

# Wiki Documentation

## 전체 위키 문서

---

### Home

---

## 블록 컴포저 Wiki

블록 컴포저는 다양한 기본 블록을 제공합니다. 이 블록들은 다음과 같은 카테고리로 나누어집니다: - 논리 (조건문 포함)  
- 반복 - 연산 - 문자열 - 리스트 - 색상 - 소리 - 제어 - 변수 - 함수 - 기타

## PDF 다운로드

전체 Wiki 를 PDF 로 다운로드할 수 있습니다:

- PDF 다운로드 (Raw)
- GitHub 에서 보기

## 기타

---

블록 코딩에서 **기타 블록**은 코드 실행 및 로봇의 동작에 영향을 주지 않는 코드들로 구성됩니다.  
주석을 달거나, 코드 실행을 종료하는 기능을 수행할 수 있습니다.

## 블록

## 주석

**주석** 블록을 사용하면 코드의 실행에는 영향을 주지 않으면서 **설명**을 추가할 수 있습니다.  
주석을 활용하면 코드 가독성이 높아지고, 유지보수가 쉬워집니다.

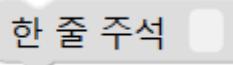


Figure 1: Image

여러 줄 주석

Figure 2: Image

### Javascript 코드

```
// 한 줄 주석  
//여러 줄 주석  
//이어서 입력
```

### Python 코드

```
# 한 줄 주석  
# 여러 줄 주석  
# 이어서 입력
```

### 링크

링크 블록을 사용하면, 주석 기능을 활용해 열고 싶은 페이지 링크를 추가할 수 있습니다.

열기 버튼을 클릭하면, 입력한 링크의 페이지로 이동할 수 있습니다.

### Javascript 코드

```
// https://google.com
```

### Python 코드

```
# https://google.com
```

### 종료하기

종료하기 블록은 프로그램 실행을 즉시 중단하고 페이지를 새로고침하여 초기 상태로 되돌립니다.

특정 조건에서 강제 종료 기능을 추가할 때 유용합니다.

