ANSWERS

9.3 Protocols

Talking to each other · Robot club · about 20 min

What this lesson is about

Agreeing what a message looks like; splitting it into words and numbers.

- Asking and waiting
- Taking a message apart
- Strict out, forgiving in

Questions

7 marks in all

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

```
words = "forward 30".split()
command, amount = words[0], float(words[1])
print(command, amount * 2)
```

Answer:

```
forward 60.0
```

`split()` gives the words `forward` and `30`. `float` turns the text `30` into the number 30.0, which can do sums.

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

```
def numbers_in(text):
    out = []
    for word in text.replace(",", " ").split():
        try:
            out.append(float(word))
        except ValueError:
            pass
    return out

print(numbers_in("ball at -12,40 ok"))
```

Answer:

```
[-12.0, 40.0]
```

Commas become spaces, so the words are ball, at, -12, 40 and ok. Only -12 and 40 turn into numbers.

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

```
text = "  Forward 30 "
words = text.lower().strip().split()
print(words)
```

Answer:

```
['forward', '30']
```

`lower()` makes Forward into forward and `strip()` removes the spaces at the ends. Forgiving in.

4. Why does `ask` give up after a second instead of waiting for a reply forever? [1 mark]

- ☐ **A** Radios miss things, and a program waiting for a reply that never comes is stuck
- ☐ **B** The Guide only speaks for one second
- ☐ **C** `messages()` stops working after a second
- ☐ **D** To save battery

> **Answer:** A. The timeout is part of the protocol. `None` means no answer came in time.

5. A message arrives saying `jump 10`, which your program does not understand. What should it do? [1 mark]

- ☐ **A** Ignore it and carry on
- ☐ **B** Stop with an error so you notice
- ☐ **C** Treat it as forward 10
- ☐ **D** Send it back to whoever sent it

> **Answer:** A. Be strict about what you send and forgiving about what you receive. A robot that crashes on an odd message is not much of a team mate.

6. `ask("next")` returns `None`. What does that mean? [1 mark]

- ☐ **A** No reply arrived within the time allowed
- ☐ **B** The Guide said none
- ☐ **C** The robot has reached the green zone
- ☐ **D** The message was sent to nobody

> **Answer:** A. `None` is not an instruction. Check for it before calling `split()`, or the program crashes.

7. What is the word for the agreement, made in advance, about what messages look like? [1 mark]

> **Answer:** protocol. Here the protocol is one word then a number, like `forward 30`, with `done` at the end.

The task: follow instructions

Ask the Guide for instructions one at a time, do each one, and stop at `done`. You will end up in the green zone.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()
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('next')
print('guide says:', reply)
```

The hint students can ask for: Send `next` and the Guide replies with one instruction: `forward <cm>`, `right <cm>` or `done`. Do each one, ask again, and stop at `done`. You end up in the green zone.

A solution

```
from bugbot import *
connect()
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
while True:
    reply = ask("next")
```

```
    print("guide says:", reply)
    if reply is None:
        continue
    words = reply.split()
    if words[0] == "forward":
        forward(50, distance=float(words[1]))
    elif words[0] == "right":
        right(50, distance=float(words[1]))
    elif words[0] == "left":
        left(50, distance=float(words[1]))
    elif words[0] == "done":
        break
stop()
```

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