단원명 : RUR-PLE

강의 : 김윤영 (구리고등학교)

1. RUR-PLE

(1) RUR-PLE은?

RUR-PLE(RUR-Python Learning Environment)는 André Roberge가 개발한 파이썬 언어를 이용한 프로그래밍 학습 환경이다.

(2) RUR-PLE의 특징

1) 로봇을 이용한 프로그래밍 학습

프로그래밍을 처음 접하는 학생들에게 추상적인 수치를 다루는 내용보다는 간단한 명령의 조작을 통해 눈에 보이는 로봇을 조작함으로써 프로그래밍을 보다 쉽게 이해할 수 있도록 한다.

2) 파이썬 언어 사용

문법은 간단하지만 강력한 기능을 지원하고 다양한 분야에 이용되는 파이썬 언어로 로봇을 제어한다.

3) 튜토리얼 제공

혼자서 학습할 수 있는 튜토리얼(영문)을 제공한다.

4) 파이썬 쉘과 에디터 제공

파이썬를 설치하지 않아도 기본적으로 파이썬 쉘(2.5.4 버전)과 에디터를 제공한다.

5) 다양한 운영체제에서 지원

Windows, OSX, Linux에서 실행할 수 있는 배포판을 제공한다.

(3) 설치하기

1) RUR-PLE 설치파일 다운로드

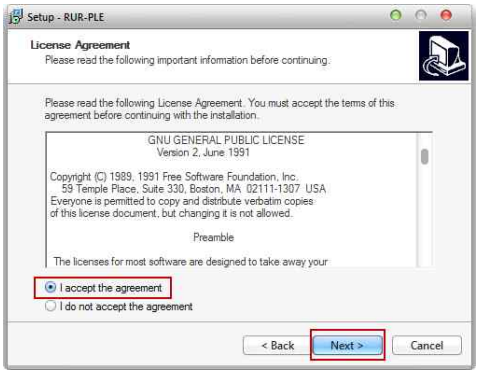
① http://rur-ple.sourceforge.net 에 접속하고 go to the download page를 클릭한다.

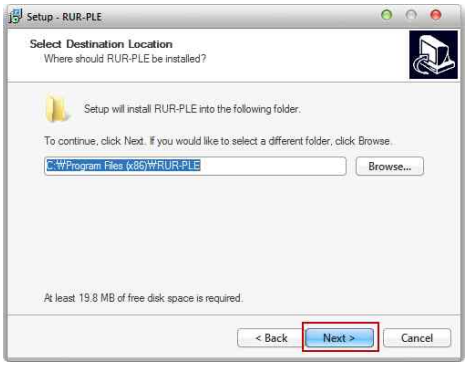
② Download windows-setup-rurple1.0rc3.exe (5.1MB)를 클릭하여 파일을 다운로드 받는다.

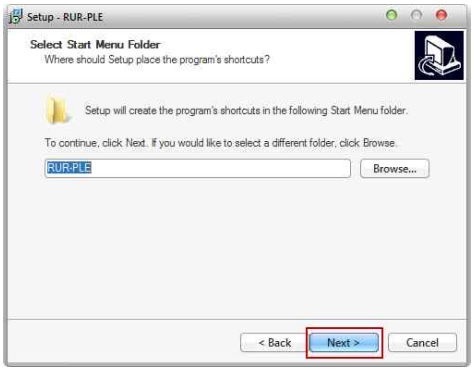
2) RUR-PLE 설치

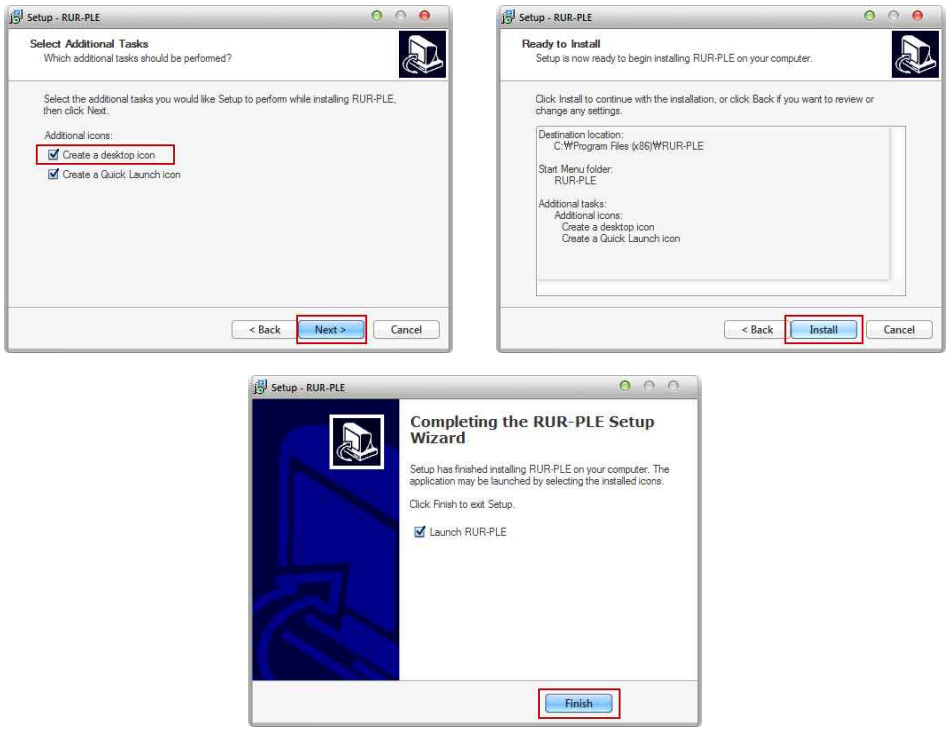
① 다운 받은 파일을 실행하고, 언어는 영어를 선택한다

② 순서에 따라 설치한다.









(4) 프로그램 구성

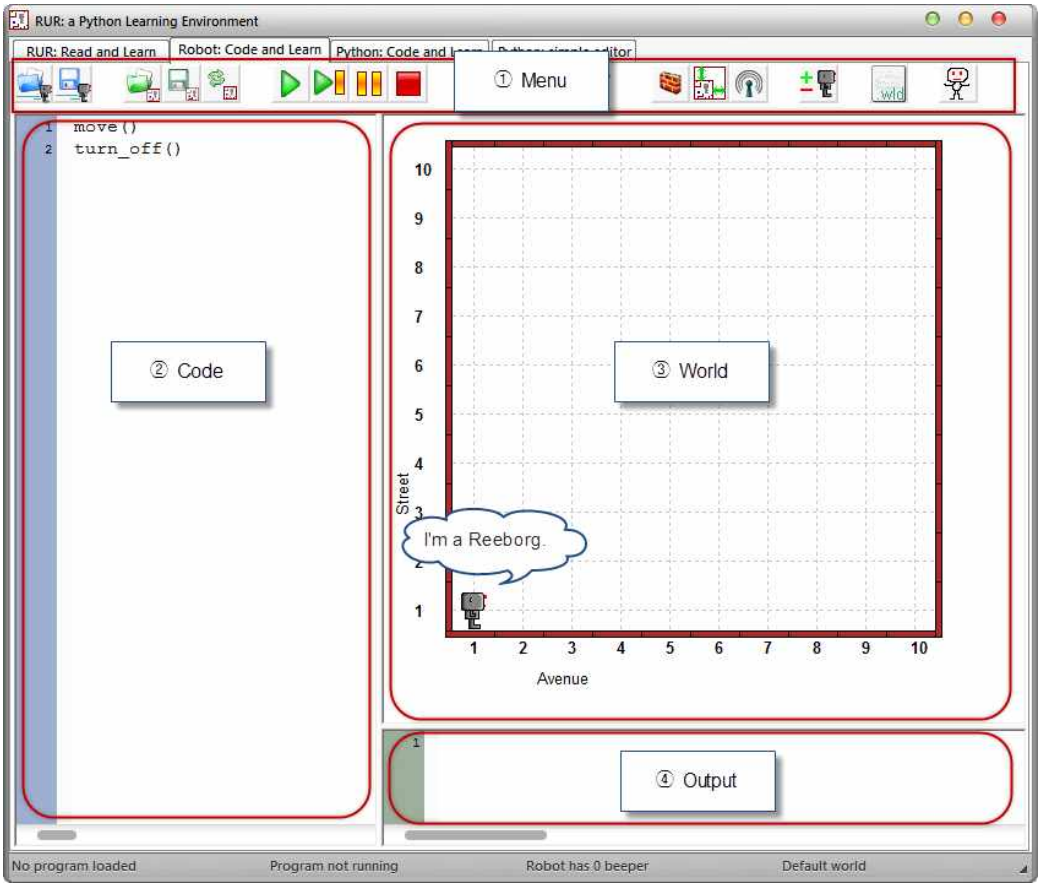
RUR-PLE은 4개의 탭으로 구성되어 있으며 각각의 텝은 다음과 같다.

- RUR: Read and Learn(튜토리얼)
- **Robot: Code and Learn(로봇을 이용한 프로그래밍 학습 환경)**
- Python: Code and Learn(파이썬 쉘)
- Python: Simple editor(파이썬 에디터)

본 교재는 4가지 탭에서 2번째 탭인 Robot을 이용한 프로그래밍 학습에 중점을 둔다.

(5) Robot 화면 구성

1) 화면구성



① Menu : 코드파일 및 월드파일 열기, 저장, 실행 등의 메뉴버튼

② Code : 리봇이 동작할 명령을 입력하는 부분

③ World : 리봇이 움직일 가상의 공간

④ Output: 데이터를 출력할 수 있는 부분

2) 메뉴구성

2. 기본 명령 I

(1) 수동 조작

키보드의 방향키를 이용하여 리봇을 이동시킬 수 있다.

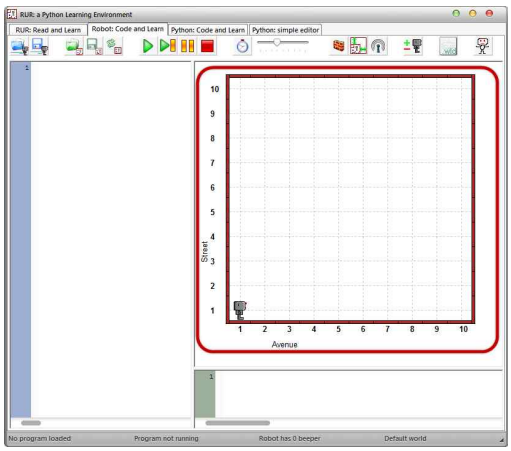
방향키는 ↑ ← 만 사용가능하다.

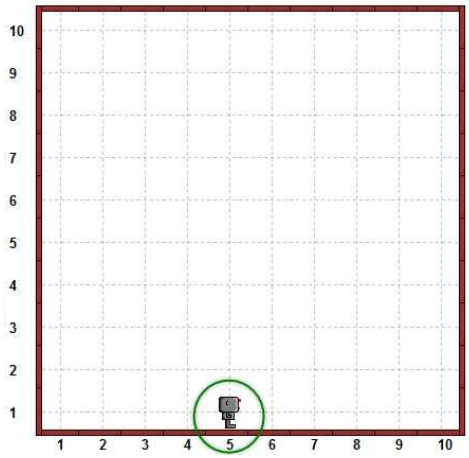
↑ : 리봇이 바라보는 방향으로 한 칸 이동

← : 제자리에서 왼쪽으로 90도 회전

[따라 하기]

1) 월드 부분을 클릭하고 방향키 ↑ 를 4번 입력한다.

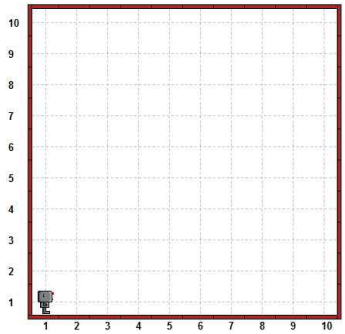


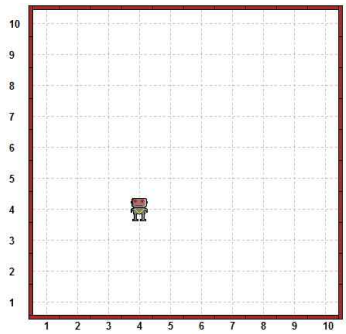


2) 결과를 확인하고  를 클릭하면 리봇이 초기 위치로 돌아간다.

[스스로 하기 1]

리봇을 다음과 같이 이동 시키시오.

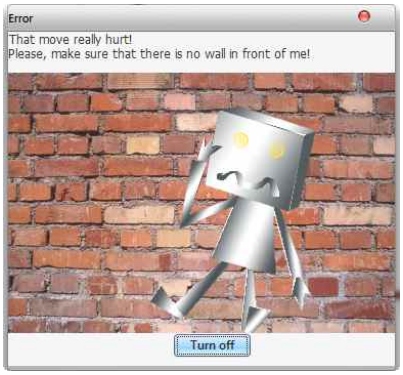




[스스로 하기 2]

리봇을 이동하다가 벽에 부딪쳐 보시오.

벽에 부딪치면 아래와 같은 에러창이 나타난다. 에러가 발생하면 리봇은 더 이상 움직이지 않기 때문에 에러가 발생하지 않도록 주의해야한다.

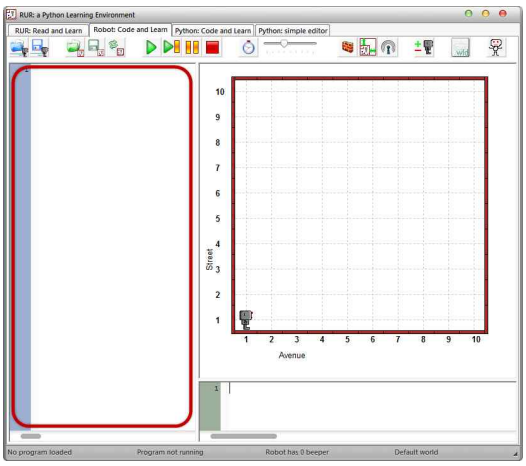


(2) move()

move()는 키보드의 방향키 ↑와 같은 역할을 하는 명령으로 리봇을 현재 방향으로 한 칸 앞으로 이동시킨다.

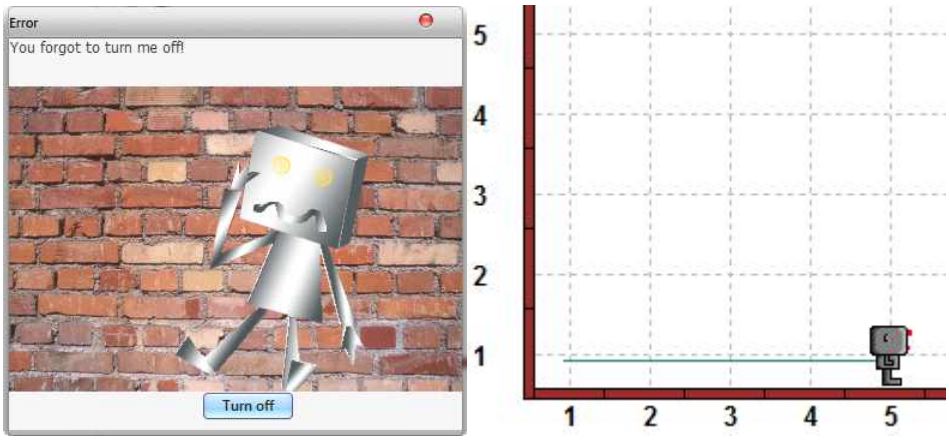
[따라 하기]

1) 코드 부분을 클릭하고 아래와 같이 입력한다.




```
1 move()
2 move()
3 move()
4 move()
```

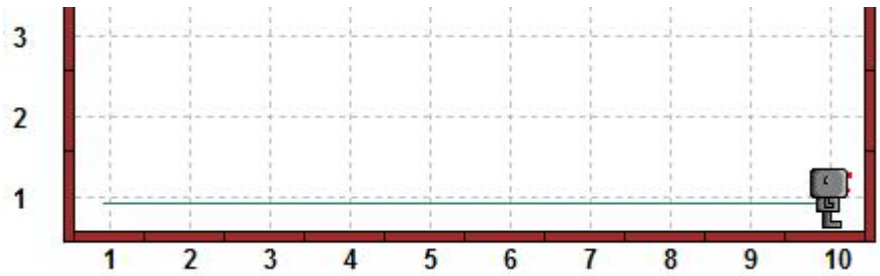
2) 상단 메뉴의  버튼을 클릭하여 코드를 실행시키고 결과를 확인한다.



