Session 8: Camel Game Activity[1]

The idea for Camel originally came from the Heath Users Group and was published in More BASIC Computer Games in 1979.

The idea is to ride your camel across the desert while being chased. You need to manage your thirst, how tired the camel is, and how far ahead of the enemies you are.

Sample Run of Camel

Here is a sample run of the game:

Welcome to Camel!
You have stolen a camel to make your way across the great Mobi desert.
Your enemies want their camel back and are chasing you down! Survive your desert trek and outrun your enemies.

A. Drink from your canteen.
B. Ahead moderate speed.
C. Ahead full speed.
D. Stop and rest.
E. Status check.
Q. Quit.
Your choice? C

You traveled 12 miles.

A. Drink from your canteen.
B. Ahead moderate speed.
C. Ahead full speed.
D. Stop and rest.
E. Status check.
Q. Quit.
Your choice? C

You traveled 17 miles.

[1] Taken and adapted from http://arcade-book.readthedocs.io/en/latest/labs/lab_04_camel/camel.html

A. Drink from your canteen.
B. Ahead moderate speed.
C. Ahead full speed.
D. Stop and rest.
E. Status check.
Q. Quit.
Your choice? e

Miles traveled: 29
Drinks in canteen: 3
Your enemies are 31 miles behind you.

A. Drink from your canteen.
B. Ahead moderate speed.
C. Ahead full speed.
D. Stop and rest.
E. Status check.
Q. Quit.
Your choice? b

You traveled 6 miles.

...and so on until...

A. Drink from your canteen.
B. Ahead moderate speed.
C. Ahead full speed.
D. Stop and rest.
E. Status check.
Q. Quit.
Your choice? C

You traveled 12 miles.
Your enemies are getting close!

A. Drink from your canteen.
B. Ahead moderate speed.
C. Ahead full speed.
D. Stop and rest.

E. Status check.
Q. Quit.
Your choice? C

You traveled 11 miles.
The enemies are getting close!
You made it across the desert! You won!

Instructions:

1. Create a new directory in your project for **Session 8** and a file called **camel.py** in that folder.
2. In that file create a new **main** function. Have it print the instructions to the screen. Do this with multiple **print** statements (i.e. don't use one **print** statement and multiple **\n** characters to jam everything on one line).

Example Welcome Message:

Welcome to Camel!
You have stolen a camel to make your way across the great Mobi desert.
Your enemies want their camel back and are chasing you down! Survive your desert trek and outrun your enemies.

1. Continue from the prior step and create a Boolean variable called **done** and set to **False**. Make sure this, and everything else, is in the **main** function.
2. Create a **while** loop that will keep looping while **done** is **False**.
3. Inside the loop, print out the following:
4. A**.** Drink **from** your canteen**.**
   B**.** Ahead moderate speed**.**
   C**.** Ahead full speed**.**
   D**.** Stop **for** the night**.**
   E**.** Status check**.**
   Q**.** Quit**.**

5. Ask the user for their choice. Make sure to add a space before the quote so the user input doesn't run into your text. That is, it should look like:

What is your choice? Q

And not:

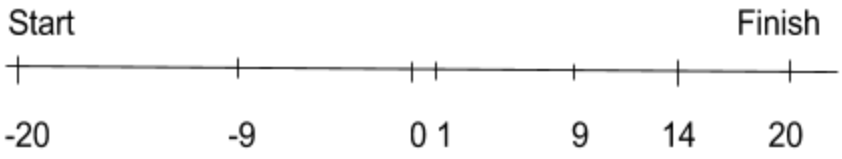
What is your choice?Q

1. If the user's choice is **Q**, then set **done** to **True**. By doing something like **user_choice.upper()** instead of just **user_choice** in your **if** statement you can make it case-insensitive.
2. Test and make sure that you can quit out of the game, and that case doesn't matter.
3. Before your loop, create variables for miles traveled, thirst, camel tiredness. Set these to zero.
4. Also before your loop, create a variable for the distance the enemies have traveled and set it to -20. (Twenty miles back.) Understanding how we track the player's distance, the enemies' distance, and the difference between the two tends to confuse some people. See the video links at the end these instructions for some hints.
5. Create and set an initial number of drinks in the canteen.

Note: Notice the difference between creating variables inside the main function versus creating them outside? It's about global versus local scope!

