WORKSHEET

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

BugBotLab

Name _____ Class _____ Date _____

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))
```

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

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

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

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

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

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)
```

Plan your program here, then type it in and press Run.

Do it on the robot
www.bugbotlab.com/learn/f4-5-libraries-and-modules/
The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Add a function `triangle(size)` to `moves.py` and use it from a Try-it cell.
2. Plan a module `sounds.py` on paper: list the functions a robot program would want from it, with their parameters.
3. Use `math.hypot` to print how far the robot is from where it started after a zigzag.