실행 종료하기

Figure 3: Image

## Javascript 코드

```
__exit(); // 종료하기
```

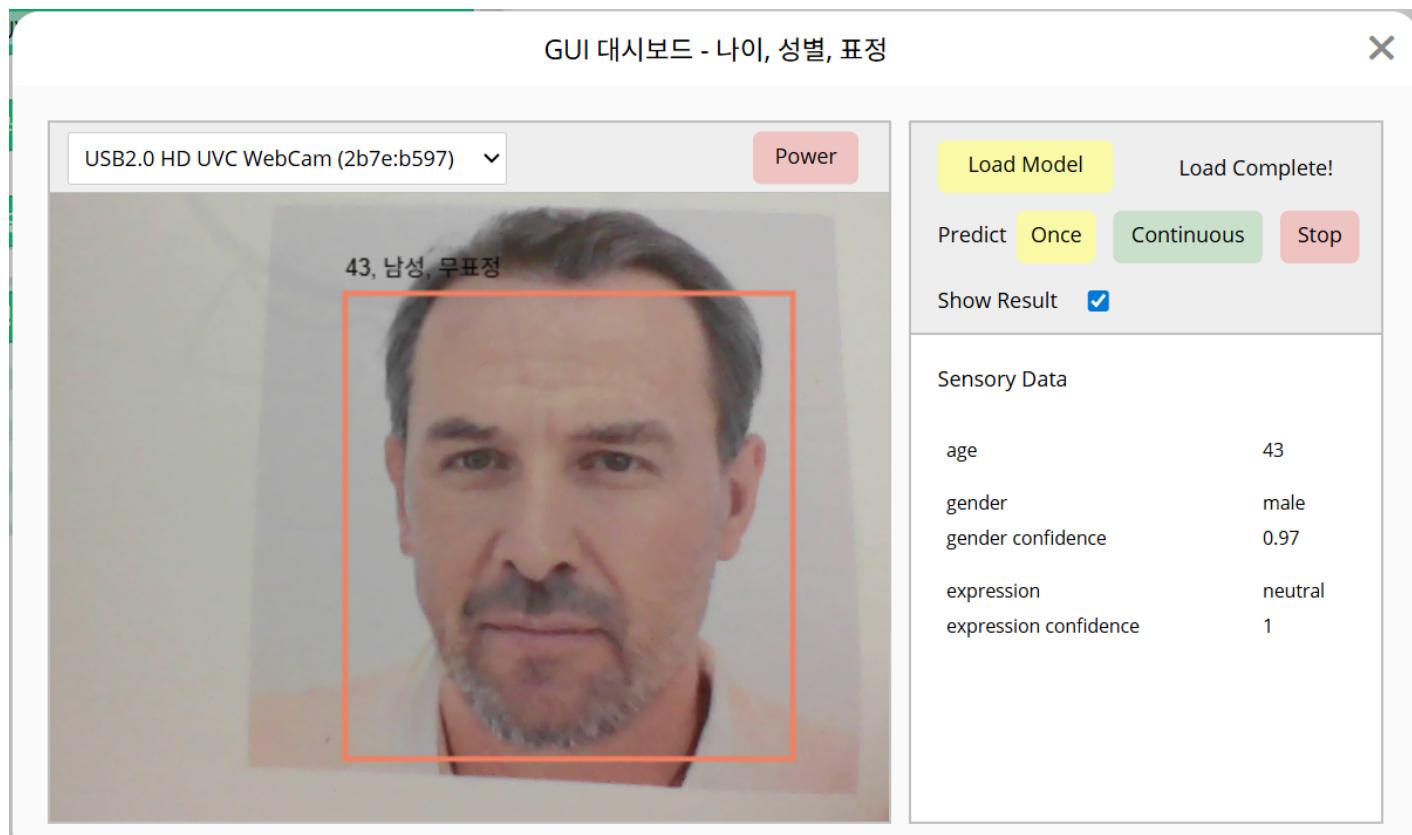
## Python 코드

```
__exit() # 종료하기
```

나이,-성별,-표정

## 대시보드 열기

대시보드 열기는 설치할 수 있는 블록은 아니지만, 확장 모듈에서 사용되는 모델이 어떠한 식으로 적용되는지 알 수 있는 대시보드를 열 수 있습니다. ## 대시보드 화면 대시보드 열기 클릭 시 다음과 같은 화면을 볼 수 있습니다.



## 세부 버튼

### Power

선택한 카메라를 키거나 끕니다.

### Load Model

학습된 나이, 성별, 표정 모델을 불러옵니다. ‘나이, 성별, 표정’확장 모듈을 사용하기 위해서 반드시 필요한 작업입니다.

### Show Result

나이, 성별, 표정 예측 결과를 카메라 화면 상으로 출력합니다.

### Detect

나이, 성별, 표정 예측을 실행하거나 멈춥니다.

Once 버튼으로 한번만 실행할지, Continuous 버튼으로 연속으로 실행할지 정할 수 있습니다.

또한, Stop 버튼을 통해 예측을 멈출 수 있습니다.

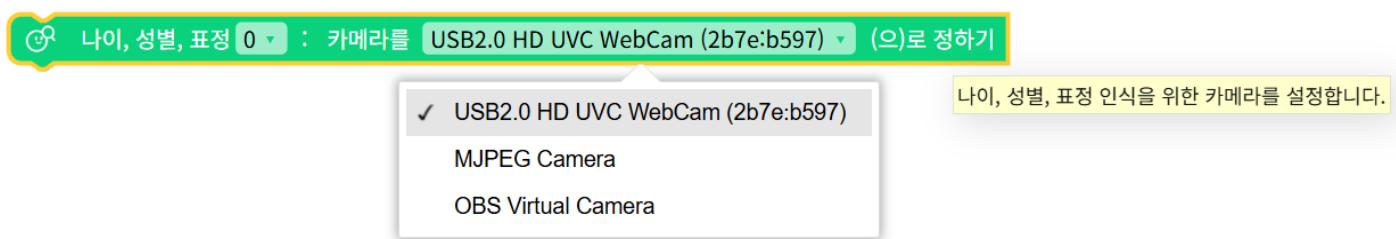
### Sensory Data

나이, 성별, 표정 예측 데이터 값을 출력합니다.

