ANSWERS

F6.4 Debugging logic errors

Robust programs · GCSE · OCR J277 2.3.2, AQA 8525 3.2.11, Edexcel 1CP2 6.1.2 · about 15 min

BugBotLab

What this lesson is about

Finding the mistake that gives no message: watching, printing, stepping and common bugs.

- Step 1: know what should happen
- Step 2: find where it first goes wrong
- Step 3: form a theory and test it
- Common logic errors
- Explaining it out loud

Questions

5 marks in all

1. What is the first step in finding a logic error? [1 mark]

- ☐ **A** Know exactly what the program should do
- ☐ **B** Delete the last line
- ☐ **C** Rewrite the program
- ☐ **D** Add more loops

> **Answer:** A. You cannot spot what is wrong until you know what right looks like.

2. A loop meant to run four times uses `range(3)`. What kind of bug is this? [1 mark]

- ☐ **A** An off-by-one error
- ☐ **B** A syntax error
- ☐ **C** A runtime error
- ☐ **D** A type error

> **Answer:** A. It runs one time too few, with no error message: a logic error.

3. Why change only one thing at a time when debugging? [1 mark]

- ☐ **A** So you know which change fixed or broke the program
- ☐ **B** Python only allows one change
- ☐ **C** It is faster to type
- ☐ **D** To keep the program short

> **Answer:** A. Several changes at once hide which one mattered.

4. Which help find where a logic error first appears? [1 mark]

Tick every answer that is true.

- ☐ **A** Printing variables inside a loop
- ☐ **B** Stepping through with a debugger
- ☐ **C** Watching the robot's trail
- ☐ **D** Adding more comments

> **Answer:** A, B, C. Comments help people read code but do not show what the program actually does.

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

```
total = 0
for x in [4, 6]:
    total = 0
    total = total + x
print(total)
```

Answer:

```
6
```

total is reset inside the loop, so it ends as just the last value.

The task: fix the square

This program should drive a square of 25 cm sides clockwise, turning right at each corner, and finish back at the start, facing the way it began. It has three logic errors. Find them with the trail, prints or Debug, and fix them one at a time.

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

side = 25
for i in range(3):
    side = 20
    forward(60, distance=side)
    turn_left(30, angle=90)
```

The hint students can ask for: Three bugs: the loop runs three times, side is reset to 20 inside the loop, and the turn goes the wrong way.

A solution

```
from bugbot import *
connect()
side = 25
for i in range(4):
    forward(60, distance=side)
    turn_right(30, angle=90)
```

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