

FUNTERS

주니어 임베디드SW 챌린저

-2차 기술 지원 교육-

2017. 08. 19

- 1차 세미나 복습하기

- 함수

- 함수의 개념과 이해
- 함수 활용 예제 실습

- 주행 제어하기

- 단위 동작 구성
- 셀 단위 주행
- 색상 감지

- 방향 전환

- 배열(array)

- 배열의 개념
- 1차원 배열의 선언과 활용
- 2차원 배열의 선언과 활용

• 리눅스 명령어

- EV3dev는 리눅스 운영체제이기 때문에 프로그램을 편집 및 수정하기 위해서는 아래 표의 명령어를 알고 있어야 한다. 리눅스에는 다양한 명령어들이 있으나 이 과정에서는 프로그램을 작성하고, 편집할 때 필요한 필수 명령어만 다룬다.

명령어	동작
vi file_name.py	텍스트 파일 편집
:wq	파일을 저장하고 vi를 종료
:q	파일을 저장하지않고 vi를 종료
chmod	권한 변경 (r : 읽기 허용, w : 쓰기 허용, x : 실행 허용)
python3 file_name.py	파일 실행
ls -al	모든 파일 리스트 표시
rm file_name	파일 삭제

• 연산자

- 연산자를 활용하면 변수에 저장된 값을 다양하게 조작하여 사용할 수 있다.
- 프로그램 상에 괄호 없이 두 개 이상의 연산자를 한 문장에서 혼합하여 사용하게 되면, 연산자의 우선 순위에 따라 실행된다.

<산술 연산자>

연산자	의미	예시	연산 결과
+	덧셈	4 + 2	6
-	뺄셈	4 - 2	2
/	나눗셈	4 / 2	2
*	곱셈	4 * 2	8
%	나머지	4 % 2	0

<관계 연산자>

연산자	의미	예시	연산 결과
<	a가 b보다 작다	4 < 2	false
>	a가 B보다 크다	4 > 2	true
==	a와 b가 같다	4 == 2	false
!=	a와 b가 같지 않다	4 != 2	true
<=	a가 b보다 작거나 같다	4 <= 2	False
>=	a가 b보다 크거나 같다	4 >= 2	true

<대입 연산자>

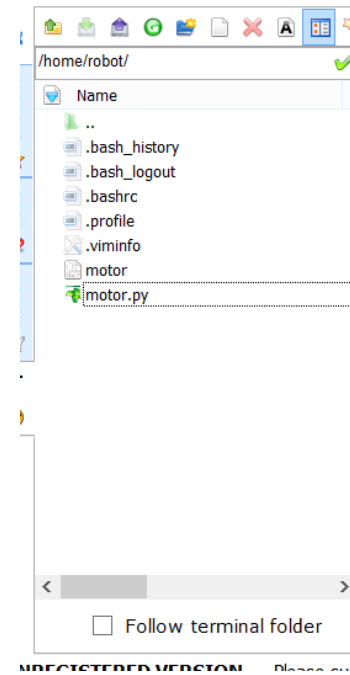
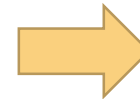
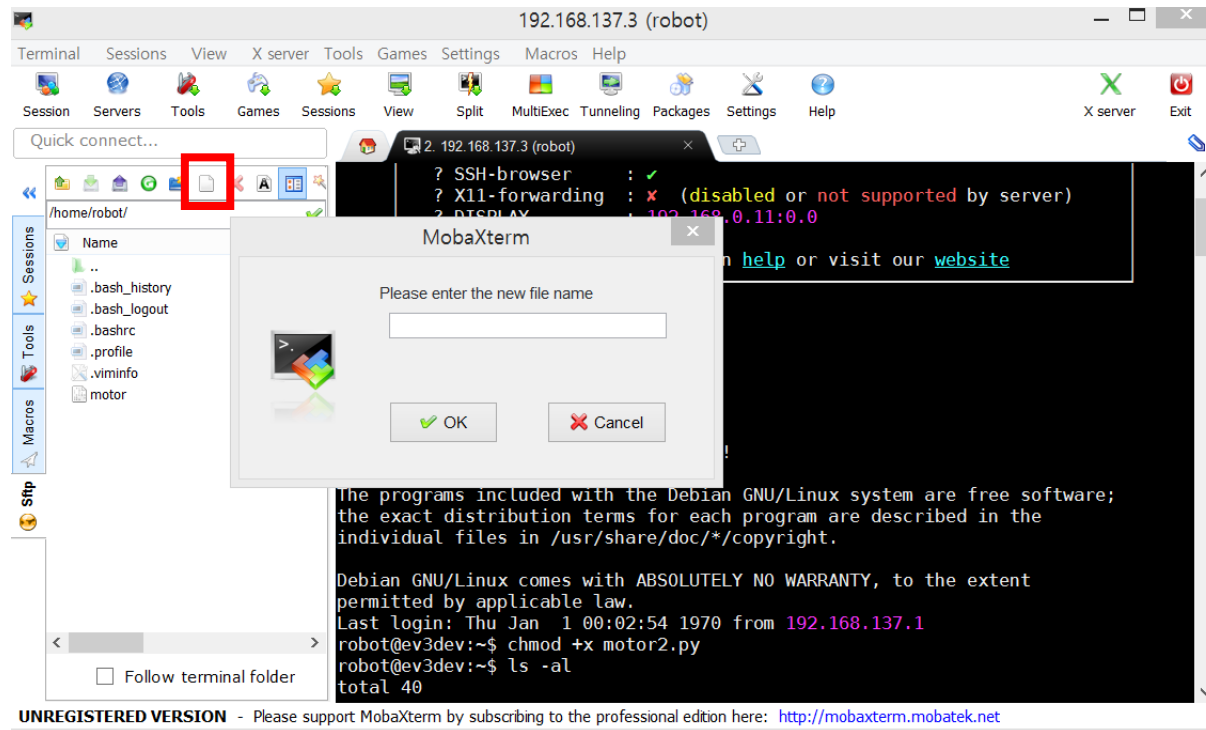
연산자	의미	예시	연산 결과
+=	덧셈 누적	a+=1	a=a+1
-=	뺄셈 누적	a-=1	a=a-1
/=	나눗셈 누적	a/=1	a=a/1
=	곱셈 누적	a=1	a=a*1

<논리 연산자>

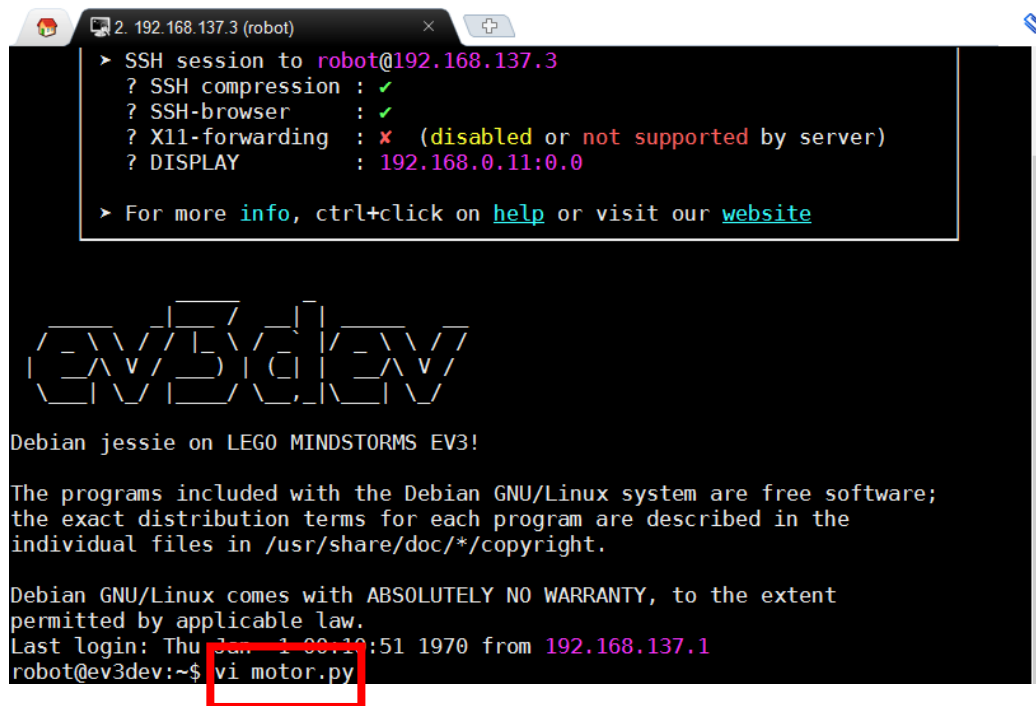
연산자	의미	예시	연산 결과
and	양쪽의 값이 모두 true인 경우 true	(2<4) and (5>8)	false
or	어느 한 쪽만 true인 경우 true	(2<4) or (5<8)	true
not	True면 false, false면 true	not(4<20)	false

• 프로그램 작성 방법

- ① MobaXterm의 왼쪽 SideBar에 Sftp항목에서 create new file을 클릭한다.
- ② File_name.py라는 형식으로 새로운 파일을 만들어준다. 예) motor.py



- ③ 이제 앞서 배운 명령어 중, vi(텍스트 파일 편집)를 이용하여 motor.py에 접근한다.



A terminal window titled "2. 192.168.137.3 (robot)". It shows an SSH session with a robot. The output includes SSH status (compression, browser, X11-forwarding, DISPLAY), a warning about X11-forwarding being disabled, and a prompt for more info. Below this is a ASCII art logo for "ev3dev" and a message "Debian jessie on LEGO MINDSTORMS EV3!". It then shows the Debian GNU/Linux license text. At the bottom, the prompt "robot@ev3dev:~\$" is followed by the command "vi motor.py", which is highlighted with a red box.