## 블럭

### 카메라 정하기

나이, 성별, 표정 예측 모듈에 사용할 카메라를 선택합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
camera	드롭다운 옵션	사용할 카메라	연결한 카메라 리스트

## 자바스크립트 코드

```
// 특정 카메라를 나이, 성별, 표정 예측을 위한 카메라로 정하기 (id 는 예시)
$('FaceExpression*0:camera.deviceId').d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb';
```

## 파이썬 코드

```
# 특정 카메라를 나이, 성별, 표정 예측을 위한 카메라로 정하기 (id 는 예시)
__('FaceExpression*0:camera.deviceId').d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb'
```

## 나이, 성별, 표정 모델 불러오기

학습된 나이, 성별, 표정 예측 모델을 불러옵니다. ‘나이, 성별, 표정’모듈의 기능을 사용하기 위해서는 이 작업이 반드시 필요합니다.



나이, 성별, 표정 0 : 나이, 성별, 표정 모델 불러오기 | 기다리기 ✓

학습된 나이, 성별, 표정 모델을 불러옵니다. '나이, 성별, 표정' 모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

## 자바스크립트 코드

```
// 나이, 성별, 표정 모델 불러오기 | 기다리기 0
$('FaceExpression*0:load_model').d = 1;
await $('FaceExpression*0:!load_model').w();

// 나이, 성별, 표정 모델 불러오기 | 기다리기 X
$('FaceExpression*0:load_model').d = 1;
```

## 파이썬 코드

```
# 나이, 성별, 표정 모델 불러오기 | 기다리기 0
__('FaceExpression*0:load_model').d = 1
await __('FaceExpression*0:!load_model').w()

# 나이, 성별, 표정 모델 불러오기 | 기다리기 X
__('FaceExpression*0:load_model').d = 1
```

## 나이, 성별, 표정을 한 번 인식하기

현재 화면에 있는 얼굴을 분석하여 예측한 나이, 성별, 표정을 딱 한번 표시합니다.



나이, 성별, 표정 0 : 나이, 성별, 표정을 한 번 인식하기

현재 화면에 있는 얼굴을 분석하여 예측한 나이, 성별, 표정을 딱 한번 표시합니다.

### 자바스크립트 코드

```
// 나이, 성별, 표정을 한 번 인식하기  
$('FaceExpression*0:predict.once').d = 1;
```

### 파이썬 코드

```
# 나이, 성별, 표정을 한 번 인식하기  
__($('FaceExpression*0:predict.once').d = 1
```

## 나이, 성별, 표정을 연속으로 인식하기

나이, 성별, 표정 연속적으로 예측하기를 시작하거나 중지합니다.

연속으로 예측하기를 시작하면, 현재 화면에 있는 얼굴을 계속 따라가며 예측한 결과를 화면상에 표시합니다.



나이, 성별, 표정 0 : 연속으로 나이, 성별, 표정 인식

시작하기 ▾

현재 화면에 있는 얼굴을 계속해서 분석하여 예측한 나이, 성별, 표정을 화면 상에 표시합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	예측하기	시작하기 (1), 중지하기 (0)