전원을 끄지 않았다는 에러가 발생한다(리봇은 코드 마지막에 전원 끄기 명령을 주어야 한다). 일단 Turn off 버튼을 클릭하면 리봇이 4칸 전진한 결과를 확인할 수 있다. 키보드로 움직인 것과 다르게 자신이 이동한 경로가 표시된다.

[스스로 하기 3]

아래와 같은 결과가 나오도록 코드를 작성하시오.



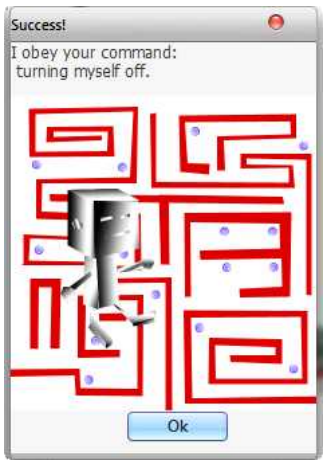
(3) turn_off()

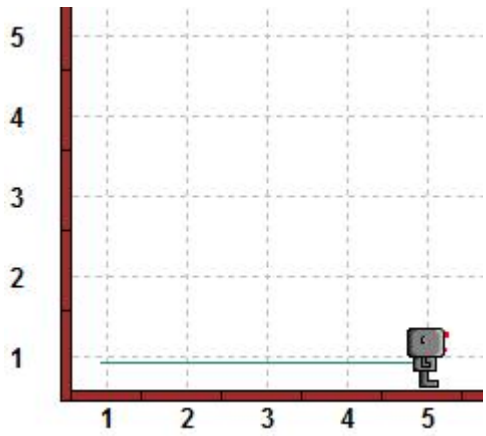
turn_off()는 리봇의 전원을 끄는 명령이다. 따라서 turn_off()가 실행되면 리봇은 더 이상 움직이지 않는다.

[따라 하기]

아래의 코드를 입력하고 실행하여 결과를 확인한다.

```
1  move()
2  move()
3  move()
4  move()
5  turn_off()
```





[스스로 하기 4]

아래 코드의 결과를 예상해 본 후, 입력하여 확인하고 예상한 결과와 비교하시오.

```
1  move()
2  move()
3  turn_off()
4  move()
5  move()
```

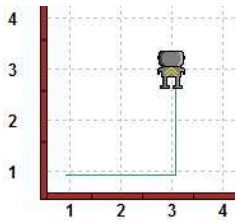
(4) turn_left()

turn_left()는 키보드의 방향키 ←와 같은 역할을 하는 명령으로 리봇을 현재 위치에서 왼쪽으로 90도 회전 시킨다.

[따라 하기]

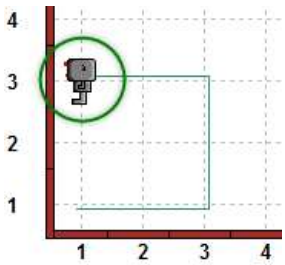
1) 아래 코드를 입력하고 실행하여 결과를 확인한다.

```
1  move()
2  move()
3  turn_left()
4  move()
5  move()
6  turn_off()
```



[스스로 하기 5]

아래와 같은 결과가 나오도록 코드를 작성하시오.



※ 리봇의 방향에 주의하시오.

[스스로 하기 6]

아래 코드의 결과를 예상해 본 후, 입력하여 확인하고 예상한 결과와 비교하시오.

```
1  move()
2  move()
3  turn_left()
4  turn_left()
5  move()
6  move()
7  turn_off()
```

(5) 주석문

주석문은 주로 코드에 대한 설명이나 일시적으로 해당 코드를 실행하고 싶지 않을 때 사용한다. # 기호 이후에 나오는 문자는 실행하지 않는다.

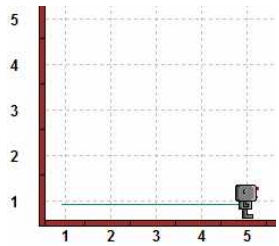
[따라 하기]

1) 아래 코드를 입력하고 실행하여 결과를 확인한다.

```
1  My first program
2  move()
3  move()
4  turn_left()
5  move()
6  move()
7  turn_off()
```

첫 번째 줄에서 에러가 발생한다. My first program 이라는 명령을 리봇은 이해하지 못한다. My first program은 리봇이 실행할 명령이 아니라 이 프로그램에 대한 설명이기 때문에 주석처리 해야 한다.

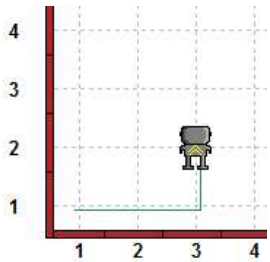
```
1  #My first program
2  move()
3  move()
4  #turn_left()
5  move()
6  move()
7  turn_off()
```



4번째 줄의 turn_left() 명령이 주석처리 되었기 때문에 실행되지 않았다.

[스스로 하기 7]

아래와 같은 결과가 나오기 위해 코드를 주석처리 하시오.




```
1  move()
2  move()
3  turn_left()
4  move()
5  turn_left()
6  move()
7  turn_off()
```

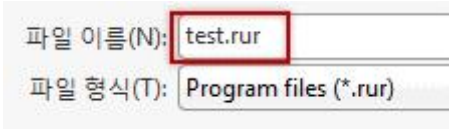
(6) 코드 저장하기

코드를 재사용하기 위해 저장한다.

[따라 하기]

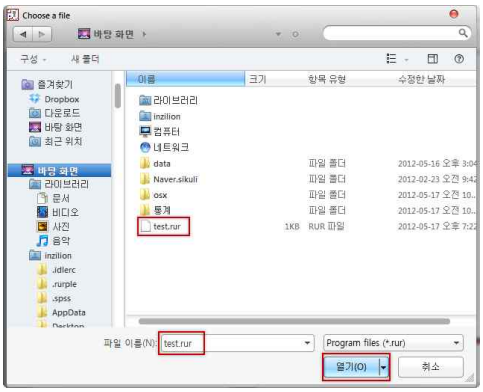
1) 상단 메뉴의  버튼을 클릭한다.

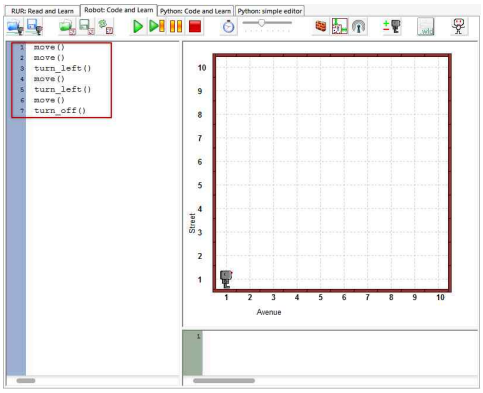
2) 저장할 폴더를 선택하고 원하는 파일명을 입력한 후 확장자를 rur로 지정하여 저장한다.



3) RUR-PLE 프로그램을 재실행한 후, 상단 메뉴의  버튼을 클릭한다.

4) 파일을 저장한 폴더로 이동하여 코드파일을 선택하고 열기버튼을 클릭한다.





3. 월드 꾸미기

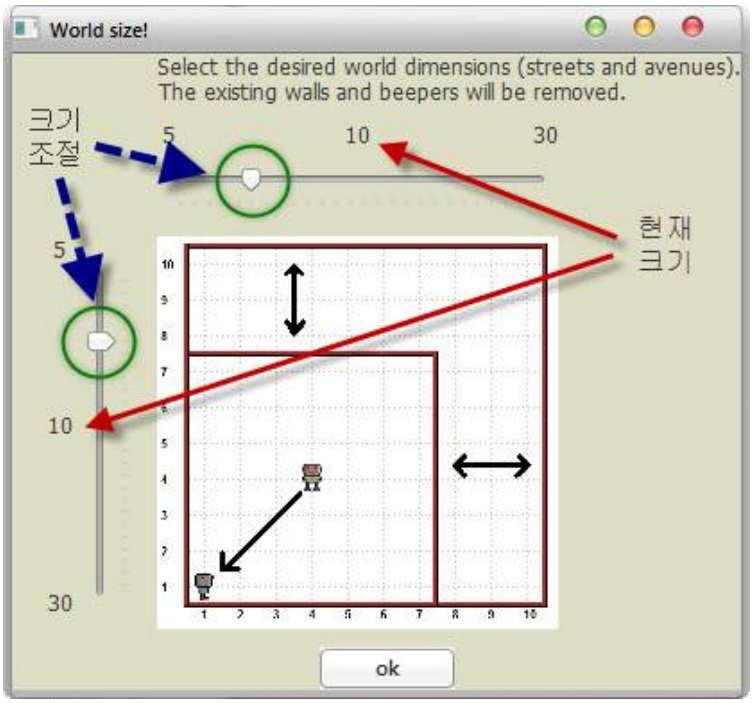
(1) 월드 크기 바꾸기

RUR-PLE의 기본 월드 크기는 10×10 이지만, 월드의 크기를 최소 5×5에서부터 최대 30×30 까지 변경할 수 있다.

[따라 하기]

1) 상단 메뉴의  버튼을 클릭한다.

2) World Size 창이 뜨면 게이지를 조절하여 월드 크기를 변경한다.



3) 상단 메뉴의  버튼을 클릭하면 월드의 초기 상태로 돌아간다.

[스스로 하기 8]

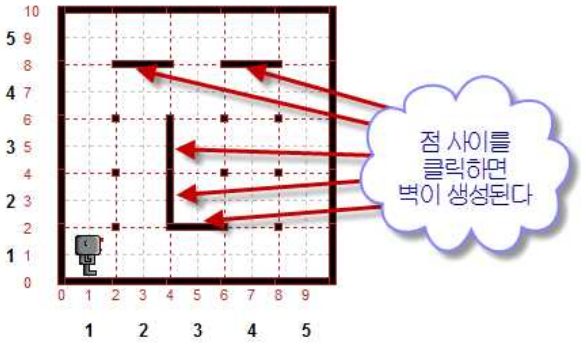
월드의 크기를 8×7로 변경하시오.

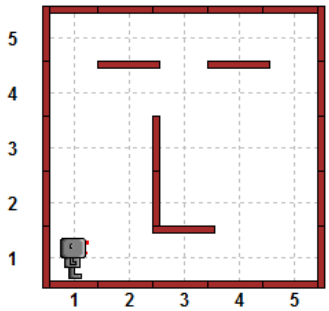
(2) 벽 만들기

RUR-PLE 월드 내에 벽을 만들어 리봇의 이동에 제약을 줄 수 있다.

[따라 하기]

1) 상단 메뉴의  버튼을 클릭하면 화면에 점이 생기고, 그 점 사이를 클릭하면 벽이 생성된다. 생성된 벽을 클릭하면 사라진다.



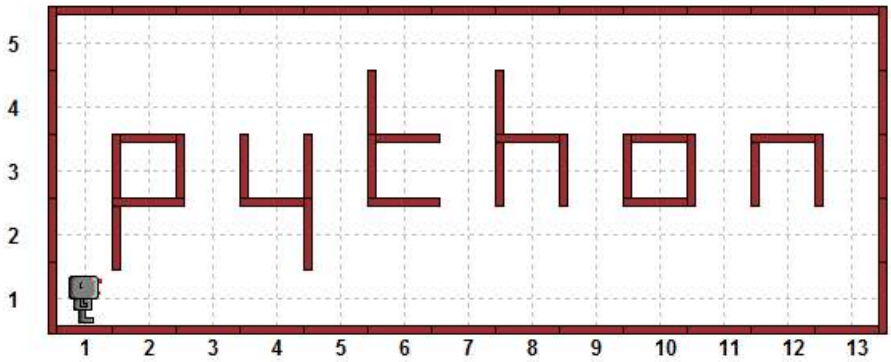


2)  버튼을 다시 클릭하면 화면의 점이 사라지고 생성된 벽만 남는다.

3) 월드를 수정하고 싶을 때  버튼을 다시 클릭한다.

[스스로 하기 9]

아래와 같은 월드를 만드시오.

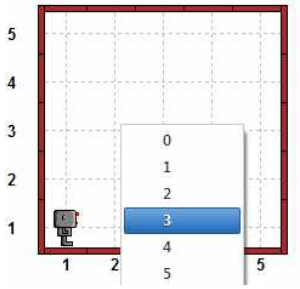


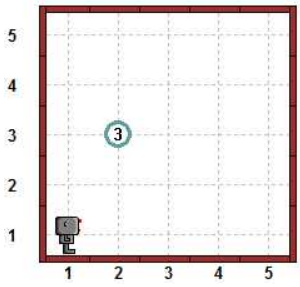
(3) 비퍼 놓기

비퍼(beeper)는 RUR-PLE 월드에 배치할 수 있는 물건이며 다양한 용도로 쓰인다. 비퍼는 리봇이 주울 수 있다(물론 다시 버릴 수 있다).

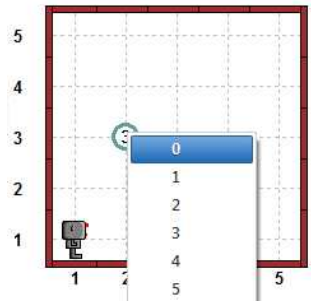
[따라 하기]

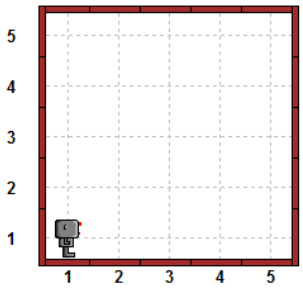
1) 월드에서 마우스 오른쪽 버튼을 클릭하면 아래(좌)와 같이 숫자가 적힌 메뉴가 나온다. 아래(우)와 같이 선택한 숫자가 적힌 비퍼가 생성된다.





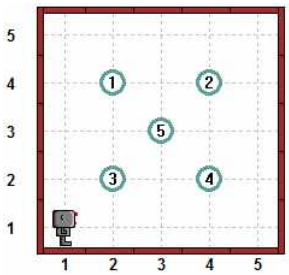
2) 다시 비퍼 위에서 마우스 오른쪽 버튼을 클릭하여 아래(좌)와 같이 0을 클릭하면 비퍼가 사라진다.





