Lesson 15. Capstone project: choose your own adventure

After reading lesson 15, you'll be able to

- Write code for a choose-your-own-adventure program
- Use branches to set up paths through the program

This capstone project is somewhat open-ended.

The problem

You'll use conditionals and branching to create a story. At each scene, the user will enter a word. The word will tell the program which path to continue following. Your program should handle all possible paths that the user might choose, but doesn't need to handle any unexpected input from the user.

The walk-through you'll see is one of many possible others; be as creative as you want with your storyline!

15.1. OUTLINING THE GAME RULES

Anytime you're getting input from users, you should be aware that they might not play by the rules. In your program, specify what you expect from them and warn them that anything else might make the program end.

A simple `print` statement will suffice, as in the following listing.

Listing 15.1. Getting user input

```
print("You are on a deserted island in a 2D world.")
print("Try to survive until rescue arrives!")
print("Available commands are in CAPITAL letters.")
print("Any other command exits the program")
print("First LOOK around...")
```

- ***1* How to play**
- ***2* Unexpected behavior closes the program.**

According to the program rules, you'll have to handle branches for any input in capital letters. There's only one option at the start of the program, to help the user get accustomed to this input format.

15.2. CREATING DIFFERENT PATHS

The general flow of the program is as follows:

- Tell users the choices they have.
- Get user input.
- If the user puts in choice 1, print a message. For this path, if the user now has more choices, indicate the choices, get input, and so on.
- Otherwise, if the user puts in choice 2, print another message. For this path, if the user now has more choices, indicate the choices, get input, and so on.
- And so on, for however many choices there are. For each path, if the user now has more choices, indicate the choices, get input, and so on.

You'll use nested conditionals to create subpaths within paths. One simple program is shown in listing 15.2. It goes only two conditionals deep; one nested conditional is inside another. The user can make at most two choices when running the program once.

The code listing begins by asking for user input. Then it makes sure the user understands the rules of the game with a conditional for the keyword LOOK.

whether the user typed LOOK. If the user did, the code gets user input again, and handles one of two possibilities from that input: that the user typed either LEFT or RIGHT. The code prints a different message for these choices.

Listing 15.2. An adventure with only one choice

```
do = input(":: ")
if do == "LOOK":
    print("You are stuck in a sand ditch.")
    print("Crawl out LEFT or RIGHT.")

    do = input(":: ")
    if do == "LEFT":
        print("You make it out and see a ship!")
        print("You survived!")
    elif do == "RIGHT":
        print("No can do. That side is very slippery.
        print("You fall very far into some weird cave
        print("You do not survive :(")
else:
    print("You can only do actions shown in capital l
    print("Try again!")
```

- **1 Input from user**
- **2 Conditional if user types LOOK**
- **3 Input from user after user LOOKed**
- **4 Conditional to check if user types LEFT**
- **5 Conditional to check if user types RIGHT**
- **6 A block to remind users that they can type only certain commands**

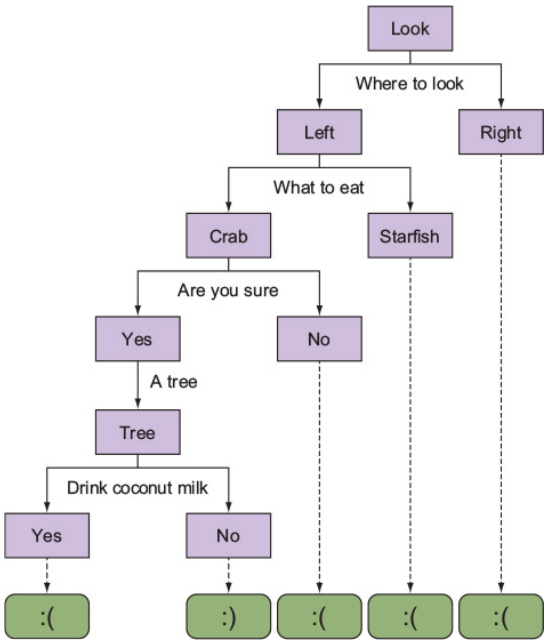
Two choices in a program don't sound fun. You can add more choices for different scenarios.

15.3. MORE CHOICES? YES, PLEASE!

A choose-your-own-adventure game should have more than one or two choices. Use many nested conditionals to create many subpaths through the code. You can make the adventure as easy or as hard as you want; for example, out of 20 possible paths through the code, maybe only one leads to survival.

Figure 15.1 shows one possible code structure. A decision is marked by the user entering a word. Depending on the chosen word, the user will see a new situation for that chosen path. The user will continue making choices until the final outcome is reached.

Figure 15.1. The boxes represent choices for the user. The text below the boxes represents the situation. The gray arrows show the path with the choice made. The dotted black lines to a smiley face or a sad face represent the end of the program, with survival or not. Out of five possible outcomes, only one leads to survival.



The following listing provides the code associated with figure 15.1. Only one path leads to survival. The user must enter LEFT, then CRAB, then YES, then TREE, and then NO. Any other choice leads to text indicating that the user has perished on the island.

Listing 15.3. One possible choose-your-own-adventure code

```
print("You are stuck in a sand ditch.")
print("Crawl out LEFT or RIGHT.")

do = input(":: ")
if do == "LEFT":
    print("You see a STARFISH and a CRAB on the sand.
    print("And you're hungry! Which do you eat?")

    do = input(":: ")
    if do == "STARFISH":
        print("Oh no! You immediately don't feel well
        print("You do not survive :(")

    elif do == "CRAB":
        print("Raw crab should be fine, right? YES or

        do = input(":: ")
        if do == "YES":
            print("Ok, You eat it raw. Fingers crosse
            print("Food in your belly helps you see a

            do = input(":: ")
            if do == "TREE":
                print("It's a coconut tree! And you'r
                print("Do you drink the coconut water

                do = input(":: ")
                if do == "YES":
                    print("Oh boy. Coconut water and
                    print("You do not survive :(")

                elif do == "NO":
                    print("Good choice.")
                    print("Look! It's a rescue plane!

        elif do == "NO":
            print("Well, there's nothing else left to
            print("You do not survive :(")

elif do == "RIGHT":
    print("No can do. That side is very slippery.")
    print("You fall very far into some weird cavern."
    print("You do not survive :(")
```

- ***1*** **First choice**
- ***2*** **Choice for if-branch of *1***
- ***3*** **Choice for elif-branch of *1***
- ***4*** **Nested choice, for if branch of *3***
- ***5*** **No choice, only one possibility**
- ***6*** **Choice for *5***
- ***7*** **Nested choice for elif-branch of *3***
- ***8*** **First choice**

SUMMARY

In this lesson, my objective was to teach you to use conditionals to write a program in which the user makes choices to try to survive the scenario outlined at the beginning of the program. To create paths for different choices that the user can make after having already made a choice, you used nested conditionals. Here are the major takeaways:

- Conditionals offer choices to the user.
- Nested conditionals are useful for offering a different set of choices after making one choice.