### 자바스크립트 코드

```
// 연속으로 나이, 성별, 표정 예측 시작하기  
$('FaceExpression*0:predict.continuous').d = 1;
```

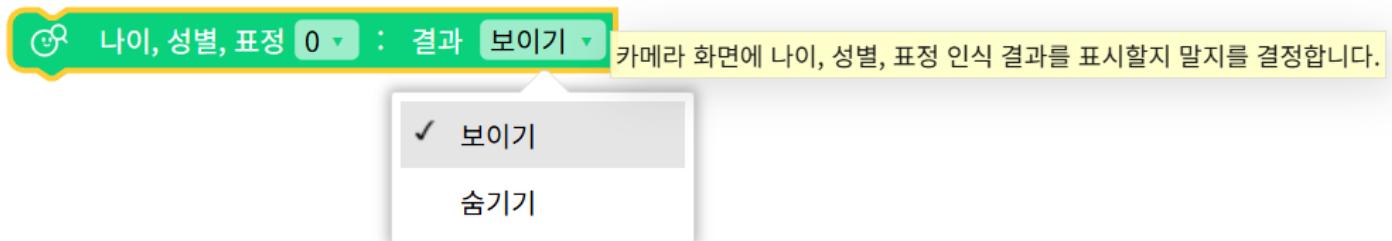
```
// 연속으로 나이, 성별, 표정 예측 중지하기  
$('FaceExpression*0:predict.continuous').d = 0;
```

## 파이썬 코드

```
# 연속으로 나이, 성별, 표정 예측 시작하기  
__('FaceExpression*0:predict.continuous').d = 1  
  
# 연속으로 나이, 성별, 표정 예측 중지하기  
__('FaceExpression*0:predict.continuous').d = 0
```

## 나이, 성별, 표정 찾기 결과 보기

카메라 화면에 나이, 성별, 표정 예측 결과를 표시할지 말지를 결정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	예측 결과 보이기 (1), 숨기기 (0)	

## 자바스크립트 코드

```
// 나이, 성별, 표정 예측 결과 보기  
$('FaceExpression*0:display').d = 1;  
  
// 나이, 성별, 표정 예측 결과 숨기기  
$('FaceExpression*0:display').d = 0;
```

## 파이썬 코드

```
# 나이, 성별, 표정 예측 결과 보기  
__('FaceExpression*0:display').d = 1  
  
# 나이, 성별, 표정 예측 결과 숨기기  
__('FaceExpression*0:display').d = 0
```

## 나이 예측 데이터

나이 예측 결과 데이터를 반환합니다.



나이

## 자바스크립트 코드

```
// 나이 예측 데이터  
$('FaceExpression*0:age').d;
```

## 파이썬 코드

```
# 나이 예측 데이터  
__('FaceExpression*0:age').d
```

## 성별 예측 데이터

성별 예측 결과 데이터를 반환합니다.

반환 데이터는 ‘남성’, ‘여성’입니다.



성별

## 자바스크립트 코드

```
// 성별 예측 데이터  
$('FaceExpression*0:gender.name').d;
```

## 파이썬 코드

```
# 성별 예측 데이터  
__('FaceExpression*0:gender.name').d
```

## 표정 예측 데이터

표정 예측 결과 데이터를 반환합니다.

반환 데이터는 ‘행복’, ‘슬픔’, ‘화남’, ‘두려움’, ‘혐오’, ‘놀람’, ‘무표정’입니다.

 나이, 성별, 표정 0 : 표정

표정

## 자바스크립트 코드

```
// 표정 예측 데이터  
$('FaceExpression*0:expression.name').d;
```

## 파이썬 코드

```
# 표정 예측 데이터  
__('FaceExpression*0:expression.name').d
```

## 성별이 인식되었는가?

성별 예측 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.

 나이, 성별, 표정 0 : 성별이 인식되었는가?

성별 인식 여부

## 자바스크립트 코드

```
// 성별이 인식되었는가?  
$('FaceExpression*0:gender.predicted').d;
```

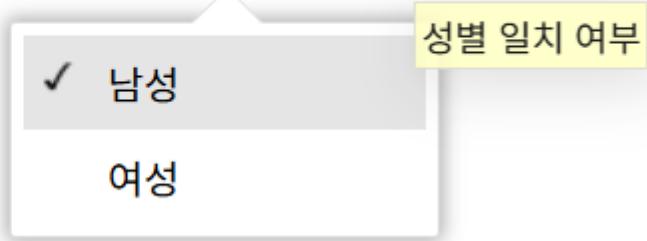
## 파이썬 코드

```
# 성별이 인식되었는가?  
__('FaceExpression*0:gender.predicted').d
```

## 성별이 ~인가?

선택한 성별과 예측 성별의 일치 여부에 따라 참 (1) / 거짓 (0) (으)로 반환합니다.