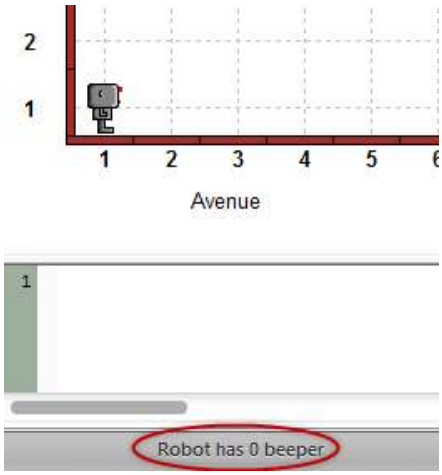
[스스로 하기 10]

아래와 같이 월드에 비퍼를 생성하시오.



(4) 리봇에게 비퍼 주기

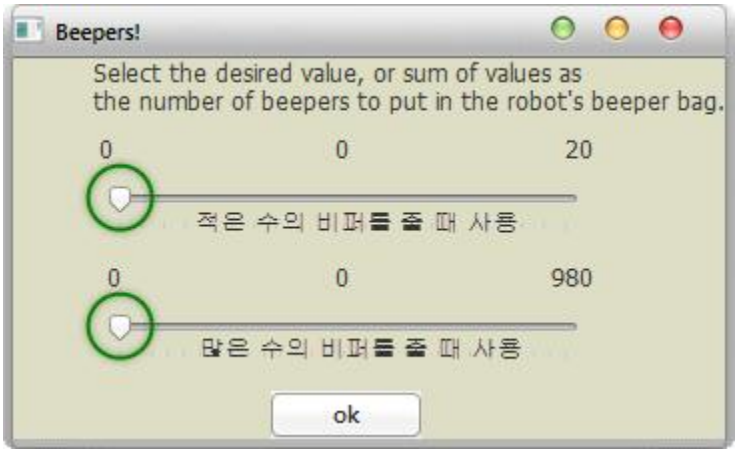
리봇은 비퍼를 보관할 수 있는 가방이 있다(보이진 않는다). 가방에 보관하고 있는 비퍼의 개수를 화면 하단의 상태 창에서 확인할 수 있다.



 버튼을 클릭하여 리봇에게 비퍼를 줄 수 있다.

[따라 하기]

1) 상단 메뉴의  버튼을 클릭하여 리봇에게 비퍼를 준다.



2) 상태 창에서 리봇이 보관하고 있는 비퍼의 수를 확인한다.

(5) 리봇 삭제/추가

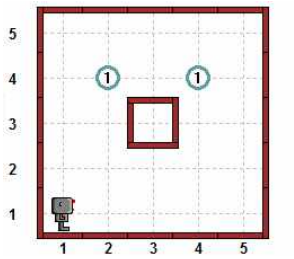
 버튼을 클릭하면 현재 월드에 리봇이 삭제/추가 된다.

(6) 월드 저장

월드를 재사용하기 위해 저장한다.

[따라 하기]

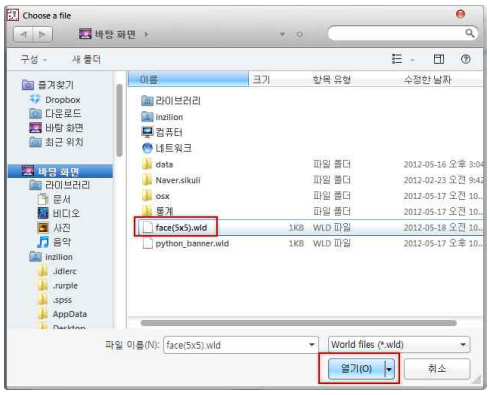
1) 아래와 같은 월드를 만들고, 상단 메뉴의  버튼을 클릭한다.

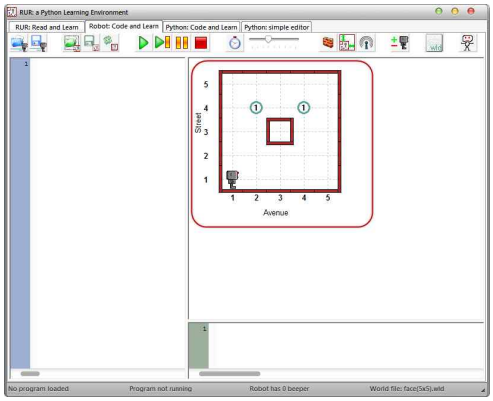


2) 저장할 폴더를 선택하고 원하는 파일명을 입력한 후 확장자를 wld로 지정하여 저장한다.

3) RUR-PLE 프로그램을 재실행한 후, 상단 메뉴의  버튼을 클릭한다.

4) 해당 폴더로 이동하여 월드파일을 선택하고 열기버튼을 클릭한다.





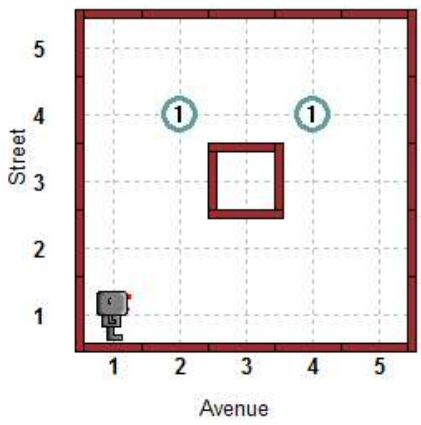
(7) 월드 파일 보기

 버튼을 클릭하여 월드 파일이 저장된 형태를 확인할 수 있다.

[따라 하기]

1) 상단 메뉴의  버튼을 클릭한다.

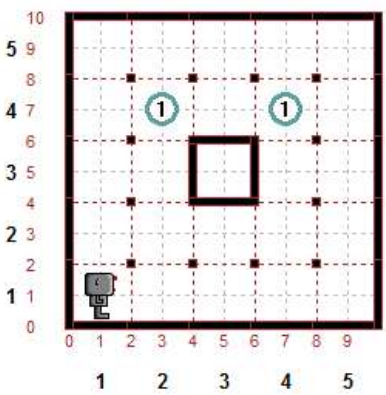
2) 아래와 같은 월드 파일의 내용을 확인할 수 있다.




```
avenues = 5
streets = 5
robot = (1, 1, 'E', 0)
walls = [
    (4, 5),
    (5, 6),
    (6, 5),
    (5, 4)
]
beepers = {
    (4, 4): 1,
    (2, 4): 1
}
```

3) 다시  버튼을 클릭하면 월드 파일 정보가 사라진다.

※ 월드파일 내용 설명



```
avenues = 5
streets = 5
robot = (1, 1, 'E', 0)
walls = [
    (4, 5),
    (5, 6),
    (6, 5),
    (5, 4)
]
beepers = {
    (4, 4): 1,
    (2, 4): 1
}
```

월드의 가로 크기
월드의 세로 크기
로봇정보(x좌표, y좌표, 방향, 비퍼 수)
벽 정보(자료형:리스트)

비퍼 정보(자료형:사전)

3. 함수(명령) 만들기

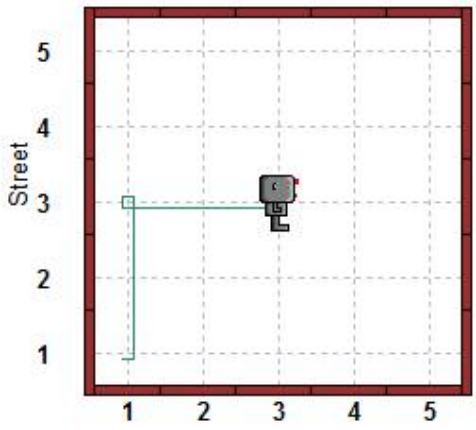
(1) turn_right()는 없다

리봇은 오른쪽으로 회전할 수 없다. 하지만 왼쪽으로 3번 회전하면 오른쪽으로 회전한 것과 같은 효과를 낼 수 있다.

[따라 하기]

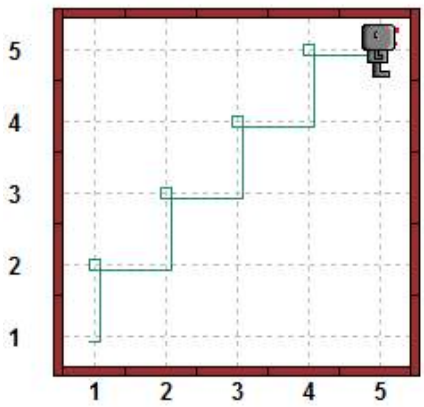
아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
1  turn_left()
2  move()
3  move()
4  turn_left()
5  turn_left()
6  turn_left()
7  move()
8  move()
9  turn_off()
```



[스스로 하기 11]

아래와 같은 결과가 나오도록 코드를 작성하시오.



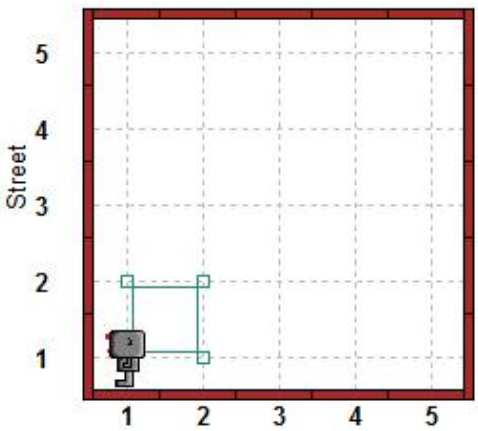
(2) def 예약어로 turn_right() 함수 만들기

오른쪽으로 회전할 때마다 turn_left()를 3번 반복하는 것은 코딩 시간도 오래 걸리고 코드를 읽기도 불편하다. def 키워드를 이용해 turn_right() 라는 새로운 명령어를 만들어 본다.

[따라 하기]

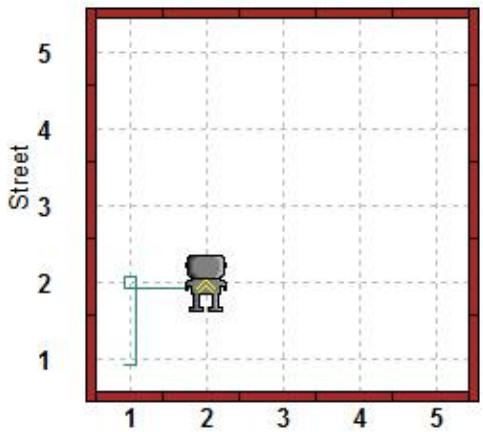
1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
def turn_right():
    turn_left()
    turn_left()
    turn_left()
turn_left()
move()
turn_right()
move()
turn_right()
move()
turn_right()
move()
turn_off()
```



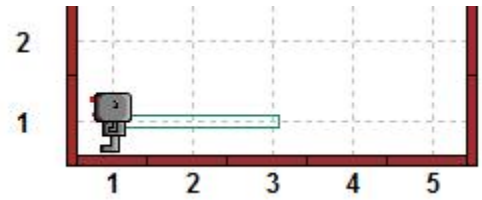
2)  버튼을 눌러 초기화 한 후,  버튼을 클릭하여 코드를 한 줄씩 실행하여 turn_right() 명령이 어떻게 실행되는지 살펴본다.

```
def turn_right():
    turn_left()
    turn_left()
    turn_left()
turn_left()
move()
turn_right()
move()
turn_right()
move()
turn_right()
move()
turn_off()
```



[스스로 하기 12]

리봇을 반대 방향으로 향하게 하는 turn_around() 함수를 정의하여 아래와 같이 이동하도록 코드를 작성하시오.

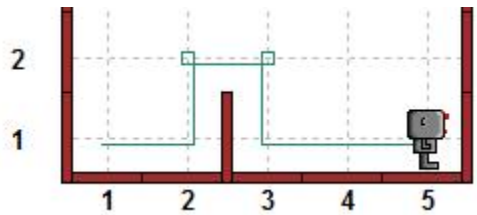


[스스로 하기 13]

turn_right()와 turn_around() 함수를 정의하여 아래와 같이 이동하도록 코드를 작성하시오.

[스스로 하기 14]

아래와 같은 결과가 나오도록 코드를 작성하시오.



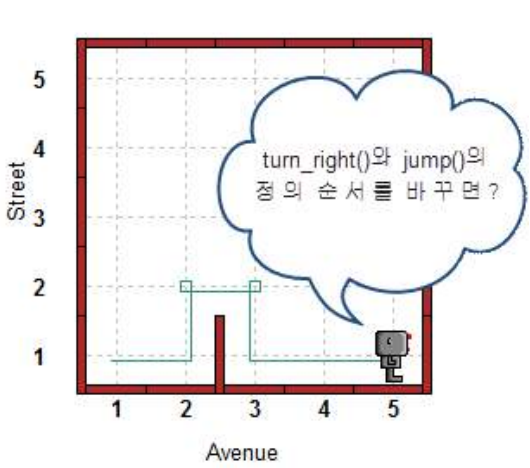
(3) 함수 활용

이미 정의한 함수를 새로운 함수에 포함하여 정의할 수 있다.

[따라 하기]

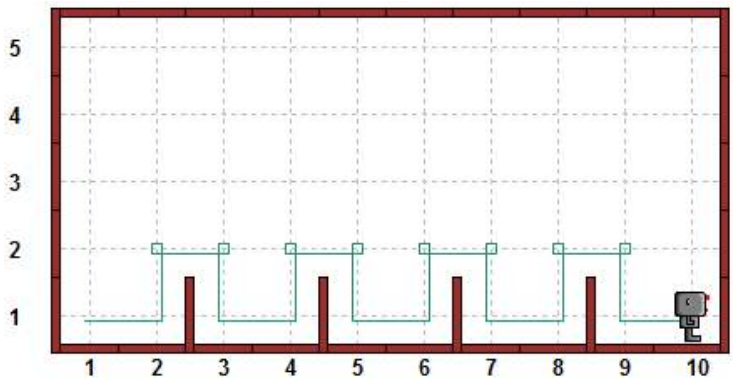
아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
def turn_right():
    turn_left()
    turn_left()
    turn_left()

def jump():
    turn_left()
    move()
    turn_right()
    move()
    turn_right()
    move()
    turn_left()

