonFrame.columnconfigure(1, weight=1)

self.btn = ttk.Button(self.buttonFrame, padding=10, style='TButton')
self.btn.grid(row=0, column=0, columnspan=2, sticky="wens", padx=20)

self.btn1 = ttk.Button(self.buttonFrame, padding=15, style='TButton')
self.btn1.grid(row=0, column=0, sticky='wens', padx=20)

self.btn2 = ttk.Button(self.buttonFrame, padding=15, style='TButton')
self.btn2.grid(row=0, column=1, sticky='wens', padx=20)

self.buttonFrame.pack()


# a variable to hold the path where we will write out the story
self.path = "";
```

^ the rest of the __init__

Here we set up two frames in the window: the story frame and the button frame

Within the story frame, we create two columns. We then create two labels. One is for one paragraph, and the other is for the second paragraph. Most of the time we only use the second (self.story) paragraph, but when there is something like the introduction to the story, we write something into the first paragraph (self.storyIntro). In each of the labels, we added styling like

the fonts we created earlier, justifying the text, and having it span both columns. We use grid to place the paragraphs in thr grid and then pack the frame.

Next is the button frame. There are two columns, like the story frame. However, the buttons were more difficult for me to configure. This is because some views require one button and others require two buttons. I tried to do this many different ways, and this was one of the hardest parts of the project. One way that was working (but had styling issues) was creating the buttons within the different view's methods. I had to clear the button frame between each switch between one vs. two button screens, or the old buttons would still show up. However, I ended up ditching that way of doing this, as creating the buttons in the __init__ made more sense organizationally and helped me solve the styling issues. I will show the functions I am using to display the correct buttons in a moment. Here, I create 3 button widgets, btn (for when there is only one button), btn1, and btn2 (for when there are two buttons). Btn is put in the grid with a column span, and the two other buttons are but in their respective columns. We then pack the button frame.

The last line in the __init__ contains a variable self.path with will hold the path that the user chooses to write the file to.

Now, we will look at the methods in this class:

```
def oneButtonGrid(self):
    self.btn1.grid_forget()
    self.btn2.grid_forget()
    self.btn.grid(row=0, column=0, columnspan=2, sticky="wens", padx=20)

def twoButtonsGrid(self):
    self.btn.grid_forget()
    self.btn1.grid(row=0, column=0, sticky='wens', padx=20)
    self.btn2.grid(row=0, column=1, sticky='wens', padx=20)
```

^ these two methods are used to set up the buttons and are called in their respective views. The oneButtonGrid() uses grid_forget on the buttons for the twoButtonsGrid to remove them from view and places btn on the grid. The twoButtonsGrid forgets btn and adds btn1 and btn2 to the grid.

```
# fucntion that creates a page with 2 button options
def newQuestionPage(self, titleText, storyText, option1, option2, function1, function2, storyIntroText="", write=False):
    self.title.config(text=titleText)
    #I had to do this to that previous single buttons wouldn't show up
    if storyIntroText != "":
        self.storyIntro.config(text=storyIntroText)
    else:
        self.storyIntro.config(text="")
    self.story.config(text=storyText)

    #set the grid to 2 buttons
    self.twoButtonsGrid()

    #configure the buttons
    self.btn1.config(text=option1, command=lambda:function1())
    self.btn2.config(text=option2, command=lambda:function2())

    if write:
        if storyIntroText:
            self.write(storyIntroText)
        self.write(storyText)
```

^ this method, the newQuestionPage method, creates a view with two buttons, with the needed information in the parameters. There are a lot of parameters, but they are the necessary content for creating a question page. This is the page in the CYOA that tells you part of the story and has you decided which way to go. The user inputs the title, story, both option texts, and both functions, which would point to the next pages of the story depending on which button the user clicks. We then have the optional parameters: story intro, and write. Write, if set to True when using the method will call the write method to write the storyText and the StoryIntroText to the designated file (bottom of screenshot). I will discuss the write method later in this document.

The rest of this method adds the content inserted in the parameters into the proper labels/buttons. Additionally, (before it sets the content of the buttons), it calls the twoButtonsgrid() to properly set the button frame to have two available buttons.