 나이, 성별, 표정 0 : 성별이 남성 인가?



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
gender	드롭다운 옵션	성별	남성 (male), 여성 (female)

## 자바스크립트 코드

```
// 예측 성별이 남성인가?  
$('FaceExpression*0:gender.name').d == '남성';  
  
// 예측 성별이 여성인가?  
$('FaceExpression*0:gender.name').d == '여성';
```

## 파이썬 코드

```
# 예측 성별이 남성인가?  
__($('FaceExpression*0:gender.name').d == '남성')  
  
# 예측 성별이 여성인가?  
__($('FaceExpression*0:gender.name').d == '여성')
```

## 성별이 ~ 일 확률 (신뢰도)

선택한 성별과 예측 성별의 일치 확률 (신뢰도)을 반환합니다.

반환값은 0 ~ 1 사이 실수입니다.

 나이, 성별, 표정 0 : 성별이 남성 일 확률(신뢰도)

선택한 성별일 확률(신뢰도)

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
gender	드롭다운 옵션	성별	남성 (0), 여성 (1)

## 자바스크립트 코드

```
// 예측 성별이 남성일 확률
$('FaceExpression*0:gender.confidence').d[0];

// 예측 성별이 여성일 확률
$('FaceExpression*0:gender.confidence').d[1];
```

## 파이썬 코드

```
# 예측 성별이 남성일 확률
__('FaceExpression*0:gender.confidence').d[0]

# 예측 성별이 여성일 확률
__('FaceExpression*0:gender.confidence').d[1]
```

## 표정이 인식되었는가?

표정 인식 여부에 따라 참 (1) / 거짓 (0) (으)로 반환합니다.



표정 인식 여부

## 자바스크립트 코드

```
// 표정이 인식되었는가?
$('FaceExpression*0:expression.predicted').d;
```

## 파이썬 코드

```
# 표정이 인식되었는가?
__('FaceExpression*0:expression.predicted').d
```

## 표정이 ~인가?

선택한 표정과 예측 표정의 일치 여부에 따라 참 (1) / 거짓 (0) (으)로 반환합니다.



표정 일치 여부

✓ 행복

슬픔

화남

두려움

혐오

놀람

무표정

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
expression	드롭다운 옵션	표정	행복 (happy), 슬픔 (sad), 화남 (angry), 두려움 (afraid), 혐오 (disgust), 놀람 (surprise), 무표정 (expressionless)

## 자바스크립트 코드

```
// 예측 표정이 행복인가?
$('FaceExpression*0:expression.name').d === '행복';

// 예측 표정이 슬픔인가?
$('FaceExpression*0:expression.name').d === '슬픔';

// 예측 표정이 화남인가?
$('FaceExpression*0:expression.name').d === '화남';

// 예측 표정이 두려움인가?
$('FaceExpression*0:expression.name').d === '두려움';

// 예측 표정이 혐오인가?
$('FaceExpression*0:expression.name').d === '혐오';
```

```
// 예측 표정이 놀람인가?  
$('FaceExpression*0:expression.name').d == '놀람';  
  
// 예측 표정이 무표정인가?  
$('FaceExpression*0:expression.name').d == '무표정';
```

## 파이썬 코드

```
# 예측 표정이 행복인가?  
__($('FaceExpression*0:expression.name').d == '행복')  
  
# 예측 표정이 슬픔인가?  
__($('FaceExpression*0:expression.name').d == '슬픔')  
  
# 예측 표정이 화남인가?  
__($('FaceExpression*0:expression.name').d == '화남')  
  
# 예측 표정이 두려움인가?  
__($('FaceExpression*0:expression.name').d == '두려움')  
  
# 예측 표정이 혐오인가?  
__($('FaceExpression*0:expression.name').d == '혐오')  
  
# 예측 표정이 놀람인가?  
__($('FaceExpression*0:expression.name').d == '놀람')  
  
# 예측 표정이 무표정인가?  
__($('FaceExpression*0:expression.name').d == '무표정')
```

## 표정이 ~ 일 확률 (신뢰도)

선택한 표정과 예측 표정의 일치 확률 (신뢰도) 을 반환합니다.