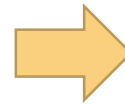
```
> SSH session to robot@192.168.137.3
? SSH compression : ✓
? SSH-browser      : ✓
? X11-forwarding   : ✗ (disabled or not supported by server)
? DISPLAY          : 192.168.0.11:0.0
> For more info, ctrl+click on help or visit our website

ev3dev

Debian jessie on LEGO MINDSTORMS EV3!

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

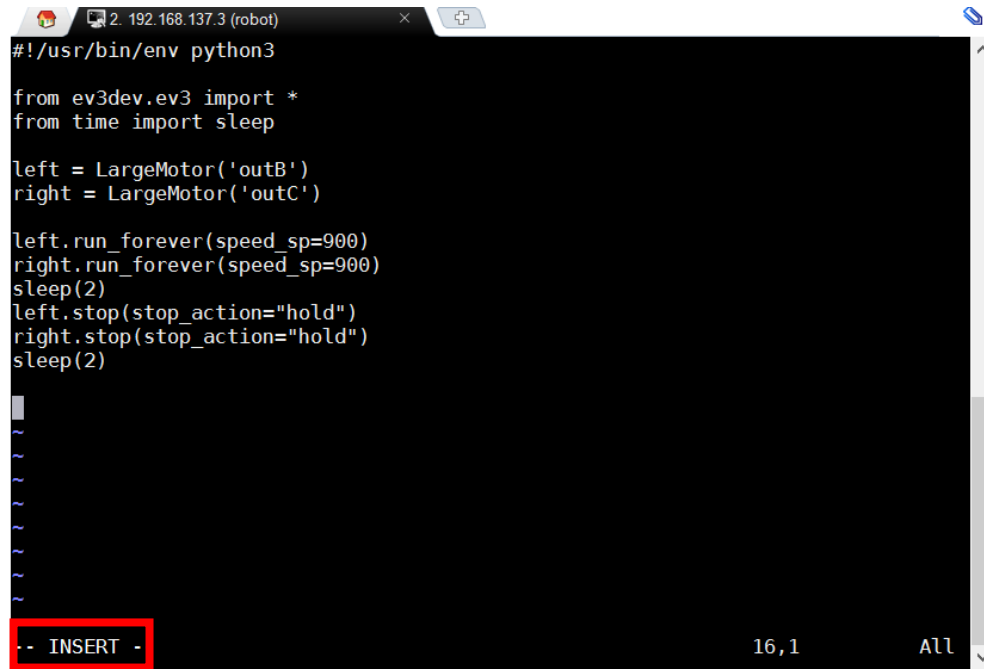
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan 1 00:10:51 1970 from 192.168.137.1
robot@ev3dev:~$ vi motor.py
```



A terminal window titled "2. 192.168.137.3 (robot)". It shows the vi editor interface. The top line shows the filename "motor.py" and the current mode "0L, 0C". The bottom line shows the current line and column "0,0-1" and the word "ALL". At the bottom of the window, there is a link to the professional edition of MobaXterm.

```
"motor.py" 0L, 0C
0,0-1 ALL
port MobaXterm by subscribing to the professional edition here: http://mobaxterm.mobatek.net
```

- ④ 키보드의 insert키(Ins)를 눌러 커서를 나타낸 후, 코드를 작성한다.
- ⑤ 작성이 완료되면, 키보드의 Esc버튼을 눌러 INSERT 작업을 마치고 :wq 명령어를 이용하여 저장한 후, 메인 화면으로 이동한다.

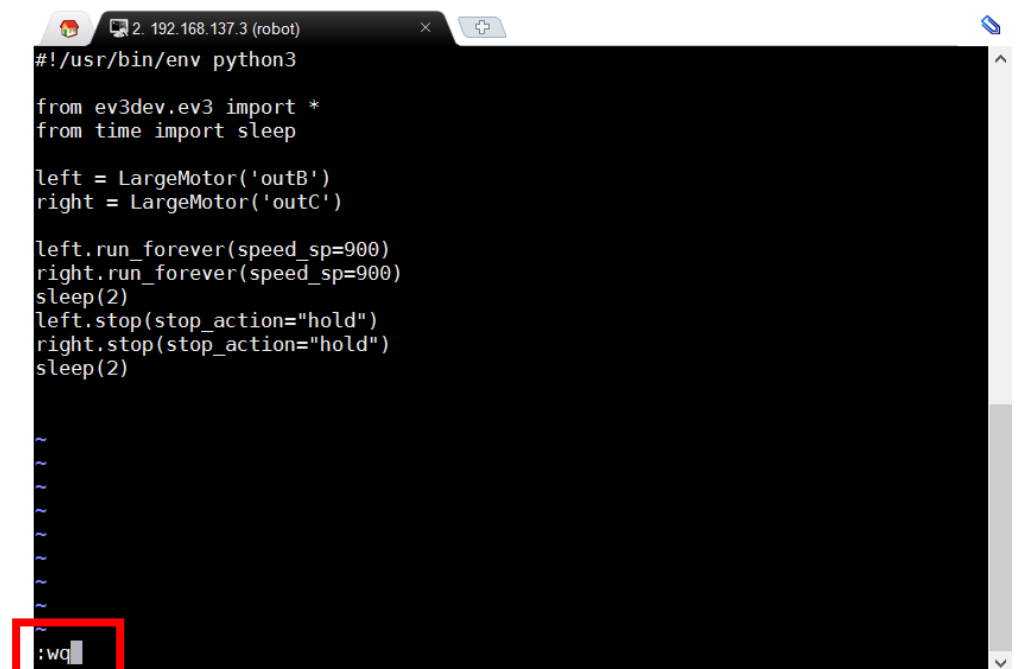


A terminal window titled '2. 192.168.137.3 (robot)' with a black background and white text. It contains Python code for controlling a robot's motors. At the bottom, the prompt '-- INSERT -' is highlighted with a red box, indicating that the terminal is in insert mode. The status bar at the bottom right shows '16,1' and 'ALL'.

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=900)
right.run_forever(speed_sp=900)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(2)
```



The same terminal window as on the left, but now the prompt ':wq' is highlighted with a red box at the bottom, indicating the user has entered the command to save and quit. The status bar at the bottom right is partially visible.

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from time import sleep

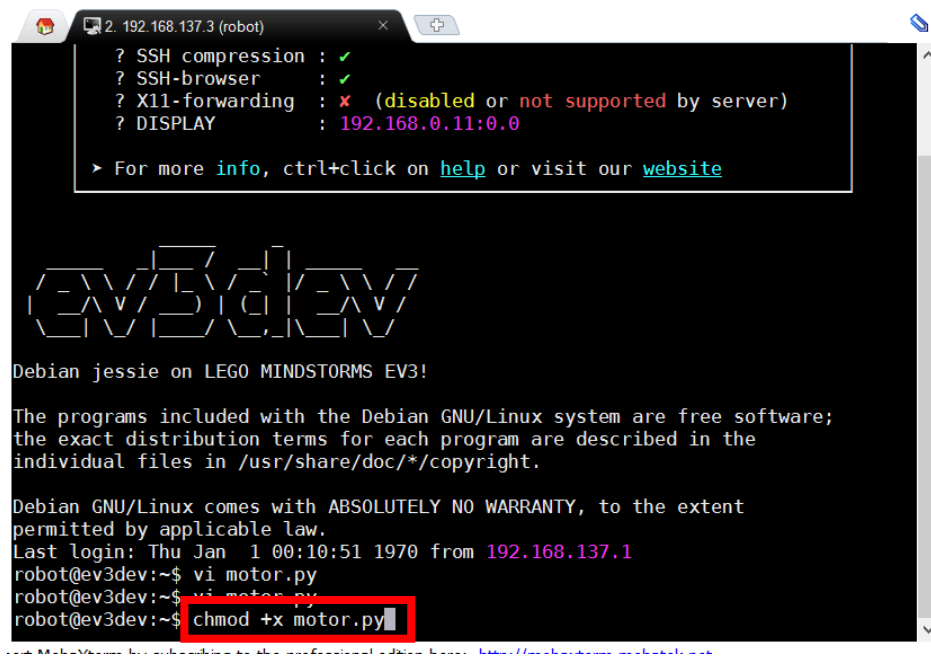
left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_forever(speed_sp=900)
right.run_forever(speed_sp=900)
sleep(2)
left.stop(stop_action="hold")
right.stop(stop_action="hold")
sleep(2)
```


- ⑥ 메인화면에서 chmod 명령어를 이용하여 motor.py 파일을 실행 파일로 설정해준다.

chmod +x motor.py 입력 후, Enter를 누른다.

- ⑦ 이제 EV3 LCD화면에서 File Browser에 들어가서 만들어진 motor.py*파일을 실행시킨다.

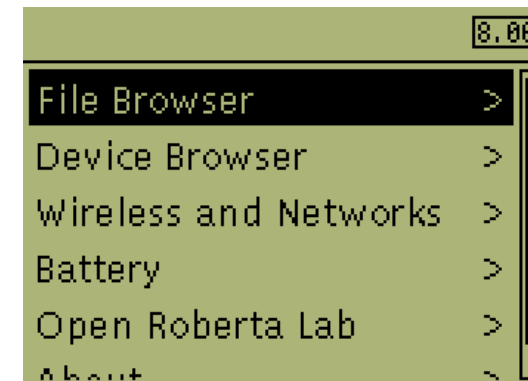


```
2. 192.168.137.3 (robot) x
? SSH compression : ✓
? SSH-browser      : ✓
? X11-forwarding   : ✗ (disabled or not supported by server)
? DISPLAY          : 192.168.0.11:0.0
> For more info, ctrl+click on help or visit our website

ev3dev
Debian jessie on LEGO MINDSTORMS EV3!

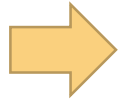
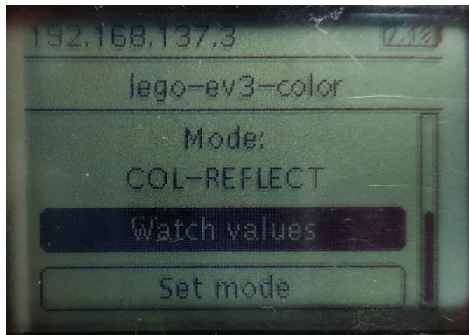
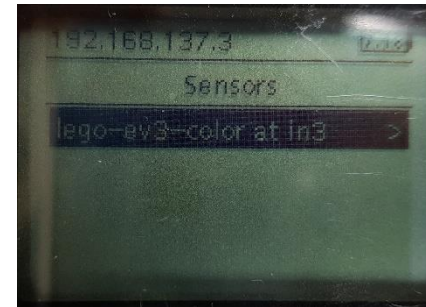
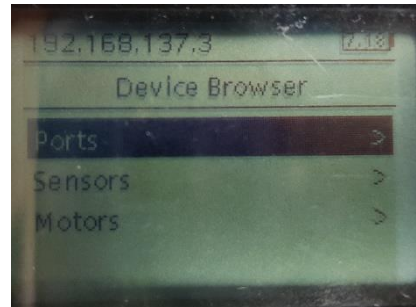
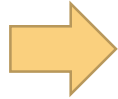
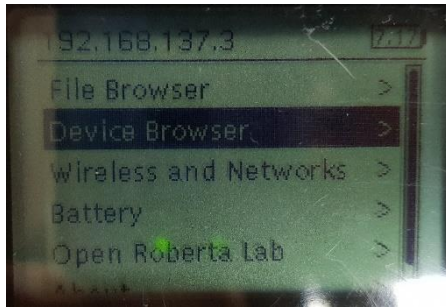
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Jan  1 00:10:51 1970 from 192.168.137.1
robot@ev3dev:~$ vi motor.py
robot@ev3dev:~$ vi motor.py
robot@ev3dev:~$ chmod +x motor.py
```



