**4. Iteration (condition loops)**

**Task 2:**

**Plan:**

1. Necessary modules are imported, and variable are assigned with their initial values.
2. A random integer is accurately generated within range 100.
3. User is instructed on what to do, e.g. “Guess number between 1 and 100”.
4. User enters their guess, there input is then validated, if the user’s guess is not valid, they are asked to enter there guess again repeatedly until their input is valid.
5. Once the users input has passed the validation test, it is check against the hidden number that was generated in step 2. If the input is greater than or lower than the hidden number, an associated message is output telling the user to “Guess lower!“ or “Guess higher!”. The program then repeats from step 4, asking the user to enter a guess.
6. Once the user has entered a guess that is equal to the hidden number, the guessing game will end and the user will be informed about how many guess it took them.

**Pseudocode:**

Import random

HiddenNumber 🡸 random.randint(1,100)

NumberGuesses 🡸 0

Guess 🡸 “0”

Print(“Guess the random number from 1 to 100:”)

While Guess != number

Guess 🡸 input(“>>> ”)

While Guess.isdigit() != True:

Guess = input(“Not a Valid Input!\n>>> ”)

[print(“Higher”) if user < number else print(“Lower”)]

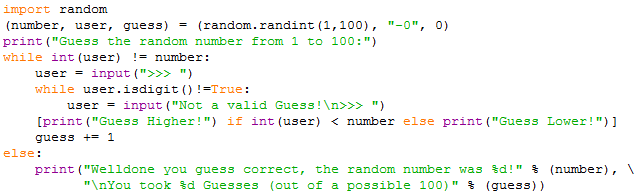
NumberGuesses += 1

Else:

Print(“Well-done you guessed correctly! You took”,NumberGuesses,”Guesses!”)

**Variables table:**

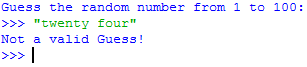
|  |  |  |
| --- | --- | --- |
| Variable Name | Data type | Comment / use |
| HiddenNumber | Integer | Stores the hidden number the user is trying to guess. |
| NumberGuesses | Integer | Counter variable, adds 1 after every Guess. |
| Guess | String | Stores the users guess. |

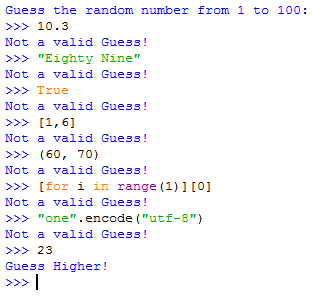
**Screenshot evidence:**

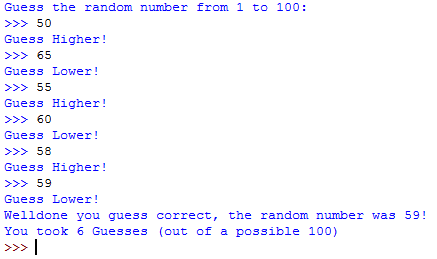
The in the screenshot above first the random module is imported; this allows the hidden number to be generated, some variable are then assigned (number, user, & guess) and the user is told to “Guess the random number from 1 to 100”. The user then enters their guess and their input is validated using the checkdigit function, if the users guess is not valid, they are asked to re-enter it until it is valid. Once the users guess is valid, it is check against some condition in the list comprehension show in the screenshot above, and an associated message is output, e.g. “Lower” or “Higher”. Then +1 is applied to a counter variable named “guess”. Finally, the users input is checked to see if it is equal to the hidden number, if it is, the program ends and a message is output telling the user how many guess it took them, else the program repeats from asking the user to input their guess.

**Sample run (Output):**

Validation error message:

the screenshot to the left, show the validation error message when an incorrect input is entered. A non-valid input can be anything that is not an integer.

The screenshot to the left shows some invalid inputs. I start by entering a float, then a string, then a Boolean True value, followed by a list, a tuple, then a list comprehension, then an encode string, until I finally enter an integer which is accepted.



The screenshot above shows a game of Guess the number, I start by entering ‘50’ and the program tells me to “Guess Higher!”, I then enter 65 and I am told to “Guess Lower!”. This narrows down the hidden number to 15 possibilities. I then enter ‘55’ and ‘60’ narrowing it down further to a possible 5 numbers. I then start by entering random numbers between 55 and 60 until I finally guess 59. Once I have guessed the hidden number “59” the program stops and tells me how many guesses I took.

**Flowcharts:**

START

END

NumberGuesses += 1

[print(“Higher”) if user < number else print(“Lower”)]

Valid END while

Not valid

Guess = input(“Not a Valid Input!\n>>> ”)

While Guess.isdigit() != True

Guess 🡸 input(“>>> ”)

Print(“Well-done you guessed correctly! You took”,NumberGuesses,”Guesses!”)

END while

(Loop) While Guess != number

Print(“Guess the random number from 1 to 100:”)

NumberGuesses 🡸 0

HiddenNumber 🡸 random.randint(1,100)

Import random