Computing

Lesson 5: Arrays and Lists

Programming Part 5: Strings and Lists

Ben Garside

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

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)
```

Code snippets

Using the sleep function

```
from time import sleep

sleep(4)
```

A for loop

```
for x in range(5):
    print(x)
```

Task 1.1: Simon Says - Predict

```
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}")
```

Write down your prediction below:

Task 1.2 - Execute the code

Execute the program by copy and pasting the code into your development environment.

Was your prediction correct?

Task 1.3 - 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   simon_says = ["Hands on head", "Hands on ears",
3                 "Right hand up", "Left hand up",
4                 "Hands on shoulders"]
5
6   print("Pick a number between 0 and 4")
7   index =
8   instruction = simon_says[index]
9
10  print(f"Simon says...{instruction}")
```


Task 1.3 - Introduce randomisation

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.

Task 1.4 - Introduce repetition

Step 1

The Simon says... game should randomly generate an instruction 10 times. Introduce a for loop to your program so that it will provide a random instruction 10 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.

Task 1.4 - Introduce repetition

Step 2

Test your code and make sure that it prints 10 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.

Task 1.5 - I didn’t say Simon says....

Task 1

If you have played Simon says 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 1.5 - I didn’t say Simon says....

Task 2

An item from intros needs to be randomly selected. Up to now we have used a random number to access a list item. We 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 1.5 - I didn’t say Simon says....

Task 3

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

```
    print(f"                    {instruction}")
```


Task 1.5 - I didn’t say Simon says....

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
>>>
```

Optional explorer tasks

This program would work great as it is but we 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
- If the program didn’t output “Simon says…” and the user entered a correct action it should output “I didn’t say Simon says”
- 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 the player performed correctly.

Code snippets

Append an item

```
shopping = []

shopping.append("Bread")
```

Remove an item

```
shopping = ["bread", "cheese", "milk"]

shopping.remove("bread")
```

Append an item using a variable

```
item = "pizza"
shopping.append(item)
```

Task 2.1 - Append and remove items

Step 1

Copy the start code below into your development environment.

```
shopping = ["bread", "cheese", "milk"]
```

Step 2

Use the append operator to add two new items to the shopping list:

- Eggs
- Flour

Task 2.1 - Append and remove items

Step 3

Use the remove operator to remove cheese from the shopping list.

Step 4

Print your list to check that your code has worked. The output should look like this:

```
['bread', 'milk', 'eggs', 'flour']
```


Task 2.2 - A shopping list program

Use the two new operators append and remove to create a shopping list program. Your program should:

- Continue to ask for new items until the user has finished
- Prompt if an item should be removed or added
- If the choice is to remove the item, it should remove it
- If the choice is to add the item, it should append it
- Display the finished list at the end of the program

On the next two slides are some example inputs and outputs to help you with your design.

Task 2.2 - A shopping list program

Example

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

The user is given a prompt	`Would you like to edit your shopping list? Y/N`
The user enters their response	`Y`
The user is given a prompt	`Would you like to add or remove an item? A/R`
The user enters their response	`A`
The user is given a prompt	`Enter an item to add:`
The user enters their response	`Eggs`
The user is given a prompt	`Would you like to edit your shopping list? Y/N`
The user enters their response	`Y`
The user is given a prompt	`Would you like to add or remove an item? A/R`
The user enters their response	`A`
The user is given a prompt	`Enter an item to add:`
The user enters their response	`Milk`

Task 2.2 - A shopping list program

The user is given a prompt	`Would you like to edit your shopping list? Y/N`
The user enters their response	`Y`
The user is given a prompt	`Would you like to add or remove an item? A/R`
The user enters their response	`R`
The user is given a prompt	`Enter an item to remove:`
The user enters their response	`Milk`
The user is given a prompt	`Would you like to edit your shopping list? Y/N`
The user enters their response	`N`
The user is shown the list of items	`['eggs']`

List methods

There are many operations you can perform on lists and their items.

oaknat.uk/comp-py-doc-lists

`list.append(item)`	add item at end of list
`list.insert(index, item)`	add item at index
`list.pop(index)`	remove item at index
`list.remove(item)`	remove item
`list.index(item)`	search for index of item
`list.count(item)`	get occurrences of item
`list.reverse()`	reverse list
`list.sort()`	sort list

List methods

There are many operations you can perform on lists and their items.

Some examples

```
numbers.append(42)
cities.insert(2, "Oslo")
last = values.pop()
countries.remove("Japan")
where = planets.index("Mars")
nb_the = words.count("the")
values.reverse()
names.sort()
```