move()
jump()
move()
move()
turn_off()
```



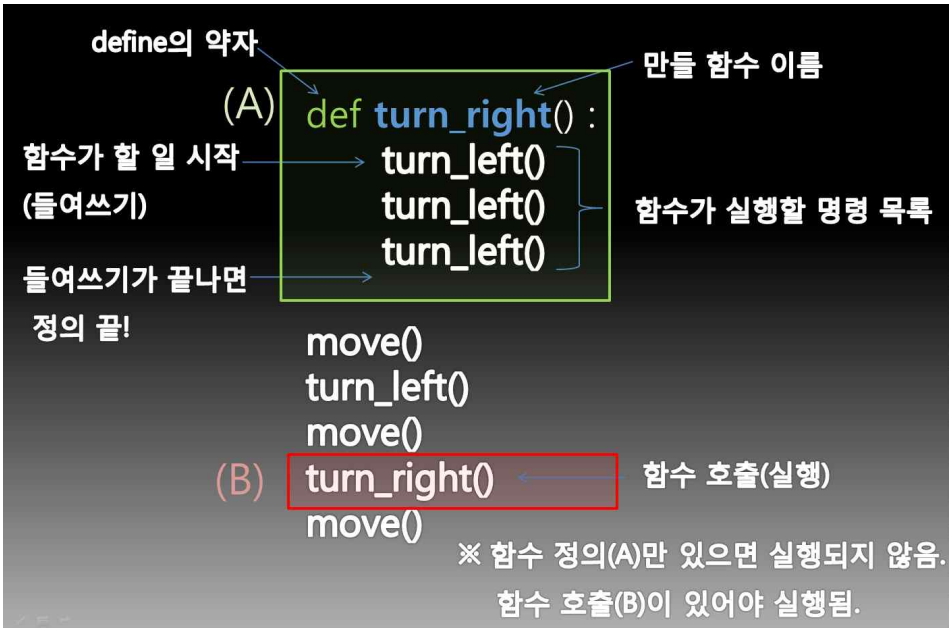
[스스로 하기 15]

아래와 같은 결과가 나오도록 코드를 작성하시오.



(4) 함수 정의에 필요한 문법 및 규칙

1) 함수 정의 문법



2) 함수 이름 만드는 규칙

① 알파벳, 숫자, _(언더바)만 사용해야 한다.

② 숫자로 시작하면 안 된다.

③ 공백이 있으면 안 된다.

④ 예약어(예:def)는 사용할 수 없다.

4. 기본 명령 II

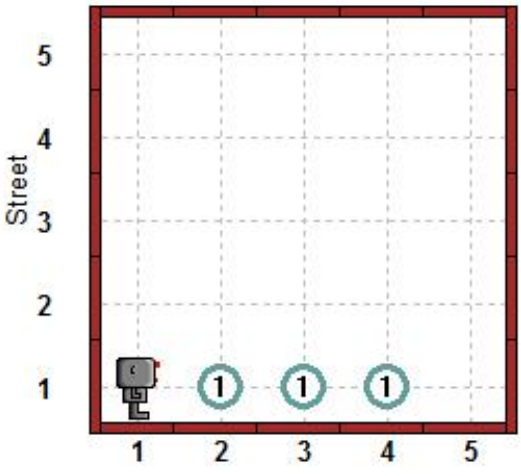
(1) pick_beeper()

pick_beeper() 함수를 이용하여 월드에 놓인 비퍼를 주울 수 있다.

[따라 하기]

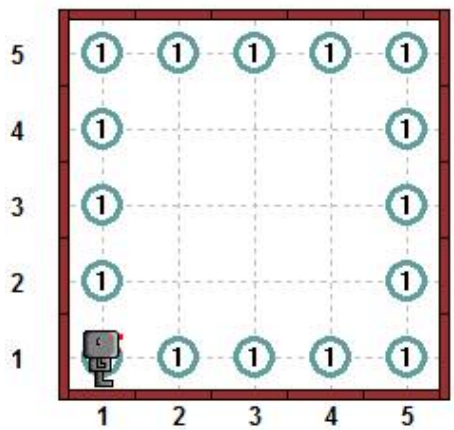
아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

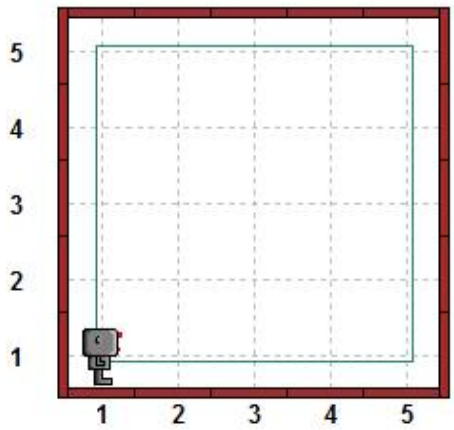
```
1  move()
2  pick_beeper()
3  move()
4  pick_beeper()
5  move()
6  pick_beeper()
7  move()
8  turn_off()
9
```



[스스로 하기 16]

아래와 같은 결과가 나오도록 코드를 작성하시오.



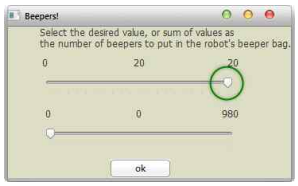


(2) put_beeper()

put_beeper() 함수를 이용하여 월드에 비퍼를 놓을 수 있다.

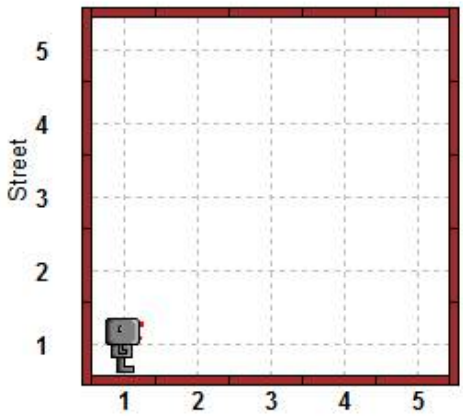
[따라 하기]

1) 상단 메뉴의  버튼을 클릭하여 리봇에게 비퍼 20개를 준다.



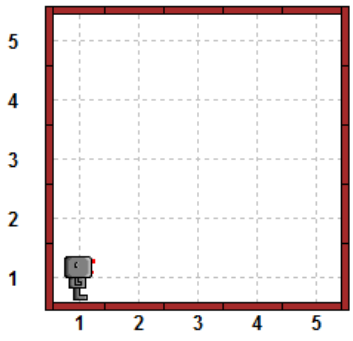
2) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

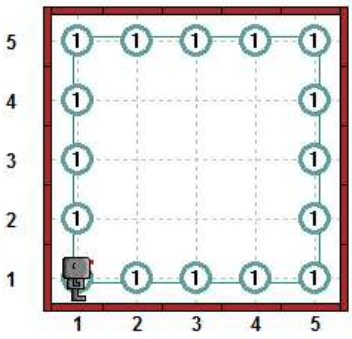
```
1  move()
2  put_beeper()
3  move()
4  put_beeper()
5  move()
6  put_beeper()
7  move()
8  turn_off()
```



[스스로 하기 17]

아래와 같은 결과가 나오도록 코드를 작성하시오.





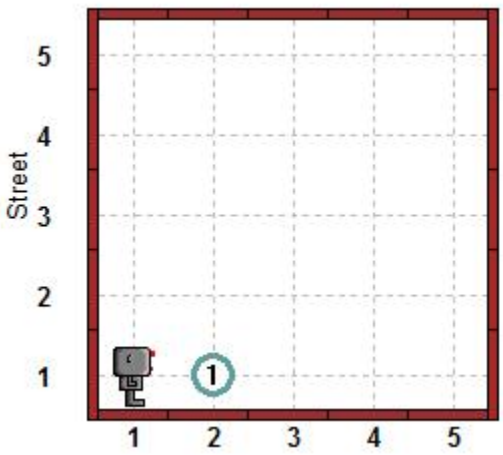
(3) 비퍼 옮기기

pick_beeper()와 put_beeper()를 사용하여 비퍼을 옮길 수 있다.

[따라 하기]

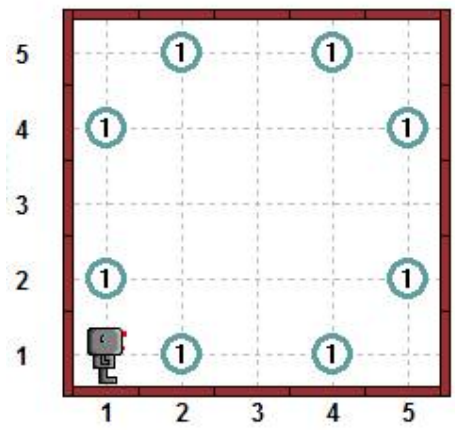
아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

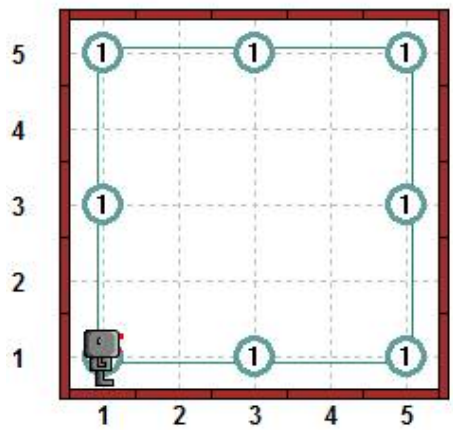
```
1  move()
2  pick_beeper()
3  move()
4  move()
5  put_beeper()
6  move()
7  turn_off()
```



[스스로 하기 18]

아래와 같은 결과가 나오도록 코드를 작성하시오.



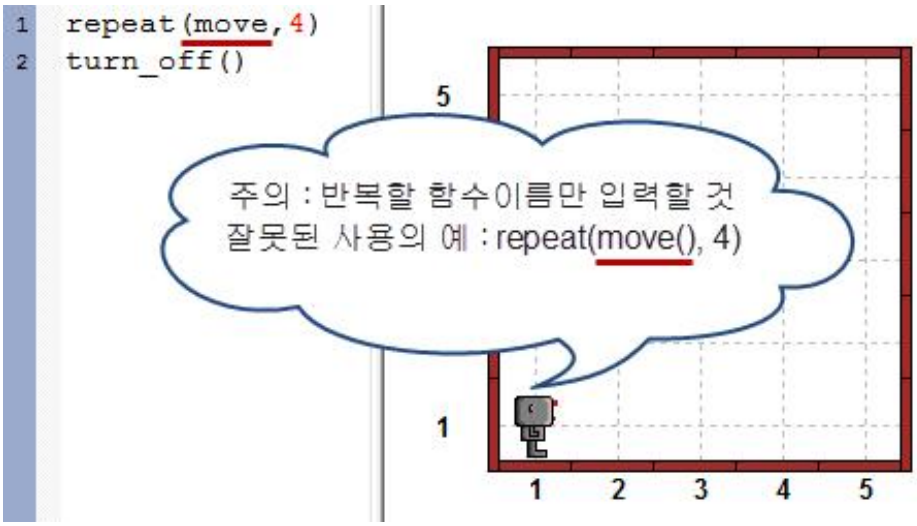


(4) repeat()

repeat() 함수를 이용하여 같은 작업을 쉽게 반복 처리 할 수 있다.

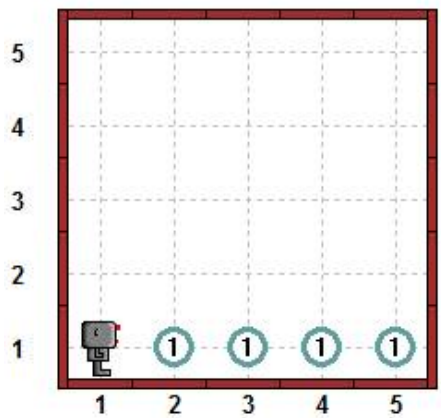
[따라 하기]

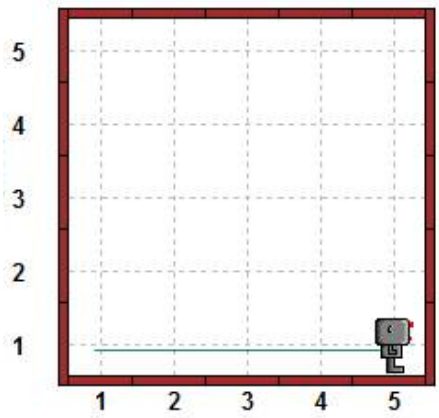
아래 코드를 작성하고  버튼을 클릭하여 실행하시오.



[스스로 하기 19]

아래와 같은 결과가 나오도록 코드를 작성하시오.





5. 선택문

(1) 에러 발생 상황

코드가 올바르게 입력되어도 명령이 수행될 수 없는 상황에서는 에러가 발생할 수 있다.

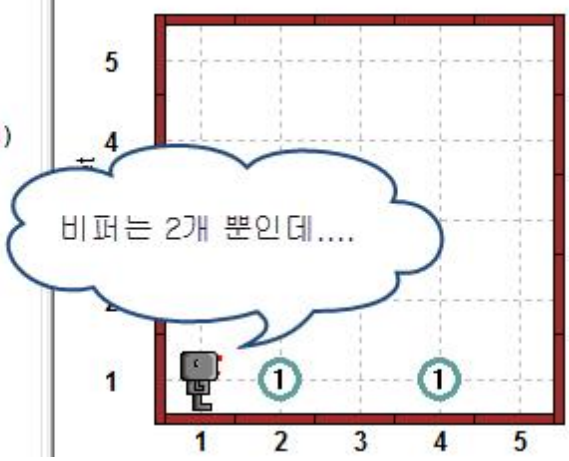
[따라 하기]

1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
1  repeat(move,10)
2  turn_off()
```

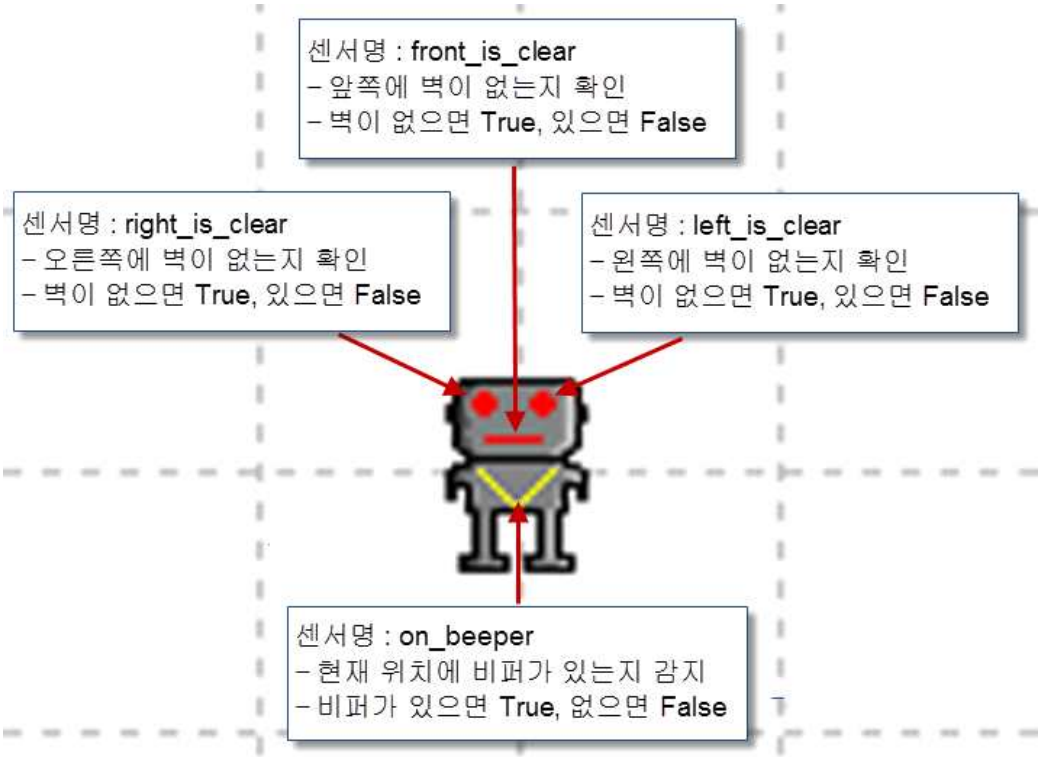
2) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
1  def move_and_pick():
2      move()
3      pick_beeper()
4
5  repeat(move_and_pick,3)
6  turn_off()
```



(2) 리봇의 센서

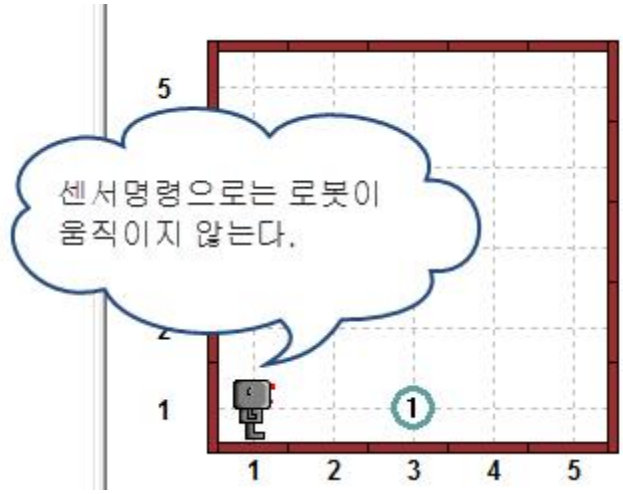
리봇은 센서를 이용하여 현재 위치의 월드 상태를 파악할 수 있다.



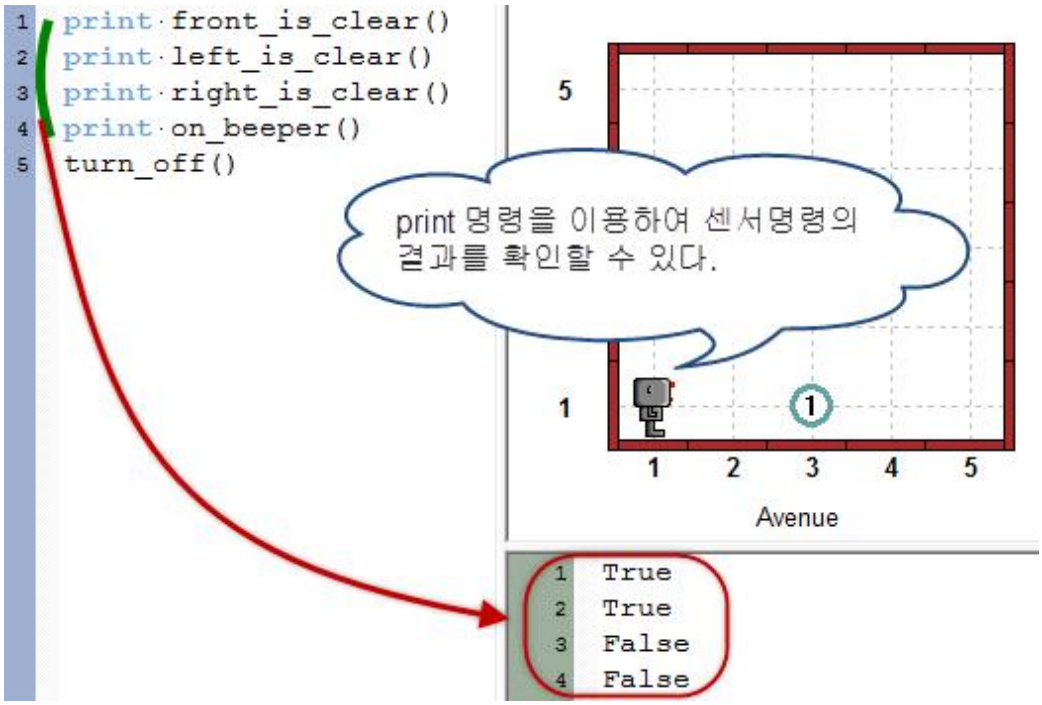
[따라 하기]

1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
1  front_is_clear()
2  left_is_clear()
3  right_is_clear()
4  on_beeper()
5  turn_off()
```

