ANSWERS

A1.4 Subroutines, parameters and passing by reference

Programming techniques and object-oriented programming · A level · OCR H446 2.2.1, AQA 7517 4.1.1.10, Eduqas A500QS 1.4 · about 20 min

What this lesson is about

Out-of-line subroutines and their interfaces, returning several values, and passing by value and by reference.

- What a subroutine is
- The interface
- Returning more than one value
- By value and by reference
- What Python does

Questions

6 marks in all

1. What does it mean for a subroutine to be out of line? [1 mark]

- ☐ **A** It is written once, apart from the code that uses it, and run by calling its name
- ☐ **B** It is written on one line
- ☐ **C** It runs at the same time as the main program
- ☐ **D** It is stored in a separate file

> **Answer:** A. A call jumps to the subroutine and comes back to the line after the call.

2. A procedure is called with an argument passed by reference, and changes its parameter. What happens to the caller's variable? [1 mark]

- ☐ **A** It changes too, because the parameter refers to the caller's variable
- ☐ **B** It stays the same, because the parameter is a copy
- ☐ **C** It is deleted
- ☐ **D** It changes only if the procedure returns it

> **Answer:** A. By reference passes the variable's location, so changes inside affect the caller.

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

```
def nudge(n, items):
    n = n + 1
    items.append(n)

count = 5
log = []
nudge(count, log)
nudge(count, log)
print(count, log)
```

> **Answer:**
>
> ```
> 5 [6, 6]
> ```
>
> Each call makes n a new local 6, so count stays 5, but append changes the one list that log and items both refer to.

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

```
def stats(values):
    low = values[0]
    high = values[0]
    for v in values:
        if v < low:
            low = v
        if v > high:
            high = v
    return low, high

low, high = stats([31, 15, 51, 20])
print(high - low)
```

Answer:

```
36
```

The function returns two values as a tuple; 51 - 15 is 36.

5. Which are advantages of passing a large array by reference rather than by value? [1 mark]

Tick every answer that is true.

- ☐ **A** No copy of the array has to be made
- ☐ **B** Less memory is used
- ☐ **C** The subroutine cannot change the caller's array
- ☐ **D** The subroutine is easier to test on its own

Answer: A, B. No copy saves time and memory; the cost is that the subroutine can change the caller's data.

6. What is the interface of a subroutine? [1 mark]

- ☐ **A** Its name, its parameters and their types, and what it returns
- ☐ **B** The code inside its body
- ☐ **C** The screen it displays
- ☐ **D** The local variables it uses

Answer: A. The interface is what a caller must know to use it; the body can change behind it.

The task: survey by reference

Write two subroutines. - The procedure `survey(readings)` takes an empty list `readings`. It takes eight distance readings, turning right 45 degrees after each one, so the robot ends facing where it started, and appends each reading to `readings`. It returns nothing, and the main program calls it on a line of its own, `survey(readings)`. - The function `nearest(readings)` takes a list of real numbers and returns two values: the index of the smallest reading (the first one if two are equal) and that reading. Find it with a loop; do not use `min`, `index` or `global`. Then print `nearest: <cm> cm at <degrees> degrees`, where the degrees are the index times 45, turn right by that many degrees to face the nearest wall, and play one note.

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

def survey(readings):
    pass

def nearest(readings):
    return 0, 0.0

readings = []
survey(readings)
index, cm = nearest(readings)
print(f"nearest: {cm} cm at {index * 45} degrees")
```

The hint students can ask for: A list passed to a subroutine is the same list the caller has, so the procedure can fill it without returning anything. For the nearest, keep the best index and the best distance so far as you go through the list, and hand both back at the end.

A solution

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

def survey(readings):
    for i in range(8):
        readings.append(distance())
        turn_right(30, angle=45)

def nearest(readings):
    best = 0
    for i in range(1, len(readings)):
        if readings[i] < readings[best]:
            best = i
    return best, readings[best]

readings = []
survey(readings)
index, cm = nearest(readings)
print(f"nearest: {cm} cm at {index * 45} degrees")
turn_right(30, angle=index * 45)
tone(880, 0.3)
```

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