How to Write a Text Adventure in Python Part 4: The Game Loop – Let's Talk Data

By Phillip Johnson

This is an abbreviated version of the book Make Your Own Python Text Adventure.

The end is near, we're almost ready to play the game! We'll finish this series by implementing the game loop and receiving input from the human player.

The Game Loop

While some applications follow a discrete set of steps and terminate, a game typically just "keeps going". The only way the program stops is if the player wins, loses, or quits. To handle this behavior, games usually run inside a loop. On each iteration, the game state is updated and input is received from the human player. In graphical games, the loop runs many times per second. Since we don't need to continually refresh the player's screen for a text game, our code will actually pause until the player provides input. Our game loop is going to reside in a new module `game.py`.

```
import world
from player import Player
def play():
    world.load_tiles() player = Player()
    while player.is_alive() and not player.victory:
```

Before play begins, we load our world from the text file and create a new `Player` object. Next, we begin the loop. Note the two conditions we check: if the player is alive and if victory has not been achieved. For this game, the only way to lose is by dying. However, there isn't any code yet that lets the player win. In my story, I want the player to escape the cave alive. If they do that, they win. To implement this behavior, we're going to add a very simple room and place it into our world. Switch back to `tiles.py` and add this class:

```
class LeaveCaveRoom(MapTile):
    def intro_text(self):
        return
    def modify_player(self, player):
        player.victory = True
```

Don't forget to include one of these rooms somewhere in your map.txt file. Now that the player can win, let's finish the game loop.

```
def play():
    world.load_tiles()
    player = Player()
    room = world.tile_exists(player.location_x, player.location_y)
    print(room.intro_text())
    while player.is_alive() and not player.victory:
        room = world.tile_exists(player.location_x, player.location_y)
        room.modify_player(player)
        if player.is_alive() and not player.victory:
            print("Choose an action:\n")
            available_actions = room.available_actions()
            for action in available_actions:
                print(action)
            action_input = input('Action: ')
            for action in available_actions:
                if action_input == action.hotkey:
                    player.do_action(action, **action.kwargs)
                    break
```

The first thing the loop does is find out what room the player is in and then executes the behavior for that room. If the player is alive and they have not won after the behavior executes, we prompt the human player for input. This is done using the built-in `input()` function. If the human player provided a matching hotkey, then we execute the associated action using the `do_action` method.

The last thing we need to include is an instruction for Python to know that `play()` should run when running the file. Include these lines at the bottom of the `game.py` module:

```
if __name__ == "__main__":
    play()
```

To run the program, navigate to the folder containing the adventuretutorial package in your console and run `python adventuretutorial/game.py`. If you get warnings about packages, try setting your PYTHONPATH environment variable manually. Have fun!

Where to go from here

Congratulations! You now have a working text adventure game. With the information learned here, you should be able to quickly add your own custom items, enemies, and tiles. If you're up for more of a challenge, here are some of the features included in *Make Your Own Python Text Adventure*:

- An easier and more flexible way to build your world (no text files or reflection!)
- A game economy where the player can buy and sell items
- The ability for players to heal during and between fights
- Difficulty settings to make the game harder or easier