```
#function that creates a page with one button option
def newContinuePage(self, titleText, storyText, buttonText, function, storyIntroText = "", write=False):

    self.title.config(text=titleText)
    if storyIntroText != "":
        self.storyIntro.config(text=storyIntroText)
    else:
        self.storyIntro.config(text="")
    self.story.config(text=storyText)

    # self.prepButtonFrame(1)
    self.oneButtonGrid()
    self.btn.config(text=buttonText, command=lambda:function())

    if write:
        if storyIntroText:
            self.write(storyIntroText)
        self.write(storyText)
```

^ this method, newContinuePage, is basically the same as the newQuestionPage, except that it only has one option parameter (called buttonText) and one function, and it sets the grid to only have one button.

```
def open_file_dialog(self):
    file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text files", "*.txt"), ("All files", "*.*")])
    self.path = file_path
    return file_path


def write(self, storyString):
    if self.path != "":
        try:
            with open(self.path, 'a') as file:
                file.write(f"{storyString}\n")
        except FileNotFoundError:
                print("ERROR")
```

^ these are the final two methods in the class.

The open_file_dialog method, uses filedialog to prompt the user to create a filename and this returns the path to the local variable file_path. We then change self.path to file_path and also return file_path incase we want to use the filepath in the main program.

The write method takes a storyString as a paraments and checks that there is a filepath, it then tries to open it (with append so as not to overwrite anything), and then write the storyString to the file.

CYOAinstance.py

```
from CYOAclass import CYOA
import tkinter as tk
```

^ The imports for this file are the class we created and tkinter

```
root = tk.Tk()
c = CYOA(root=root)
```

^ at the top of the file we create the root and create an instance of the class with this root.

```
intro()
root.mainloop()
```

^ this is at the end of the class. It calls the first function in this file which starts of the list of connected functions and it runs the root's main loop.

```
def start():
    c.newQuestionPage("Save Story", "As you follow the story, we will be writing the results to a file. Please create a file to save your s
                      "Create File", "Quit", createFile, Quit)

def createFile():
    c.open_file_dialog()
    drawbridge()

def Quit():
    c.newContinuePage("Byeeeeee", "Thank you for your time :)", "BYE", root.destroy)

def intro():
    # the adventure starts here
    c.newQuestionPage("Start Adventure","Hi! welcome the the choose your own adventure following the adventures of a deaf dog named Cooper!
                      "Start", "Quit", start, Quit)

