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 continue

```
*_*_*_*_*_*_*_*_*_WELCOME TO PYTHONLAND_*_*_*_*_*_*_*_*_*

~~~~~~~The fantasy adventure game no one asked for~~~~~~~

(Game is in development. Progress will not be saved.)


Enter in START to begin: start
Loading:* * * * * * * * * * * * * * * * * * * *
Loading complete

~You feel yourself awakening from a deep slumber...
~Your memory is hazy, but you remember your name...
~Who are you?

>>> IAN
~Ah, that's right, you're IAN.
```

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

```
Loading:* * * * * * * * * * * * * * * * * * * *
BEACH LOADED

~You find yourself on a beach
~Towards the water, you see a stranded sailor
~To your left, you see a treasure chest
~To your right, there is a flock of birds
~Where will you go?
A) To the sailor
B) To the treasure chest
C) To the flock of birds

*Type A, B or C
>>> c
~You approach the birds
~The birds suddenly look at you, they look as if they haven't eaten in days...
~They take to the air, and begin to swarm...
~You have perished...
G A M E   O V E R
~Enter YES to restart the location.

>>>yes
Loading:* * * * * * * * * * * * * * * * * * * *
Restarting...

~You feel yourself awakening from a deep slumber...
~You find yourself on a beach
~Towards the water, you see a stranded sailor
~To your left, you see a treasure chest
~To your right, there is a flock of birds
~Where will you go?
A) To the sailor
B) To the treasure chest
C) To the flock of birds

*Type A, B or C
>>>
```

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

```
~You arrive at the forest...
~Ahead, you see a small pig.
~To your left you see a hollowed out stump.
~To your right you see a pack of wolves
~What do you do?
A) Approach the pig
B) Approach the stump
C) Approach the wolves

*Type A, B or C
>>> a
You approach the pig
~Luckily, you have your Pig Latin translator with you
~The pig begins to speak to you
Igpay: Oday ouyay eakspay Igpay Atinlay?
~"Do you speak Pig Latin", it said
~Say anything back to it...
*Type words separated by spaces to translate to pig latin

>>> python is fun
Translation: ythonpay isway unfay
Igpay: upersay oolcay!
~Translate another phrase?
*Enter YES to keep translating

>>>yes
*Type words separated by spaces to translate to pig latin

>>> my name is ian
Translation: ymay amenay isway ianway
Igpay: upersay oolcay!
~Translate another phrase?
*Enter YES to keep translating

>>>no
Igpay: Eyhay!! Ouyay areway usingway away anslatortray!!
Igpay: Ovepray otay emay atthay ouyay ancay eakspay Igpay Atinlay!!
Igpay: Atwhay olorcay amway Iway?
~You try to answer its question in Pig Latin...
```

```
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()
```