ANSWERS

A3.1 Arrays, records and tuples

Data structures · A level · OCR H446 1.4.2, AQA 7517 4.2.1.1, Eduqas A500QS 1.1 · about 20 min

BugBotLab

What this lesson is about

Arrays in one, two and three dimensions, records and fields, tuples and lists, and static structures.

- Arrays
- Two and three dimensions
- Records and fields
- Tuples and lists

Questions

6 marks in all

1. Which of these describes an array? [1 mark]

- ☐ **A** A fixed-size, indexed collection of elements of the same data type
- ☐ **B** A collection of named fields that can have different data types
- ☐ **C** An immutable ordered collection of values of any types
- ☐ **D** A collection of key-value pairs

> **Answer:** A. An array has a fixed size, one data type and an index for each element. The others describe a record, a tuple and a dictionary.

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

```
scans = [[[1, 2], [3, 4]], [[5, 6], [7, 8]], [[9, 10], [11, 12]]]
print(len(scans), len(scans[0]), len(scans[0][0]))
print(scans[2][0][1])
```

> **Answer:**
>
> ```
> 3 2 2
> 10
> ```
>
> The array is 3 by 2 by 2. scans[2] is the third block, [0] its first row, and [1] the second element of that row: 10.

3. A two-dimensional array with 8 columns is stored in row-major order, starting at position 0. At what position is element [3][5]? [1 mark]

> **Answer:** 29. Row-major order stores whole rows one after another, so [r][c] is at r x columns + c = 3 x 8 + 5 = 29.

4. BugBot's position() returns a tuple rather than a list. Which is the best reason? [1 mark]

- ☐ **A** A tuple is immutable, so the x and y cannot be changed separately by mistake
- ☐ **B** A tuple can hold more values than a list
- ☐ **C** A tuple is sorted automatically
- ☐ **D** A tuple can only hold numbers

> **Answer:** A. A tuple cannot be changed once made, which suits a pair of values that belong together.

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

```
grid = [[0] * 2] * 3
grid[1][0] = 7
print(grid)
```

> **Answer:**
>
> ```
> [[7, 0], [7, 0], [7, 0]]
> ```
>
> Multiplying the outer list makes three references to the same row, so changing one row changes all three.

6. Which of these are true of a record? [1 mark]

Tick every answer that is true.

- ☐ **A** It groups fields that can have different data types
- ☐ **B** Each field has its own name
- ☐ **C** All of its fields must have the same data type
- ☐ **D** A file is often a collection of records with the same fields

> **Answer:** A, B, D. A record describes one thing with named fields of any types, and a file of records keeps many of them on storage.

The task: the depth cube

`scans` is a 3D array of three saved 4 by 4 depth scans, indexed `scans[scan][row][col]`, each value a whole number of centimetres from 1 to 100. Using nested loops and `len()`, print: 1. `size: 3 x 4 x 4`, working the three sizes out with `len()`. 2. For each scan `s` from 0 to 2, `scan <s> average: <mean>`, the mean of its 16 values rounded to 1 decimal place with `round(value, 1)`, for example `scan 0 average: 56.0`. 3. `nearest: <cm> cm at [<scan>][<row>][<col>]`, the smallest value and its three indexes. The smallest value appears only once. The robot does not move.

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

# scans[scan][row][col]: three saved 4 by 4 depth scans, in cm
scans = [
    [[45, 44, 43, 50], [40, 38, 39, 41], [60, 58, 57, 59], [80, 79, 81, 82]],
    [[44, 42, 41, 49], [39, 36, 12, 40], [61, 57, 56, 60], [78, 80, 80, 83]],
    [[46, 43, 42, 51], [41, 37, 38, 42], [59, 59, 55, 58], [81, 78, 82, 80]],
]
```

The hint students can ask for: Three nested loops reach every reading: the outer one picks the scan, the middle one the row, the inner one the column. Keep a running total for each scan, and keep the smallest value seen so far together with the three indexes where you saw it.

A solution

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

# scans[scan][row][col]: three saved 4 by 4 depth scans, in cm
scans = [
    [[45, 44, 43, 50], [40, 38, 39, 41], [60, 58, 57, 59], [80, 79, 81, 82]],
    [[44, 42, 41, 49], [39, 36, 12, 40], [61, 57, 56, 60], [78, 80, 80, 83]],
    [[46, 43, 42, 51], [41, 37, 38, 42], [59, 59, 55, 58], [81, 78, 82, 80]],
]

print("size:", len(scans), "x", len(scans[0]), "x", len(scans[0][0]))

best = scans[0][0][0]
where = [0, 0, 0]
for s in range(len(scans)):
    total = 0
    count = 0
    for r in range(len(scans[s])):
        for c in range(len(scans[s][r])):
            value = scans[s][r][c]
            total = total + value
            count = count + 1
            if value < best:
                best = value
                where = [s, r, c]
    print(f"scan {s} average: {round(total / count, 1)}")
print(f"nearest: {best} cm at [{where[0]}][{where[1]}][{where[2]}]")
```

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