Robot Software Education Institute
로봇 SW 교육원

로봇활용 소프트웨어 교육

파이썬 #3

광운대학교 로봇학부
박광현

일정

튜플, 딕셔너리, 모듈	13:00 ~ 14:30
미로 주행	14:40 ~ 16:40

튜플, 딕셔너리, 모듈

튜플(Tuple)

간결함을 추구하는 파이썬에서
리스트와 비슷한 튜플을 굳이 왜 만들었을까?

```
empty = []
numbers = [1, 2, 3]
texts = [‘hello’, ‘hamster’]
mixed = [1, ‘hello’, 2, ‘hamster’]
embedded = [1, 2, [1, 2]]
```

```
empty = ()
numbers = (1, 2, 3)
texts = (‘hello’, ‘hamster’)
mixed = (1, ‘hello’, 2, ‘hamster’)
embedded = (1, 2, (1, 2))
```

(1,) 콤마가 필요한 이유는?

1, 2, 3

```
numbers = [1, 2, 3, 4, 5, 6]
numbers[3:5]
numbers[3:-1]
numbers[3:]
numbers[:3]
```

```
numbers = (1, 2, 3, 4, 5, 6)
```

(4, 5)만 뽑아 내기

```
print numbers[3:5]

print numbers[3:-1]

print numbers[3:]

print numbers[:3]
```

(4, 5, 6)까지

(1, 2, 3) 뽑아 내기

```
test = (1, (2, 'hello', 'hamster'), (3, 4, 5))
```

('hello', 'hamster') 빼야 내기

```
print test[1][1:]
```

(3, 4) 빼야 내기

```
print test[-1][:2]
```

```
print [1, 2, 3] + [‘hello’, ‘hamster’]
print [1] * 10
print (1, 2, 3) * 3
```

```
(1, 2, 3)
(‘hello’, ‘hamster’)
```

```
(1, 2, 3, ‘hello’, ‘hamster’)
```

```
(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

```
numbers = [1, 2, 3]   →   [1, 5, 3]

numbers[1] = 5
print numbers


numbers = (1, 2, 3)   →   (1, 5, 3)
```

딕셔너리(Dictionary)

키-밸류 쌍

```
{key1:value1, key2: value2}

score = {'park': 10, 'kim': 20}
print score['park']

menu = { 1: 'fish', 2: 'pasta' }
print menu[1]

menu[3] = 'chicken'

