Computing

Lesson 1: While Loops

Programming Part 3: Iteration

Rebecca Franks

[1] *Materials from the Teach Computing Curriculum created by the National Centre for Computing Education*

Guess the number game

Syntax checklist

Use this **checklist** to help you fix any errors in your code.

- [ ] misspelled `print`, `if`, `elif`, `else` or `while` (this includes using capitals)
- [ ] forgot the colon `:` at the end of a line containing `while`, `if`, `elif` or `else`
- [ ] neglected to **indent** statements in the `while`-block, `if`-block, `elif`-block or `else`-block
- [ ] indented `while`, `if`, `elif` or `else` by mistake
- [ ] used `=` instead of `==` in a condition, to check if two values are equal
- [ ] used quotes around the name of a variable
- [ ] forgot to use quotes around a string literal (like `"Monday"`)
- [ ] forgot to use the `f` to display an fstring `print (f"{guess}")`

Task 1: Open the start program

Open this starting program **oaknat.uk/comp-ks4-guessnumlivecode** in Repl.it.

Note that this is the same program that I just live coded with you.

```
number = 4
print("Guess a number between 1 and 10")
guess = int(input())
while guess != number:
  print("Incorrect")
  print("Guess a number between 1 and 10")
  guess = int(input())
print("Correct")
```

Task 2: Introduce a variable to count the guesses

Checklist:

- [ ] Initialise the variable `guesses` first at the top of your program.
- [ ] Decide what line of code will increment the variable by 1 after each guess.
- [ ] Place the line of code in the most appropriate place.
- [ ] Test your new code by **printing** the variable to see if it increments after each guess.

Tip: in order to test your code is working you should print the variable `guesses` within your loop. When you are testing your code it should output the number of guesses made and this should increment each time. See the sample output on the next slide.

Task 2: Introduce a variable to count the guesses

```
Guess a number between 1 and 10
2
1 <- this displays what is being held in guesses
Incorrect
Guess a number between 1 and 10
8
2 <- this displays what is being held in guesses
```

Task 3: Display the number of guesses

- [ ] At the end of the game it should reveal the number of guesses made by the user
- [ ] Test your program using the table below as a guide

Example

Note: Use this example to check your program. Given the input you see in this sample interaction, this is the output your program should produce.

The program displays a prompt and waits for keyboard input.	`Guess a number between 1 and 10`
The user types in a reply.	`7`
The program checks if 7 is not equal to 4. This is True so it displays.	`Incorrect` `Guess a number between 1 and 10`
The user types in a new number.	`4`
The program checks if 4 is not equal to 4. This is False so it displays.	`Correct` `You guessed it in 2 attempts`

Task 4: Only allow 3 guesses

Currently your program will allow an infinite number of guesses. This needs to now be limited to just 3 guesses.

- **Change** the condition on the while loop so that it will only continue to the next iteration if they haven't guessed correctly **and** their number of guesses is less than 4.
- Use the table on the next slide to test your program.

* A while loop isn't constantly evaluating the condition. It executes the entire sequence within the loop before re-evaluating the condition. This will affect the value that you use for the number of guesses in the condition.

Task 4: Only allow 3 guesses

Example

Note: Use this example to check your program. Given the input you see in this sample interaction, this is the output your program should produce.

The program displays a prompt and waits for keyboard input.	`Guess a number between 1 and 10`
The user types in a reply.	`7`
The program checks if 7 is not equal to 4 **and** if guesses are less than 3. This is True so it displays.	`Incorrect` `Guess a number between 1 and 10`
The user types in a new number.	`7`
The program checks if 7 is not equal to 4 **and** if guesses are less than 3. This is True so it displays.	`Incorrect` `Guess a number between 1 and 10`
The user types in a new number.	`7`
The program checks if 7 is not equal to 4 **and** if guesses are less than 3. This is False* so it displays.	`Correct` `You guessed it in 3 attempts`

Note that this output is not what you might have expected. This will be addressed in the next task.

Task 5: Displaying the correct message at the end of the game

Currently your program displays a message stating that the user has guessed correctly, even when they haven't. This is because these two lines of code are executed when the loop terminates.

```
print("Correct")
print(f"You guessed it in {guesses} attempts")
```

Implement selection to output `Correct, and in n guesses!` When the user guessed correctly. When the user didn't guess correctly it should output `Incorrect, guess limit reached.`

Copy and complete testing table on the next slide to help you test your program.

Task 5: Displaying the correct message at the end of the game

Test Number	Test Input	Expected Outcome	Actual Outcome
1	3	`Incorrect, guess a number between...`	
2	4	`Correct, you guessed it in 2 attempts`	
3			
4			
5			
6			

Optional extension task

You can now make the game much more interesting by introducing a **random number** rather than always having the number set to 4.

In lesson 1 of the programming part 2 unit we learnt about the random function.

Here are some code snippets to help you with this task.

```
from random import randint
number = randint(1,10)
```


Parson’s Puzzle

Instructions

Take a look at the code on the next slide. It contains all of the code needed to create a simple **guess the word** game.

Your job is to rearrange the lines of code so that the program will:

- ask for a word to guess
- check if the word is not equal to Raspberry
- continue to check if the word is not equal to Raspberry
- when the word is equal to Raspberry it should display, ‘Well done, the word was Raspberry!’

Note: You might need to add indents if they are needed

Parson’s Puzzle

```
print(f"Well done, the word was {word}!")
print("Try again...")
print("Guess the word")
word = input()
word = input()
while word != "Raspberry":
```