# Chapter 5 - Oregon Trail Text Adventure Game

## Contents



Time required: 90 minutes

- Comment each line of code as shown in the tutorials and other code examples.
- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

## The Game Plan

Computer games started in the 80's with text adventure games. Zork, The Dreamhold, The Hobbit, Spider and Web just to name a few. It wasn't about the glitzy graphics; it was and still is about the story.

We are going to build a text-based choose your own adventure game. This is a sample of how a text adventure game works.


Now as you can see above in the game map, we first **start** the game. Tell the player a story like "You are standing in a dark room. There is a door to your left and right, which one do you take? (l or r)". If the player types "l", then we lead him to the **bear_room**, or if he/she types "r", then we lead him to the **monster_room** like that.

You can easily guess how the game works by looking at the map alone. To build this game in python, we need to take the `input()` from the user after showing some prompts like "you are in a _ room". Then lead the player according to his inputs. To make our work again simple, we are going to use `functions` in python3.

## What You Will Learn

As part of this project, you will learn the following

- How to work with `functions` in python3.
- How to take `input()`.
- How to `print()` output.
- `if`, `elif`, `else` statements.
- `==` equality operator.
- `lower()` function to convert the string into a lower case.
- And much much more.

## What are Functions?

Imagine you have a **cake-making robot**! For the robot to make a cake, you should give certain commands. Assume that the following codes are those "certain commands". If you want to type this, feel free to do so.

```
print("Mix Ingredients for one cake")
print("Add Vanilla flavor")
print("Bake the cake")
print("Serve the cake")
```

Let's assume if we call the `print()` function, the robot will do that thing

By using the above commands, the robot will make only **one cake** of **vanilla flavor**, right? What would you do if you want **5** cakes?

You can do like this:

```
print("Mix Ingredients for one cake")
print("Add Vanilla flavor")
print("Bake the cake")
print("Serve the cake")

print("Mix Ingredients for one cake")
print("Add Vanilla flavor")
print("Bake the cake")
print("Serve the cake")

print("Mix Ingredients for one cake")
print("Add Vanilla flavor")
print("Bake the cake")
print("Serve the cake")

print("Mix Ingredients for one cake")
print("Add Vanilla flavor")
print("Bake the cake")
print("Serve the cake")

print("Mix Ingredients for one cake")
print("Add Vanilla flavor")
print("Bake the cake")
print("Serve the cake")
```

This doesn't look like a very efficient way to create a program. We are going to **extract the pieces of code that needs repetition** and put it under a **particular name** like below:

```
def make_cake():
  print("Mix Ingredients for one cake")
  print("Add Vanilla flavor")
  print("Bake the cake")
  print("Serve the cake")
```

This is a `function`. To define a function in python, we use the `def` keyword, and following that we give the `name` of our function and the brackets - `()`. Then after semicolon(`:`), we give the function body from the next line with **indentation**.

If we want **5 cakes**, you don't have to hard code them. Instead, you can call the name of the function **5** times!

```
def make_cake():
    print("Mix Ingredients for one cake")
    print("Add Vanilla flavor")
    print("Bake the cake")
    print("Serve the cake")

make_cake()
make_cake()
make_cake()
make_cake()
```

What would you do if you want **5** cakes with **5** different flavors like **vanilla**, **chocolate**, **orange**, **banana**, and **strawberry**?

If you are using `functions`, you can do it like this:

```
def make_cake(flavor):
    print("Mix Ingredients for one cake")
    print("Add " + flavor + " flavor")
    print("Bake the cake")
    print("Serve the cake")

make_cake("vanilla")
make_cake("chocolate")
make_cake("orange")
make_cake("banana")
make_cake("strawberry")
```

The thing inside the brackets is what's called an `argument`. At the time of calling the function `make_cake()`, we can supply that `argument` like `make_cake(argument)`. We can use it inside the function body wherever you want.

```
def make_cake(flavor):
  print("Mix Ingredients for one cake")
  print("Add" + flavor + "flavor")
  print("Bake the cake")
  print("Serve the cake")
```

argument

argument in function

```
def make_cake(flavor):
  print("Mix Ingredients for one cake")
  print("Add" + flavor + "flavor")
  print("Bake the cake")
  print("Serve the cake")
```

