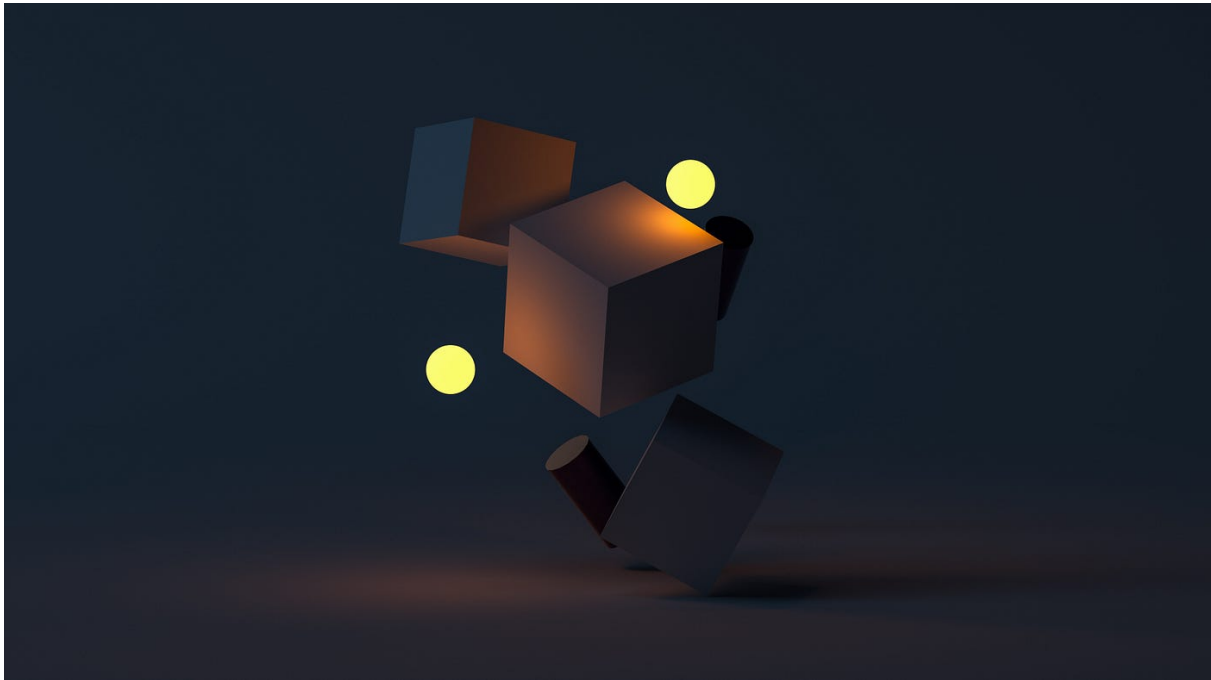


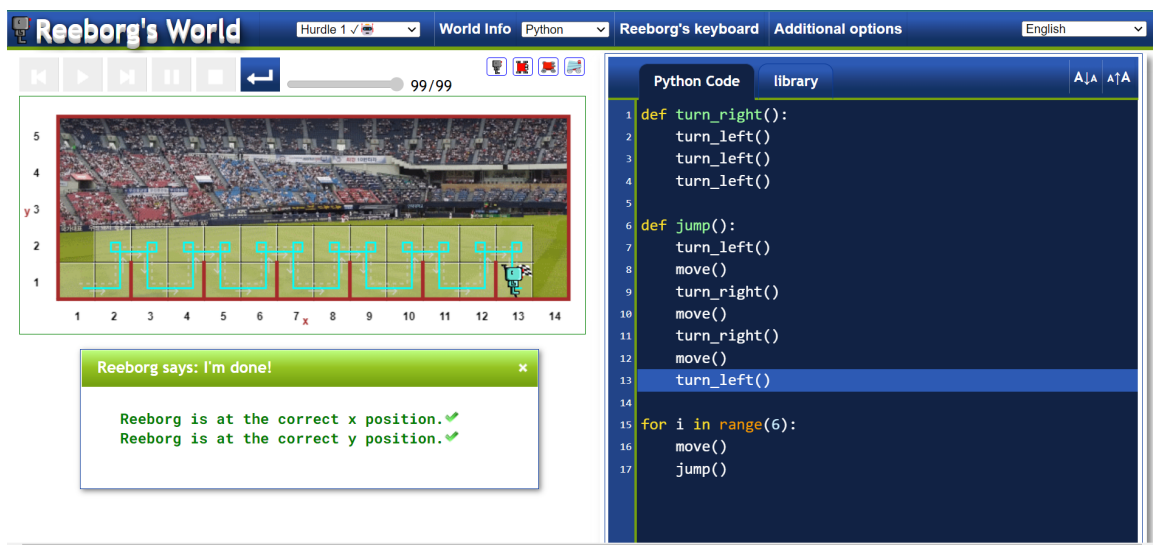