del menu[1]
```

모듈

adder.py

```
def sum(a, b):
      return a + b
```

test.py

```
import adder
print adder.sum(1, 2)
```

모듈

test.py

```
from adder import sum
print sum(1, 2)
```

미션 #1

제어 모듈

controller.py

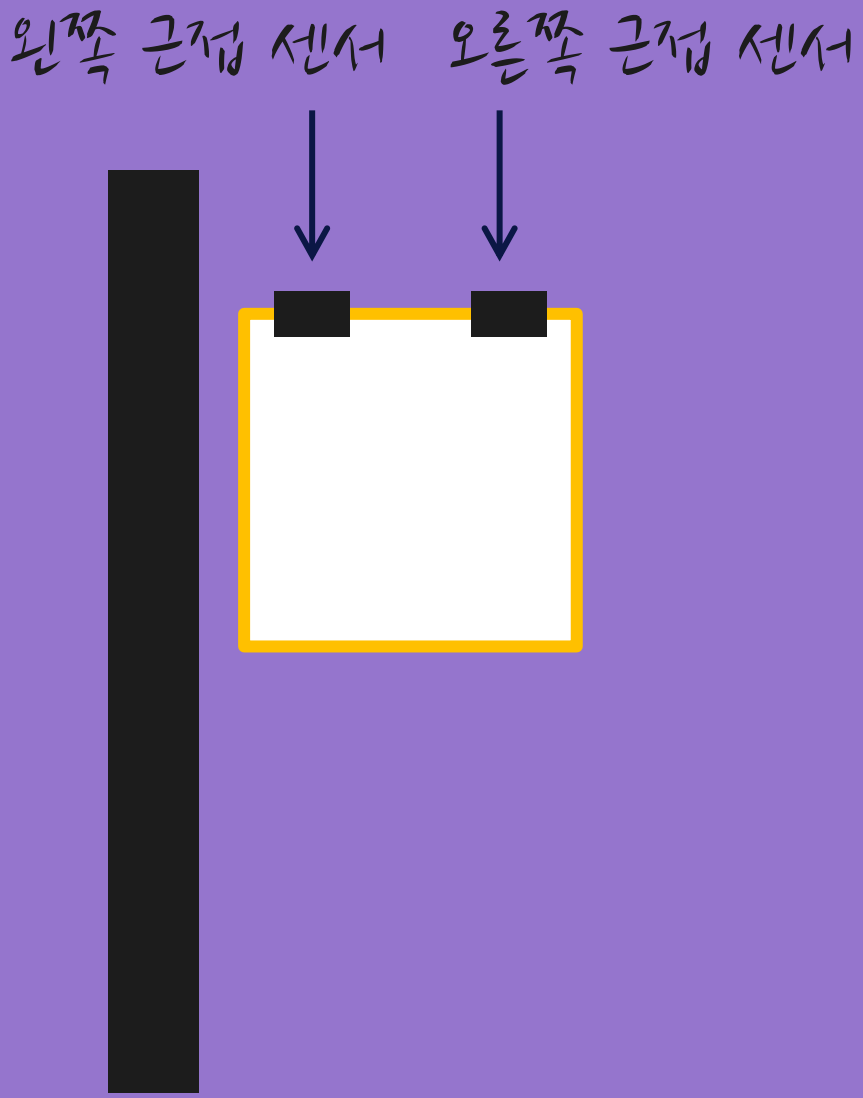
```
def motors(left, right)
def leds(left, right)
```

test.py

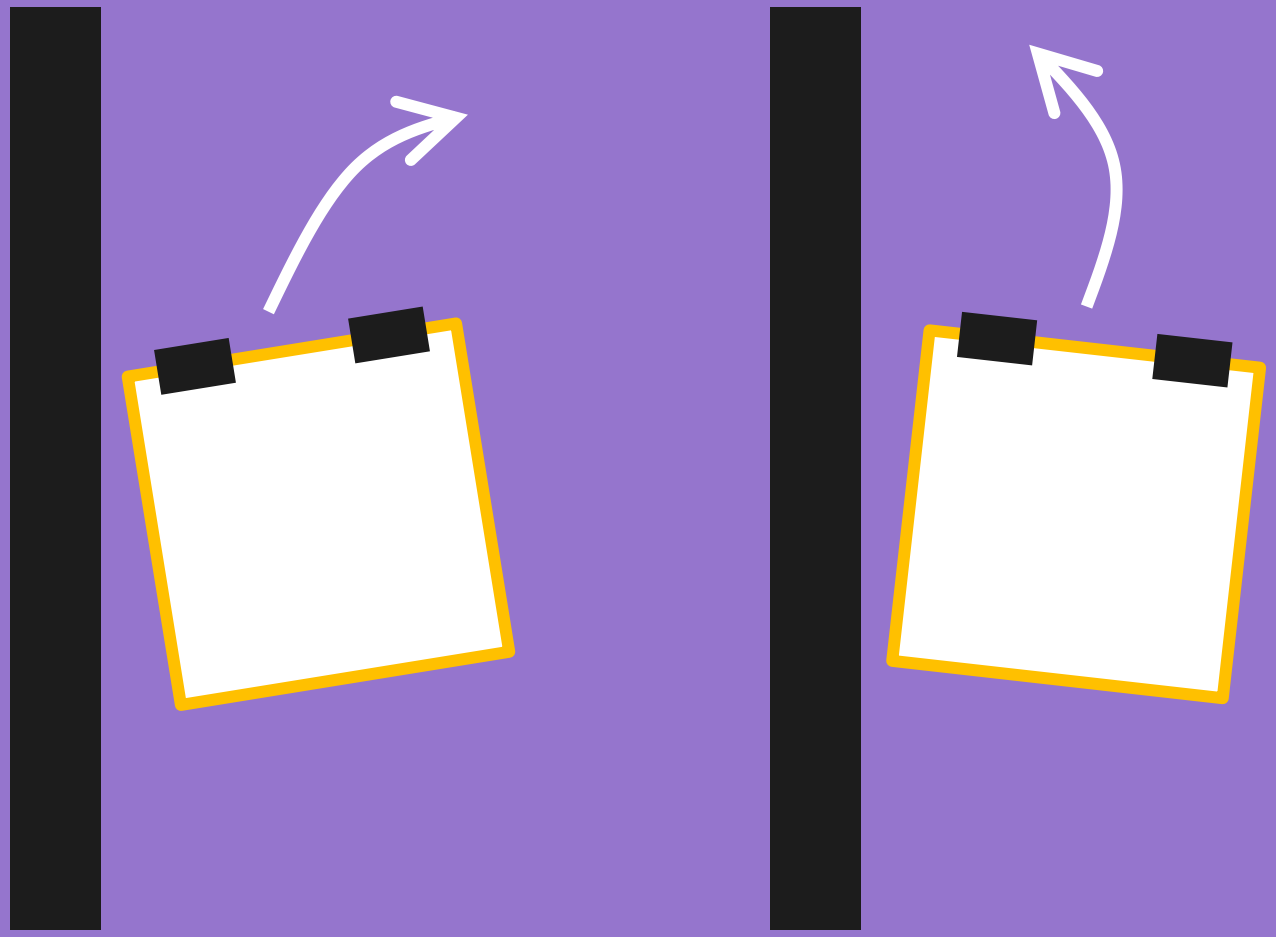
```
import controller
