ANSWERS

A15.1 How your A level is assessed

Exam preparation · A level · OCR H446 2.2.1, AQA 7517 4.4.1.2, Eduqas A500QS 1.8 · about 35 min

What this lesson is about

The papers and the project for OCR H446, AQA 7517 and Eduqas A500QS, the assessment objectives, and a timing plan built with integer arithmetic.

- Three boards, one shape
- The assessment objectives
- Marks are the clock
- The project runs alongside
- Using this module

Questions

5 marks in all

1. Which A level paper is sat on a computer, with a skeleton program released before the exam? [1 mark]

- ☐ **A** AQA paper 1
- ☐ **B** OCR paper 1
- ☐ **C** OCR paper 2
- ☐ **D** AQA paper 2

> **Answer:** A. AQA's paper 1 is on screen, with preliminary material and a skeleton program; the other papers are written.

2. What percentage of the A level is the programming project worth, on every board? [1 mark]

> **Answer:** 20. Each board's non-exam assessment is 20%; the two exams make up the other 80%.

3. A question says: "Write a program that returns the robot home using a stack." Which assessment objective does it mainly target? [1 mark]

- ☐ **A** AO3: design, program and evaluate solutions
- ☐ **B** AO1: knowledge and understanding
- ☐ **C** AO2: apply knowledge to analyse problems
- ☐ **D** None of them

> **Answer:** A. Writing a program to solve a problem is AO3; recalling facts is AO1 and applying them to analyse a problem is AO2.

4. An OCR paper has 140 marks in 150 minutes. To the nearest minute, how long should a 12-mark question take? [1 mark]

> **Answer:** 13. 150 / 140 is about 1.07 minutes a mark, and 12 x 1.07 is about 12.9, so 13 minutes.

5. What does this timing calculation print? [1 mark]

```
start = 9 * 60
finish = start + round(35 * 1.4)
print(f"{finish // 60:02d}:{finish % 60:02d}")
```

> **Answer:**
>
> ```
> 09:49
> ```
>
> 35 x 1.4 is 49 minutes after 09:00, which is 9 hours 49 minutes: // gives the hours and % the minutes.

The task: the timing plan

Make a timing plan for a mock paper. `START` is the start time as a string `"hh:mm"` in 24-hour time. `MINUTES` is the length of the paper in whole minutes, `CHECKING` is the whole number of minutes kept back at the end for checking, and `marks` is a list of the whole-number marks for each question in order. 1. Write `clock_time(minutes)`, which takes a whole number of minutes after midnight and returns the time as `"hh:mm"`, with two digits each. Use `//` for the hours and `%` for the minutes. 2. The minutes a mark are `(MINUTES - CHECKING)` divided by the total of `marks`. Print `<n> minutes a mark`, to one decimal place. 3. For each question, keep a running total of the marks so far. The question should be finished by the start time plus that total times the minutes a mark, rounded to the nearest whole minute with `round`. Print `Q<n> (<marks> marks): finish by <hh:mm>`, numbering from 1. 4. Print `checking from <hh:mm>`, the start plus `MINUTES - CHECKING`, and then `end <hh:mm>`, the start plus `MINUTES`. The robot stays still.

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

START = "09:00"
MINUTES = 150
CHECKING = 10
marks = [8, 12, 15, 5, 20, 10, 18, 12]
```

The hint students can ask for: Turn the start time into minutes after midnight first, so all the arithmetic is on whole numbers. The time per mark comes from the time left after checking divided by the total marks. Keep a running total of marks, and turn each finishing time back into hours and minutes only when you print it.

A solution

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

START = "09:00"
MINUTES = 150
CHECKING = 10
marks = [8, 12, 15, 5, 20, 10, 18, 12]

def to_minutes(text):
    return int(text[:2]) * 60 + int(text[3:])

def clock_time(minutes):
    return f"{minutes // 60:02d}:{minutes % 60:02d}"

start = to_minutes(START)
per_mark = (MINUTES - CHECKING) / sum(marks)
print(f"{per_mark:.1f} minutes a mark")
done = 0
for n, m in enumerate(marks, start=1):
    done += m
    print(f"Q{n} ({m} marks): finish by {clock_time(start + round(done * per_mark))}")
print("checking from", clock_time(start + MINUTES - CHECKING))
print("end", clock_time(start + MINUTES))
```

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