WORKSHEET

3.6 Project: precision parking

Control · Robot club · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

Round a box and into a bay facing east, on the worst robot in the course.

- The plan
- Leg one as a function
- Leg two: sideways, holding two things
- Leg three: turn and hold
- Where next

Questions

6 marks in all

1. In `hold_line(0, 67)`, what does x = 0 mean? [1 mark]

- ☐ **A** The line the robot started on
- ☐ **B** The left edge of the mat
- ☐ **C** The middle of the box
- ☐ **D** The parking bay

2. In `slide_right`, which part of the `drive` is the speed? [1 mark]

```
drive((target_y - y) * 10, 70, wrapped(0 - heading()) * 4)
```

- ☐ **A** The sideways part, 70
- ☐ **B** The forward part, holding y
- ☐ **C** The rotation part, holding the heading
- ☐ **D** There is no speed, it uses distance

3. `hold_heading(90)` is running and the robot faces 60. What rotation does `max(-60, min(60, error * 3))` ask for? [1 mark]

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

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

target = 90
for heading in [350, 80, 120]:
    error = wrapped(target - heading)
    print(heading, max(-60, min(60, error * 3)))
```

5. Put the parking program's main lines in order, after the functions have been defined. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `hold_line(0, 67)`

___ `print("facing", round(heading(), 1))`

___ `hold_heading(90)`

___ `slide_right(67, 55)`

6. Why does `hold_heading` loop `while abs(error) > 2` instead of until the error is exactly 0? [1 mark]

- ☐ **A** The heading is never exactly on target, so it would never stop
- ☐ **B** abs() cannot give 0
- ☐ **C** An error of 2 is the same as 0
- ☐ **D** Python cannot compare with 0

The task: precision parking

Reach the bay without touching the box and finish facing 90, within 6 degrees.

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

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

def hold_line(target_x, until_y):
    while position()[1] < until_y:
        # where am I? (cm from where I started)
        x, y = position()
        # forward, sideways, rotation: -100 to 100 each, until the next command
        drive(70, (target_x - x) * 10, wrapped(0 - heading()) * 4)
        # pause 0.1 s (the robot keeps doing what it was told)
        wait(0.1)
    # all motors off
    stop()

hold_line(0, 67)
```

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

Do it on the robot

www.bugbotlab.com/learn/3-6-project-parking/

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

Challenges

1. Park in under 10 seconds.
2. Go round the *right* of the box instead and arrive from below.
3. Write `go_to(x, y)` that drives to any point using `math.atan2` for the heading and the two-loop line holder, and park with two calls to it.