Ian Sharff

COSC 010: Introduction to Computer Science

Project 1 Report

**SECTION 1**

*Overview*

In the game, titled PYTHONLAND, the user wakes up on a beach without knowing where they are, and they are prompted with several choices. Only one pathway of choices can lead to them winning the game. The adventure game contains an introduction section where user must enter “start” to continue. Then, a character name section is run that collects the users name to be used in other sections. The main part of the game is divided into two (more could be added) locations contained in functions which loop when the user loses. This is done through calling a “game over” function that returns True if they want to continue and False if not, either continuing the loop or closing the program respectively.

*Expected inputs/outputs*

Start command, user’s name, adventure decisions and game over continue/end decisions are stored to execute certain conditionals that either continue the game, lead to the player losing, and/or allowing the user to keep playing or quit.

Outputs include the various narration function calls, and the pig latin translator included in the forest section. The player can continuously translate words until they don’t enter “yes” into the console. Other formatted outputs include the PYTHONLAND intro, and the GAME OVER animation, and the loading bar simulation

*Concept descriptions*

User input used throughout to make decisions. Various print statements and outputs from functions like the pig latin translator. String indexing list appending were used in this translator. The list comprehension was used in the sailors math problem to sum the squares of the user’s favorite numbers. While and for loops used for various purposes to either iterate lists or generate graphical outputs and break/continue was used accordingly. Conditional structures were used widely throughout to simulate natural response to user’s choices.

*Limitations and Expansions*

I have not found anything that explicitly breaks the program. The pig latin translator could be better and probably more efficient. It can’t deal with the word “rhythm”. Ideally I would have liked for the user to be able to return to any location at will. With better knowledge of classes I would have made a Locations class that would better organize the code and allow user to “move” and change locations or go back and repeat sections at will.

The location functions are messy and it is difficult to work around the issue of scope when putting everything in functions. For example, I wanted to have a “deathcount” variable that would tell the user at the end how many times they had repeated the sequence, but taking that data outside of two function scopes was tricky to think about. In hindsight a list could be created and passed in as a parameter and appended since they are mutable and change even within function scope.

**SECTION 2**

Intro display, requiring user to enter “start” to continueText

Description automatically generated

Beginning of beach function, choice that results in player losing, and then restarting the program after player enters “yes”

Text

Description automatically generated

Beginning of forest function showing pig response and repeatable pig latin translator.

Text

Description automatically generated

import sys

import time

def main():

#main function contains a while loop with flag variable, passed into the first

#"dream" function, which then returns true when complete. Not sure if this is

#redundant or not.

is\_dream = True

intro()

name = character\_name()

while is\_dream:

beach\_complete = beach(is\_dream)

forest\_complete = forest(beach\_complete)

if forest\_complete and beach\_complete:

is\_dream = False

victory(name)

sign\_off()

#SYSTEM FUNCTIONS\_\_\_\_\_\_\_\_\_

def pause(arg = 1):

#customizable pause function, default time multiplier is 1 but can be specified

#along with the delay time

delay = 1

time.sleep(arg \* delay)

def narration(message, choice = None):

#message parameter is printed along with a pause() function from above

print(message)

pause()

def loading\_screen(message = "") :

#simulated loading screen, can print with message or not, default is empty string

print("Loading:", end = "")

for i in range(20): #simulated loading bar

print("\* ", end = "")

pause(0.1)

print("")

if bool(message): #if message chagned from empty string it prints

print(f"{message}")

pause()

def decision(optA, optB, optC):

#used in the three-way decisions in the location functions. returns the result

#chosen by user. ideally would be able to return a function call to each decision

#group of code but it got complicated with function scope and the ability to restart

#with game\_over function

while True:

choice = input("\*Type A, B or C\n>>> ").upper()

if choice == "A":

result = optA

break

if choice == "B":

result = optB

break

if choice == "C":

result = optC

break

else:

narration("\*Invalid, try again...") #repeats if answer not A, B or C

return result

def pig\_latin():

#pig latin translator (rules commented below) to be used in the forest() function

#doesn't take a phrase as a parameter, but repeatedly takes user input

narration("\*Type words separated by spaces to translate to pig latin")

consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"

vowels = "aeiouAEIOU" #distinction made between vowels/consonants and case

platin\_phrase = [] #to be appended with translated english words

while True:

english = input(">>> ")

eng\_no\_space = english.replace(" " ,"") #removes spaces to

if not eng\_no\_space.isalpha(): #checks all values in string are alpha

narration("\*Please enter only spaces and letters")

else: break #continues with translation when condition met

english\_phrase = english.split() #creates list of strings for iteration

for eng\_word in english\_phrase:

first\_letter = eng\_word[0] #to check if vowel or consonant

first\_sound = first\_letter #concatenated with letters of first word sound

#one letter words ("a", "I") and words that start with vowels just

#have "way" concatenated

if (first\_letter in vowels) or (len(eng\_word) == 1):

platin\_word = eng\_word + "way"

platin\_phrase.append(platin\_word) #appends translated word

#if y is second letter, most likely used as vowel, so first letter goes to

#end and "ay" is concatenated

elif eng\_word[1] == "y":

platin\_word = eng\_word[1] + first\_letter + "ay"

platin\_phrase.append(platin\_word)

#if these conditions aren't met, then more than one consonant at beginning.