• 컬러 센서 활용

- 컬러 센서로 색상의 값을 측정하는 방법.

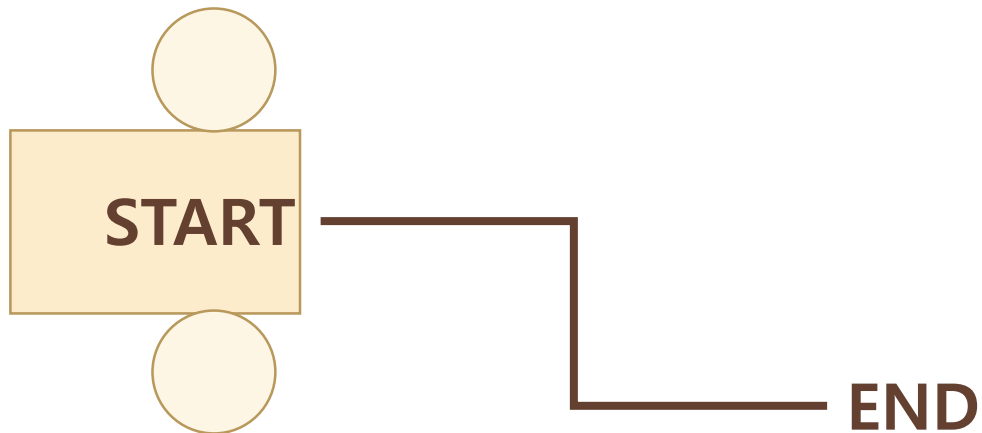


- 함수란?

- ✓ 특정한 기능만으로 분리된 단위 프로그램
- ✓ 인자(parameter)를 전달받아서 특정한 작업(기능)을 수행하고, 그 결과를 전달하는 구조
- ✓ 항상 인자를 전달받거나 결과를 전달하는 것은 아니지만 그 기능은 항상 나타남

- 함수를 활용할 때의 장점

- ✓ 반복되는 코드를 매번 중복하여 작성하지 않아도 됨 → 코드가 간결해지고 이해가 쉬워짐.
- ✓ 프로그램을 관리하고 수정하는 것이 용이함 → 수정 사항이 발생할 때, 함수 내용만 수정하면 됨.
- ✓ 팀원 간에 협업을 하는 것이 편리함
- ✓ 함수 내부의 지역변수는 함수가 끝날 때 소멸되기 때문에 메모리 낭비가 적음



```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

left.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
right.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
left.wait_while('running')
right.wait_while('running')

left.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
left.wait_while('running')

left.run_timed(speed_sp=300, time_sp=500, stop_action="hold")
right.run_timed(speed_sp=300, time_sp=500, stop_action="hold")
left.wait_while('running')
right.wait_while('running')

right.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
right.wait_while('running')

left.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
right.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
left.wait_while('running')
right.wait_while('running')
```

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

def turn_left():
    right.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
    right.wait_while('running')

def turn_right():
    left.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
    left.wait_while('running')

def go():
    left.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
    right.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
    left.wait_while('running')
    right.wait_while('running')

go()

turn_right()

go()

turn_left()

go()
```

• 함수의 선언

	함수명	인수
ex)	long_straight()	
	turn_left()	

- ✓ 함수명 : 사용자가 임의로 정하는 함수 이름으로, 함수의 기능을 알 수 있게 만드는 것을 권장함.
 - 숫자는 첫 글자로 사용할 수 없음.
 - 대소문자를 구분함.
 - 공백을 쓸 수 없음.
- ✓ 인수(매개변수) : 함수를 실행하는데 필요한 정보(데이터)를 전달받거나, 실행 결과를 넘겨주기 위한 변수들. 여러 개인 경우에는 콤마로 구분

• 함수의 기능 작성

- ✓ 함수가 처리해야 하는 일을 프로그래밍 함.

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

def turn_left():
    right.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
    right.wait_while('running')

def turn_right():
    left.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
    left.wait_while('running')

def go():
    left.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
    right.run_timed(speed_sp=300, time_sp=1500, stop_action="hold")
    left.wait_while('running')
    right.wait_while('running')
```

```
go()
turn_right()
go()
turn_left()
go()
```

• 함수의 호출

- ✓ 함수의 호출이란 선언된 함수를 사용하는 것
 - 호출하는 함수에게 인수를 넘겨준 후, 동작을 시킴
(함수가 기능을 실행함)
 - 함수 실행이 완료되면, 다시 호출했던 함수로 돌아와서
그 다음 명령을 수행함.
- ✓ 함수의 기능이 정확히 수행되도록, 순서에 맞게 필요한
데이터의 값을 지정함
- ✓ 리턴값은 변수로써 활용할 수 있음. (함수 = 기능+변수)

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

def turn_left():
    right.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
    right.wait_while('running')

def turn_right():
    left.run_timed(speed_sp=300, time_sp=1200, stop_action="hold")
    left.wait_while('running')

def go(sp, time):
    left.run_timed(speed_sp=sp, time_sp=time, stop_action="hold")
    right.run_timed(speed_sp=sp, time_sp=time, stop_action="hold")
    left.wait_while('running')
    right.wait_while('running')

go(300, 1500)

turn_right()

go(300, 500)

turn_left()

go(300, 1200)
```

• 함수의 유형

	함수명	인수
ex)	go()	spd, time

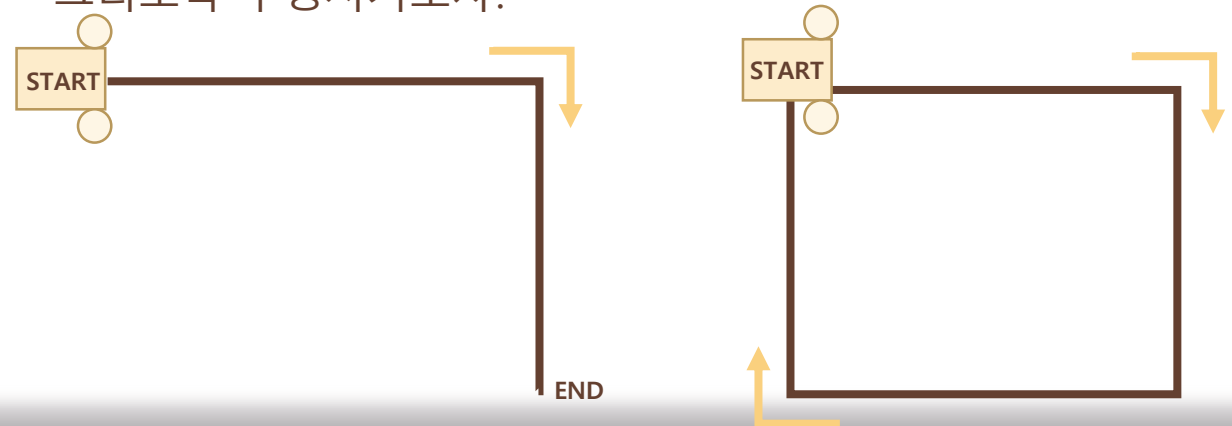
✓ return하는 값의 유형과 동일해야 함.

• 인수(매개변수) 정의

✓ 함수 수행에 필요한 정보를 선언함.

• 로봇의 이동 거리와 속도를 바꿀 수 있는 함수로 수정해보자!

✓ 인수를 다양하게 바꾸어 입력하면서, 로봇이 더 커다란 모양을 그리도록 주행시켜보자!



함수의 개념과 이해

함수의 선언 / return

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

def go2sec():
    left.reset()

    left.run_timed(speed_sp=300, time_sp=2000, stop_action="hold")
    right.run_timed(speed_sp=300, time_sp=2000, stop_action="hold")
    left.wait_while('running')
    right.wait_while('running')

    pos=left.position
    return pos

gopos = go2sec()

left.run_to_rel_pos(speed_sp=-300, position_sp=-gopos, stop_action="hold")
right.run_to_rel_pos(speed_sp=-300, position_sp=-gopos, stop_action="hold")
```

• 함수의 유형

	함수명	Return 값
ex)	go2sec()	pos

✓ return하는 값의 유형과 동일해야 함.

• return값 처리

✓ 함수가 저장하는 값. 호출 함수로 되돌려줄 때 사용.

- return은 해당 함수를 종료함.



- 함수를 호출하는 방법으로 출발점에서 도착점까지 주행하기!

