ANSWERS

2.6 Project: the maze

BugBotLab

Sensing · Robot club · about 25 min

What this lesson is about

Walls, corners and a goal. Everything in this module in one run.

- The maze
- The plan
- One leg as a function
- Two legs
- Staying off the side walls
- Where next

Questions

7 marks in all

1. Put the maze legs in order. [1 mark]

Number the lines 1 to 5 to put them in the right order.

___ `Drive along the top until the right-hand wall is close`

___ `Turn right`

___ `Drive down the middle channel into the goal`

___ `Drive up the left channel until the top wall is close`

___ `Turn right again`

Answer:

```
Drive up the left channel until the top wall is close
Turn right
Drive along the top until the right-hand wall is close
Turn right again
Drive down the middle channel into the goal
```

Three legs and two corners. Each leg is a "drive until distance() is small" loop.

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

```
def drive_to_wall(gap):
    reading = 50
    while reading > gap:
        reading = reading - 10
    print("stopped at", reading)

drive_to_wall(15)
```

Answer:

```
stopped at 10
```

The reading goes 50, 40, 30, 20, 10. The loop only checks between steps, so it stops at the first reading that is not over 15, which is 10.

3. In the side-wall check, `left_side` is 6 cm. What does the program set `sideways` to, and why? [1 mark]

```
sideways = 0
if left_side < 8:
    sideways = 30
elif right_side < 8:
    sideways = -30
```

- ☐ **A** 30, to slide right, away from the left wall
- ☐ **B** -30, to slide left towards the wall
- ☐ **C** 0, because the gap ahead is still big
- ☐ **D** 60, to drive forward faster

> **Answer:** A. Positive sideways is right. The left wall is too close, so the robot nudges right.

4. Why write one leg as `drive_to_wall(gap)`? [1 mark]

- ☐ **A** The same few lines drive every leg, with the gap passed in
- ☐ **B** Functions make the robot drive straighter
- ☐ **C** The maze task will not run without a function
- ☐ **D** It stops the robot touching the walls

> **Answer:** A. Every leg is the same job, so write it once and call it three times. A fix then happens in one place.

5. Which ideas from Module 2 does the maze use? [1 mark]

Tick every answer that is true.

- ☐ **A** The distance sensor to find walls
- ☐ **B** Turns at the corners
- ☐ **C** The depth grid to stay off the side walls
- ☐ **D** Correction so long runs stay straight
- ☐ **E** The camera to read signs

> **Answer:** A, B, C, D. The maze brings the whole module together. The camera comes later, in Module 4.

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

```
legs = [("up", 15), ("right", 15), ("down", 15)]
for leg in legs:
    print(leg[0], "until", leg[1], "cm")
```

> **Answer:**
>
> ```
> up until 15 cm
> right until 15 cm
> down until 15 cm
> ```
>
> Each item in the list is a pair. `leg[0]` is the first part and `leg[1]` the second, just like `position()[0]`.

7. How does the left-hand rule get a robot through a maze it has not seen? [1 mark]

- ☐ **A** It always keeps the left wall close
- ☐ **B** It always turns left at every corner, wall or not
- ☐ **C** It remembers the whole maze first
- ☐ **D** It drives towards the furthest reading

> **Answer:** A. Follow one wall and you work your way round the maze until you reach the way out.

The task: the maze

Up the left channel, right along the top, down the middle channel into the goal, without touching either wall. Forty-five seconds.

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

def drive_to_wall(gap):
    while distance() > gap:
        # drive forward at 60 (keeps going until the next command)
        forward(60)
        # pause 0.1 s (the robot keeps doing what it was told)
        wait(0.1)
    # all motors off
    stop()

drive_to_wall(15)
# turn, second leg, turn, third leg
```

The hint students can ask for: Up the left channel, right along the top, down the middle channel into the goal. Use distance() to find each wall and turn_right(30, angle=90) at the corners.

A solution

```
from bugbot import *
connect()
def drive_to_wall(gap):
    while distance() > gap:
        forward(60)
        wait(0.1)
    stop()

drive_to_wall(15)
turn_right(30, angle=90)
drive_to_wall(15)
turn_right(30, angle=90)
drive_to_wall(12)
```

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