6. Add an **elif** in your main program loop and see if the user is asking for status. If so, print out the miles traveled, the drinks in the canteen, and how far the enemies are behind you. If you aren't sure how to calculate that, see these videos: https://youtu.be/I9dJDDBe27c and https://youtu.be/tHjwHP-lD3I

HINT: It will really help you to diagram this out as in the picture below:



Stage 0:
p_p=0
e_p=-20
m_t=0
e_t=0
e_d=20

Stage 1:
p_p=9
e_p=-9
m_t=9
e_t=11
e_d=18

Stage 2:
p_p=14
e_p=1
m_t=5
e_t=10
e_d=13

Legend for the above diagram:

- p_p: player's current position
- e_p: enemies' current position
- m_t: miles traveled by player
- e_t: miles traveled by enemies
- e_d: distance between player and the enemies

Stage 0 is the start: neither the player nor the enemy has traveled, but the enemy starts at a **magnitude of 20 miles** (meaning, the absolute distance from 0, so we can ignore the negative sign when stating the final distance) **behind** the player.

Stage 1 is an illustration of a scenario likely to play out after the player's first move: say the player decides to move forward. The player has now moved randomly by 9 miles (so they are at +9), and the enemies have also moved a random distance, namely 11 miles (so they are at -20 +11 = -9). The way we now calculate e_d is p_p, the player's current position, minus the enemies' current position, e_p. In this case, that's e_d = p_p -e_p = +9 - (-9) = 18. The negative signs will cancel each other out and become a plus sign, thus adding up to a distance between player and enemy of 18 miles.

Stage 2 works the same, even if at this point both player and enemy are into the positive numbers. In this case, the player has moved forward a random distance of 5 miles, the enemies 10, so they are at +14 (9+5) miles and +1 (-9 + 10) respectively, making for a distance of 13 miles (14-1).

1. Add an **elif** in your main program loop and handle if the user wants to stop for the night. If the user does, reset the camel's tiredness to zero. Print that the camel is happy, and move the enemies up a random amount from 7 to 14 or so.

HINT: remember the random module you learned last session!

2. Add an **elif** in your main program loop and handle if the user wants to go ahead full speed. If the user does, go forward a random amount between 10 and 20 inclusive. Print how many miles the user traveled. Add 1 to thirst. Add a random 1 to 3 to camel tiredness. Move the enemies up 7 to 14 miles.
3. Add an **elif** in your main program loop and handle if the user wants to go ahead moderate speed. If the user does, go forward a random amount between 5 and 12 inclusive. Print how many miles the user traveled. Add 1 to thirst. Add 1 to camel tiredness. Move the enemies up 7 to 14 miles.
4. Add an **elif** in your main program loop and handle if the user wants to go ahead drink from the canteen. If the user does, make sure there are drinks in the canteen. If there are, subtract one drink and set the player's thirst to zero. Otherwise print an error.
5. In the loop, print "You are thirsty." if the user's thirst is above 4.
6. Print "You died of thirst!"" if the user's thirst is above 6. Set **done** to **True**. Make sure you create your code so that the program doesn't print both "Your are thirsty" and "You died of thirst!" Use **elif** as appropriate.
7. Print "Your camel is getting tired." if the camel's tiredness is above 5.
8. Print "Your camel is dead." if the camel's tiredness is above 8. Like the prior steps, print one or the other. It is a good idea to include a check with the done variable so that you don't print that your camel is getting tired after you died of thirst.
9. If the enemies have caught up, print that they caught the player and end the game.
10. Else if the enemies are less than 15 miles behind, print "The enemies are getting close!"
11. If the user has traveled 200 miles across the desert, print that they won and end the game. Make sure they aren't dead before declaring them a winner. If they land on mile marker 201 instead of 200, make sure they still win the game. See the videos below.
12. Add a one-in-twenty chance of finding an oasis. Print that the user found it, refill the canteen, reset player thirst, and rest the camel. Make sure a person can't find the oasis unless they are traveling.

13. Play the game and tune the numbers so it is challenging but not impossible. Fix any bugs you find.

Reminders:

- Remember to indent your code correctly, or it will not run properly!
    - If you have a main function defined as your game “loop” (not the same as the loops from session 6, don’t confuse the two!), all of your code for the game should go below that definition and be indented.
    - For instance, everything you want repeated should go into your while loop.
    - If you have nested conditionals (i.e. one conditional inside of another), make sure those are properly indented.
- Somewhat related, remember that code runs one command at a time, from top to bottom. This may affect where you put certain commands in your script.
- Remember global versus local variables when you are defining and calling functions, as well as initializing variables (i.e. setting them to an initial value)!
- Make sure to comment your code, including what your variables and functions do!