ANSWERS

7.6 Passing

Hands and feet · Robot club · about 20 min

What this lesson is about

Aim at a team mate's tag, judge the power from its distance.

- Find the receiver
- Aim
- Range
- Receiving

Questions

7 marks in all

1. A kick goes about 45 cm, and the ball starts about 8 cm in front of your centre. How far from you, in cm, should the receiver be? [1 mark]

> **Answer:** 53 (accept within 2). 45 + 8 = 53 cm. Farther and the pass drops short; nearer and it rolls past.

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

```
dist = 70
if dist > 56:
    print("forward", dist - 53)
elif dist < 50:
    print("backward", 53 - dist)
else:
    print("kick from here")
```

> **Answer:**
>
> ```
> forward 17
> ```
>
> 70 is more than 56, so carry the ball 70 - 53 = 17 cm closer before kicking.

3. Put the steps of a pass in order. [1 mark]

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

___ `gripper("close")`

___ `Carry the ball to about 53 cm from the receiver`

___ `aim_at(101) again`

___ `aim_at(101)`

___ `kick()`

Answer:

```
gripper("close")
aim_at(101)
Carry the ball to about 53 cm from the receiver
aim_at(101) again
kick()
```

Hold the ball, aim so the distance is measured along the right line, get to range, aim again because moving changes the bearing, then kick.

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

```
error = 20
print(max(-30, min(30, error * 3)))
error = -4
print(max(-30, min(30, error * 3)))
```

Answer:

```
30
-12
```

20 x 3 = 60 is clamped to 30 so the robot never spins too fast. -4 x 3 = -12 is inside the limits, so it turns gently anticlockwise.

5. Why does the passing code call `aim_at(101)` a second time after moving? [1 mark]

- ☐ **A** Driving forward or back changes the bearing a little
- ☐ **B** The tag's id changes when you move
- ☐ **C** `kick()` only works after two aims
- ☐ **D** The gripper opens when the robot drives

Answer: A. Small sideways drift while carrying the ball is enough to miss, so check the aim just before kicking.

6. You are the receiver. What should your program do? [1 mark]

- ☐ **A** Face the passer, wait, and close the jaws when `distance()` drops to a few centimetres
- ☐ **B** Close the jaws before the kick so they are ready
- ☐ **C** Drive at the ball as fast as you can
- ☐ **D** Wait until `holding()` says red, then close the jaws

Answer: A. Closed jaws cannot catch anything, and `holding()` only says red after a grip. The distance reading is the moment to close.

7. The receiver is 70 cm away and you kick without moving. What happens? [1 mark]

- ☐ **A** The ball stops about 17 cm short of the receiver
- ☐ **B** The ball rolls past the receiver
- ☐ **C** The ball reaches the receiver
- ☐ **D** The kick is stronger to make up the distance

> **Answer:** A. The ball always stops about 53 cm from you, whatever the receiver's distance, and 70 - 53 = 17.

The task: pass to your partner

Grip the ball, get to kicking range of the blue robot, aim, and kick so the ball stops within the square around it.

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

# close the jaws (it waits while they move)
gripper("close")
# kick: one turn of the kicker winds the spring and lets it go
kick()
# pause 5 s (the robot keeps doing what it was told)
wait(5)
```

The hint students can ask for: Your team mate is waiting up to the right with a blue dome. Grip the ball, carry it to about 53 cm from it (a kick goes 45 cm and the ball starts 8 cm ahead of you), turn until it is centred, and kick.

A solution

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

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

def bearing_of(cx):
    return (cx - 160) * 120 / 320

def go_to(x, y, speed=60):
    # one tick of sliding towards (x, y), positions relative to the start, holding heading
0
    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=5):
    px, py = position()
    return math.hypot(x - px, y - py) < cm

def find_ball(colour):
    # spin until a ball of this colour is in view
    set_cv('blob', colour)
    while not blobs():
        turn_right(40)
        wait(0.1)
    stop()
    wait(0.3)

def fetch(colour):
    # approach the ball with the camera, then grip it; returns True when holding it
    set_cv('blob', colour)
    for tick in range(200):
        found = blobs()
        if not found:
            turn_right(30)
        else:
```

```
            cx = found[0][0]
            if distance() < 6:
                stop()
                gripper("close")
                if holding():
                    return True
                backward(30)
                wait(0.3)
            else:
                drive(45 if abs(bearing_of(cx)) < 10 else 20, 0, bearing_of(cx) * 3)
        wait(0.1)
    stop()
    return False
def partner():
    return [r for r in robots() if r[0] == 'blue']

def aim_at_partner():
    for tick in range(100):
        seen = partner()
        if not seen:
            turn_right(30)
        else:
            error = bearing_of(seen[0][1])
            if abs(error) < 1.5:
                stop()
                wait(0.3)
                again = partner()
                if again and abs(bearing_of(again[0][1])) < 1.5:
                    return
            else:
                drive(0, 0, max(-30, min(30, error * 3)))
        wait(0.05)
    stop()
gripper("close")
aim_at_partner()
dist = partner()[0][3]
if dist > 56:
    forward(40, distance=dist - 53)
elif dist < 50:
    backward(40, distance=53 - dist)
aim_at_partner()
kick()
wait(5)
```

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