Day 6 - Reeborg's World

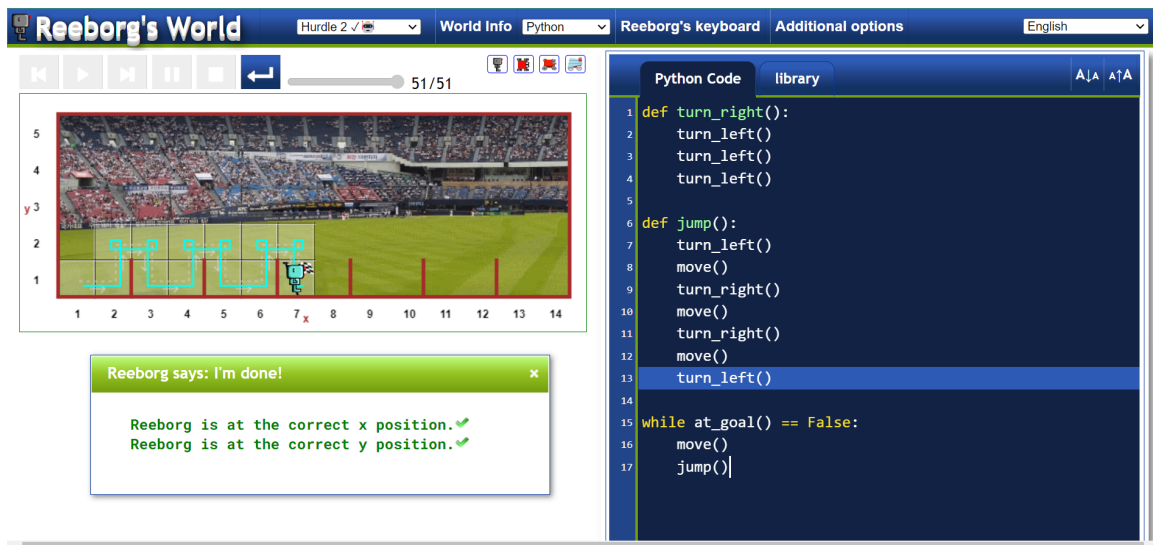


Activities:

1. Making the borg move through the hurdles course by defining a function `jump()`:

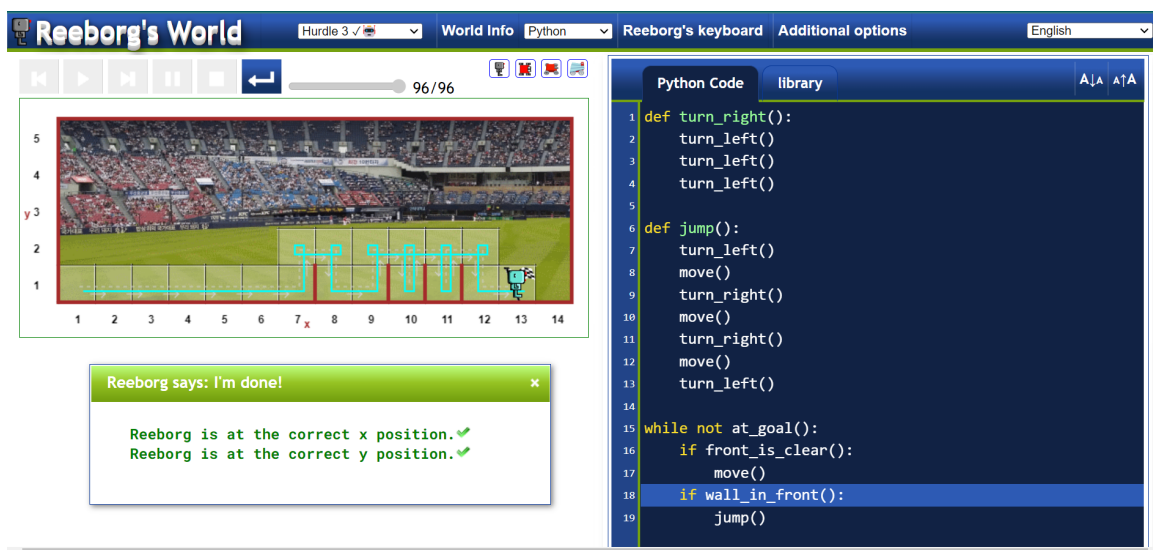


2. Now in each run, the flag is randomly set after any of the hurdles. We need to stop when we reach the goal, which is given by the predefined function `at_goal()`. So, I used a while function:



{Can also do - `while not at_goal():` }

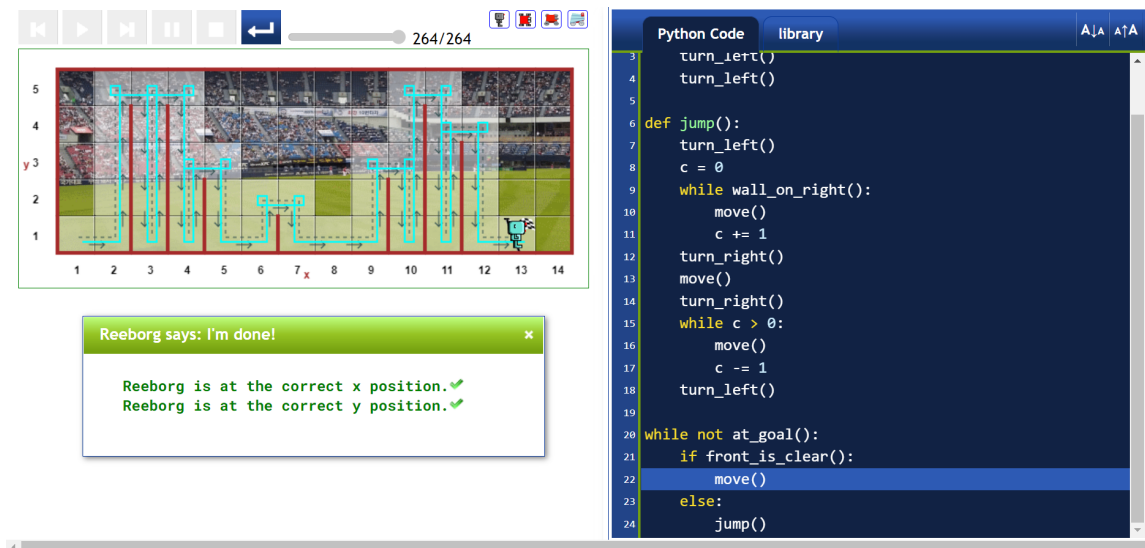
3. This is a hurdles course with a variable number and variable position of walls:



{More efficiently, I could use an if-else}

```
while not at_goal():
    if front_is_clear():
        move()
    else:
        jump()
```

4. Now, the hurdles course has a variable height of walls, too; I modified the jump() function:



The screenshot shows the Reeborg's World interface. On the left, a 14x5 grid represents the world with a blue robot at (7,1). A path of red arrows shows the robot's movement, jumping over walls of varying heights. A green box at the bottom left says "Reeborg says: I'm done!" and "Reeborg is at the correct x position. ✓" and "Reeborg is at the correct y position. ✓". On the right, the Python Code editor shows the following code:

```
Python Code library A1A A7A
3 turn_left()
4 turn_left()
5
6 def jump():
7     turn_left()
8     c = 0
9     while wall_on_right():
10        move()
11        c += 1
12    turn_right()
13    move()
14    turn_right()
15    while c > 0:
16        move()
17        c -= 1
18    turn_left()
19
20 while not at_goal():
21     if front_is_clear():
22         move()
23     else:
24         jump()
```

The instructor used the predefined function front_is_clear():

```
def jump():
    turn_left()
    while wall_on_right():
        move()
    turn_right()
    move()
    turn_right()
    while front_is_clear():
        move()
    turn_left()
```

Final Project

Lost in a maze

Reeborg was exploring a dark maze and the battery in its flashlight ran out.