#second for loop adds subsequent consonants to first\_sound, stops when

#it reaches a vowel, and that index is used to mark beginning of the

#pig latin word

else:

rest = eng\_word[1:]

for i in range(len(rest)):

if rest[i] in consonants:

first\_sound += rest[i]

else:

platin\_word = rest[i:] + first\_sound + "ay"

platin\_phrase.append(platin\_word)

break

#if word has no vowel, then "ay" added to the end

#does not account for the word "rhythm"

else:

platin\_word = first\_sound + "ay"

platin\_phrase.append(platin\_word)

#prints list as words separated by spaces using \*

print("Translation:", end = " ")

print(\*platin\_phrase, sep = " ")

def game\_over():

#displays game over message. if user inputs yes, returns True and the loop it's

#in is continued. if not, sign\_off() run and program closes

narration("~You have perished...")

for char in "GAME OVER": #slowly each letter separated by space

print(char, end = " ")

pause(.5)

pause()

narration("\n~Enter YES to restart the location.")

#returns boolean value

if input(">>>").upper() == "YES":

loading\_screen("Restarting...\n")

narration("~You feel yourself awakening from a deep slumber...")

return True

else:

return False

def amegay\_overway():

#i just made the same function but in pig latin for the pig section

print("~Ouyay avehay erishedpay...")

pause()

for char in "AMEGAY OVERWAY":

print(char, end = " ")

pause(.5)

pause()

print("\n~Enterway ESYAY otay estartray.")

pause()

if input(">>>").upper() == "ESYAY":

loading\_screen("Estartingray...\n")

narration("~You feel yourself awakening from a deep slumber...")

return True

else:

return False

#STORY FUNCTIONS\_\_\_\_\_\_

def intro():

#intro ~graphics~ and startgame input from user

narration("\*\_\*\_\*\_\*\_\*\_\*\_\*\_\*\_\*\_WELCOME TO PYTHONLAND\_\*\_\*\_\*\_\*\_\*\_\*\_\*\_\*\_\*\n")

narration("~~~~~~~The fantasy adventure game no one asked for~~~~~~~\n")

narration("(Game is in development. Progress will not be saved.)\n")

startgame = (input("Enter in START to begin: ")).upper()

#if not START then sign\_off() runs

if startgame != "START" :

sign\_off()

else :

loading\_screen("Loading complete\n")

def character\_name():

#character name stored to be used in victory function

narration("~You feel yourself awakening from a deep slumber...")

narration("~Your memory is hazy, but you remember your name...")

narration("~Who are you?")

input\_name = input(">>> ")

narration(f"~Ah, that's right, you're {input\_name}.", input\_name)

narration("~You get up...")

return input\_name

def beach(loop):

#beach location with while loop that allows for repetition upon game\_over being called.

#loop parameter passed as is\_dream variable from main() each "if" block indented

#once from while loop corresponds to a user choice. the sailor is the only way

#to continue to next function

loading\_screen("BEACH LOADED\n")

while loop:

narration("~You find yourself on a beach")

narration("~Towards the water, you see a stranded sailor")

narration("~To your left, you see a treasure chest")

narration("~To your right, there is a flock of birds")

narration("~Where will you go?")

narration("A) To the sailor\nB) To the treasure chest\nC) To the flock of birds")

beach\_object = decision("sailor", "treasure", "birds")

narration(f"~You approach the {beach\_object}")

if beach\_object == "birds": #if birds player loses

narration("~The birds suddenly look at you, they look as if they haven't eaten in days...")

narration("~They take to the air, and begin to swarm...")

if game\_over():

continue

else:

sign\_off()

if beach\_object == "treasure": #if treasure chest immediately player loses

narration("~The chest appears to be locked...")

narration("~You try to pry it open, and hear a click, followed by a hiss...")

narration("~The chest was booby trapped, and exploded!")

if game\_over():

continue

else:

sign\_off()

if beach\_object == "sailor": #only way to win

narration("Sailor: Please! Help me fix my boat")

narration("~Will you help him? \*Enter YES to help")

sailor\_choice = input(">>> ").upper()

#if doesn't help player loses

if sailor\_choice != "YES":

narration(f"Sailor: What do you mean {sailor\_choice}???")

narration("~The sailor swings his oar...")

if game\_over():

continue

else:

sign\_off()

else:

narration("Sailor: Great! Now give me a list of your favorite numbers!")

narration("~You decide to go along...")

input\_string = input("\*input numbers separated by spaces\n>>>")

string\_list = input\_string.split()

#if invalid input, player loses

try:

#split list of strings converted to float with mapping

numbers\_list = list(map(float, string\_list))

except ValueError:

narration("Sailor: Those aren't all numbers!!")

narration("~The sailor swings his oar...")

if game\_over():

continue

else:

sign\_off()

#list comprehension for finding sum of squares

num\_squared\_list = [num \*\* 2 for num in numbers\_list]

sum\_of\_squares = sum(num\_squared\_list)

narration("Sailor: Now, what is the sum of the squares of the numbers you chose?")