def end():
    c.write("THE END")
    c.newQuestionPage("THE END", "Thank you for playing! Would you like to start the story again?", "Start", "Quit", drawbridge, Quit)
```

^ these are the more organizational functions. Intro is what runs automatically, and it points to either start or Quit, using the class's newQuestionPage method to create the view. Start prompts the user to create a file (or Quit), and the createfile calls the class's open_file_dialog method, and once that is complete it point to drawbridge, the first page function in the CYOA. Quit used the newContinuePage method as it's view and when the user clicks the button it destroys the running of the program. The last function in this screenshot is end, which is called from the last page/function in a story (there are a couple different endings that call this), and it prompts the user to either start the series again or Quit.

The rest of the functions in this file are similar. They each represent a different page of the CYOA and use either the newContinuePage or newQuestionPage method from the class to create their views, linking to the next page/function depending on which button the user clicks on. Each function that contains a piece of the story uses write=True in the parameter list of the class method they are calling in order to write their part of the story into the designated file. Below are

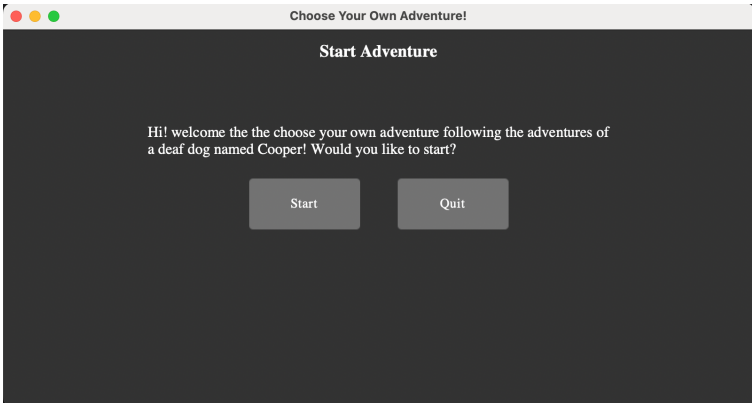
the story/page functions:

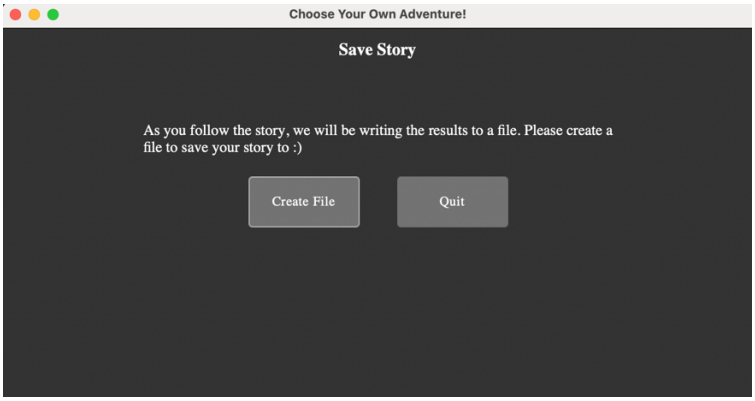
```
def drawbridge():
    c.newQuestionPage("Drawbridge", "cooper heads to the castle door, but the drawbridge is up! What does he do?",
                      "A. Use puppy eyes to get the guard to open the drawbridge", "B. bite through the rope to let down the bridge",puppyEyes,
                      "A long time ago in a kingdom far far away, there lived a dog named Cooper. Cooper had lived a good long life. Throughou
                      write=True)

def puppyEyes():
    c.newQuestionPage("Puppy Eyes", "Cooper gives the guard on duty his best puppy-eyed stare. The guards melts, as they all do, and walks to
                      "A. snatch a piece of steak and run away", "B. Bite at the cooks heels", stealSteak, biteHeels, write=True)

def biteRope():
    c.newQuestionPage("Crossing the Drawbridge", "Cooper dashes forward and bites the rope in 2. The drawbridge opens and Cooper runs off to
                      "A. left", "B. right", left, right, write=True)

def stealSteak():
    c.newQuestionPage("Steak Theif", "While the cook is distracted, Cooper takes the opportunity to snatch a piece of steak and dart away as
                      "A. left", "B. right", hallwayLeft, hallwayRight, write=True)

def biteHeels():
    c.newQuestionPage("Biting", "Cooper begins biting at the cooks heels, which angers him even more. In a rage, the cook scoops up the littl
                      "A. left", "B. right", left, right, write=True)

def hallwayLeft():
    c.newQuestionPage("Dead End", "Cooper turns left. Alas! Dead end!  In a rage, the cook scoops up the little dog and throws him out of the
                      "A. left", "B. right", left, right, write=True)

def hallwayRight():
    c.newContinuePage("King", "Cooper decides to go right, immidiately running into the kind-hearted king who is obsessed with him. He is pic
                      "THE END", end, write=True)