controller.motors(30, 30)
controller.leds(2, 3)
```

미로 주행

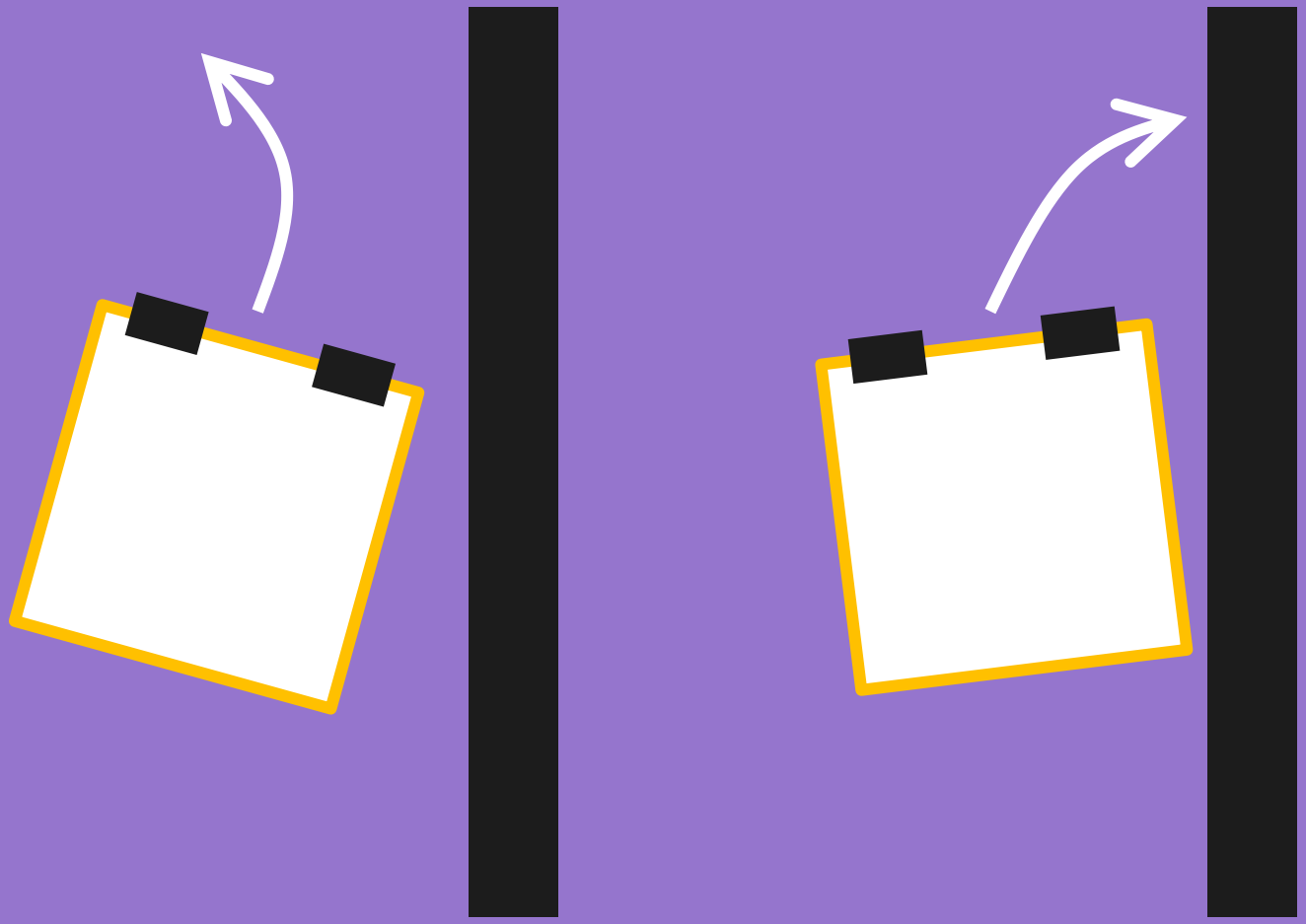
왼쪽 벽 따라가기



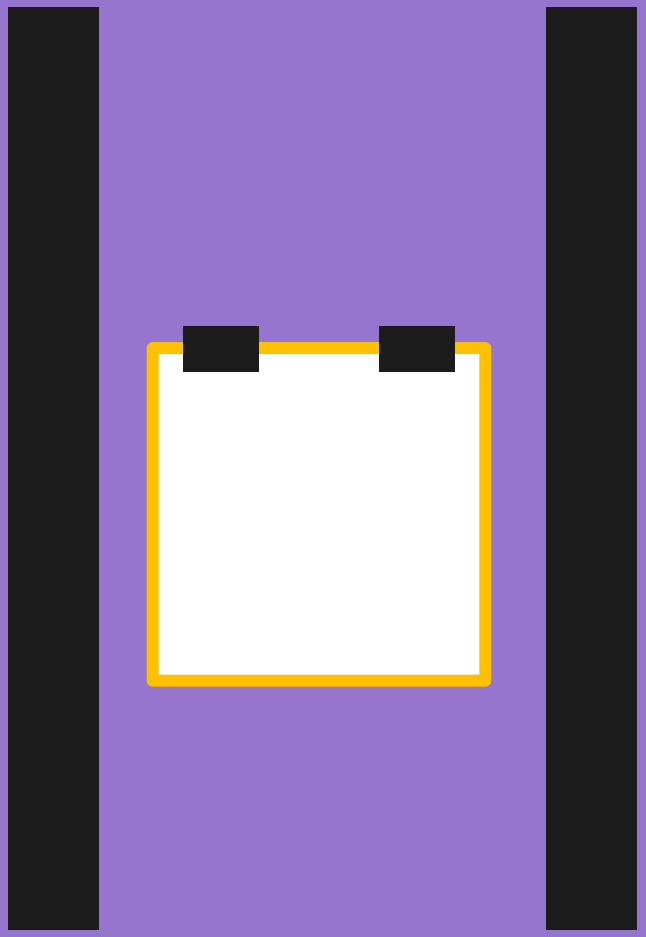
왼쪽 벽 따라가기



오른쪽 벽 따라가기



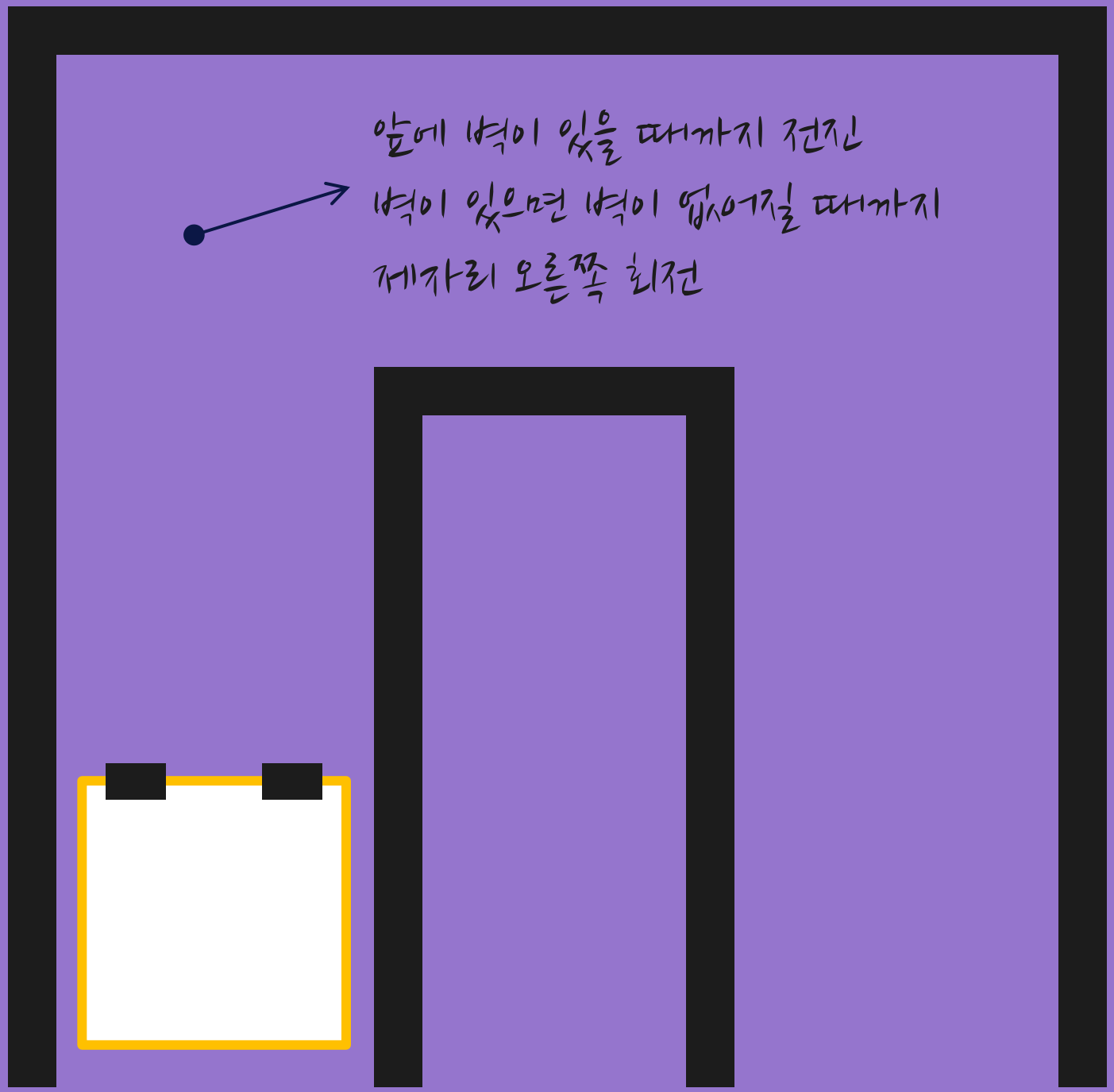
