ANSWERS

F3.3 Lists: one-dimensional arrays

Strings, lists and records · GCSE · OCR J277 2.2.3, AQA 8525 3.2.6, Edexcel 1CP2 6.3.1 · about 15 min

What this lesson is about

Making, indexing and changing a list; append, remove, in, split and join.

- Making and reading a list
- Using items
- Changing a list
- An empty list that fills up
- Is it in the list?
- Lists and strings

Questions

6 marks in all

1. What does this program print? [1 mark]

```
legs = [20, 15, 30]
print(legs[0], legs[-1], len(legs))
```

Answer:

```
20 30 3
```

Index 0 is the first item, -1 the last, and len counts the items.

2. What does this program print? [1 mark]

```
notes = [262, 294, 330]
notes[0] = 523
notes.append(349)
print(notes)
```

Answer:

```
[523, 294, 330, 349]
```

Assigning to an index replaces that item; append adds one to the end.

3. `legs = [20, 15, 30]`. What happens with `print(legs[3])`? [1 mark]

- ☐ **A** IndexError: there is no index 3
- ☐ **B** It prints 30
- ☐ **C** It prints 0
- ☐ **D** It prints None

Answer: A. Three items have indexes 0, 1 and 2.

4. Why use an array rather than separate variables such as reading1, reading2, reading3? [1 mark]

- ☐ **A** One name holds them all, and a loop can process every item
- ☐ **B** Arrays use less electricity
- ☐ **C** Separate variables cannot hold numbers
- ☐ **D** Arrays are always sorted

> **Answer:** A. An array groups values under one identifier, so the same code works however many values there are.

5. What does this program print? [1 mark]

```
sides = "30,20".split(",")
print(sides)
print(int(sides[0]) + int(sides[1]))
```

> **Answer:**
>
> ```
> ['30', '20']
> 50
> ```
>
> split makes a list of strings; int is needed before adding them.

6. What does this program print? [1 mark]

```
colours = ["red", "green"]
print("green" in colours, colours.index("green"))
```

> **Answer:**
>
> ```
> True 1
> ```
>
> in asks whether an item is in the list, and index gives its position.

The task: a changing tune

Start with `notes = [262, 294, 330]`. Change the first note to 523, add 349 to the end, print how many notes there are as `4 notes`, then play every note in the list for 0.3 seconds. Change the list with list operations, not by typing a new list.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

notes = [262, 294, 330]
```

The hint students can ask for: A list can be changed in place: replace one item by its position, and add another on the end. Then loop over the list to play it.

A solution

```
from bugbot import *
connect()
notes = [262, 294, 330]
notes[0] = 523
notes.append(349)
print(len(notes), "notes")
for note in notes:
    tone(note, 0.3)
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.