ANSWERS

8.2 Collecting data

Learning · Robot club · about 20 min

What this lesson is about

Readings with labels, and why bad data makes bad robots.

- Four situations
- Getting to the spots
- Bad data makes bad robots

Questions

6 marks in all

1. In this module, what is a sample? [1 mark]

- ☐ **A** A view (the 16 level readings) paired with a label
- ☐ **B** Just the 16 level readings
- ☐ **C** The label the robot guessed
- ☐ **D** The robot's position when it recorded

> **Answer:** A. A sample is the numbers plus a word for what they mean. The model learns the link between the two.

2. Which of these produce a robot that does the wrong thing with total confidence? [1 mark]

Tick every answer that is true.

- ☐ **A** Recording the wall once with the label open
- ☐ **B** Recording open only at the start
- ☐ **C** Recording three headings at each spot
- ☐ **D** Using labels that mean where the robot is, not what it sees
- ☐ **E** Recording after the robot has settled

> **Answer:** A, B, D. Wrong labels, one place only, and labels that mean where instead of what. Recording from more headings and after settling are the fixes.

3. Why does `drive_to` end with `wait(0.4)` before a sample is recorded? [1 mark]

- ☐ **A** So the readings are taken after the robot has settled, not while it is still sliding
- ☐ **B** The depth sensor needs 0.4 s to switch on
- ☐ **C** To give you time to type the label
- ☐ **D** `tof_grid()` only works after a wait

> **Answer:** A. A view taken while sliding is a mixture of places. Settle first, then record.

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

```
data = []
data.append(([54, 52, 51], "open"))
data.append(([16, 16, 16], "wall-ahead"))
print(len(data), data[-1][1], data[0][0][1])
```

> **Answer:**
>
> ```
> 2 wall-ahead 52
> ```
>
> Two samples. `data[-1][1]` is the last sample's label; `data[0][0][1]` is the second reading of the first sample.

5. Who decides the label on a sample? [1 mark]

- ☐ **A** You do, and if you get it wrong the model learns your mistake
- ☐ **B** The robot works it out from the readings
- ☐ **C** The depth sensor measures it
- ☐ **D** The model fixes wrong labels during training

> **Answer:** A. The label is your judgement. The model can only be as right as the labels you give it.

6. What does the label `gap-left` mean? [1 mark]

- ☐ **A** The wall ends on my left, wherever I happen to be
- ☐ **B** The robot is standing on the left spot, (-38, 45)
- ☐ **C** Turn left now
- ☐ **D** There is a gap somewhere on the left of the mat

> **Answer:** A. Labels describe what the robot sees, not where it is. The same view anywhere should get the same label.

The task: collect data

Visit the four spots in order, record the level rows at each with its label, and print `recorded <label>` each time.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
# maths: atan2, hypot, sin, cos, radians
import math

def wrapped(h):
    return (h + 180) % 360 - 180

def go_to(x, y, speed=60):
    # where am I?
    px, py = position()
    a = math.radians(wrapped(math.degrees(math.atan2(x - px, y - py)) - heading()))
    # forward, sideways, rotation: -100 to 100 each, until the next command
    drive(speed * math.cos(a), speed * math.sin(a), wrapped(0 - heading()) * 3)

def near(x, y, cm=4):
    # where am I?
    px, py = position()
    return math.hypot(x - px, y - py) < cm

def drive_to(x, y):
    while not near(x, y):
        go_to(x, y)
        # pause 0.1 s (the robot keeps doing what it was told)
        wait(0.1)
    # all motors off
    stop()
    # pause 0.4 s (the robot keeps doing what it was told)
    wait(0.4)
drive_to(0, 10)
print('recorded', 'open')
```

The hint students can ask for: Visit the four spots in order (open, wall-ahead, gap-left, gap-right), record the level rows at each with its label, and print `recorded <label>` each time. Positions relative to the start: (0, 10), (0, 45), (-38, 45), (38, 45).

A solution

```
from bugbot import *
connect()
import math

def wrapped(h):
    return (h + 180) % 360 - 180

def go_to(x, y, speed=60):
    px, py = position()
    a = math.radians(wrapped(math.degrees(math.atan2(x - px, y - py)) - heading()))
    drive(speed * math.cos(a), speed * math.sin(a), wrapped(0 - heading()) * 3)

def near(x, y, cm=4):
```

```
    px, py = position()
    return math.hypot(x - px, y - py) < cm

def drive_to(x, y):
    while not near(x, y):
        go_to(x, y)
        wait(0.1)
    stop()
    wait(0.4)
data = []
spots = [("open", 0, 10), ("wall-ahead", 0, 45), ("gap-left", -38, 45), ("gap-right", 38,
45)]
for label, x, y in spots:
    drive_to(x, y)
    data.append((tof_grid()[16:32], label))
    print("recorded", label)
print("samples:", len(data))
```

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