반환값은 0 ~ 1 사이 실수 입니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
expression	드롭다운 옵션	표정	행복 (0), 슬픔 (1), 화남 (2), 두려움 (3), 혐오 (4), 놀람 (5), 무표정 (6)

## 자바스크립트 코드

```
// 예측 표정이 행복일 확률
$('FaceExpression*0:expression.confidence').d[0];

// 예측 표정이 슬픔일 확률
$('FaceExpression*0:expression.confidence').d[1];

// 예측 표정이 화남일 확률
$('FaceExpression*0:expression.confidence').d[2];

// 예측 표정이 두려움일 확률
$('FaceExpression*0:expression.confidence').d[3];

// 예측 표정이 혐오일 확률
$('FaceExpression*0:expression.confidence').d[4];

// 예측 표정이 놀랄일 확률
$('FaceExpression*0:expression.confidence').d[5];

// 예측 표정이 무표정일 확률
$('FaceExpression*0:expression.confidence').d[6];
```

## 파이썬 코드

```
# 예측 표정이 행복일 확률
__($('FaceExpression*0:expression.confidence').d[0]

# 예측 표정이 슬픔일 확률
__($('FaceExpression*0:expression.confidence').d[1]

# 예측 표정이 화남일 확률
__($('FaceExpression*0:expression.confidence').d[2]

# 예측 표정이 두려움일 확률
__($('FaceExpression*0:expression.confidence').d[3]

# 예측 표정이 혐오일 확률
__($('FaceExpression*0:expression.confidence').d[4]

# 예측 표정이 놀랄일 확률
__($('FaceExpression*0:expression.confidence').d[5]

# 예측 표정이 무표정일 확률
__($('FaceExpression*0:expression.confidence').d[6]
```

## 나이, 성별, 표정 모델 로딩 상태값

나이, 성별, 표정 모델 로딩 상태를 반환합니다.

아직 불러오지 않았다면 0, 불러오는 중이면 1, 불러오기를 완료했다면 2 를 반환합니다.

## ⌚ 나이, 성별, 표정 0 : 나이, 성별, 표정 모델 로딩 상태

나이, 성별, 표정 모델 로딩 상태를 반환합니다.

아직 불러오지 않았으면 0, 불러오는 중이면 1, 불러오기를 완료했으면 2를 반환합니다.

### 자바스크립트 코드

```
// 나이, 성별, 표정 모델 로딩 상태 값  
$('FaceExpression*0:model_state').d;
```

### 파이썬 코드

```
# 나이, 성별, 표정 모델 로딩 상태 값  
__('FaceExpression*0:model_state').d
```

### 논리

**불리언 (boolean)** 은 두 가지 값을 갖는 간단한 수학적 시스템입니다:

- 참
- 거짓

논리 블록은 일반적으로 조건 블록과 반복 블록을 제어하는 데 사용됩니다.

다음은 예시입니다:



Figure 4: Image

변수 **x**의 값이 100 보다 크면 조건은 **참**이 되어 “큰 수입니다.”라는 텍스트가 출력됩니다. 만약 **x**의 값이 100 보다 크지 않으면 조건은 **거짓**이 되어 “크지 않은 수입니다.”가 출력됩니다.

불리언 값은 변수에 저장하거나 함수로 전달할 수 있으며, 숫자, 텍스트, 리스트 값과 동일한 방식으로 사용할 수 있습니다.