def left():
    c.newQuestionPage("Beach", "Cooper trots towards the beach, taking in the amaxing weather. As he reaches the beach, he also reaches a con
                      "A. play in the water", "B. dig in the sand", water, sand, write=True)
```

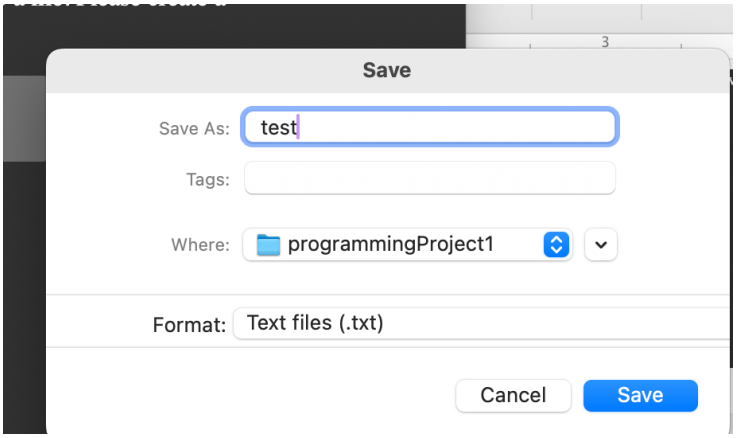
```
def right():
    c.newQuestionPage( "Dark Forest", "Cooper heads to where he imagines the most ad
                      "A. follow the singing", "B. stay on the path", singing, stay, write=True)

def singing():
    c.newContinuePage("Singing", "Did you foget that Cooper's deaf? He can't hear the singing, idiot.",
                      "Go Back", right)

def stay():
    c.newQuestionPage("Smells", "He trots along the path, then he smells sweets, he strays off the path following the smell. he comes across a
                      "A. towards the sweets", "B. towards the bacon", sweets, bacon, write=True)

def sweets():
    c.newContinuePage("Sweets","He follows the sweers, finding an old lady with a sinister smile, stirring a large potions. The door shut ominou
                      "THE END", end, write=True)

def bacon():
    c.newContinuePage("Bacon", "following the scent of bacon, Cooper comes across a beshevled youngster along the side of the path. Excited to h
                      "THE END", end, write=True)

def water():
    c.newContinuePage("Ocean", "excited, Cooper bounds to the water, splashing around, pretending to be a pirate lost at sea. However, Cooper st
                      "THE END", end, write=True)

def sand():
    c.newQuestionPage("Digging", "As most dogs are inclined to do, Cooper excitedly starts digging a hole. He imagines he is a pirate digging fo
                      "A. keep digging", "B. make a better hole somewhere else", treasure, dig, write=True)

def treasure():
    c.newContinuePage("Treasure", "Cooper keeps digging, hoping for the best. What he uncovers is a small metal chest full of gold and jewelry.
                      "THE END", end, write=True)

