Lesson 5: Create your own Guessing Game

In this lesson, we'll use what we learned from the previous lesson to make a new game!

This game will be similar to the previous guessing game, but will also involve *functions*.

Like we learned earlier, functions are pieces of code that can be run, or *called*, to perform certain tasks, like printing, or making random numbers.

Here, we'll learn how to actually define and use a function.

PART 1: Defining a Function

Before we define our function, we'll first `import` the random module into this program.

After that, we can define a function using the `def` command, like below.

Try typing the following into a new program:
(Try making your own theme and messages for this guessing game!)

```
import random

def play():
    print("You are walking in an ice storm back to camp.")
    print("You see 3 ice bridges ahead. They look dangerous.")
```

By itself, defining a function won't run the code inside, so you won't be able to see the messages appear until you call the `play()` function.

PART 2: Creating a Loop

Like in the number guessing game, we'll need to use a loop, and repeatedly ask for the user to guess a number.

This time though, we'll have a smaller selection of numbers, since choosing the same number as the randomly-created one will cause the player to lose!

Try adding the following code to the end of your program (inside the function):

```
def play():
    .  .  .
    print("You see 3 ice bridges ahead. They look dangerous")
    alive = True
    score = 0
    while alive:
        number = random.randint(1,3)
        print("Choose bridge 1, 2, or 3.")
        guess = int(input())
```

Make sure the lines you add are indented properly!
Also, don't forget the `:` at the end of the `while` line.

So far, the program will only be able to keep asking which bridge to try crossing.

PART 3: Adding a Conditional to the Loop

In order to make our program into a game, we need to make a way for the user to lose. We can do this by checking if the user's guess equaled the losing bridge number.

Try adding this code to the end of your program (inside the while loop):

```
def play():
    .  .  .
    while alive:
        .  .  .
        guess = int(input())
        if (guess == number):
            print("Crack -- Crash -- Bye, byeeeeeeeeee!")
            alive = False
```

Now, our program has a way to tell if the user has lost, and it can exit the loop as well, because the `while` loop will no longer repeat when `alive` is set to `False`.

After they lose, the program will run whatever code is after the `while` loop's block.

PART 4: Tracking the Player's Score

If the user chose a safe bridge, we want to let them know, and increase their score. To do this, we'll use an `else` statement after checking if they guessed incorrectly.

Try typing this code after the if statement:

```
def play():
    .  .  .
    while alive:
        .  .  .
        if (guess == number):
            .  .  .
            alive = False
        else:
            print("Nice job! You are safe for now...")
            print("There are more bridges ahead.")
            score += 1
    print("Game Over!  You scored " + (str)(score) + ".")
```

Make sure the last line starts at the same indentation as the while loop!

We can use the form `variable += 1` to add 1 to a variable.
This is the same as typing `variable = variable + 1`

Now, when the user guesses a number, they'll either gain a point and be congratulated, or they'll lose, and get a game over message showing their final score.

PART 5: Calling the Play Function

We're almost done, but we still have to add a bit more code to make our program work!

Before our program will run our game, we have to actually call the `play()` function

Try typing in this line at the end of your program:

```
def play():
    .  .  .
    print("Game Over!  You scored " + (str)(score) + ".")
play()
```

Make sure the last line is indented all the way to the left!

By adding the `play()` line, we're telling Python to run the code inside the function.

When we *defined* the play() function earlier, we were only telling our program what to do if the function got called.

When we use `import`, Python loads the functions defined in a module. Just like `import`, using `def play()` won't run any code by itself. This is why we have to call the `play()` function after it's defined.

Now you should be able to play your own guessing game!

PART 6: BONUS: Adding a Replay Option

Since we set up our game to run whenever we call the `play()` function, we can now add the option of replaying the game when the user loses.

In order to do this, we'll just add some code around the place where we call `play()`.

Try modifying the end of your program to look like this:

```
def play():
    .  .  .
    print("Game Over!  You scored " + (str)(score) + ".")
playgame = True
while playgame:
    play()
    again = input("Would you like to play again? ")
    if(again != "yes"):
        playgame = False
```

Notice that code for calling the `play()` function is now indented!

Here, we've added a new variable to represent whether we want to play the game.

We can use this in a new while loop, which asks the user whether they want to replay the game after it finishes the first time.

If they don't type yes as their response, the playgame variable will be changed to False, which will cause the loop to exit, and the program to close.

Now, if you test your program, you should be able to replay the game after losing!

PART 7: BONUS: Creating your own Game

Using what you've learned from Lesson 5, you can now make your own guessing game! In this new game, you can use a new theme and reward the player for making progress.

Make a new file and save it with a new name.

Have your game incorporate the following:

- Player is in an exciting setting.
- Player must choose from 3-5 paths or items on each level. All but one of these will provide a reward. The other one will end the game.
- Player can decide at each level if they wish to stop or continue playing.
- A player's total score is shown after each choice (level).

As an example, your game could use the following story:

"You are in an old castle. There are many floors. Each floor has 4 doors. Behind 3 of the doors there is a random amount of money. Behind the other door is a dragon who will eat you and take your money. If you open a door and collect money, you may decide to continue for more bounty or leave with your winnings. Good luck."

Try using what you've learned from Lesson 5 to create a new game!

Here are some hints and tips:

- Add code to award random amount of money to player for each correct guess

- Add code to let the user type 0 if they wish to end the game:

```
if (guess == 0)
   break
```

- Add code to print one of two messages at end of game:

```
if alive == True:
   print(some sort of goodbye to player who leaves game by pressing 0)
else:
   print(some sort of goodbye to player who lost game)
```