Lab: Functions

For this lab the exercises involve the *number_guess_game* you created in the last couple of labs:

Take the number guess game and break it up into a number of functions.

There is not necessarily a right or wrong way to do this; look for functions that are meaningful to you within the code, for example:
1. You could create a function to obtain the guesses from the user. This function could verify if the input is a number.
2. You could have a function to obtain the y or n input from the user when deciding whether to continue to play the game. This function could verify that the input is either y or n.
3. You could create another function that will implement the main game playing loop.
4. You could have a function to welcome the user to the game
5. You could create a function to display the end of game message.
6. You could define a function that will display a set of instructions to the user. This function could check to see if the user wants the instructions printed or not.
7. You could create a function to generate the initial number to guess from a range (such as generate_number_to_guess(start,end). The start and end could have default values etc.

The end result should appear to behave in exactly the same way as the original version; however, it should now be more modular and easier to maintain.

Next ensure that you have a function that allows the user to enter their guess. So far we have assumed that the input will be a string that can be converted into an integer.

We can now add some logic to check that the input is valid. To do this write a function to obtain the user input (for example get_user_input(prompt)).

This function can check the input using the isdigit() operation on a string, for example:

```
if not user_input.isdigit():
    print('Input must be a number')
```

The function should not return a value until the user has entered a valid value.

Now check any other functions you have that allow the user to enter information and put in place any appropriate validation code.