def dig():
    c.newContinuePage("More Diggin", "Disgruntled at the unfortunate piece of metal occupying his perfectly good hole, Cooper moves a couple fe
                      "THE END", end, write=True)
```

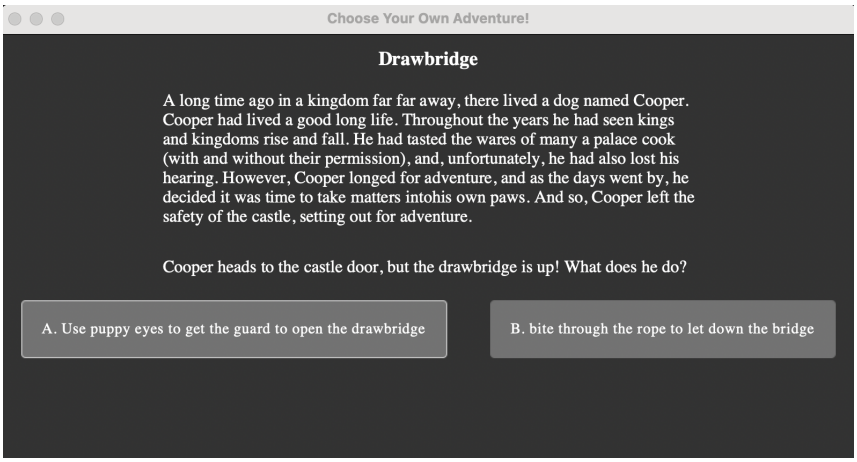
Note: this is the exact same story I used in Exercise 4, just implemented differently.

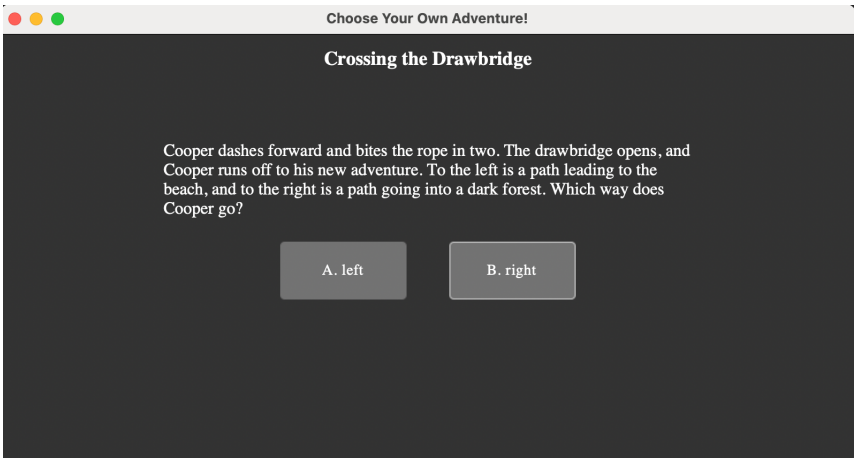
Example of the Code Running

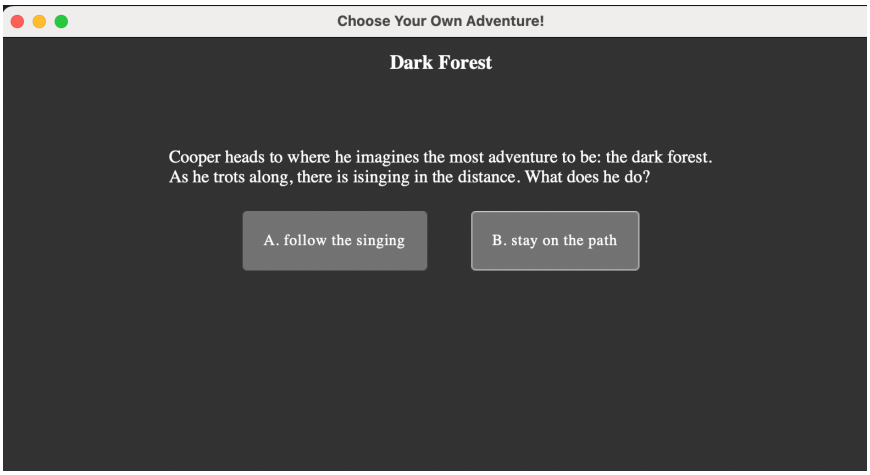


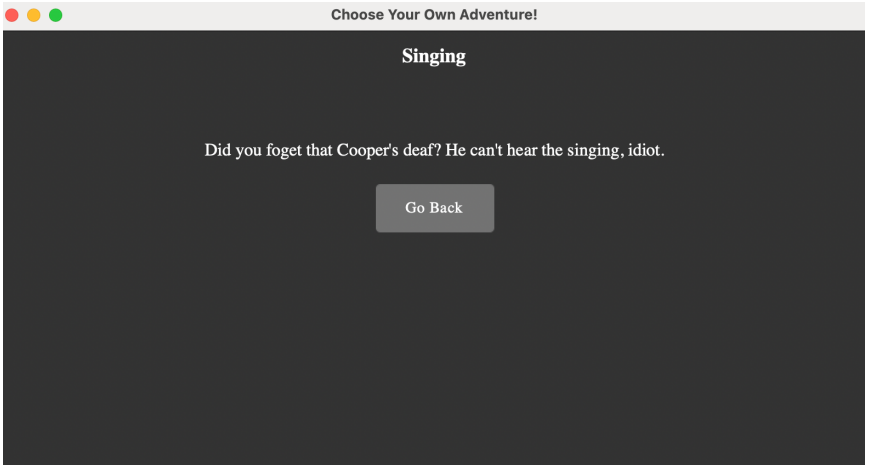


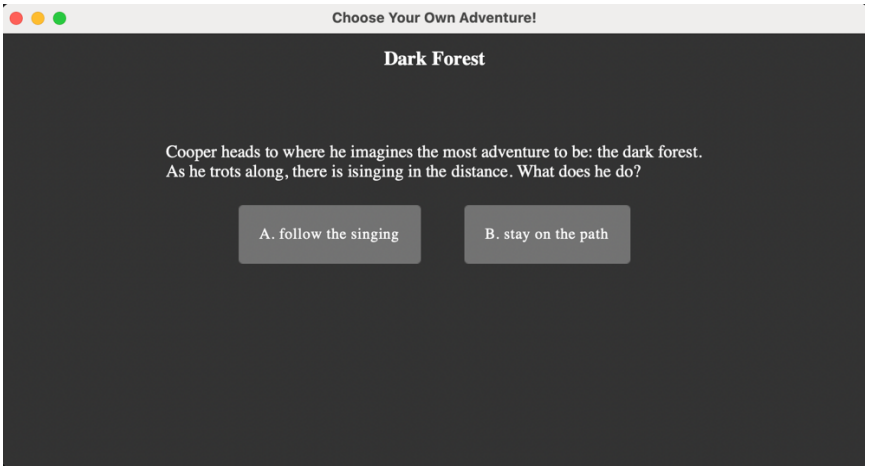


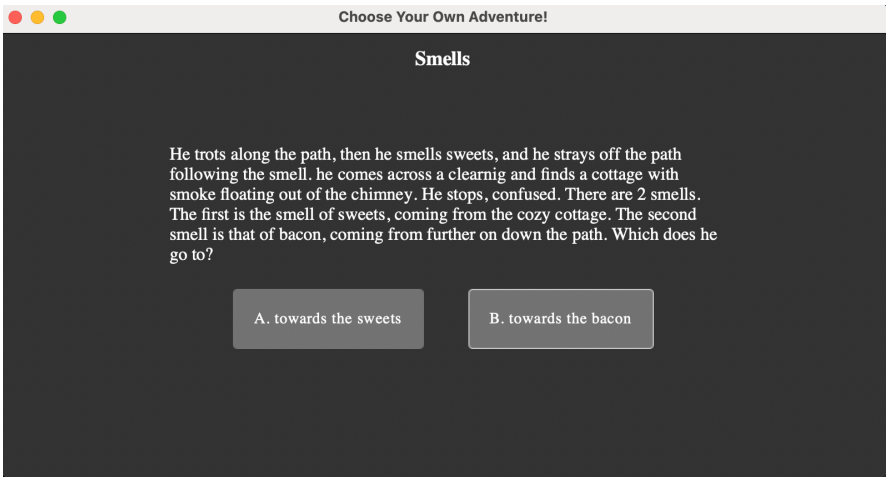

Choose Your Own Adventure!
Drawbridge
A long time ago in a kingdom far far away, there lived a dog named Cooper. Cooper had lived a good long life. Throughout the years he had seen kings and kingdoms rise and fall. He had tasted the wares of many a palace cook (with and without their permission), and, unfortunately, he had also lost his hearing. However, Cooper longed for adventure, and as the days went by, he decided it was time to take matters intohis own paws. And so, Cooper left the safety of the castle, setting out for adventure.
Cooper heads to the castle door, but the drawbridge is up! What does he do?
A. Use puppy eyes to get the guard to open the drawbridge