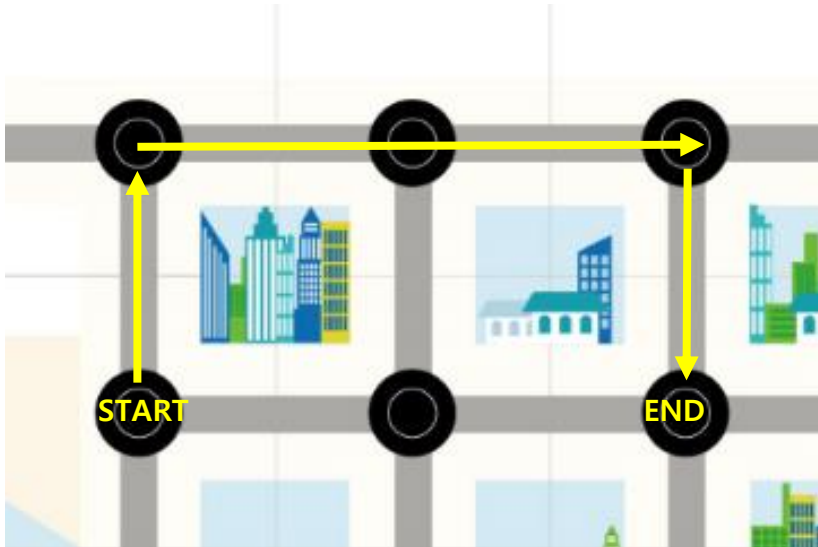
- ✓ 메인에는 함수를 호출하는 프로그램만 작성!
ex)

```
goandturn(1500, 1300)  
goandturn(2000, 0)
```

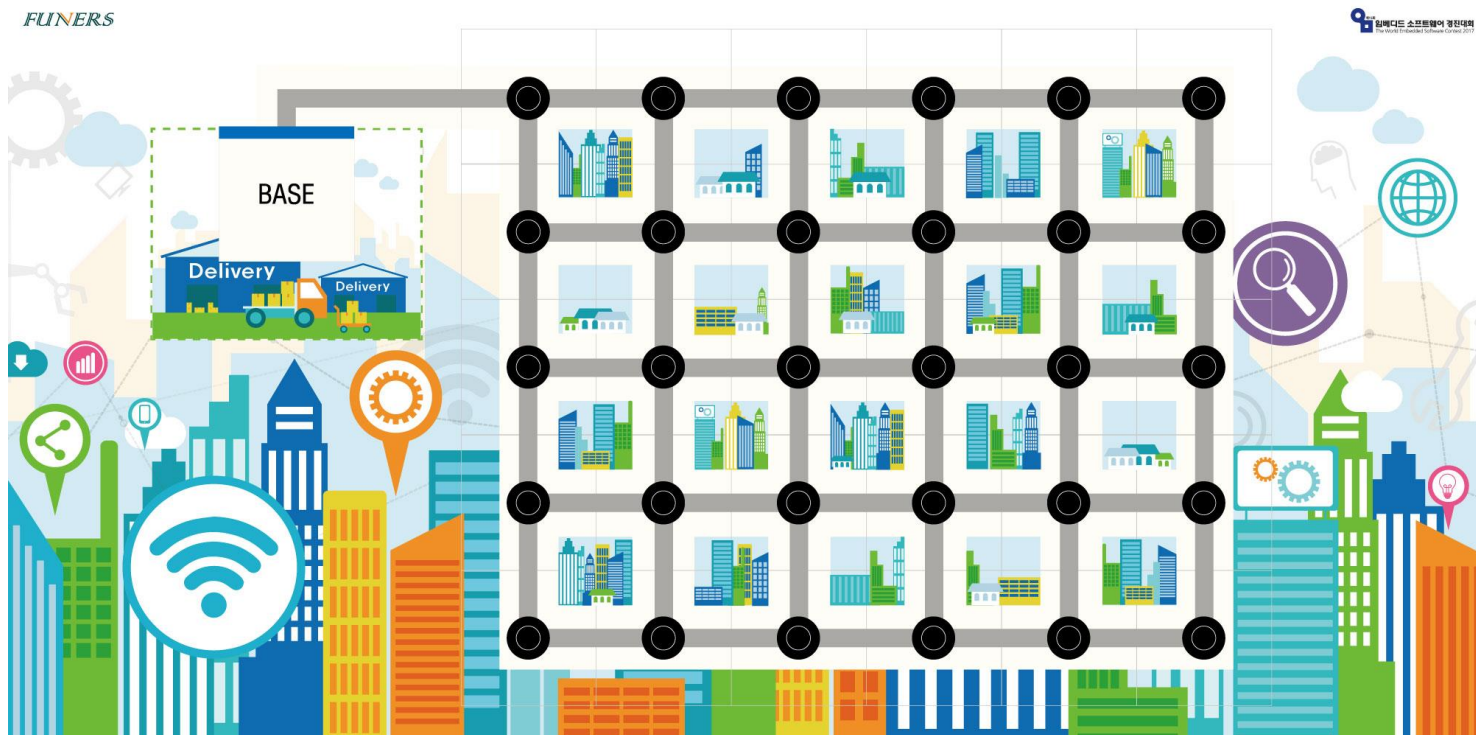
- ✓ 로봇이 일정 거리만큼 직진하는 기능과 정확히 90도로 우회전하는 기능을 포함하는, **단 1개의 함수만** 작성!
- ✓ 함수의 인수는 시간 값을 입력 받도록 되도록 정의!

함수 활용 예제 실습

함수 활용 예제



```
def goandturn(go, turn) :  
    left.run_timed(speed_sp=300, time_sp=go, stop_action="hold")  
    right.run_timed(speed_sp=300, time_sp=go, stop_action="hold")  
    left.wait_while('running')  
    right.wait_while('running')  
  
    left.run_timed(speed_sp=300, time_sp=turn, stop_action="hold")  
    right.run_timed(speed_sp=0, time_sp=turn, stop_action="hold")  
    left.wait_while('running')  
    right.wait_while('running')
```



• 주행 제어의 필요성

- ✓ 로봇이 맵을 탐색하고, 원하는 위치로 이동하기 위해 로봇의 주행 제어는 필수적으로 요구됨
- ✓ 주어진 맵의 특성을 잘 파악하여 로봇의 주행 방법을 적절히 설계해야 주어진 임무를 완수할 수 있음



• 주행 제어에 필요한 단위 동작의 예시

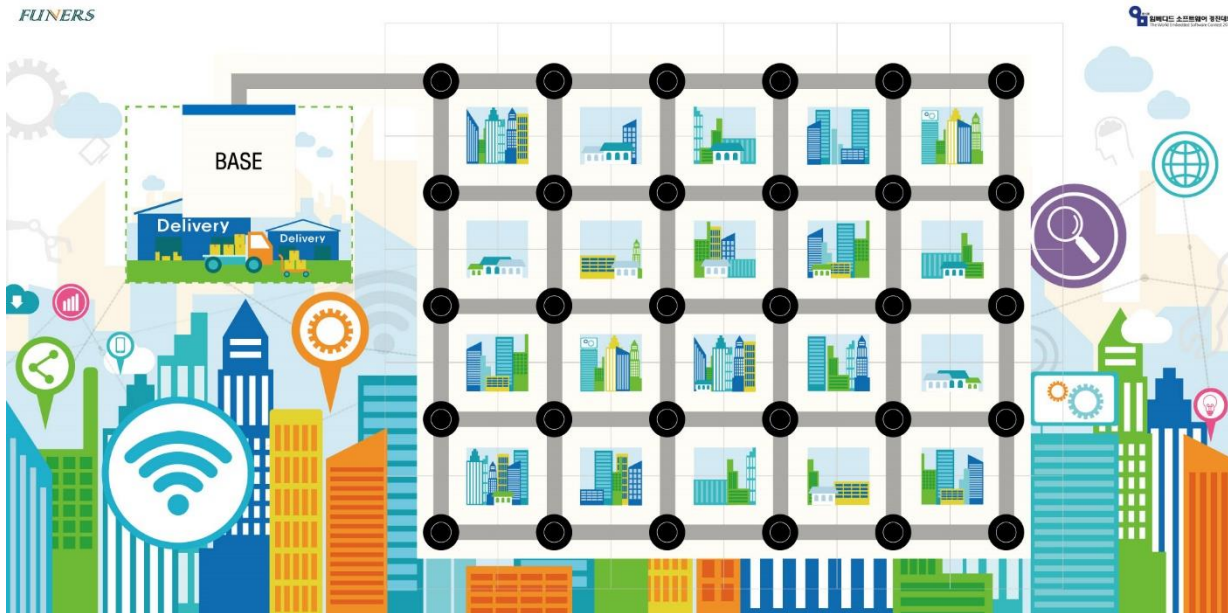
✓ 셀 단위 주행

- 라인을 따라 이동하여 다음 셀까지 주행

✓ 색깔 감지

✓ 방향 전환

- 직진 / 좌회전 / 우회전 / U-turn 중 택일하여 움직임(이 때, 컬러 센서의 위치는 라인트레이싱이 이어질 수 있는 위치에 도달하도록 해야 함!)

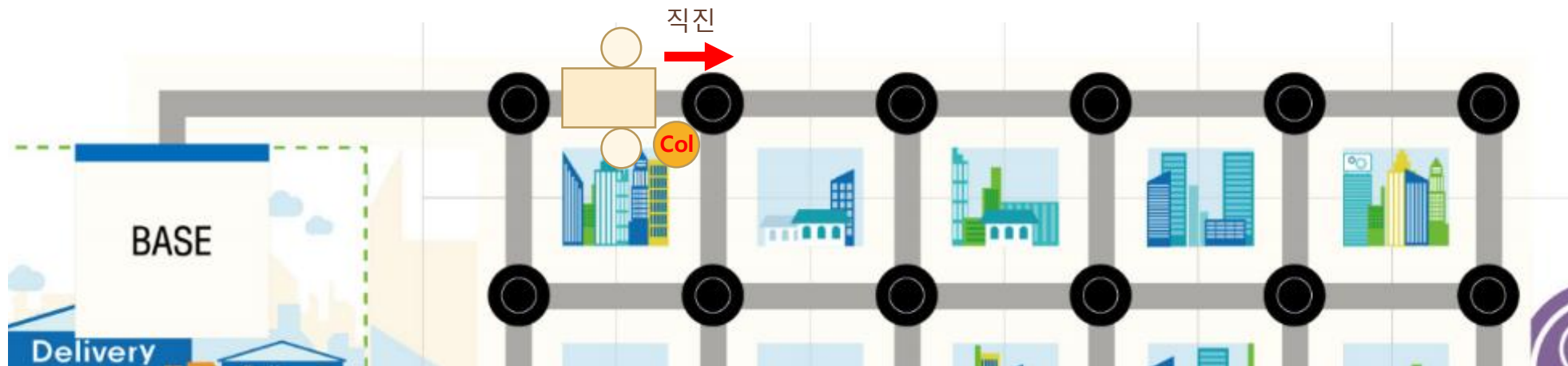


주행 제어하기

셀 단위 주행/셀과 셀 사이를 어떻게 이동할까?