## 블록

블록이 불리언 값을 입력으로 요구하면, 입력이 없으면 일반적으로 거짓으로 해석됩니다. 아래에 예시가 제공됩니다. 불리언 값이 기대되는 곳에 비불리언 값을 직접 연결할 수는 없지만, 비불리언 값을 변수에 저장한 뒤 그것을 입력으로 전달하는 것은 가능합니다. 다만, 이는 권장되지 않으며, 향후 버전에서 동작이 달라질 수 있습니다.

## 값

참 또는 거짓을 지정하는 드롭다운이 있는 단일 블록을 사용하여 불리언 값을 얻을 수 있습니다:



Figure 5: Image

## Javascript 코드

```
true // 참  
false // 거짓
```

## Python 코드

```
True # 참  
False # 거짓
```

## 비교

여섯 가지 비교 연산자가 있습니다. 각 연산자는 두 개의 입력 (보통 숫자) 을 받아, 입력값들이 서로 어떻게 비교되는지에 따라 참 또는 거짓을 반환합니다.

여섯 가지 연산자는: 같다, 같지 않다, 작다, 크다, 작거나 같다, 크거나 같다입니다.

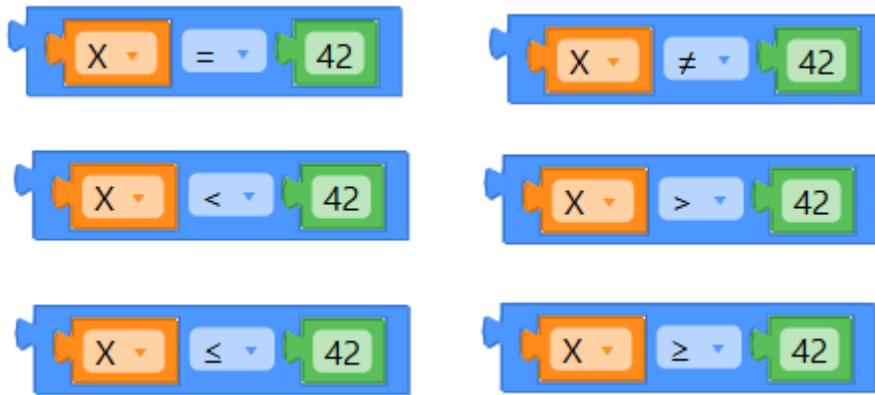


Figure 6: Image

### Javascript 코드

```
x == 42    // 같다
x != 42   // 같지 않다
x < 42    // 작다
x > 42    // 크다
x <= 42   // 작거나 같다
x >= 42   // 크거나 같다
```

### Python 코드

```
x == 42    # 같다
x != 42   # 같지 않다
x < 42    # 작다
x > 42    # 크다
x <= 42   # 작거나 같다
x >= 42   # 크거나 같다
```

### 논리 연산

그리고 블록은 두 입력이 모두 참일 때만 참을 반환합니다.

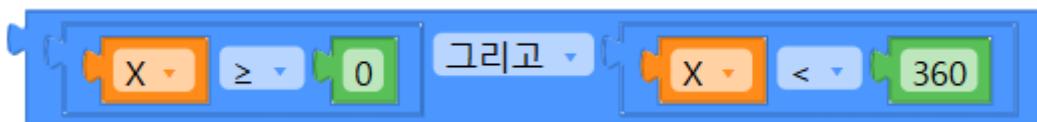


Figure 7: Image

또는 블록은 두 입력 중 하나라도 참이면 참을 반환합니다.

### Javascript 코드

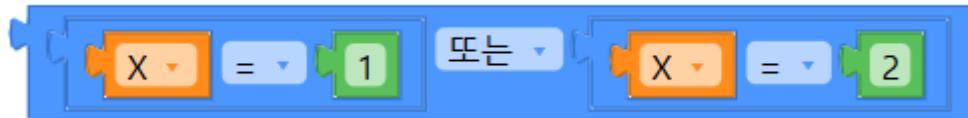


Figure 8: Image

```
x >= 0 && x < 360 // 그리고
x == 1 || x == 2 // 또는
```

## Python 코드

```
x >= 0 and x < 360 # 그리고
x == 1 or x == 2 # 또는
```

**아닙니다**

**아닙니다** 블록은 불리언 입력을 그 반대로 변환합니다. 예를 들어, 다음과 같은 결과는:



Figure 9: Image

**거짓이 됩니다.**

위에서 언급한 바와 같이, 입력이 없으면 **참** 값이 기본값으로 간주되므로, 아래 블록은 **거짓** 값을 반환합니다:

