WORKSHEET

F6.7 Project: the fail-safe controller

BugBotLab

Robust programs · GCSE · about 25 min

Name _____ Class _____ Date _____

What this lesson is about

A command program that authenticates, validates every command, and is tested against a plan.

- The brief
- Decompose it
- Step 1: validate one command
- Step 2: log in
- Step 3: the main loop
- Test plan for the whole program

Questions

4 marks in all

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

```
def parse(command):
    parts = command.strip().lower().split()
    if len(parts) != 2 or parts[0] not in ["forward", "back"]:
        return None
    if not parts[1].isdigit() or not 1 <= int(parts[1]) <= 50:
        return None
    return (parts[0], int(parts[1]))

print(parse("Forward 20"), parse("forward 60"), parse(""))
```

2. Why put all the command checks in one function, parse? [1 mark]

- ☐ **A** Every command goes through the same checks, and they can be tested on their own
- ☐ **B** Functions run faster
- ☐ **C** Python requires validation in a function
- ☐ **D** It avoids using a loop

3. In the controller's test plan, an empty command is which kind of test data? [1 mark]

- ☐ **A** Erroneous
- ☐ **B** Normal
- ☐ **C** Boundary
- ☐ **D** Valid

4. Which comes first in the controller, and why? [1 mark]

- ☐ **A** Authentication, so no commands are accepted from someone who is not allowed
- ☐ **B** Validation, so the password is sensible
- ☐ **C** Testing, so the robot is safe
- ☐ **D** The main loop

The task: the fail-safe controller

Build the controller from the brief with `operators = {"ada": "robot42"}`. Print `access denied` for a wrong login, `welcome <name>` for a right one, `invalid command` for every command it refuses, and `bye` on quit. The task types: `ada` and `oops`, then `ada` and `robot42`, then the commands `forward 20`, `fly 10`, `right 500`, an empty line, `right 15`, `beep` and `quit`.

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

operators = {"ada": "robot42"}
MOVES = ["forward", "back", "left", "right"]
```

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

Do it on the robot

www.bugbotlab.com/learn/f6-7-project-fail-safe-controller/

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

Challenges

1. Refuse a `forward` command that would take the robot closer than 10 cm to a wall, using `distance()`.
2. Keep a log of every command in a list, with whether it was obeyed, and print it at `quit`.
3. Log out automatically after five invalid commands in a row, and ask for the login again.