Practise using selection

Worked example Greeting

This is an example of the Python program that you have developed so far: it prompts the user for their name and reserves a special greeting for anyone named Elizabeth.

```
print("What’s your name?")
user = input()
if user == "Elizabeth":
  print("Good morning Your Majesty")
else:
  print("Hello", user)
```

Syntax checklist

If you encounter an error message, read it and try to fix the problem. Use the list below to check for common errors (and tick ✓ if you find yours).

- ☐ misspelt `if` or `else` (this includes using capitals)
- ☐ forgot the colon `:` after the `if` condition or after `else`
- ☐ forgot to **indent** statements in the `if` block or the `else` block
- ☐ indented `if` or `else` by mistake
- ☐ used = instead of == in the condition for `if`, to check if two values are equal
- ☐ used quotes around the name of a variable
- ☐ forgot to use quotes around a string literal (like `"Elizabeth"`)

Testing your program

Once you manage to run your program successfully, test it at least twice, once for every possible **branch** of the `if`, `else` statement.

Tip: In every task, the problem statement includes sample interactions between the user and the program. Use the values provided in these examples to test your program.

Task 1 Film critic

You are going to make a program that asks for the user's favourite film. The program will either react enthusiastically to one particular film or display a generic comment.

Example

Note: The result displayed depends on user input, so it will not always be the same.

The program displays a prompt and waits for keyboard input.	`Best film ever?`
The user types in a reply.	`Star Wars`
The program displays the result.	`Star Wars is not too bad`

Example

Note: The result displayed depends on user input, so it will not always be the same.

The program displays a prompt and waits for keyboard input.	`Best film ever?`
The user types in a reply.	`BFG`
The program displays the result.	`BFG is my favourite too!`

Step 1

Open this [incomplete program](ncce.io/py-critic-30) (ncce.io/py-critic-30) in your development environment:

```
1 print("Best film ever?")
2 film = input()
3 if              :
4 print(film, "is not too bad")
5 else:
6 print(film, "is my favourite too!")
```

Step 2

Complete line 3 with the **condition** that your program will need to check.

Tip: Use == to check if two values are equal, or != to check if two values are different.

Step 3

Indent any line(s) of code that you believe should be indented.

Step 4

Once you manage to **run** your program successfully, **test** it.

Task 2 Lucky number

Open the Python program below (ncce.io/py-lucky-30) in your development environment. It picks a **specific** 'lucky number' and displays it to the user.

```
lucky = 13
print("My lucky number is", lucky)
```

Step 1

Extend this program into a number guessing game. The program should ask the user to guess the lucky number, and then it should display a message, depending on whether or not the user guessed the lucky number.

Example

Note: Use these numbers to test that your program works correctly. In general, the messages displayed will depend on user input and will not always be the same.

The program displays a prompt and waits for keyboard input.	`Guess my lucky number:`
The user types in a reply.	13
The program displays a message that the user's guess is correct.	`Amazing, that's right!`

Example

Note: Use these numbers to test that your program works correctly. In general, the messages displayed will depend on user input and will not always be the same.

The program displays a prompt and waits for keyboard input.	`Guess my lucky number:`
The user types in a reply.	7

The program displays a message that the user's guess is incorrect. It also displays the lucky number.	`Sorry, it's not 7`
	`My lucky number is 13`

Tip

Introduce a variable called `guess`, to refer to the number entered by the user.

Tip

Don't forget that the user's guess should be an integer. You will need to use `int` to convert user input from the keyboard to an integer.

Tip

Use `==` to check if two values are equal and `!=` to check if they are different. Do not confuse `==` with `=`, which is used in assignments.

Step 2

Extend the program that you created in the previous task so that, **regardless of the outcome**, this message is displayed at the end of the game:

```
Nice playing with you
```

Step 3: Checklist

Perform each of the tests below (and tick ✓ the boxes when you have finished them).

- ☐ When the user guesses the lucky number, does the program display a message that the guess is correct?
- ☐ When the user fails to guess the lucky number, does the program display a message that the guess is incorrect?
- ☐ Does the program display a message that reveals the magic number **only** when the user's guess is incorrect?
- ☐ Does the program **always** display a goodbye message to the user, regardless of the outcome of the game?

Explorer task Eligible to vote

You are going to make a program that asks for the user's age and displays a message that says whether or not the user is eligible to vote.

In the UK, you are eligible to vote when you are 18 or over.

Example

Note: Use these numbers to test that your program works correctly. In general, the result displayed will depend on user input.

The program displays a prompt and waits for keyboard input.	`How old are you?`
The user types in a reply.	`21`
The program displays a message.	`You are eligible to vote`

Example

Note: Use these numbers to test that your program works correctly. In general, the result displayed will depend on user input.

The program displays a prompt and waits for keyboard input.	`How old are you?`
The user types in a reply.	`14`

The program displays a message.	`You are not eligible to vote`
	`4 more years to go`

Step 1

Write your program, run it, and test it. Use the code from the worked example and the previous tasks as points of reference.