B. bite through the rope to let down the bridge

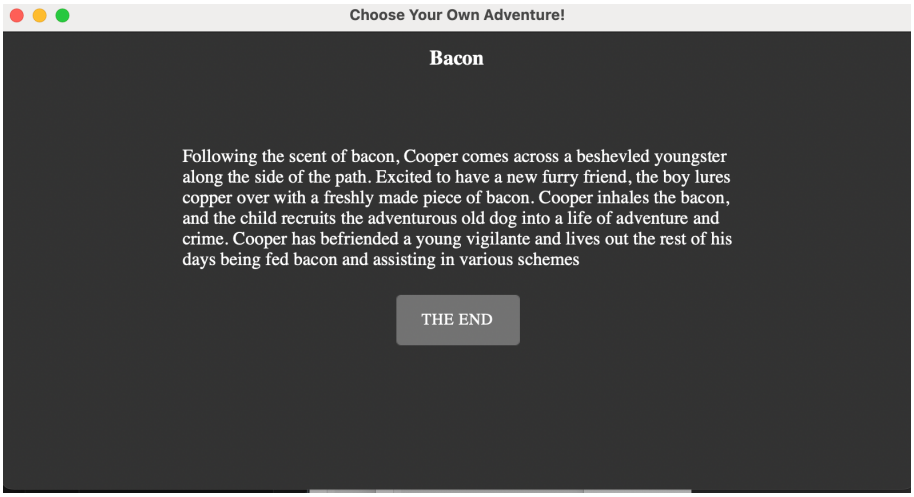

Choose Your Own Adventure!
Crossing the Drawbridge
Cooper dashes forward and bites the rope in two. The drawbridge opens, and Cooper runs off to his new adventure. To the left is a path leading to the beach, and to the right is a path going into a dark forest. Which way does Cooper go?
A. left
B. right


Choose Your Own Adventure!
Dark Forest
Cooper heads to where he imagines the most adventure to be: the dark forest. As he trots along, there is isinging in the distance. What does he do?
A. follow the singing
B. stay on the path


Choose Your Own Adventure!
Singing
Did you foget that Cooper's deaf? He can't hear the singing, idiot.
Go Back


Choose Your Own Adventure!
Dark Forest
Cooper heads to where he imagines the most adventure to be: the dark forest.
As he trots along, there is isinging in the distance. What does he do?
A. follow the singing
B. stay on the path





GitHub Link

https://github.com/mikaylahubbard/CYB220/tree/65498a064fbb93db773e5e6cc567ca53c56a4b1e/programmingProject1