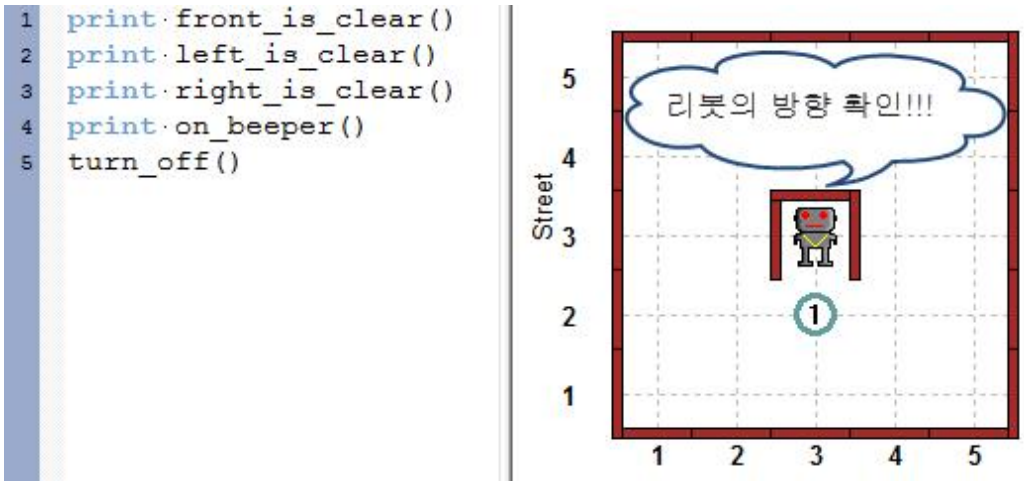


2) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.



[스스로 하기 20]

아래 코드의 결과를 예측해보고, 입력한 후 실행하여 결과를 확인하시오.

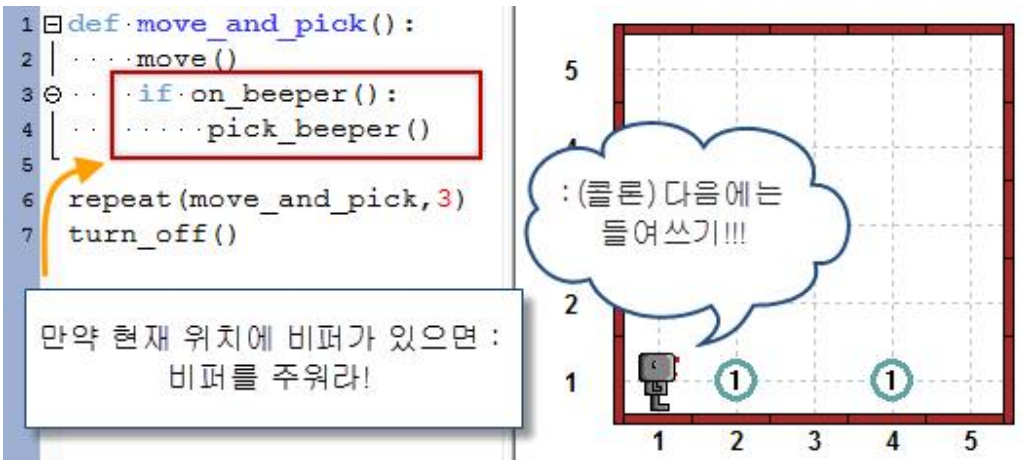


(3) if 예약어

센서에서 감지한 True나 False 값을 if 예약어와 결합시키면, 상황에 따라 리봇이 다른 동작을 선택할 수 있다.

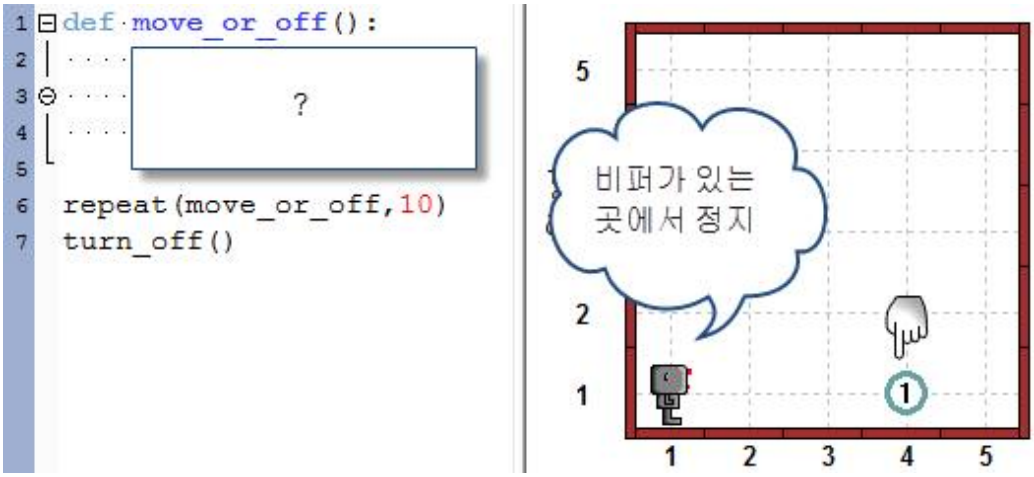
[따라 하기]

아래 코드를 작성하고  버튼을 클릭하여 실행하시오.



[스스로 하기 21]

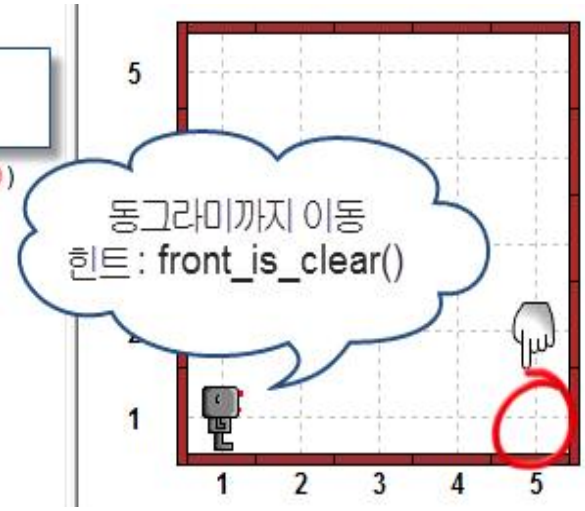
주어진 결과가 나오도록 코드를 추가하시오.



[스스로 하기 22]

주어진 결과가 나오도록 코드를 추가하시오.

```
def move_or_stop():
    ?

repeat(move_or_stop,10)
turn_off()
```



[스스로 하기 23]

주어진 결과가 나오도록 코드를 추가하시오.

```
def move_and_pick():
    ?

repeat(move_and_pick,10)
turn_off()
```

(4) not 예약어

리봇의 앞쪽이 비어 있지 않으면(벽이 있으면), 왼쪽으로 회전하려 한다.

[따라 하기]

아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

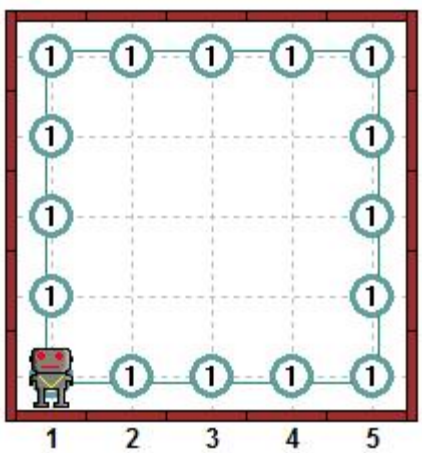
```
def move_and_turn():
    if not front_is_clear():
        turn_left()
    move()

repeat(move_and_turn,16)
turn_off()
```

not 은 True는 False로, False는 True로 바꾼다.

[스스로 하기 24]

아래와 같은 결과가 나오도록 코드를 작성하시오.

(단, 초기 비퍼의 위치는 임의로 바꿀 수 있다.)



(5) else 예약어

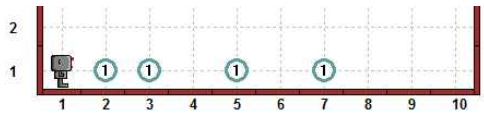
else 예약어는 if와 함께 양자택일(兩者擇一)에 사용된다.

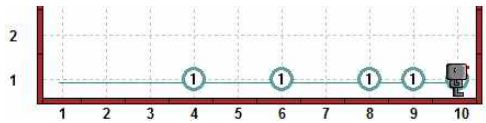
[따라 하기]

아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
def pick_or_put_and_move():
    move()
    if on_beeper():
        pick_beeper()
    else:
        put_beeper()

repeat(pick_or_put_and_move,9)
turn_off()
```

if 절의 조건이 False이 되면 else 절을 실행한다.

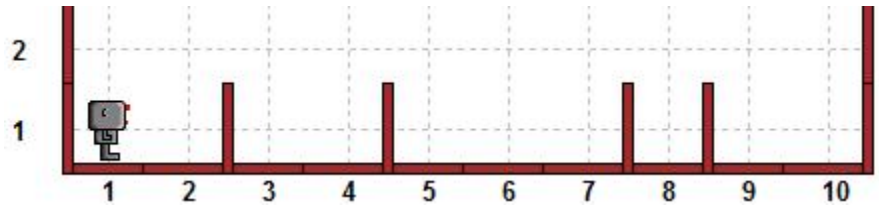


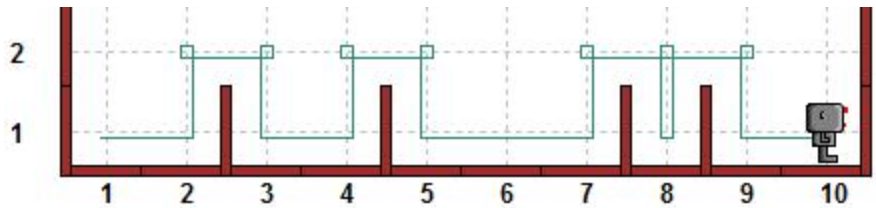


[스스로 하기 25]

아래와 같은 결과가 나오도록 코드를 작성하시오.

(단, 초기 벽의 위치는 임의로 바꿀 수 있다.)





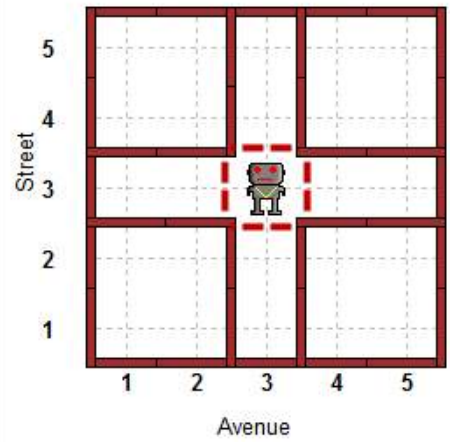
(6) elif 예약어

elif 예약어는 if와 함께 사용되며 else 와는 달리 선택 조건을 넣을 수 있다.

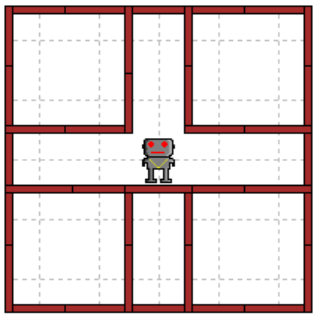
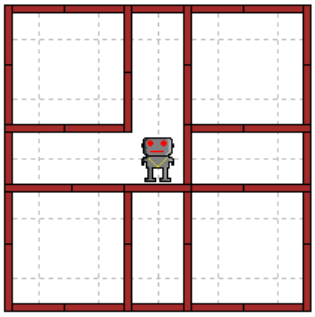
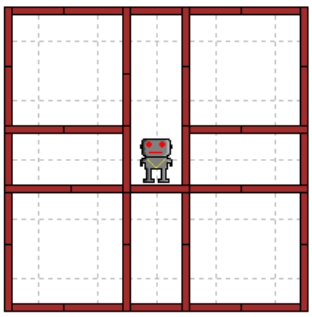
[따라 하기]

1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
def turn_right():
    repeat(turn_left,3)
def turn_around():
    repeat(turn_left,2)
def search_and_move():
    if front_is_clear():
        pass
    elif left_is_clear():
        turn_left()
    elif right_is_clear():
        turn_right()
    else:
        turn_around()
    move()

repeat(search_and_move,2)
turn_off()
```



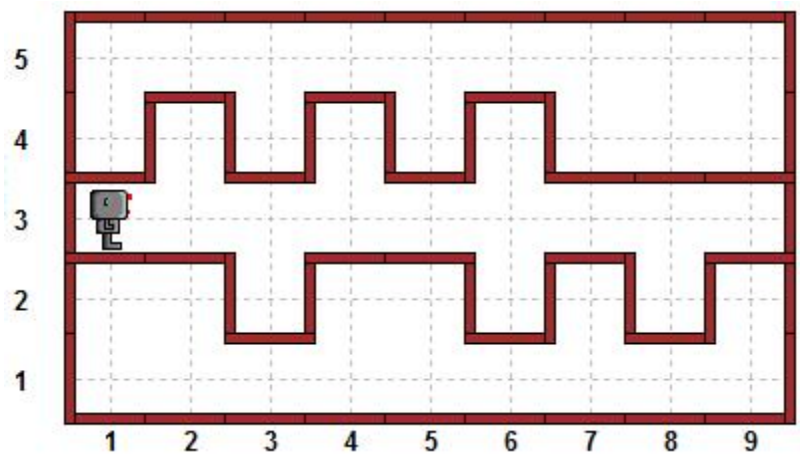
2) 1)번의 월드를 아래와 같이 바꾸고  버튼을 클릭하여 실행하시오.