방법 1) 직진을 활용한 경로 주행

- ✓ 셀과 셀 사이를 이동할 때, 미리 알아둔 Encoder 값(position_sp)을 활용하여 직진과 90도 회전을 수행하는 방법
- ✓ 구현하기는 비교적 쉬우나, 주행 횟수가 늘어날 때마다 오차가 누적되어 갈수록 원래 경로에서 벗어날 확률이 높아짐.



주행 제어하기

셀 단위 주행/셀과 셀 사이를 어떻게 이동할까?

방법 2) 라인트레이싱을 활용한 셀 단위 주행

- ✓ 셀과 셀 사이를 이동할 때, 컬러 센서를 활용해 한쪽 라인을 타고 가는 라인트레이싱을 활용하는 방법
- ✓ 구현하기는 다소 까다롭지만, 누적 오차의 영향이 적음
- ✓ 셀과 셀 사이에서 라인트레이싱을 시작하는 지점과 끝나는 지점을 어떻게 판단할 것인가에 대한 기준을 잘 세워야 함



주행 제어하기

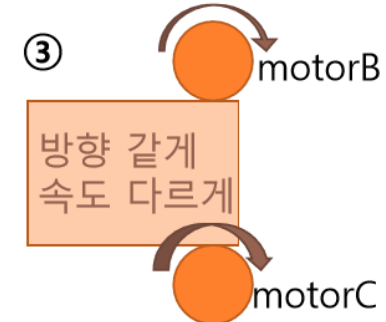
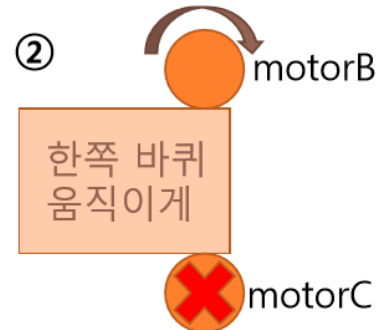
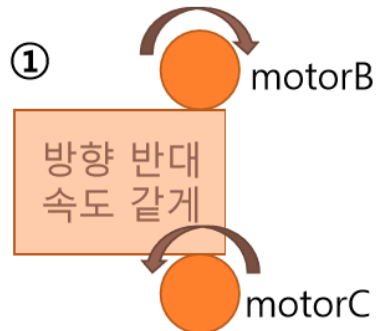
셀 단위 주행/셀과 셀 사이를 어떻게 이동할까?

- 고려해야할 점

- 셀과 셀 사이에서 라인트레이싱을 시작하는 지점은 어디인가?
- 라인트레이싱이 끝나는 지점은 어떻게 판단할 것인가?



- 라인트레이싱은 어떤 주법을 사용하여 진행할 것인가?



주행 제어하기

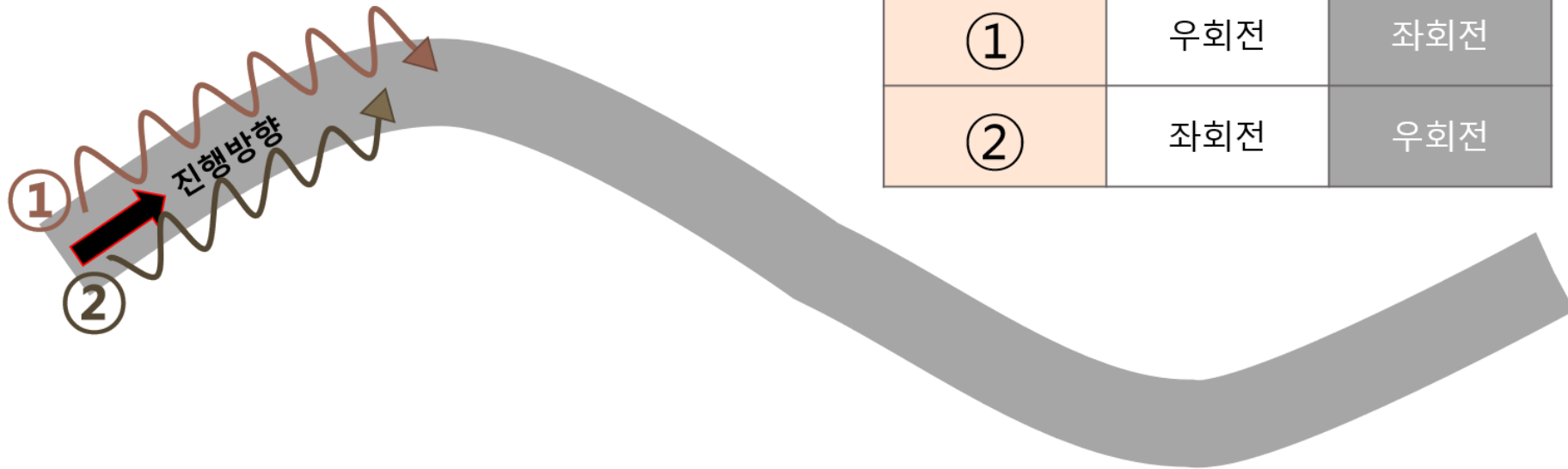
셀 단위 주행/셀과 셀 사이를 어떻게 이동할까?

- 라인트레이싱의 시작 지점과 끝나는 지점 고려하기



- 라인트레이싱

- 컬러 센서를 활용한 라인트레이싱



- 컬러 센서를 활용한 라인트레이싱 <Zig-Zag Line tracing>

- Button 함수

- b = Button() : 버튼을 변수 b로 설정한다.

ex) btn=Button()

- b.xxx : 버튼의 상태를 설정한다.

상태 : up, down, left, right, enter, backspace

Ex) btn.backspace

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

color = ColorSensor('in3')
color.mode='COL-REFLECT'

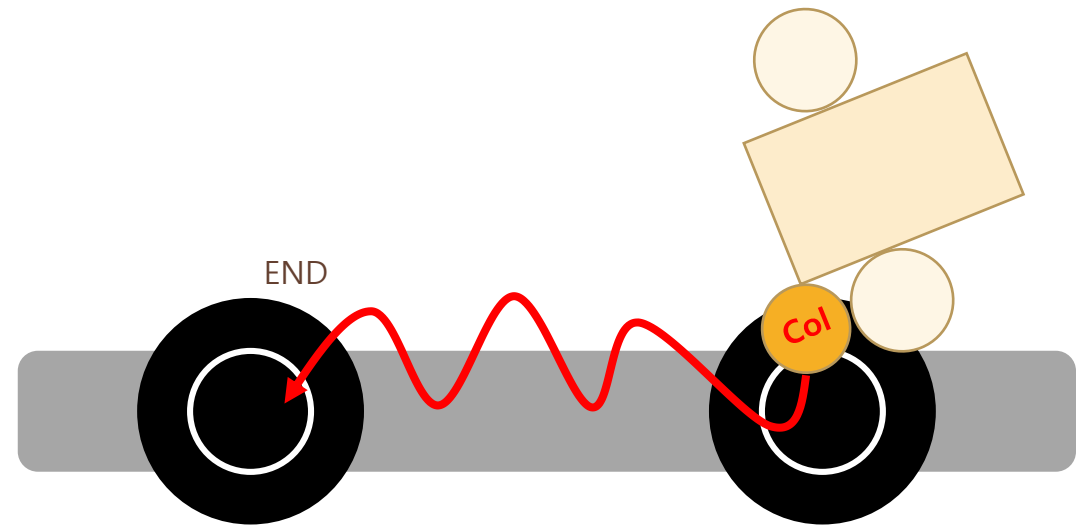
btn = Button()

