



5.4 Priorities

Name _____

Class _____

Date _____

What this lesson is about

Avoid beats seek: behaviours that take over and run to completion.

- Seek alone
- Avoid takes over
- The rule

Questions

7 marks in all

1. With behaviours ordered by priority, who controls the robot each tick? [1 mark]

- ☐ A The most urgent behaviour that wants control
- ☐ B All of them, averaged together
- ☐ C The one that was in charge last tick
- ☐ D The lowest priority behaviour, always

2. Put these layers in order, highest priority first. [1 mark]

Number the lines 1 to 3 to put them in the right order.

- ___ Seek: head for the goal
- ___ Emergency: the battery is low, so stop
- ___ Avoid: something is close ahead, so back off and slide

3. Once avoid takes over, it runs for ten ticks whatever the sensor says. Why? [1 mark]

- ☐ A So the robot actually gets clear instead of flickering between avoid and seek
- ☐ B Because the distance sensor only works every ten ticks
- ☐ C So seek can finish its turn first
- ☐ D To save battery

4. What does this program print?

[1 mark]

```
def choose(battery, dist, avoiding):
    if battery < 20:
        return "stop"
    elif avoiding or dist < 22:
        return "avoid"
    else:
        return "seek"

print(choose(80, 50, False))
print(choose(80, 15, False))
print(choose(10, 15, True))
print(choose(80, 60, True))
```

5. What does this program print?

[1 mark]

```
state = "seek"
avoid_ticks = 0
for d in [50, 20, 18, 30, 15, 60]:
    if state == "seek" and d < 22:
        state = "avoid"
        avoid_ticks = 0
    if state == "avoid":
        avoid_ticks += 1
        if avoid_ticks >= 3:
            state = "seek"
print(d, state)
```

6. Does `steer_to`, the seek behaviour, need to know that avoid exists?

[1 mark]

- ☐ A No: it just finds itself somewhere new and heads for the goal again
- ☐ B Yes: it has to undo the sideways slide
- ☐ C Yes: it must turn avoid off when it is done
- ☐ D No, because avoid never moves the robot

7. Why does `steer_to` only drive forward once the robot is facing within 25 degrees of the goal?

[1 mark]

- ☐ A So the distance sensor looks where the robot is going
- ☐ B Because the robot cannot turn and drive at once
- ☐ C So it reaches the goal faster in a straight line
- ☐ D Because `drive` ignores speed while turning

The task: seek, but safely

Reach the green zone at (82, 82) without touching either wall.

```
# 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 steer_to(x, y, speed=60):
    # where am I?
    px, py = position()
    bearing = math.degrees(math.atan2(x - px, y - py))
    error = wrapped(bearing - heading())
    # forward, sideways, rotation: -100 to 100 each, until the next command
    drive(speed if abs(error) < 25 else 0, 0, max(-60, min(60, error * 3)))

def near(x, y, cm=6):
    # where am I?
    px, py = position()
    return math.hypot(x - px, y - py) < cm

goal = (82 - 15, 82 - 15)
# do this 390 times (tick counts from 0)
for tick in range(390):
    if near(*goal, cm=5):
        # leave the loop
        break
    steer_to(*goal, speed=60)
    # pause 0.1 s (the robot keeps doing what it was told)
    wait(0.1)
# all motors off
stop()
```

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



Do it on the robot

www.bugbotlab.com/learn/5-4-priorities/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Slide left or right depending on which side of the depth grid has more room.
2. Add a third layer above avoid: if the battery is under 20, stop and print a warning. (Change the number to test it.)
3. Count how many times avoid took over, and print it at the end.