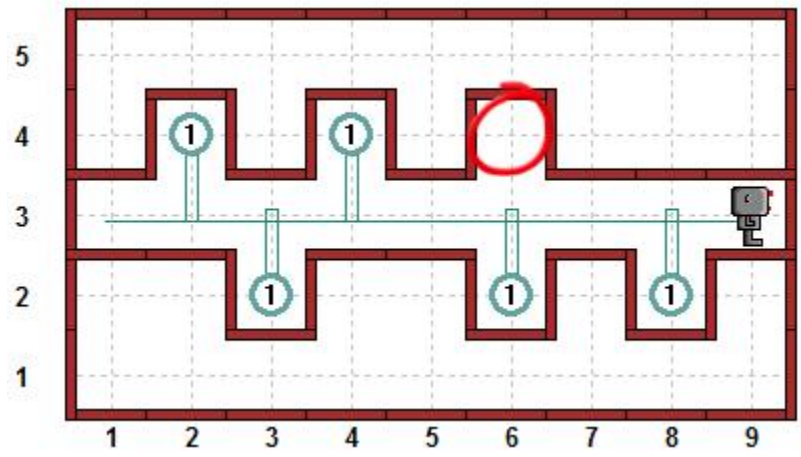




3) 1)번 코드의 6행, 8행, 10행의 조건을 서로 바꾸어 실행하시오.

[스스로 하기 26]

복도에 6개의 방이 있다. 방에 비퍼를 넣고 복도 끝으로 이동하도록 코드를 작성하시오.(단, 방의 위치는 임의로 바뀔 수 있다. 마주 보고 있는 방에는 한 쪽 방에만 비퍼를 넣는다.)





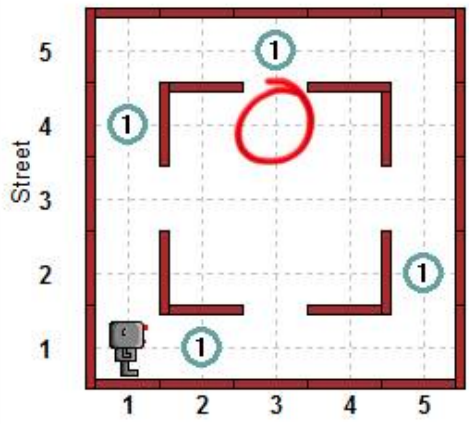
(7) if 안에 if 있다

동시에 만족해야 하는 조건이 있을 경우 if를 중첩하여 사용할 수 있다.

[따라 하기]

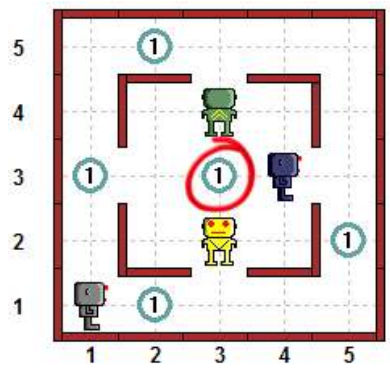
아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

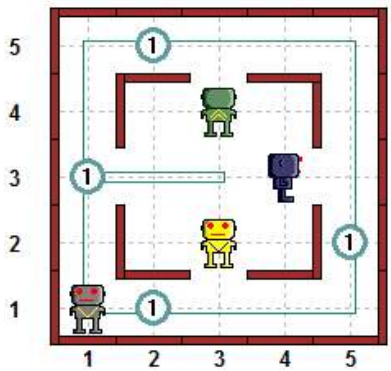
```
def move_and_check():
    if on_beeper():
        if left_is_clear():
            turn_left()
            move()
            turn_off()
    if front_is_clear():
        move()
    else:
        turn_left()

repeat(move_and_check,19)
turn_off()
```



[스스로 하기 27]

4개의 출입구가 있는 방 한가운데 보물이 놓여 있고 3개의 출입구를 고렘이 지키고 있다. 고렘이 없는 출입구를 찾아 보물을 가지고 돌아오는 코드를 작성하시오(단, 고렘의 위치는 바뀔 수 있고, 고렘이 없는 출입구 앞에는 비퍼가 놓여 있다).





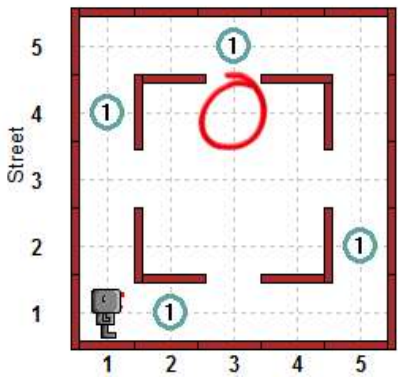
(8) and / or

and 와 or 키워드를 사용하면 여러 가지 조건을 쉽게 표현할 수 있다.

[따라 하기]

1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

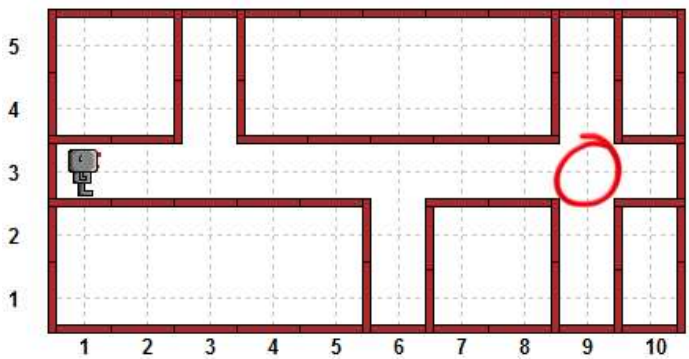
```
def move_and_check():
    if on_beeper() and left_is_clear():
        turn_left()
        move()
        turn_off()
    if front_is_clear():
        move()
    else:
        turn_left()

repeat(move_and_check,19)
turn_off()
```



2) 1)번 코드의 밑줄 친 and 를 or 로 바꾸어 실행하고 결과를 확인하시오.

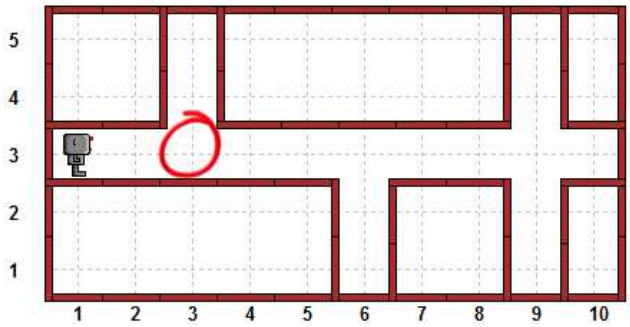
[스스로 하기 28]

리봇이 4거리에서 정지하는 코드를 작성하시오(단, 사거리의 위치는 바뀔 수 있다.)



[스스로 하기 29]

리봇이 첫 갈림길에서 정지하는 코드를 작성하시오(단, 갈림길의 위치는 바뀔 수 있다.)

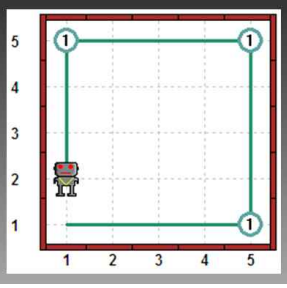


(9) 들여쓰기의 중요성

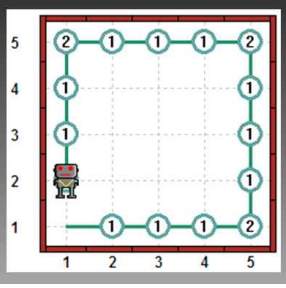
[따라 하기]

put_beeper()의 들여쓰기를 아래와 같이 변경하여 코딩하고  버튼을 클릭하여 실행하시오.

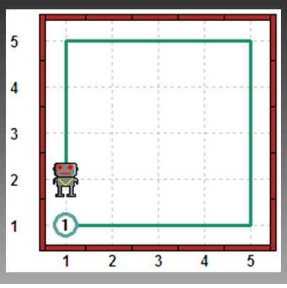
```
def move_or_turn():
    if front_is_clear():
        move()
    else:
        turn_left()
        put_beeper()

repeat(move_or_turn,18)
turn_off()
```



```
def move_or_turn():
    if front_is_clear():
        move()
    else:
        turn_left()
    put_beeper()

repeat(move_or_turn,18)
turn_off()
```



```
def move_or_turn():
    if front_is_clear():
        move()
    else:
        turn_left()
put_beeper()

repeat(move_or_turn,18)
turn_off()
```



6. 반복문

(1) repeat()의 문제점

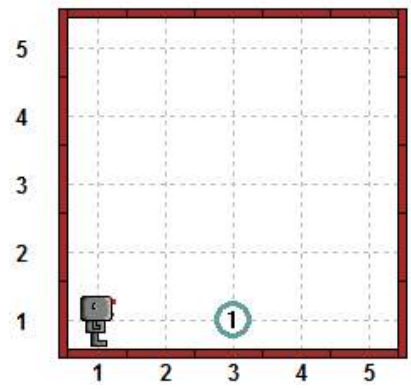
① 명령의 실행 회수를 미리 알고 있어야 한다.

② 지정된 회수만큼 무조건 실행한다.

[따라 하기]

아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

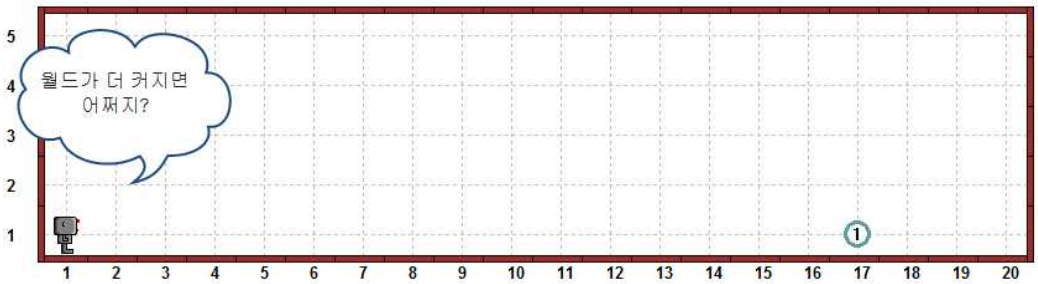
```
1 def move_or_stop():
2     if on_beeper():
3         turn_off()
4     move()
5
6 repeat(move_or_stop,4)
7 turn_off()
```



[스스로 하기 30]

아래와 같은 월드에서 비퍼까지 움직이는 코드를 작성하시오.

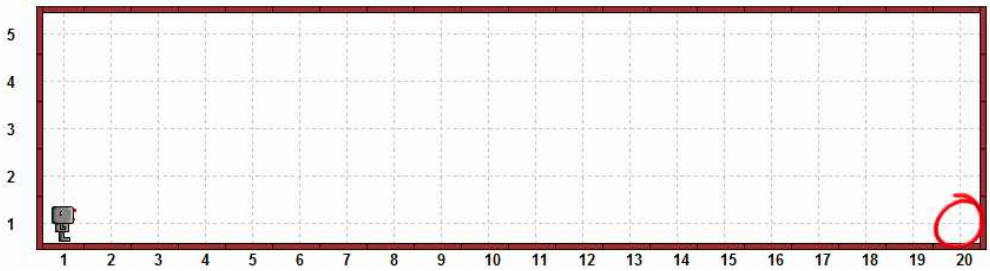
(단, 비퍼의 위치는 임의로 바뀔 수 있다.)



[따라 하기]

아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
def move_or_stop():
    if front_is_clear():
        move()

repeat(move_or_stop,10000)
turn_off()
```



[스스로 하기 31]

무한대의 크기를 가진 월드에서 비퍼를 탐색하기 위한 코드를 어떻게 작성할 수 있을지 생각해보시오.

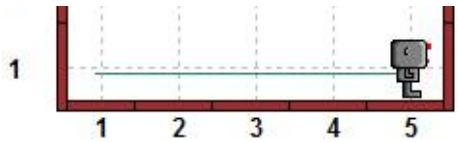
(2) while 예약어

조건에 맞으면 반복한다.

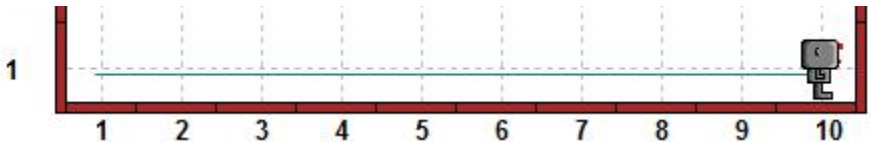
[따라 하기]

1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
while front_is_clear():
    move()
turn_off()
```

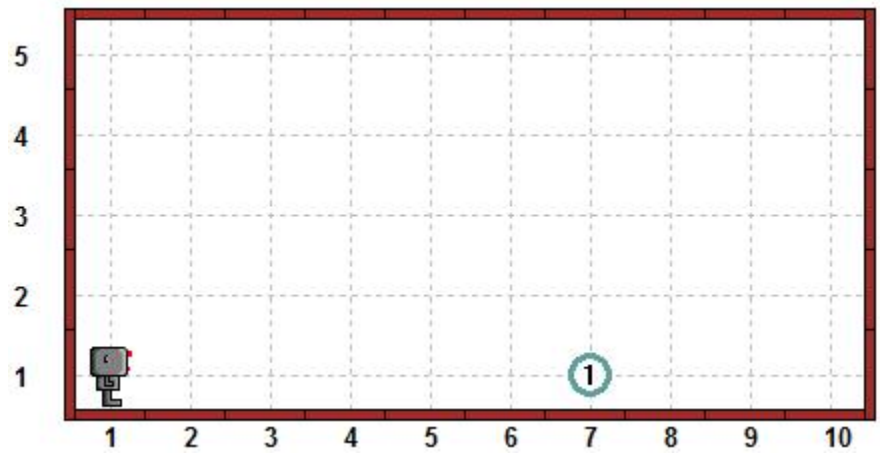


2) 월드의 Avenue 크기를 아래와 같이 변경하고 1)번의 코드로 실행하시오.



[스스로 하기 32]

아래와 같은 월드에서 비퍼가 있는 위치에 리봇이 정지하는 코드를 작성하시오. (단, 비퍼의 위치는 임의로 바꿀 수 있다.)