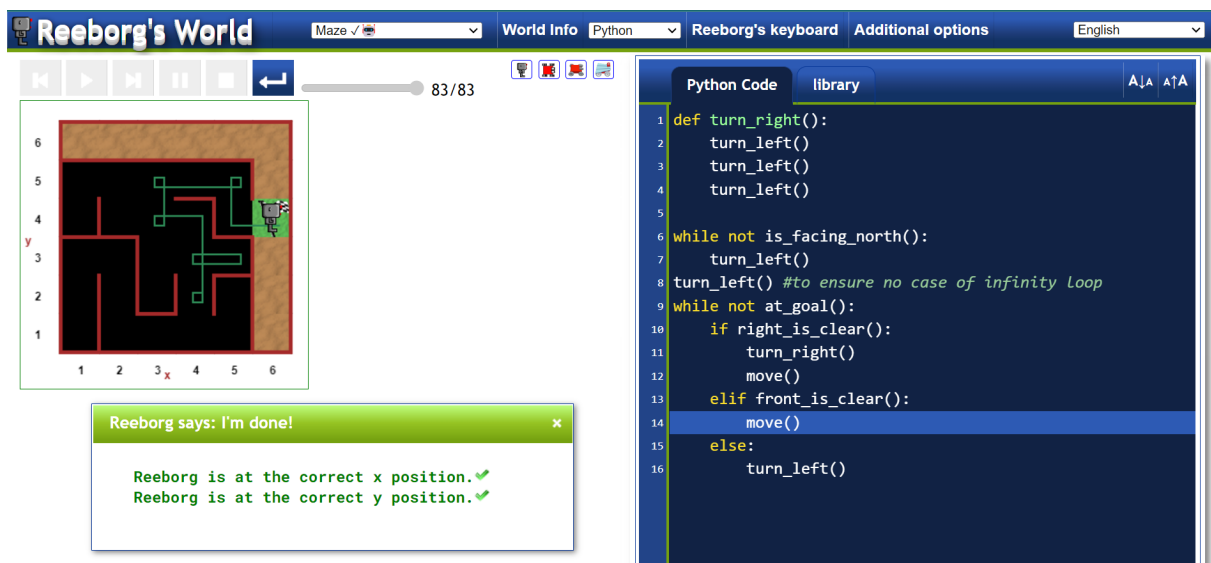
Write a program using an **if/elif/else** statement so Reeborg can find the exit. The secret is to have Reeborg follow along the right edge of the maze, turning right if it can, going straight ahead if it can't turn right, or turning left as a last resort.

What you need to know

- The functions **move()** and **turn_left()**.
- Either the test **front_is_clear()** or **wall_in_front()**, **right_is_clear()** or **wall_on_right()**, and **at_goal()**.
- How to use a **while** loop and **if/elif/else** statements.
- It might be useful to know how to use the negation of a test (**not** in Python).

My Solution:

Given that we need to follow the strategy of keeping along the right edge of the maze:



NOTE: My strategy to avoid an infinite loop in the central-ish region of the maze proved to be a lot more efficient than the instructor's solution in terms of the number of steps required before reaching the goal. This held true for all the 3 test cases provided by the instructor. My strategy was that an infinite loop will be avoided if I ensure that the borg necessarily faces the left before starting.

The instructor's strategy was to ensure that the borg has a wall in front of it before it starts:

```
1 def turn_right():
2     turn_left()
3     turn_left()
4     turn_left()
5
6 while front_is_clear():
7     move()
8     turn_left()
9
10 while not at_goal():
11     if right_is_clear():
12         turn_right()
13         move()
14     elif front_is_clear():
15         move()
16     else:
17         turn_left()
18
```