**Games.py – part 3**

The final file to create is ‘games.py’ and this is the shortest since all it does is act as a wrapper for the two games we’ve built. There are only three functions and two of them should be quite familiar by now. Before I go over each function we’ll start as always by constructing the skeleton.

import sys

import number\_guess

import rps

def menu\_ui():

def menu():

def main():

main()

First we import the ‘sys’ module. There is a good reason for this and I’ll explain why a little later. We also import ‘number\_guess’ and ‘rps’, the two games we’ve built.

**Note**

For anyone using your own python environment on windows, mac or linux, you’ll need to add a file called ‘\_\_init\_\_.py’ in the same folder as the three scripts we’ve made. This is a necessary step if you want python to be able to find your own modules. For those using repl.it this will be done automatically.

‘menu\_ui()’ and ‘menu()’ handle our menu interface and logic respectively while ‘main()’ will determine which game to play based on the player’s choice in ‘menu()’.

We’ll build ‘menu\_ui()’ first since it is the simplest.

return '''

----------------------------------

What would you like to play?

Enter a number to select an option

----------------------------------

1 - Guess the Number

2 - Rock Paper Scissors

3 - Quit

----------------------------------

'''

Now we can concentrate on ‘menu()’ which is the longest function in this script. Again it is practically identical to ‘difficulty()’ and ‘choice()’ from before but with a few changes. Since I’ve already broken it down twice I’ll give you the complete code all at once.

print(menu\_ui())

while(True):

# Handle invalid input type

try:

user\_input = int(input("-> "))

except:

print("Invalid input, please try again")

continue

# If input is a string or is not 1-3 display error

if((user\_input < 1) or (user\_input > 3)):

print("Invalid input, please try again")

else:

if(user\_input == 1):

print("Guess the Number selected\n")

return 1

elif(user\_input == 2):

print("Rock Paper Scissors selected\n")

return 2

else:

print("Quit selected\nExiting...")

sys.exit()

Aside from the adjusted ‘print’ statements you may notice that we call ‘sys.exit()’ if the player chooses option 3. This function allows us to exit the script and return to the command line (if you’re using your own environment at least).

Finally we can complete the last function in our application, ‘main()’. Type the following inside it.

while(True):

game\_choice = menu()

We enter an infinite loop so that once the player has completed playing one game they’ll be able to choose to play it again or play a different one. Once the player has had enough they can quit by choosing option 3. So we get the player’s choice from ‘menu()’ which will be either 1 or 2 and from this we can send the player to the appropriate game. Within the loop add the following.

# Play game depending on choice

if(game\_choice == 1):

number\_guess.guess\_the\_number()

else:

rps.rock\_paper\_scissors()

And with that our little application is complete. You should be able to run ‘games.py’ and hopefully it work as intended. As a final note this application can be easily expanded with other games or utilities. All you would need to do is import your scripts and modify the functions to handle more options.