while not btn.backspace:
```

?

- 셀 단위 주행 프로그램을 작성하고 테스트하기

- ✓ 흰색, 회색, 검정색에서의 컬러센서 반사값을 측정하여 라인트레이싱을 위한 Threshold 값 설정
- ✓ Threshold와 센서값을 비교하여 작동하는 라인트레이싱 프로그램 작성
- ✓ 종료 지점으로 판단되면 라인트레이싱 끝!



- 셀 단위 주행 프로그램을 작성하고 테스트하기

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

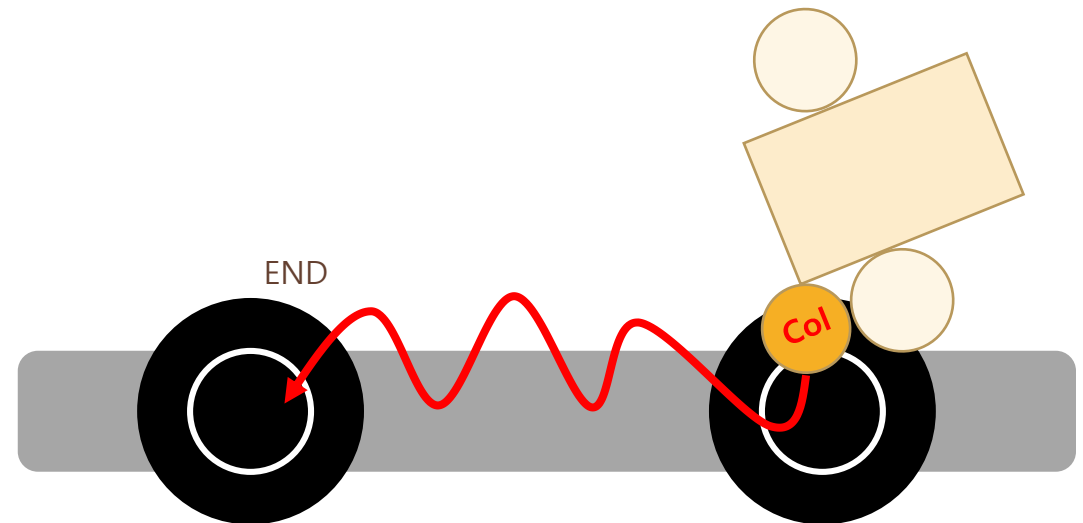
left = LargeMotor('outB')
right = LargeMotor('outC')

color = ColorSensor('in3')

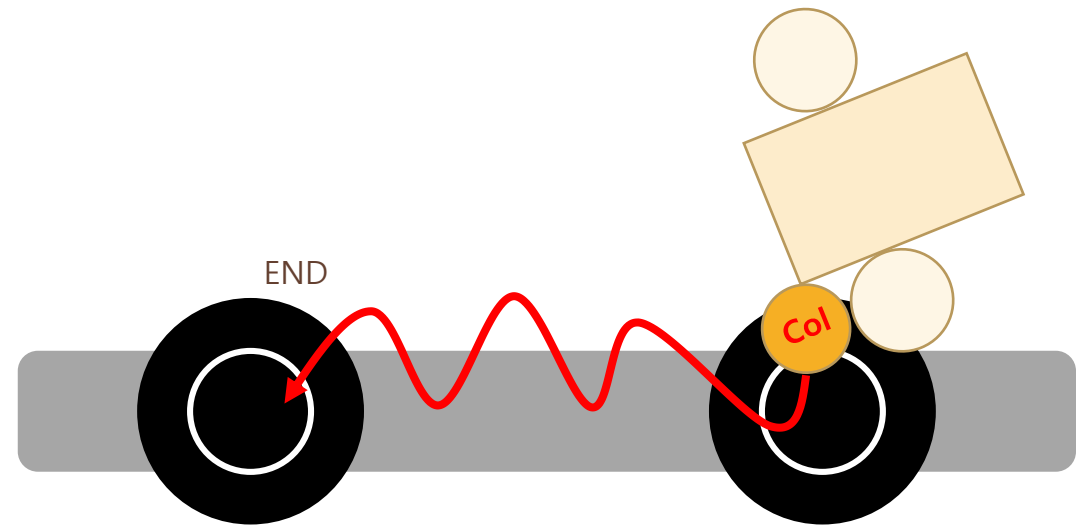
btn = Button()

while not btn.backspace :
```

?



- 셀 단위 주행 프로그램을 함수로 만들고, 테스트하기
 - ✓ 2개의 모터 파워(빠른 파워, 느린 파워)를 인수로 전달받는 함수 작성하기
 - ✓ 파워를 조절하면서 라인트레이싱이 부드럽게 진행되도록 테스트하기
 - ✓ 라인트레이싱이 끝나는 지점에서 로봇이 향하는 방향을 확인하기

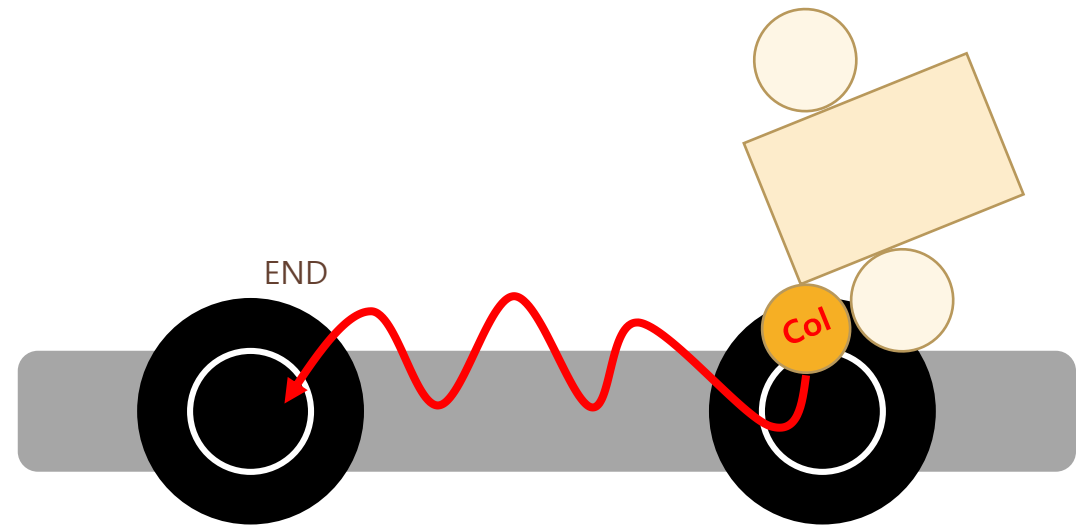


- 셀 단위 주행 함수를 만들고, 테스트하기

```
#!/usr/bin/env python3  
  
from ev3dev.ev3 import *  
from time import sleep  
  
left = LargeMotor('outB')  
right = LargeMotor('outC')  
  
color = ColorSensor('in3')  
  
btn = Button()
```

?

```
while not btn.backspace :  
    linetrace(200, 300)
```

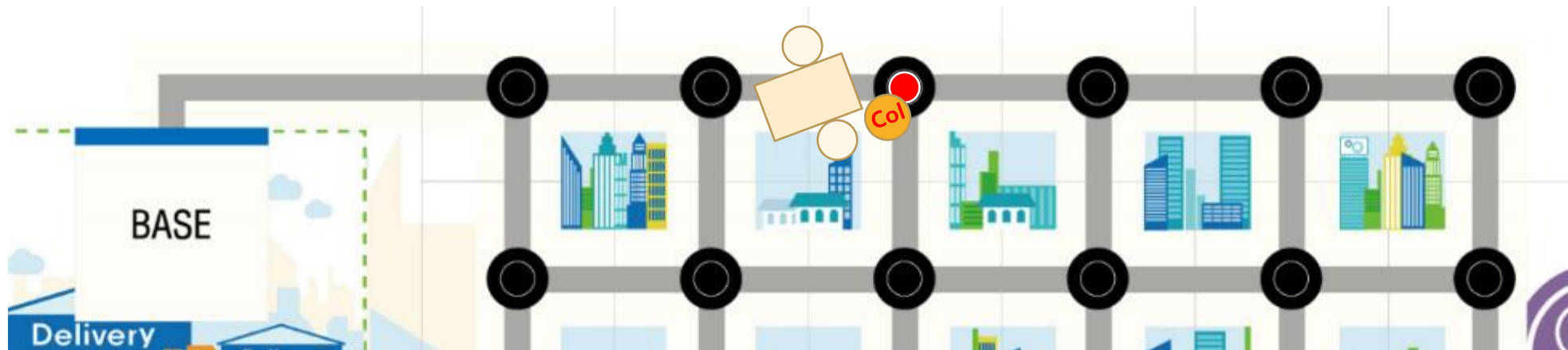


주행 제어하기

색깔 감지/교차점에 도착했을 때 필요한 동작

- 색깔 정보 감지

- ✓ 컬러 센서를 'COL-COLOR'(컬러모드)로 변경하고, 바닥의 색깔을 확인해야 함.



```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

left = LargeMotor('outB')
right = LargeMotor('outC')

color = ColorSensor('in3')

btn = Button()

while not btn.backspace :
    color.mode = 'COL-COLOR'

    while True :
        if color.value() == 7 :
            left.run_forever(speed_sp=0)
            right.run_forever(speed_sp=0)
        else :
            left.run_forever(speed_sp=300)
            right.run_forever(speed_sp=300)
```

• 색깔 정보 읽기

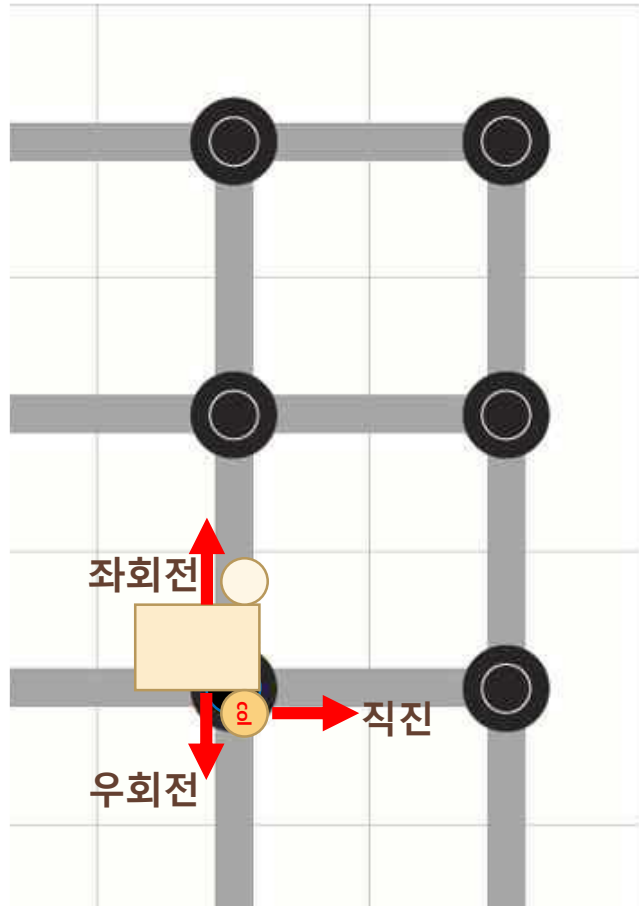
- ✓ 색깔 정보를 읽기 위해 센서가 색상 패치를 향하도록 로봇의 방향을 틀어줌
- ✓ 컬러 센서를 'COL-COLOR'(컬러모드)로 변경함
→ color.mode = 'COL-COLOR'
- ✓ 바닥의 색상을 읽어 변수에 저장