양쪽 벽 따라가기

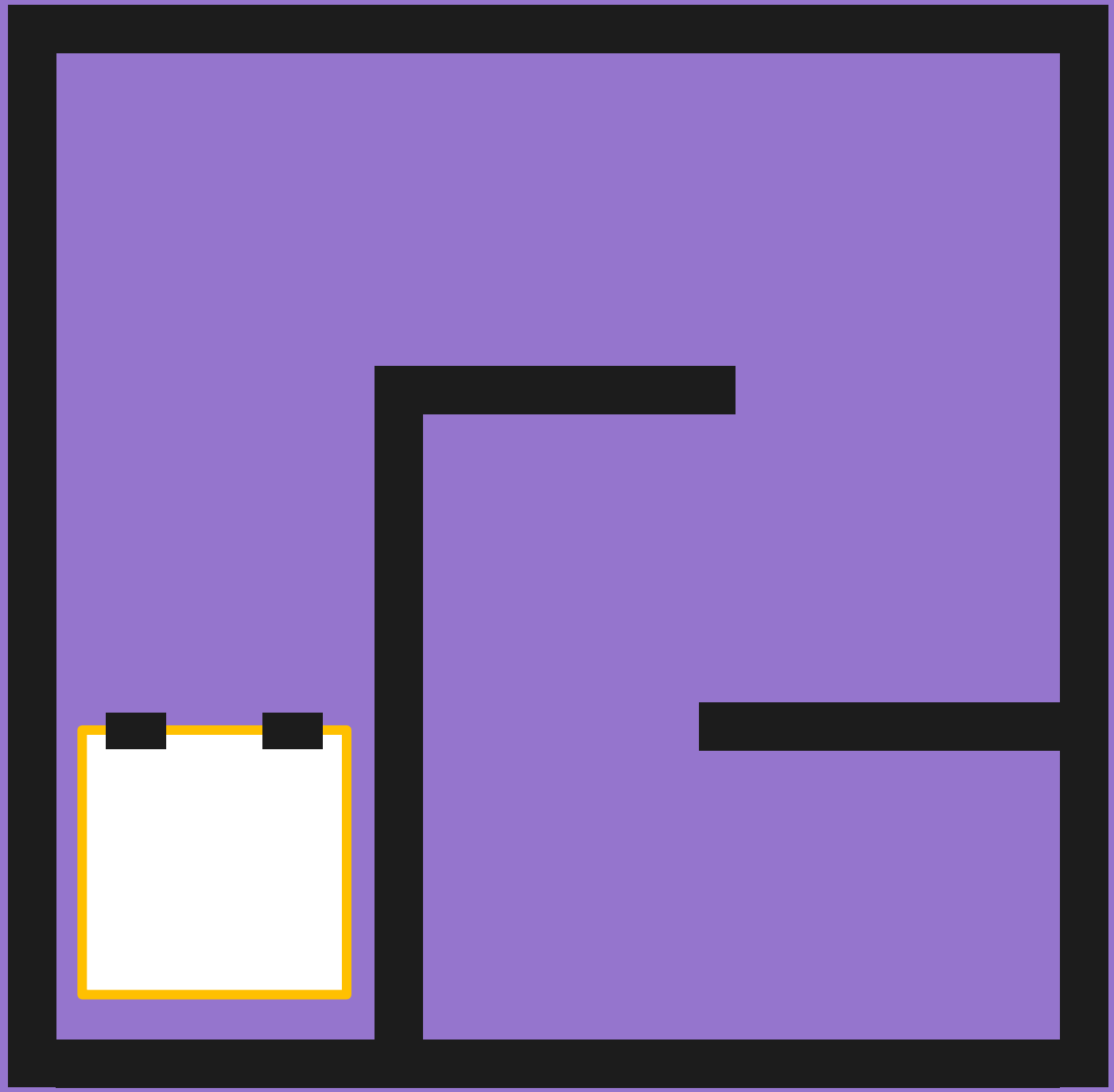


힌트

```
leftProximity = robot.read(Hamster.LEFT_PROXIMITY)
rightProximity = robot.read(Hamster.RIGHT_PROXIMITY)
diff = leftProximity - rightProximity

robot.write(Hamster.LEFT_WHEEL, a + diff * b)
robot.write(Hamster.RIGHT_WHEEL, a - diff * b)
```


앞에 벽이 있을 때까지 전진
벽이 있으면 벽이 없어질 때까지
제자리 오른쪽 회전



수고하셨습니다 !

akaii@kw.ac.kr