ANSWERS

F13.4 Programming questions

Exam preparation · GCSE · about 20 min

What this lesson is about

Taking a question apart, the jobs that come up again and again, and how the marks are given.

- Read the question like a specification
- The jobs that come up again and again
- How the marks are given
- Worked example

Questions

5 marks in all

1. You cannot finish a programming question. What is the best thing to do? [1 mark]

- ☐ **A** Write what is left as comments or pseudocode
- ☐ **B** Leave it blank
- ☐ **C** Rub out what you have
- ☐ **D** Copy an earlier answer

> **Answer:** A. Marks are given in steps, and a correct approach scores.

2. Which of these earns marks in a written programming answer? [1 mark]

- ☐ **A** Meaningful variable names and clear structure
- ☐ **B** Very short variable names
- ☐ **C** No comments at all
- ☐ **D** Perfect punctuation

> **Answer:** A. The marker must be able to follow what you meant.

3. A question says to reject an input over 60. What is that worth? [1 mark]

- ☐ **A** A mark: handle the failure case
- ☐ **B** Nothing
- ☐ **C** Only if the program runs
- ☐ **D** A mark only in an on-screen exam

> **Answer:** A. Every rule in the question usually carries a mark.

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

```
done = 0
steps = 0
while done < 32:
    step = min(7, 32 - done)
    done = done + step
    steps = steps + 1
print(steps, done)
```

> **Answer:**
>
> ```
> 5 32
> ```
>
> Four steps of 7 and a last step of 4.

5. What should you do before writing any code in a long question? [1 mark]

- ☐ **A** Underline the inputs, outputs and rules
- ☐ **B** Write the last line first
- ☐ **C** Choose variable names for everything
- ☐ **D** Count the marks

> **Answer:** A. Then write the structure: input, process, output.

The task: an exam-style program

Write the program for this question. The robot must drive `far` centimetres in steps of `step` centimetres, where the last step is whatever is left over. After each step, print `so far: <n> cm`. At the end print `steps: <n>` and `total: <n> cm`. Use the values given; do not type the answers.

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

far = 32
step = 7
```

The hint students can ask for: Keep a running total of how far you have driven and loop while it is less than the target. Each step is the step size, or whatever is left if that is smaller. Count the steps as you go.

A solution

```
from bugbot import *
connect()
far = 32
step = 7
done = 0
count = 0
while done < far:
    this = min(step, far - done)
    forward(50, distance=this)
    done = done + this
    count = count + 1
    print(f"so far: {done} cm")
print("steps:", count)
print(f"total: {done} cm")
```

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