• x.mode = 'COL-xxx'

- ✓ 센서의 모드를 변경하는 함수
- ✓ 한 번 명령하면 이후에는 해당 모드로 작동하며, 모드를 변경하고자 할 때는 다시 명령을 작성해야 함.
- ✓ 'COL-REFLECT' : 반사광 모드
- ✓ 'COL-COLOR' : 컬러 모드
- ✓ 'COL-AMBIENT' : 주변광 모드

주행 제어하기

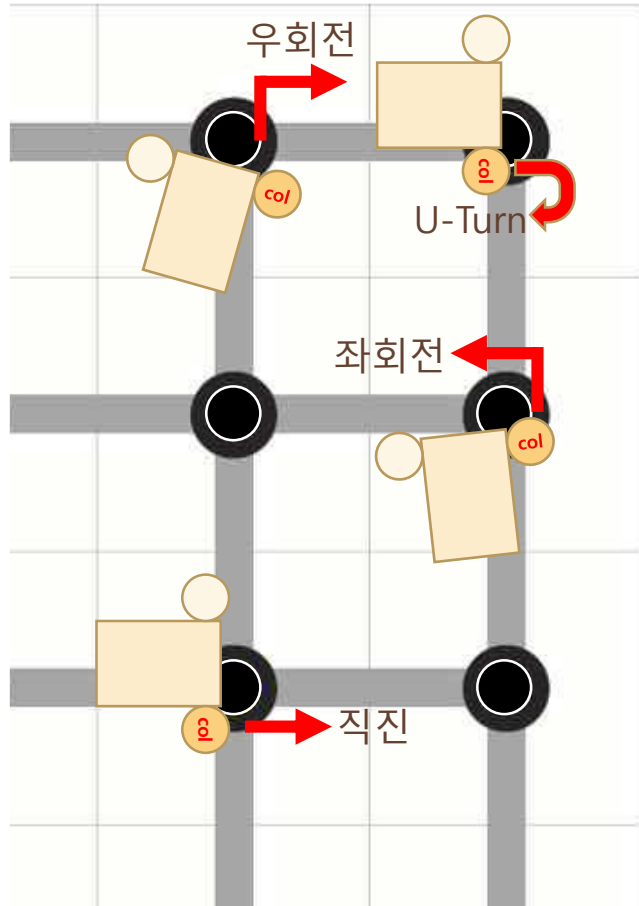
방향 전환/로봇의 주행 방향을 결정하고 움직이기



- 색깔 정보를 읽은 후, 다음 셀로 이동하기 전에 현재 위치에서 어떤 방향으로 이동할 것인지 결정해야 함.
- 원하는 주행 방향으로 가장 쉽고 효과적으로 움직일 수 있도록 프로그램을 작성하는 방법은?
 - ✓ 방법1) 전진 / 우회전 / 좌회전 / 유턴의 함수를 모두 별도로 작성하여 필요한 함수를 불러오는 방법
 - ✓ 방법2) 하나의 함수를 이용하여 command에 따라 전진 / 우회전 / 좌회전 / 유턴 중의 1가지 동작을 실행하는 방법

주행 제어하기

방향 전환/로봇의 주행 방향을 결정하고 움직이기

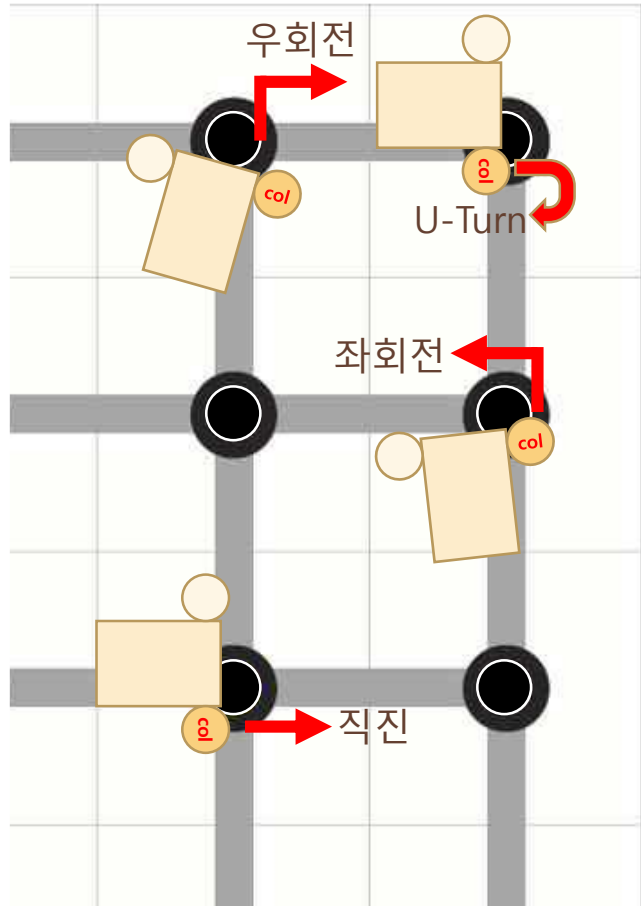


• 명령에 따라 직진, 좌회전, 우회전, U-Turn 하기

- ✓ 교차점에 도착했을 때, 로봇이 원하는 진행 방향에 따라 작동하도록 하는 기능을 가진 1개의 함수 작성하기
 - 로봇은 함수가 호출될 때 전달된 파라미터에 따라 직진, 좌회전, 우회전, U-Turn 중 하나를 선택하여 작동해야 함.

TIP☺ 함수를 작성할 때, if-elif를 사용하면 파라미터로 전달한 값에 따라 여러 동작 중 1개의 동작 만을 수행하도록 할 수 있음

주행 제어하기 방향 전환/실습



```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

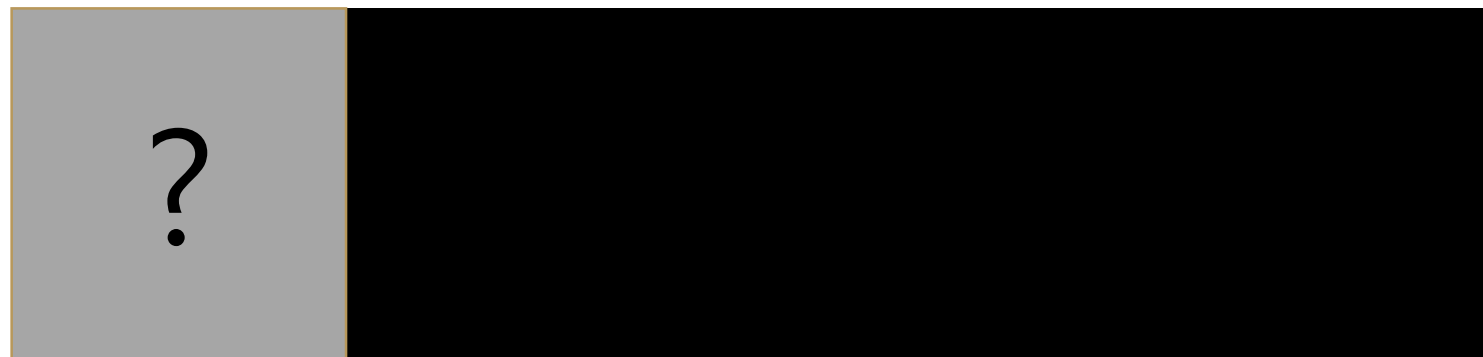
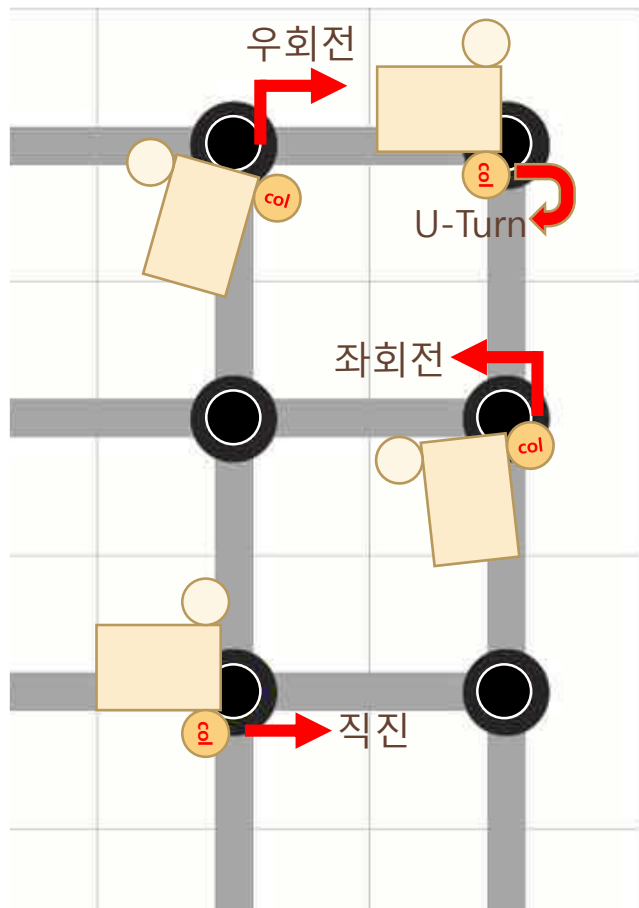
left = LargeMotor('outB')
right = LargeMotor('outC')

cmd_forward = 0
cmd_rigth = -1
cmd_left = 1
cmd_urn = 2

def MoveOneCell(command) :
```