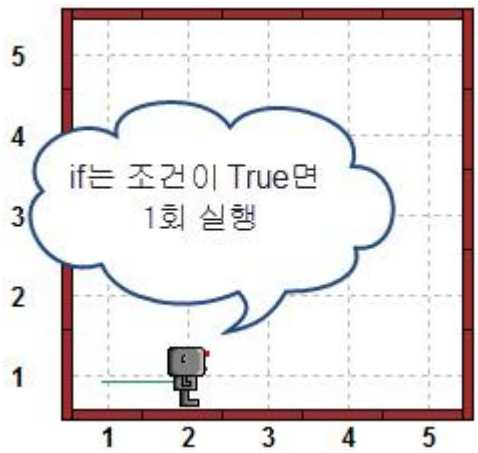
(3) while VS if

while과 if의 차이는 실행 회수이다.

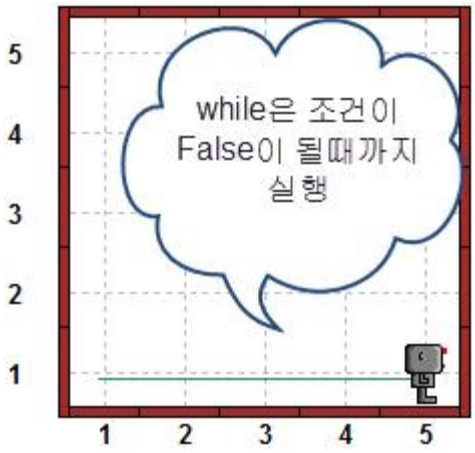
[따라 하기]

1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
if front_is_clear():
    move()
turn_off()
```




```
while front_is_clear():
    move()
turn_off()
```



[스스로 하기 33]

주어진 결과가 나오도록 코드를 추가하시오.

```
while                :
    
    
    
turn_off()
```

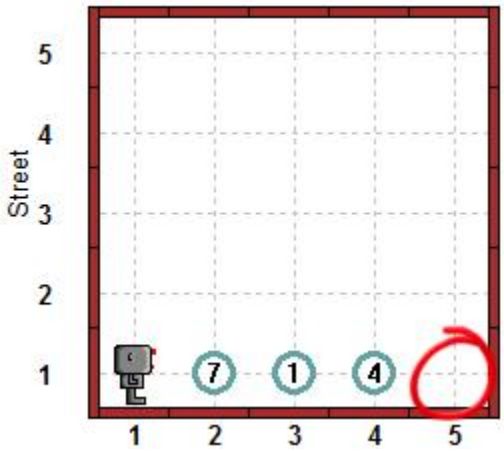
(4) while 안에 while 있다

while을 중첩하여 사용할 수 있다.

[따라 하기]

아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
while front_is_clear():
    move()
    while on_beeper():
        pick_beeper()
turn_off()
```

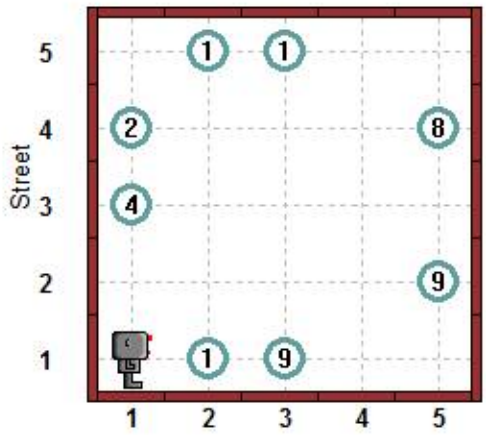


[스스로 하기 34]

월드의 모든 비퍼를 줍고 출발지점으로 돌아오도록 코드를 추가하시오

(단, 비퍼의 위치와 개수는 바뀔 수 있다).

```
def move_and_pick():
    while 
        
        :
        
        
    repeat(move_and_pick,4)
    turn_off()
```



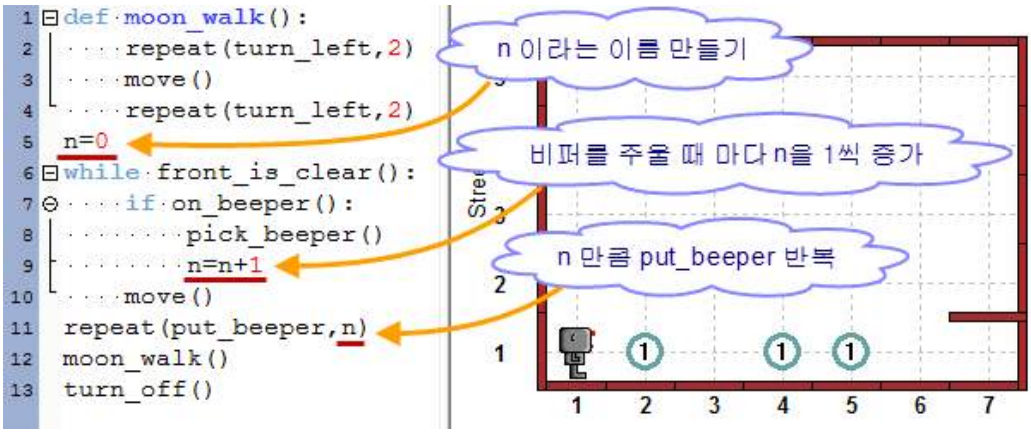
7. 이름

(1) 이름(name)의 필요성

이름에 값이나 명령을 기억해 두었다가 다시 사용할 수 있다. 이름 작성 규칙은 함수 작성 규칙과 동일하다.

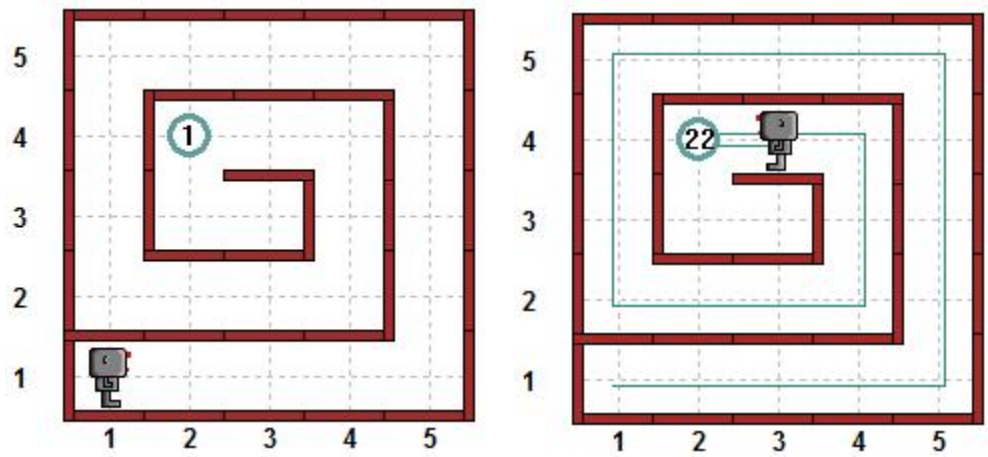
[따라 하기]

아래 코드를 작성하고  버튼을 클릭하여 실행하시오.



[스스로 하기 35]

비퍼가 놓여있는 지점까지 이동하기 위한 리봇의 최소 이동 회수를 나타내는 코드를 작성하시오.(단, 비퍼의 위치는 임의로 바뀔 수 있다.)



이름을 비교 연산자와 함께 사용하여 if문이나 while문에 사용할 수 있다.

※비교 연산자 : 크다(>),작다(<),크거나 같다(>=),작거나 같다(<=),같다(==), 다르다(!=)를 말하며 실제 값이나 이름이 나타내는 값을 비교하여 참(True) 또는 거짓(False)으로 계산된다.

[따라 하기]

1) 아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
n=0
while front_is_clear():
    move()
    while on_beeper():
        if n<9:
            pick_beeper()
            n=n+1
        else:
            break
    turn_off()
```

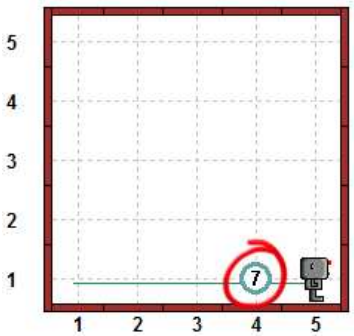
※break 예약어가 실행되면 가장 가까운 반복문을 빠져나간다.

2)아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
x=0
while front_is_clear():
    n=0
    x=x+1
    move()
    while on_beeper():
        if n<x:
            pick_beeper()
            n=n+1
        else:
            break
    turn_off()
```

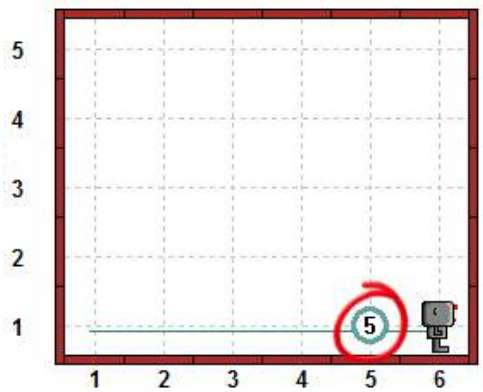
3)아래 코드를 작성하고  버튼을 클릭하여 실행하시오.

```
def how_many_beepers():
    n=0
    while on_beeper():
        pick_beeper()
        n=n+1
    return n

def put_beepers(n):
    repeat(put_beeper,n)

move()
a=how_many_beepers()
move()
b=how_many_beepers()
move()
if a>b:
    put_beepers(a)
else:
    put_beepers(b)
move()
turn_off()
```



[스스로 하기 36]

아래와 같이 가장 많은 비퍼의 개수를 찾아내는 코드를 작성하시오(단, 비퍼의 개수는 임의로 바뀔 수 있다).

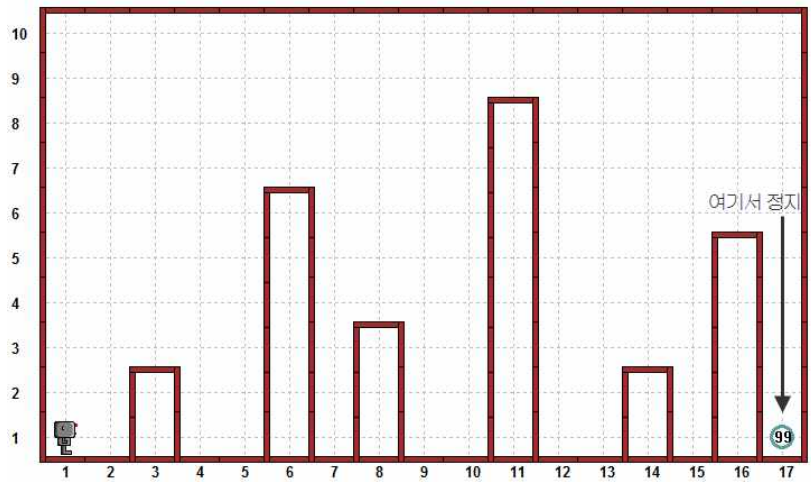


8. 종합 문제

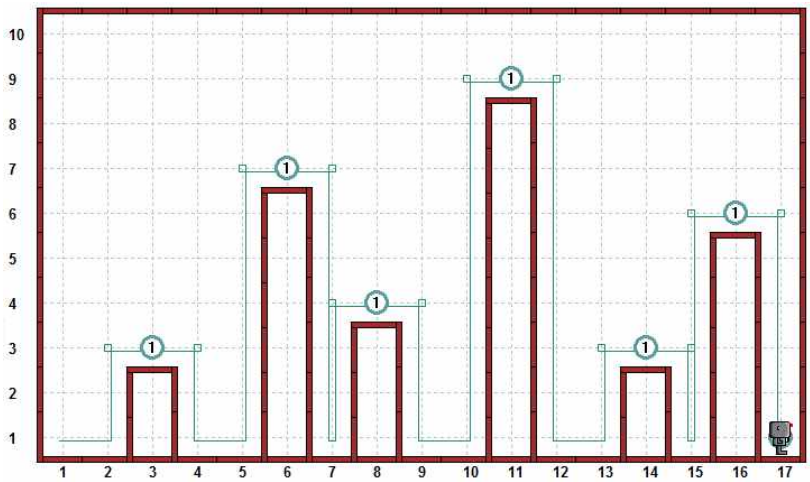
[문제 1]

빌딩 옥상에 안테나를 설치하는 리봇 제작을 의뢰 받았다. 아래와 같이 각 빌딩 옥상에 안테나(비퍼)를 설치하도록 코드를 작성하시오(단, 빌딩의 개수와 높이는 임의로 바뀔 수 있다).

[초기 상태 예시]



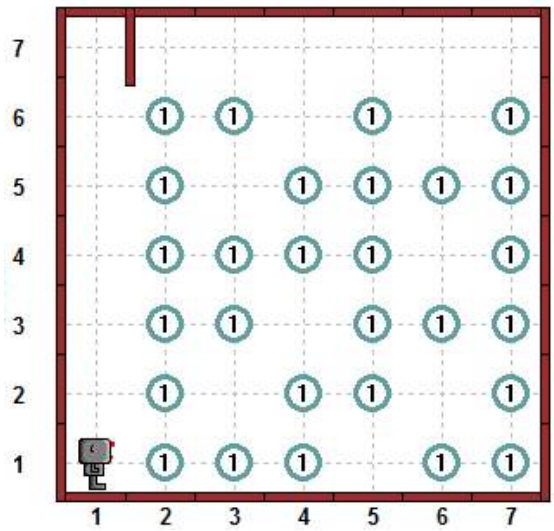
[실행 결과 예시]



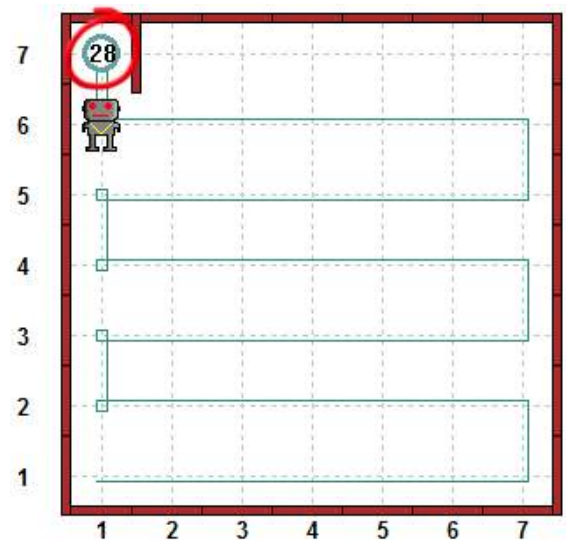
[문제 2]

농작물을 수확하는 리봇 제작을 의뢰 받았다. 농작물(비퍼)을 수확하여 창고에 모두 저장하는 코드를 작성하시오(단, 농작물의 개수는 임의로 바뀔 수 있다).