making use of the argument

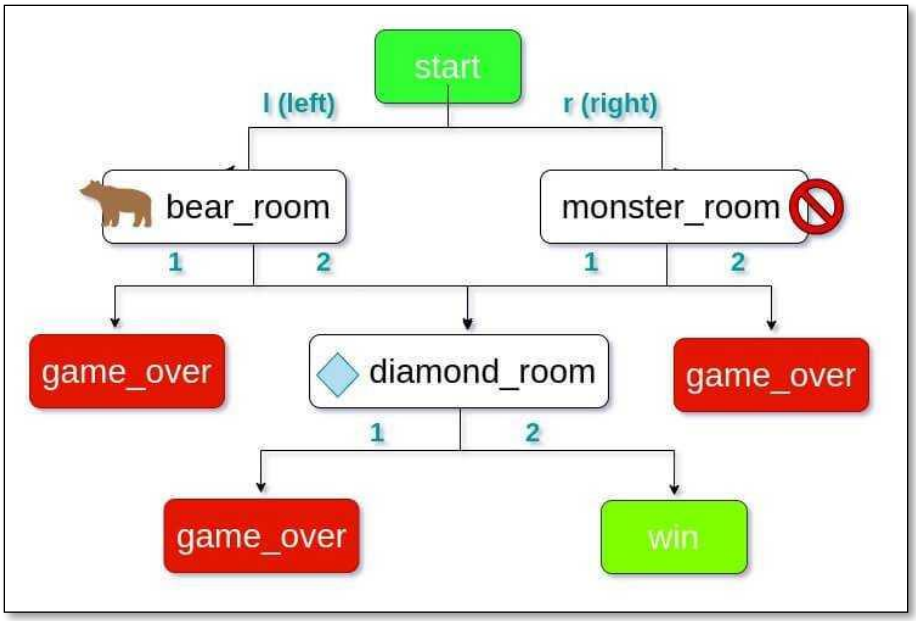
making use of the argument inside function body

You can give more than one `argument` like this - `make_cake(flavor, baking_time, something_else, something_else)`.

## The Game

We are going to make our game according to the game map. Feel free to be creative with the story, game room names, etc.

Dragon Realm is a similar text-based adventure tutorial. It has some random choices in it, which makes a game much more interesting. Look at it to get some ideas on how to make this tutorial more interesting.




If you look at it, you can see there are many boxes. Think of each box as a **room** and as a **function** except the **win** box. Let's start by creating a function called `start()`.

1. Create a Python file named **adventure.py**
2. Type the following code inside the `adventure.py` file:

```
#---------------------- START -----------------------#
def start():
    # Prompt the user for input
    print("\nYou are standing in a dark room.")
    print("There is a door to your left and right")
    print("Which one do you take? (l or r)")

    # Convert the player's input() to lower_case
    answer = input("> ").lower()

    if "l" in answer:
        # If player typed "left" or "l" lead him to bear_room()
        bear_room()
    elif "r" in answer:
        # Else if player typed "right" or "r" lead him to monster_room()
        monster_room()
    else:
        # else call game_over() function with the "reason" argument
        game_over("Don't you know how to type something properly?")


# If a standalone program, call the main function
# Else, use as a module
if __name__ == '__main__':
    main()
```

## What is Going on Here?

- The `start()` function is the beginning of the game. We are prompting the player where he/she is currently standing and letting them know the options available to him/her.
- We take the `input()` whatever the player types and convert it into a `lower()` case string.
- Check if `"l"` is in the player's input. If the player typed `"left"` or `"l"`, lead him to the `bear_room()`.
- If the player typed `"right"` or `"r"`, lead him to the `monster_room()`.
- Else, if the player typed something else, call `game_over()` function with an `argument` called `reason`. We have to call this `game_over()` function in so many places whenever the player's game is over. The `reason` may be different in each situation. That's why we have to take it as an `argument`.
- In the main function, activate the `start()` function to begin the game.

We are not going to run this game yet. If we do so, we will get a bunch of errors. Because we have only called the `bear_room()`, `monster_room()`, and `game_over()` functions but haven't created it yet.

## Bear Room

Let's create the `bear_room()`. Type the following code above the `start()` function definition:

```python
#---------------------- BEAR ROOM -----------------------#
def bear_room():
    # Tell the story to this point
    # Prompt the user for input
    print("\nThere is a bear here.")
    print("Behind the bear is another door.")
    print("The bear is eating tasty honey!")
    print("What would you do? (1 or 2)")
    print("(1). Take the honey.")
    print("(2). Taunt the bear.")

    # Get user input
    answer = input("> ")

    if answer == "1":
        # The player is dead!
        game_over("The bear killed you.")
    elif answer == "2":
        # Lead him to the diamond_room()
        print("\nThe bear moved from the door. You can go through it now!")
        diamond_room()
    else:
        # Else call game_over() function with the "reason" argument
        game_over("Don't you know how to type a number?")
```

- We give some messages to the player to describe the current situation.
- We take the player's choice as `input()`.
- We check if the player typed `"1"` or `"2"` or anything else.
- If he/she typed `"1"`, then the game is over. We call the `game_over()`.
- Else if he/she typed `"2"`, lead them to the `diamond_room()`. We have to create it too.
- Else, call the `game_over()`.