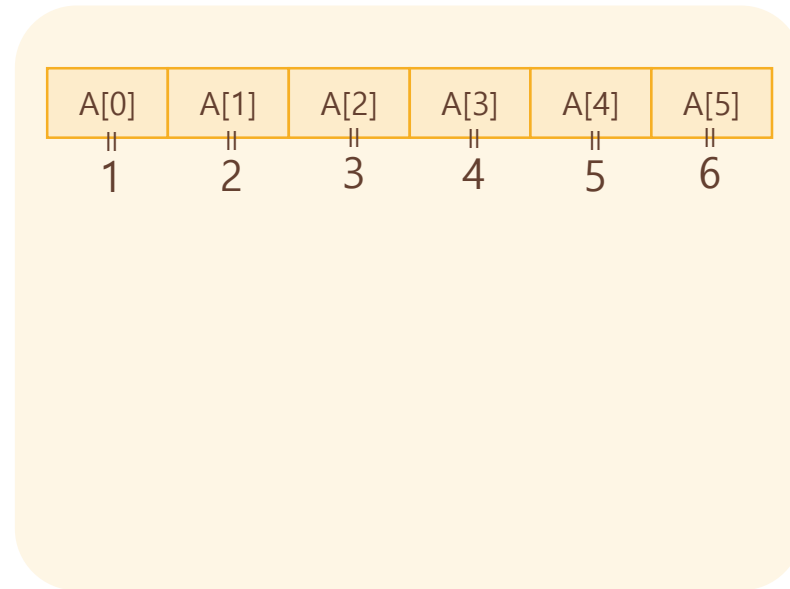
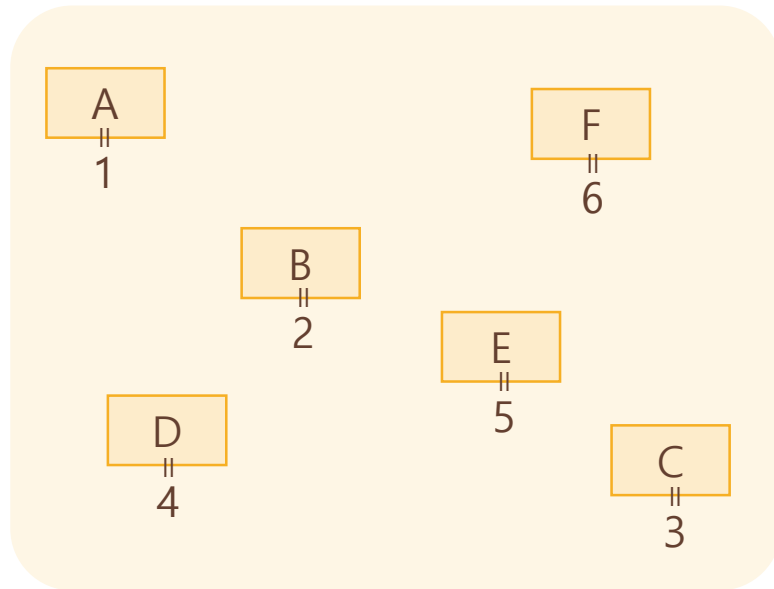
?

주행 제어하기 방향 전환/실습



• 배열이란?

- ✓ '같은 자료형'을 가진 동일한 유형의 묶음 형태의 변수
- ✓ 어떤 한 유형의 변수 여러 개가 순서대로 묶여진 형태
- ✓ 일반 변수와 배열의 차이점? → 1가구 주택 vs 아파트



• 배열이란?

- ✓ 인덱스를 활용하여 동일한 속성을 갖는 여러 개의 데이터를 한꺼번에 다룰 수 있음.
- ✓ 복잡하거나 많은 양의 자료를 다룰 때 유용함.
- ✓ 전체 크기를 선언하여야 함.
- ✓ 인덱스를 이용하여 특정 위치를 지정하여 사용함.
- ✓ 1차원, 2차원 및 다차원 배열을 사용 가능함.

1	2	3.5
---	---	-----

1	a	2	b	3
---	---	---	---	---

1	0	12	15	2
26	12	2	65	93
3	30	40	7	33
34	43	2	9	8

- 배열의 유형은 실수, 문자, 문자열, boolean등에 상관없이 선언 가능함.
- 전체 크기를 선언하여야 함.

A=[1,2,3.5]

B=[1,"a",2,"b",3]

C=[[1,0,12,15,2],[26,12,2,65,93],
[3,30,40,7,33],[34,43,2,9,8]]

a

1	2	3.5
---	---	-----

b

1	a	2	b	3
---	---	---	---	---

c

1	0	12	15	2
26	12	2	65	93
3	30	40	7	33
34	43	2	9	8

행

열

- 배열의 위치를 지정하여 사용
- 시작 위치의 번호는 반드시 0부터 시작함.

A=[1,2,3.5]

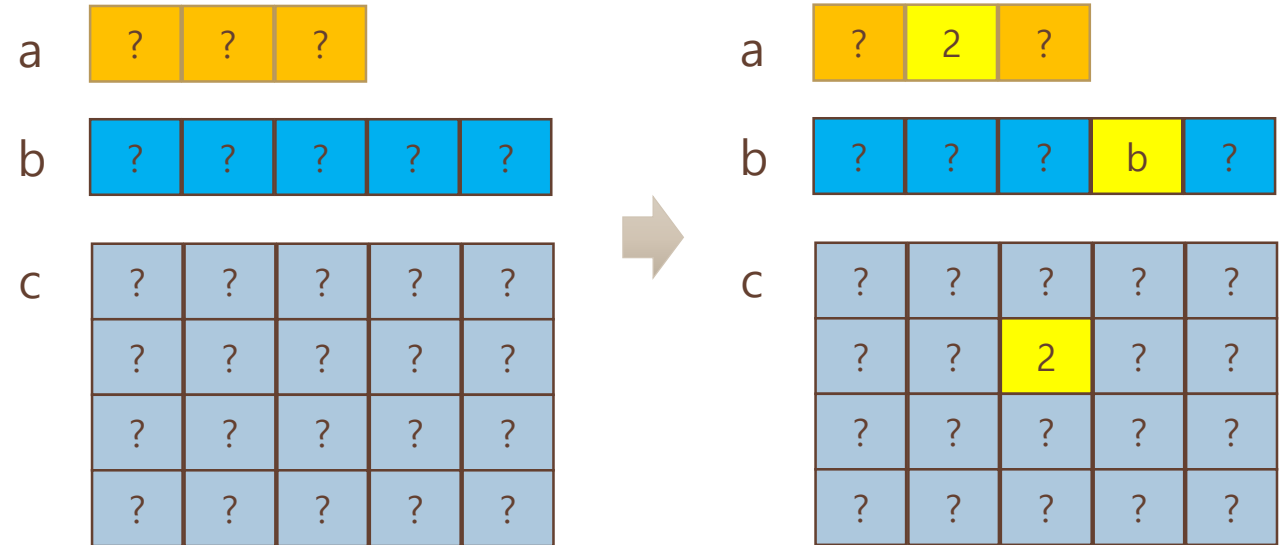
B=[1,"a",2,"b",3]

**C=[[1,0,12,15,2],[26,12,2,65,93],
[3,30,40,7,33],[34,43,2,9,8]]**

A[1] = 2

B[3] = b

C[1][2] = 2




```
a = [1,2,3,4,5]
b = [1,"a",2,"b",3]
print(a[0])
print(b[1])
```



실행

```
robot@ev3dev:~$ python3 array.py
1
a
```

- 1차원 배열의 형식 : 배열명=[값1, 값2...] ex) A=[1,2,3,4,5]
 - ✓ 배열명 : 배열의 이름(변수 명명 규칙과 동일하게 적용)
 - ✓ 값 : 실수, 문자, 문자열, boolean등의 값을 쉼표로 구분
 - ✓ 배열 index의 개수는 값이 늘어날 수록 자동으로 증가함.
(주의!! 배열의 index는 항상 [0]부터 시작)

배열 1차원 배열/평균 반사값 구하기

- 첫번째 반사값 읽고 2초 후 두번째 반사값 읽기

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

color = ColorSensor('in3')
color.mode = 'COL-REFLECT'

i = color.value()
sleep(2)
j = color.value()

a = [i,j]

print(a)
```

- 평균 반사값 구하고 평균 반사값보다 작은 반사값에서 빨강 LED 켜기

```
#!/usr/bin/env python3

from ev3dev.ev3 import *
from time import sleep

color = ColorSensor('in3')
color.mode = 'COL-REFLECT'

i = color.value()
sleep(2)
j = color.value()

a = [i,j]
```

?

- LED 함수

- Leds.set_color(group, color) : EV3 P브릭의 LED를 제어한다.
 - group : Leds.LEFT, Leds.RIGHT
 - color : Leds.RED, Leds.YELLOW, Leds.GREEN, Leds.ORANGE
- Ex) Leds.set_color(Leds.Left, Leds.RED)
- Leds.all_off() : EV3 P브릭의 LED를 전부 끈다.

배열

2차원 배열의 선언과 활용

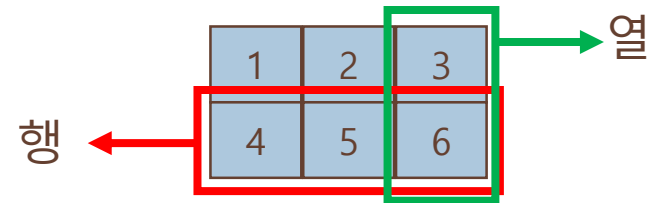
```
A = [[1,2,3],[4,5,6]]  
print(A[1][2])
```



실행

```
robot@ev3dev:~$ python3 array2.py  
6
```

- 2차원 배열의 형식 : 배열명 **[(1행)], [2행),...]** ex) A=**[[1,2,3], [4,5,6]]**
 - ✓ 2차원 배열은 행과 열로 이루어진 MATRIX 형태
 - ✓ 배열의 index는 항상 **[0]부터** 시작하기 때문에, 위의 예시와 같은 경우, 마지막 원소는 **[1][2]**가 된다.
 - ✓ ex) A=**[[1,2,3], [4,5,6]]**을 좀더 이해하기 쉽게 풀어 쓰면
= **[[1,2,3],**
 [4,5,6]] 이다.



- 참고 사이트

- EV3 Python 학습 사이트 : www.ev3python.com
- API 참조 : <http://python-ev3dev.readthedocs.io/en/latest/spec.html>

- 교육, 기술 지원 및 경기장 구매 문의

퓨너스 (www.funers.com) T.070-8670-8911