[초기 상태 예시]



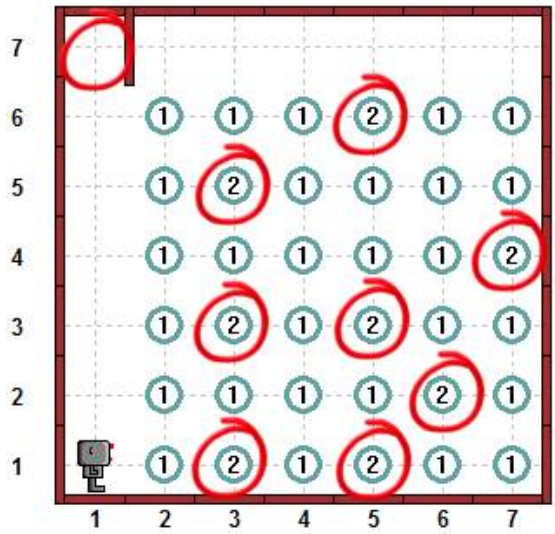
[실행 결과 예시]



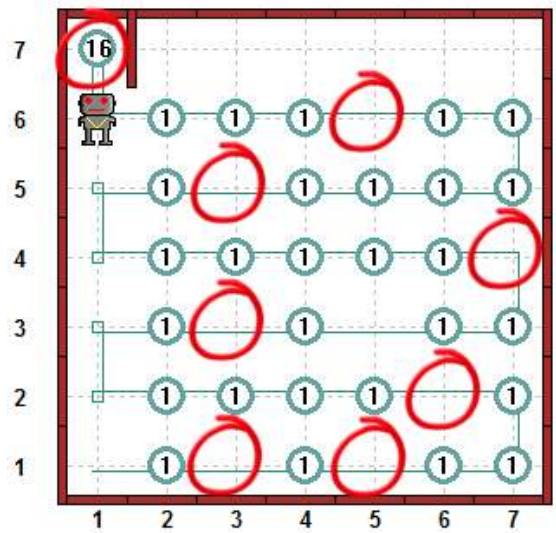
[문제 3]

딸기만(②) 수확하여 창고에 모두 저장하는 코드를 작성하시오(단, 딸기의 위치는 임의로 바뀔 수 있다).

[초기 상태 예시]



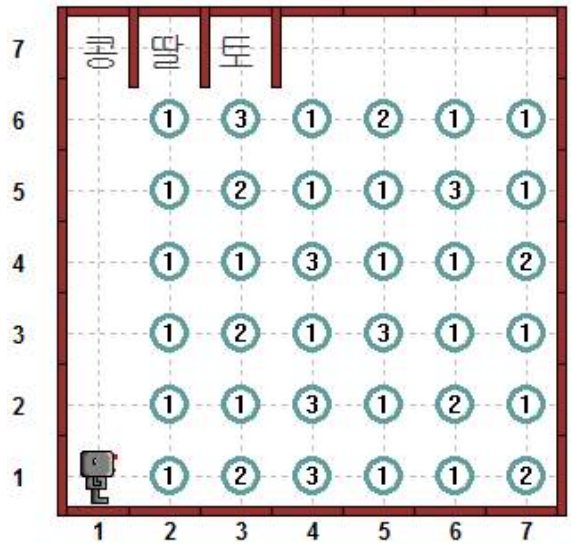
[실행 결과 예시]



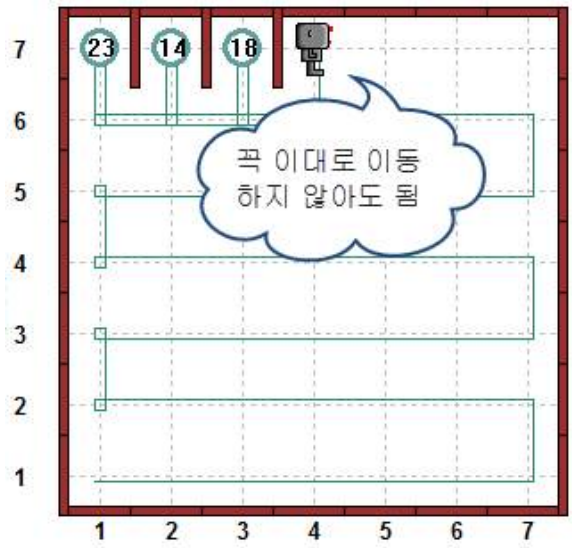
[문제 4]

콩(①), 딸기(②), 토마토(③)를 수확하여 각각 창고에 저장하는 코드를 작성하시오(단, 농작물의 위치와 개수는 임의로 바꿀 수 있다).

[초기 상태 예시]



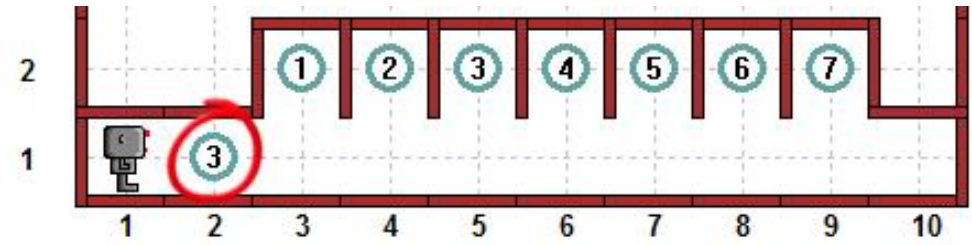
[실행 결과 예시]



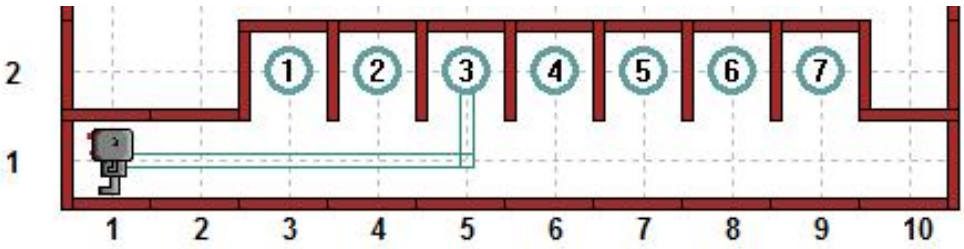
[문제 5]

식당에서 손님의 주문을 받는 리봇 제작을 의뢰 받았다. 아래와 같이 호출한 방의 번호를 확인하고, 해당 방으로 이동하여 주문을 받아 돌아오는 코드를 작성하시오(단, 호출한 방의 번호는 임의로 바뀔 수 있다).

[초기 상태 예시]



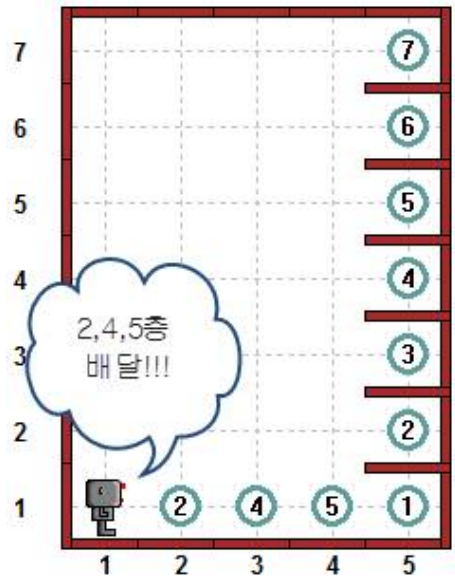
[실행 결과 예시]



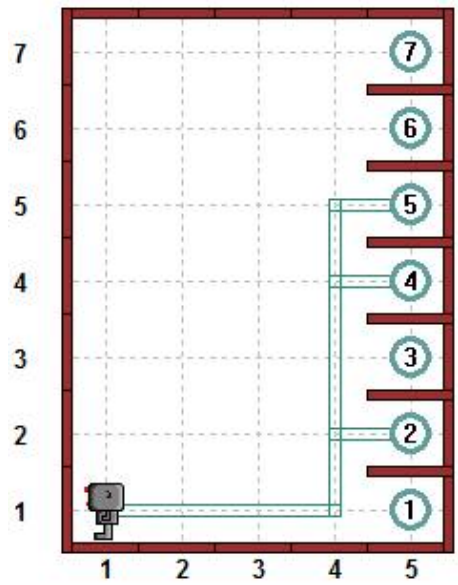
[문제 6]

피자를 배달하는 리봇 제작을 의뢰 받았다. 아래와 같이 주문한 층 번호를 확인하고 해당 층으로 이동하여 피자를 배달하고 돌아오는 코드를 작성하시오 (단, 배달하는 곳은 항상 3곳이며, 임의로 층수가 바뀔 수 있다).

[초기 상태 예시]



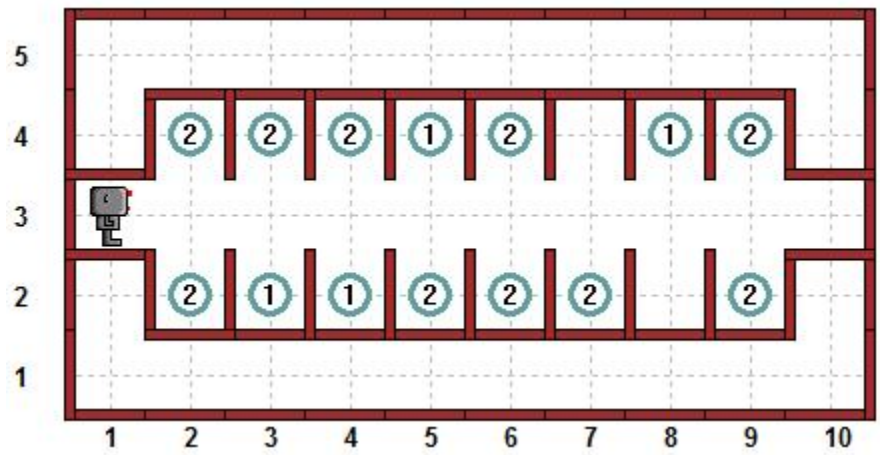
[실행 결과 예시]



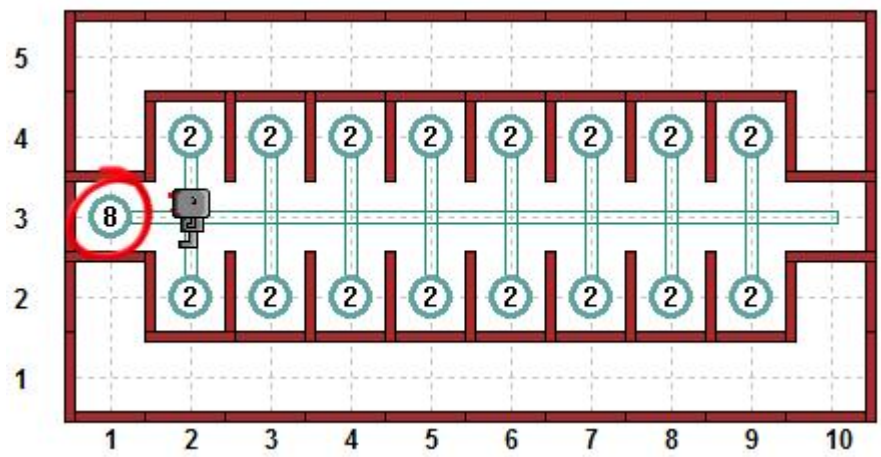
[문제 7]

호텔 룸서비스 용 리봇 제작을 의뢰받았다. 호텔 객실에는 수건(비퍼)이 2개씩 비치되어야 하는데, 가끔 일부 몰지각한 사람들이 수건을 가져가는 경우가 있다. 따라서 리봇은 아래와 같이 객실을 순회하며 분실된 수건을 채워 넣고, 분실된 수건이 몇 장인지 총지배인에게 알려주어야 한다. 이와 같은 동작을 하는 코드를 작성하시오(단, 수건의 개수는 임의로 바뀔 수 있다).

[초기 상태 예시]



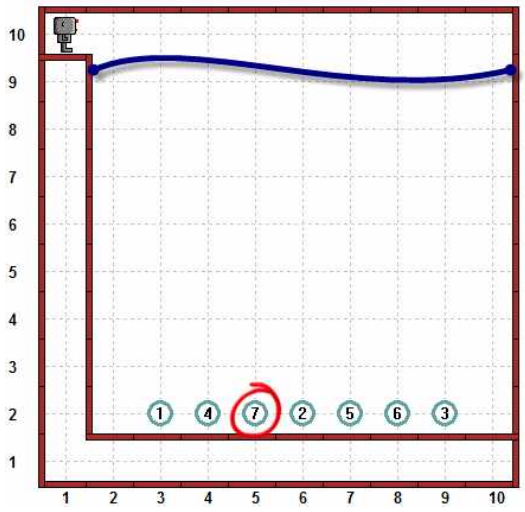
[실행 결과 예시]



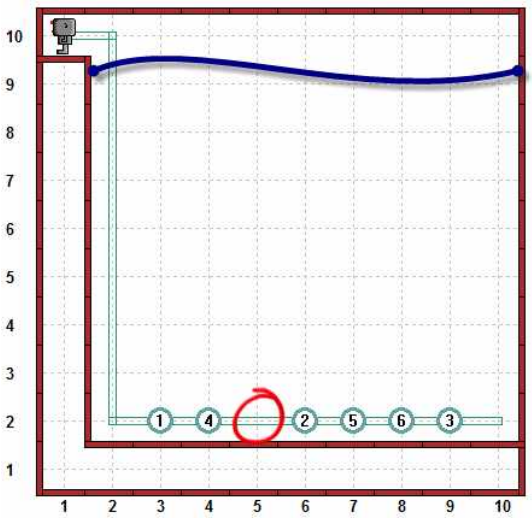
[문제 8]

해저 탐험 용 리봇 제작을 의뢰 받았다. 리봇은 바다 밑을 탐색하여 보물(비퍼)을 건져 와야 한다. 그러나 안타깝게도 보물은 하나만 건져 올 수 있다고 한다. 가장 가치가 높은 보물(비퍼의 숫자가 놓을 수로 가치가 높은 보물이다)을 건져 오는 코드를 작성하시오.

[초기 상태 예시]



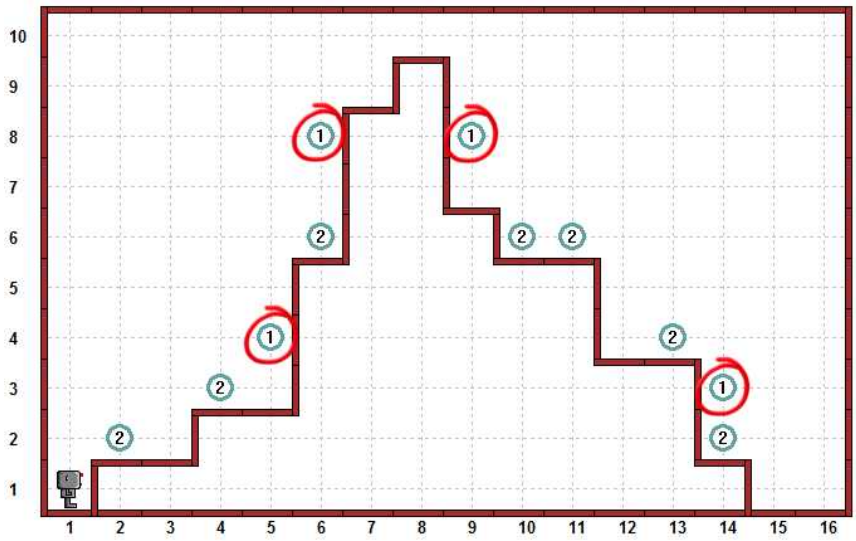
[실행 결과 예시]



[문제 9]

산악 구조 용 리봇 제작을 의뢰받았다. 리봇이 산을 등반하면서 조난당한 등산객(①)을 구조하여 내려올 수 있도록 코드를 작성하시오(단, 조난당한 등산객의 위치는 임의로 바뀔 수 있다).

[초기 상태 예시]



[실행 결과 예시]