The `bear_room()` is ready, let's head towards creating the `monster_room()`, `diamond_room()`, and the `game_over()`.

## Game Over

Now create the `game_over()` function above the previous function:

- See this has that `play_again()` function too.

```
#---------------------- GAME OVER -----------------------#
def game_over(reason):
    # print the "reason" in a new line (\n)
    print("\n" + reason)
    print("Game Over!")
    # ask player to play again or not by activating play_again() function
    play_again()
```

## Play Again

Create the `play_again()` function above the `game_over()` function:

```
"""
    Name: adventure.py
    Author:
    Created:
    Purpose: A text adventure tutorial
"""
import sys

def main():
    # Start the game
    start()


#---------------------- PLAY AGAIN -----------------------#
def play_again():
    print("\nDo you want to play again? (y or n)")

    # convert the player's input to lower_case
    answer = input(">").lower()

    if "y" in answer:
        # if player typed "yes" or "y" start the game from the beginning
        start()
    else:
        # if user types anything besides "yes" or "y", exit() the program
        sys.exit()
```

You have just created an awesome yet simple text-based choose your own adventure game in Python using functions!

Hooray, it's time to run the game!

## Assignment: Create Your Own Adventure Game

You can use the example above to get started or start on your own adventure game!

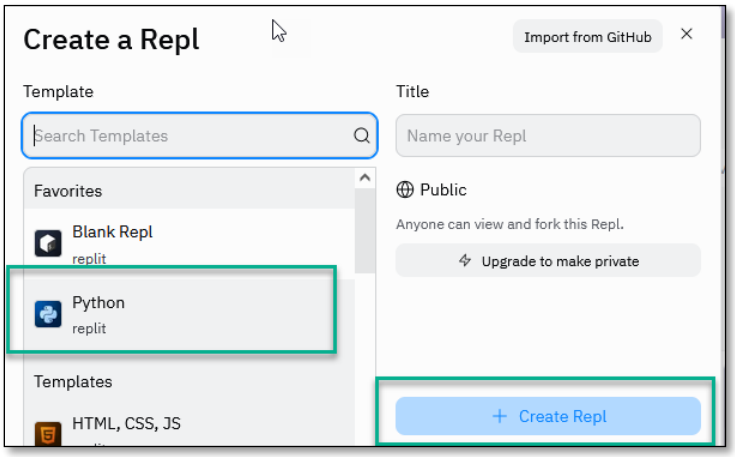
1. Your game must have a minimum of 4 rooms.
2. Sketch out your adventure game. Attach to the assignment.

3. Be creative with your game.

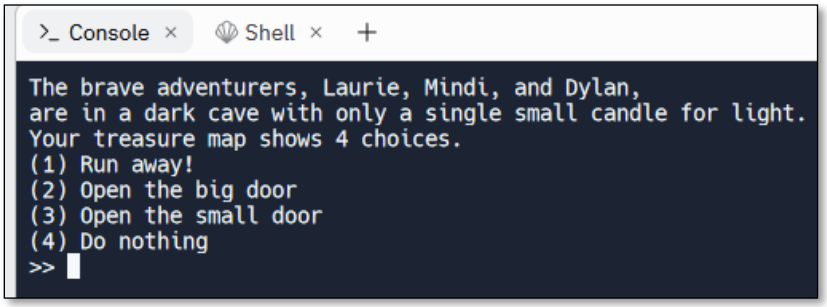
## Repl.it

Many times, people have a lot of fun with this assignment and want to share what they have done with friends and family.

1. Go to https://replit.com/
2. Sign Up for a free account.
3. When you login you should see a **Create** button on the left-hand side.
4. Choose **Python** → **Create Repl**. It will take a few moments for the rpl to boot.

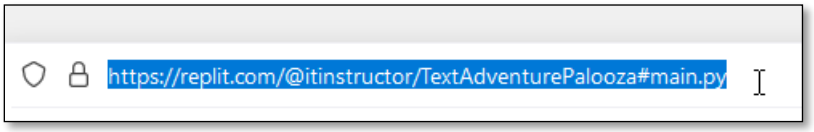



1. Repl automatically creates a main.py file.
2. Copy the code from your assignment → Paste it into the file.
3. Click the **Run** button. Your program will run on the right hand side.




4. In the address bar at the top of your browser → you will see the link to your program.

5. Copy the link. You can email or send this link for anyone to see your text adventure.




## Assignment Submission

- Attach the game rooms sketch.
- Attach the pseudocode.
- Attach the program files.
- Attach screenshots showing the successful operation of the program.
- Submit in Blackboard.