ANSWERS

F4.5 Libraries and your own modules

Functions and structured code · GCSE · OCR J277 2.2.3, AQA 8525 3.2.10, Edexcel 1CP2 6.6.1 · about 15 min

What this lesson is about

import, Python's own libraries, and a module file of your own.

- import
- Three ways to import
- Libraries you already know
- Your own module
- Why write modules?

Questions

5 marks in all

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

```
import math
print(math.sqrt(49))
```

Answer:

```
7.0
```

sqrt from the math library gives a real number, so 7.0.

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

```
from math import floor, ceil
print(floor(2.7), ceil(2.1))
```

Answer:

```
2 3
```

from ... import brings in just those names; floor rounds down and ceil rounds up.

3. Which are benefits of using a library? [1 mark]

Tick every answer that is true.

- ☐ **A** It saves writing the code yourself
- ☐ **B** Its code has already been tested
- ☐ **C** You can use complex features without knowing how they work
- ☐ **D** It makes every program shorter than 10 lines

Answer: A, B, C. Libraries save time and effort and are usually well tested.

4. A program says `import moves` and calls `moves.square(20)`. Where is `square` defined? [1 mark]

- ☐ **A** In a file called moves.py
- ☐ **B** Inside the program itself
- ☐ **C** In the math library
- ☐ **D** Nowhere: it causes an error

> **Answer:** A. import moves loads the module moves.py, and moves.square reaches the function in it.

5. Why does moves.py need its own `from bugbot import *` line? [1 mark]

- ☐ **A** A module gets the robot's commands only by importing them itself
- ☐ **B** Every Python file must start with it
- ☐ **C** It connects to the robot twice
- ☐ **D** It does not need it

> **Answer:** A. Each file has its own names, so a module that uses forward must import it.

The task: use your module

Finish `square(size)` in `moves.py` above so it drives a square and ends where it started (replace `pass`). Then, in the task, import `moves` and use it to drive a square of 20 cm and then a square of 35 cm. Do not write the square's drives in the task program itself.

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

import moves

moves.square(20)
```

The hint students can ask for: Put the driving function in the other file, with the size as its parameter. The task program imports that file and calls the function twice with different sizes.

A solution

```
{'program': 'from bugbot import *\nconnect()\nimport
moves\n\nmoves.square(20)\nmoves.square(35)\n', 'files': {'moves.py': '# moves.py: driving
moves, for any program that imports it\nfrom bugbot import *\n\ndef zigzag(length):\n
"""Drive a zigzag: forward and right, then forward and left."""\n    forward(60,
distance=length)\n    right(60, distance=length)\n    forward(60, distance=length)\n
left(60, distance=length)\n\ndef square(size):\n    """Drive a square with sides of size
cm, ending where it started."""\n    forward(60, distance=size)\n    right(60,
distance=size)\n    backward(60, distance=size)\n    left(60, distance=size)\n'}}
```

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