ANSWERS

9.4 Asking

Talking to each other · Robot club · about 20 min

What this lesson is about

A question and a reply, with a timeout; going where you are told.

- The same map
- Going there
- Ask again

Questions

6 marks in all

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

```
import math
sx, sy = 30, 40
x, y = 0, 0
print(f"the scout is {math.hypot(sx - x, sy - y):.0f} cm away")
```

Answer:

```
the scout is 50 cm away
```

`hypot` gives the straight-line distance: 30 across and 40 up is 50 cm.

2. The Scout says `at 25,55`. You stop 13 cm in front of it, aiming for `(sx, sy - 13)`. What y, in cm, do you drive to? [1 mark]

Answer: 42. 55 - 13 = 42. Aiming short by a robot's length stops you driving into the Scout.

3. Before two robots can share positions, what must they agree? [1 mark]

- ☐ **A** The same origin and the same directions for x and y
- ☐ **B** The same speed
- ☐ **C** The same colour LED
- ☐ **D** The same heading

Answer: A. A position is only useful if both measure it the same way. On this mat both count from your start.

4. You drove to where the Scout said it was, but the Scout had moved. What should your program have done? [1 mark]

- ☐ **A** Asked again, because the answer was only true when it was sent
- ☐ **B** Driven faster
- ☐ **C** Trusted the first answer
- ☐ **D** Sent its own position to the Scout

Answer: A. A robot that moves needs asking again, over and over.

5. In the Scout task, a student writes `others()` to find the Scout. What happens? [1 mark]

- ☐ **A** It does not work: `others()` exists only in games, so in a task you have to ask by radio
- ☐ **B** It returns the Scout's position
- ☐ **C** It returns the Scout's reply text
- ☐ **D** It sends `where are you?` for you

> **Answer:** A. Games hand out every robot's position for free. A real team, and this task, gets it only by asking.

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

```
import math
sx, sy = 30, 40
print(round(math.hypot(sx - 0, (sy - 13) - 0)))
```

> **Answer:**
>
> ```
> 40
> ```
>
> The stopping point is (30, 27), which is about 40 cm from the start: sqrt(900 + 729) is 40.4.

The task: meet the scout

Ask the Scout where it is and stop in the green square, 13 cm in front of it, without touching it.

```
# 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 ask(question, seconds=1.0):
    # send a question and return the first reply that arrives within `seconds`, or None
    send(question)
    for tick in range(int(seconds * 10)):
        # pause 0.1 s (the robot keeps doing what it was told)
        wait(0.1)
        for sender, text in messages():
            return text
    return None

def numbers_in(text):
    # every number in a message, as floats: "at 25,55" -> [25.0, 55.0]
    out = []
    for word in text.replace(",", " ").split():
        try:
            out.append(float(word))
        except ValueError:
            pass
    return out
reply = ask('where are you?')
print('scout says:', reply)
```

The hint students can ask for: Ask the Scout `where are you?` and it answers `at <x>,<y>` measured from your start. Go and stop about 13 cm in front of it (the green square), without touching it.

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 ask(question, seconds=1.0):
    # send a question and return the first reply that arrives within `seconds`, or None
    send(question)
    for tick in range(int(seconds * 10)):
        wait(0.1)
        for sender, text in messages():
            return text
    return None

def numbers_in(text):
    # every number in a message, as floats: "at 25,55" -> [25.0, 55.0]
    out = []
    for word in text.replace(",", " ").split():
        try:
            out.append(float(word))
        except ValueError:
            pass
    return out
reply = ask("where are you?")
print("scout says:", reply)
sx, sy = numbers_in(reply)
target_x, target_y = sx, sy - 13                  # a robot's length short of it
while not near(target_x, target_y):
    go_to(target_x, target_y)
    wait(0.1)
stop()
```

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