Simon says...

# Worked example and code snippets .

Accessing a list item using an expression that evaluates as an integer

|  | days = ["Monday", "Tuesday",  "Wednesday", "Thursday",  "Friday", "Saturday",  "Sunday"]  day = 3  print(days[day-1]) |
| --- | --- |

Generating a random number

|  | from random import randint  random\_number = randint(2,4) |
| --- | --- |

Using the sleep function

|  | from time import sleep  sleep(4) |
| --- | --- |

A for loop

|  | for x in range(5):  print(x) |
| --- | --- |

# Make a prediction .

| 1  2  3  4  5  6  7  8  9 | simon\_says = ["Hands on head", "Hands on ears",  "Right hand up", "Left hand up",  "Hands on shoulders"]  print("Pick a number between 0 and 4")  index = int(input())  instruction = simon\_says[index]  print(f"Simon says...{instruction}") |
| --- | --- |

Make a prediction about this program. If the user enters a 2 when prompted, what will be the output?

Prediction

|  |
| --- |

# Execute the code .

Execute the program by copy and pasting the code into your development environment. Was your prediction correct?

Answer

|  |
| --- |

Introduce randomisation .

# **Step 1** Currently the user needs to type in a value in order to reveal what Simon is saying. Modify your program so that it picks an instruction at **random**. The areas that you need to modify are highlighted below on line 1 and line 7. Use the code snippet on page 1 to support you.

| 1  2  3  4  5  6  7  8  9  10 | .  simon\_says = ["Hands on head", "Hands on ears",  "Right hand up", "Left hand up",  "Hands on shoulders"]  print("Pick a number between 0 and 4")  index = .  instruction = simon\_says[index]  print(f"Simon says...{instruction}") |
| --- | --- |

# **Step 2** Test your program. When you execute the code, it should output a random ‘Simon says…’ instruction. Try it a few times to see if the instructions change.

# **There is now an additional instruction for the user that isn’t required. Don’t forget to remove it.**

Introduce repetition .

# **Step 1**

The ‘Simon says…’ game should randomly generate an instruction ten times. Introduce a for loop to your program so that it will provide a random instruction ten times.

**Note:** You do not need to include the list inside the for loop.

**Tip**: Use the code snippet on page 1 to help you.

# **Step 2**

Test your code and make sure that it prints ten lines of instructions from Simon.

# **Step 3**

The game wouldn’t be very effective if it listed all of the instructions at the same time. Introduce the sleep function to your for loop to pause the program between each instruction.

**Tip**: Use the code snippet on page 1 to help you.

# I didn’t say Simon says... .

**Task 1**

If you have played this game before, you will know that if the leader doesn’t say “Simon says”, then the action should not be performed by the players. The code needs to be adjusted so that it will randomly say “Simon says…” or be kept blank “”.

Introduce a new list to your program underneath the current list. It should look like the list below:

|  | intros = ["Simon says...", ""] |
| --- | --- |

**Task 2**

An item from intros needs to be randomly selected. Up to now you have used a random number to access a list item. You can also use another function from random, called choice. This will randomly select an item from a list.

The code for this is below:

|  | from random import choice  intro = choice(intros) |
| --- | --- |

Place the import statement at the top of your program and the other line inside your for loop above the print statement.

**Task 3**

Modify the print statement so that it now displays the two randomly generated strings.

|  | print(f" {instruction}") |
| --- | --- |

**Task 4**

Test your program. It should now either say “Simon says..”, or be blank and then say the instruction. See the example output below:

Simon says...Left hand up

Hands on head

Hands on ears

Hands on head

Simon says...Hands on head

Right hand up

Hands on ears

Hands on shoulders

Simon says...Left hand up

Hands on shoulders

>>>

# 

# Explorer task .

This program would work great as it is, but you could make it more interactive for the user.

Adapt the program so that it:

* Uses instructions that could be carried out by keyboard input instead of physical movements
* Checks if the user has entered the input correctly
* Outputs “I didn’t say Simon says”, if the program didn’t output “Simon says…” but the user has entered a correct action

# Further explorer task .

Use a while loop instead. Keep track of how many times the player has correctly followed the actions. As soon as the player performs an action that Simon didn’t say, end the game and reveal how many actions the player performed correctly.