ANSWERS

3.1 What a controller is

BugBotLab

Control · Robot club · about 15 min

What this lesson is about

The error and its sign, wrapping headings, bang-bang control, overshoot.

- The error
- The simplest controller: bang-bang
- Overshoot
- Fixing it the crude way

Questions

7 marks in all

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

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

print(wrapped(200))
print(wrapped(350))
```

Answer:

```
-160
-10
```

Headings wrap round. 200 degrees is the same as 160 the other way, and 350 is the same as 10 to the left, so they become -160 and -10.

2. A controller uses `error = target - measured`. The target heading is 90 and the robot faces 60. What is the error? Give a whole number. [1 mark]

Answer: 30. 90 minus 60 is 30. It is positive, so the robot is short of the target and should turn clockwise.

3. What is bang-bang control? [1 mark]

- ☐ **A** Looking at the sign of the error and going that way at a fixed speed
- ☐ **B** Pushing harder the bigger the error is
- ☐ **C** Driving into a wall and backing off
- ☐ **D** Turning on and off at random

Answer: A. It is how a thermostat works: fully on one way or the other, with nothing in between.

4. The bang-bang loop calls `stop()` when the error is under 2 degrees, but the robot ends further off. Why? [1 mark]

- ☐ **A** The robot keeps turning for a moment after stop()
- ☐ **B** The heading sensor is wrong
- ☐ **C** wrapped() gives the wrong answer
- ☐ **D** abs() removes the sign

Answer: A. It coasts in rotation just as it does driving forward. The faster the spin, the bigger the overshoot.

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

```
for error in [30, -10]:
    speed = 60 if abs(error) > 25 else 20
    print(error, speed)
```

Answer:

```
30 60
-10 20
```

`abs()` ignores the sign. 30 is more than 25 so it gets 60, while -10 is only 10 away so it gets the slow speed of 20.

6. What does `break` do inside `while True:`? [1 mark]

- ☐ **A** Leaves the loop
- ☐ **B** Stops the robot's motors
- ☐ **C** Pauses the loop for a moment
- ☐ **D** Starts the loop again from the top

Answer: A. `while True` would repeat for ever. `break` is how you get out once the error is small enough.

7. What is the problem with the crude fix: fast when far, then short slow bursts with a pause? [1 mark]

- ☐ **A** It works, but it is slow at the end
- ☐ **B** It never gets within 2 degrees
- ☐ **C** It spins the wrong way
- ☐ **D** It only works for heading 0

Answer: A. Stopping and settling between bursts takes time. Proportional control, in lesson 3.2, does better.

The task: face north

Spin until the robot faces 0 within 3 degrees, printing `error:` and the error every time round the loop. Do not drive anywhere.

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

print("heading:", heading())
```

The hint students can ask for: Spin until heading() is within 3 degrees of 0, printing `error:` and the error each time round the loop. Do not drive anywhere.

A solution

```
from bugbot import *
connect()
def wrapped(h):
    return (h + 180) % 360 - 180
while True:
    error = wrapped(heading())
    print('error:', round(error, 1))
    if abs(error) <= 2:
        break
    speed = 60 if abs(error) > 25 else 20      # fast when far, careful when close
    if error > 0:
        turn_left(speed)
    else:
        turn_right(speed)
    wait(0.1)
    if abs(error) <= 25:
        stop()
        wait(0.3)     # let it settle before looking again
```

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