#if doesn't input a number, player loses

try:

sailor\_guess = float(input("\*Enter a numerical value\n>>>"))

pause()

except ValueError:

narration("Sailor: That isn't a number!!")

narration("~The sailor swings his oar...")

if game\_over():

continue

else:

sign\_off()

#if more than a 100 off, player loses

difference = abs(sailor\_guess - sum\_of\_squares)

if difference > 100:

narration("Sailor: That's not even close!!")

narration("~The sailor swings his oar...")

if game\_over():

continue

else:

sign\_off()

elif difference > 0:

narration("Sailor: Eh, close enough")

else:

narration("Sailor: That's it!!!")

narration("Sailor: Here, take this key, it'll open that treasure chest!")

narration("~The sailor sails off into the sunset.")

narration("~You approach the treasure chest and attempt to open it...")

narration("~You put the key into the lock and turn...")

narration("~You found a map to a nearby forest!")

narration("~You walk towards the spot on the map...")

loop = False

return True #for the next function in main

def forest(loop):

#similar to the previous function. pig is the only way to win, all others result

#in game\_over(). The loop repeats only for forest, not from the beach

loading\_screen("FOREST LOADED\n")

while loop:

narration("~You arrive at the forest...")

narration("~Ahead, you see a small pig.")

narration("~To your left you see a hollowed out stump.")

narration("~To your right you see a pack of wolves")

narration("~What do you do?")

narration("A) Approach the pig\nB) Approach the stump\nC) Approach the wolves")

forest\_object = decision("pig", "stump", "wolves")

narration(f"You approach the {forest\_object}", forest\_object)

#if wolves player loses

if forest\_object == "wolves":

narration("~The wolves begin to snarl...")

narration("~Like the birds, they too have not eaten in days...")

narration("~They come bounding over...")

if game\_over():

continue

else:

sign\_off()

#if stump player loses

if forest\_object == "stump":

narration("~You hear something moving from inside the stump...")

narration("~Suddenly, a giant spider emerges")

narration("~It drags you into its den...")

if game\_over():

continue

else:

sign\_off()

#pig is only way to win

if forest\_object == "pig":

narration("~Luckily, you have your Pig Latin translator with you")

narration("~The pig begins to speak to you")

narration("Igpay: Oday ouyay eakspay Igpay Atinlay?")

narration('~"Do you speak pig latin", it said')

narration("~Say anything back to it...")

still\_translating = True #flag variable for repeating translator

while still\_translating:

pig\_latin()

narration("Igpay: upersay oolcay!")

narration("~Translate another phrase?")

narration("\*Enter YES to keep translating")

if input(">>>").upper() == "YES":

continue

else:

still\_translating = False

#pig announces he knows player is using translator and asks user what

#color he is to prove user knows pig latin. the correct answer is "inkpay"(pink)

narration("Igpay: Eyhay!! Ouyay areway usingway away anslatortray!!")

narration("Igpay: Ovepray otay emay atthay ouyay ancay eakspay Igpay Atinlay!!")

narration("Igpay: Atwhay olorcay amway Iway?")

narration("~You try to answer its question in Pig Latin...")

pig\_response = input(">>>").upper()

#if doesn't enter inkpay player loses

if pig\_response != "INKPAY":

narration(f"Igpay: Atwhay oday ouyay eanmay {pig\_response}?!")

narration("~The pig shreiks, and a herd of pigs comes charging from the forest...")

narration("~There are too many of them...")

if amegay\_overway():

continue

else:

sign\_off()

#escapes forest function

else:

narration("Igpay: OKway, ouyay ancay ebay ustedtray...")

narration("~The pig leads you through the forest to the opening of a cave...")

narration("TO (probably not) BE CONTINUED")

loop = False #ends loop

return True #returns True for possible additional locations/ victory function

def sign\_off():

#says bye and exits program

narration("Thanks for playing")

narration("Signing off")

narration("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

sys.exit(0)

def victory(name):

#signifies victory

narration(f"Congratulations {name}!")

narration("You win!")

narration("Stay tuned for more")

if \_\_name\_\_ == '\_\_main\_\_' : main()