Note: I couldn't figure out a good way to include the Azure folder. I zipped it and put it in the GitHub folder, so you can download the folder from there. You could also get the Azure files from it's GitHub at https://github.com/rdbende/Azure-ttk-theme

References

- https://stackoverflow.com/questions/76932493/tkinter-using-buttons-to-progress-a-text-adventure-game
- https://www.geeksforgeeks.org/difference-between-fill-and-expand-options-for-tkinter-pack-method/
- https://stackoverflow.com/questions/11949391/how-do-i-use-tkinter-to-create-line-wrapped-text-that-fills-the-width-of-the-win
- https://stackoverflow.com/questions/4174575/adding-padding-to-a-tkinter-widget-only-on-one-side
- https://tkdocs.com/tutorial/grid.html
- https://www.youtube.com/watch?v=mop6g-c5HEY
- https://python-forum.io/thread-26250.html
- https://www.reddit.com/r/Python/comments/lps11c/how_to_make_tkinter_look_modern_how_to_use_themes/
- https://github.com/rdbende/Azure-ttk-theme
- https://www.geeksforgeeks.org/tkinter-fonts/
- https://www.youtube.com/watch?v=9eljXrx7fCM
- https://www.geeksforgeeks.org/how-to-hide-recover-and-delete-tkinter-widgets/
- https://www.geeksforgeeks.org/python-forget_pack-and-forget_grid-method-in-tkinter/
- Mikayla Hubbard's Exercise 4 part 2 (included in the ‘no longer in use’ folder)
- CYB 220 resources (mostly slides)