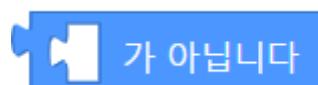


Figure 10: Image

입력을 비워두는 것은 권장되지 않습니다.

## Javascript 코드

```
!true // 또는 false
```

## Python 코드

```
not True # 또는 False
```

## 삼항 연산자 (Ternary operator)

삼항 블록은 간단한 조건문 블록처럼 동작합니다. 세 개의 입력을 받습니다. 첫 번째 입력은 테스트할 불리언 조건이고, 두 번째 입력은 조건이 참일 경우 반환할 값입니다. 세 번째 입력은 조건이 거짓일 경우 반환할 값입니다. 아래 예시에서 **x** 변수가 10 보다 작으면 **color** 변수는 빨간색으로 설정되고, 그렇지 않으면 **color** 변수는 초록색으로 설정됩니다.



Figure 11: Image

삼항 블록은 항상 조건문 블록으로 대체할 수 있습니다. 아래 두 예시는 정확히 동일합니다.



Figure 12: Image

## Javascript 코드

```
x == 0 ? 'Game Over!' : 'Keep Playing'
```

## Python 코드

```
'Game Over' if x == 0 else 'Keep Playing'
```

라쿤

## 블록

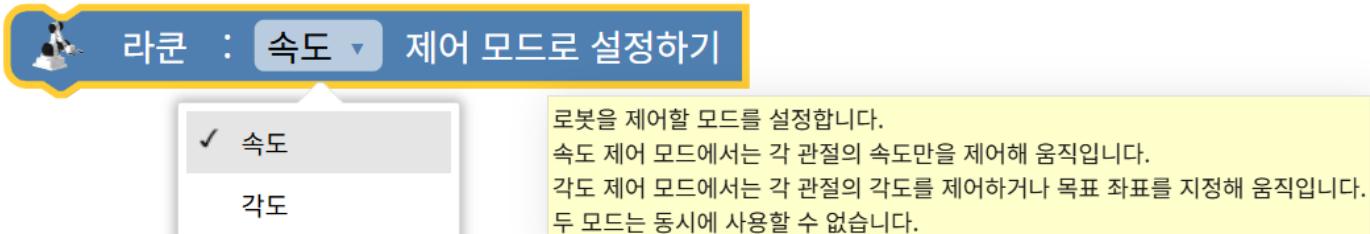
### 제어 모드 설정하기

라쿤을 제어할 모드를 설정합니다.

속도 제어 모드에서는 각 관절의 속도만을 통해 관절을 움직입니다.

각도 제어 모드에서는 각도 제어 속도, 각 관절 각도를 통해 제어하거나 목표 좌표를 통해 움직입니다.

두 모드는 동시에 사용할 수 없습니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
mode	드롭다운 옵션	라쿤 제어 모드	속도 (0), 각도 (1)

### 자바스크립트 코드

```
// 라쿤 속도 제어 모드로 설정하기
$(`'Raccoon4*0:mode').d = 0;

// 라쿤 각도 제어 모드로 설정하기
$(`'Raccoon4*0:mode').d = 1;
$(`'Raccoon4*0:angle.joints').d = [$(`'Raccoon4*0:encoder.joint_1').d, $(`'Raccoon4*0:encoder.joint_2').d, $(`'Raccoon4*0:encoder.joint_3').d, $(`'Raccoon4*0:encoder.joint_4').d];
```

### 파이썬 코드

```
# 라쿤 속도 제어 모드로 설정하기
__(`'Raccoon4*0:mode').d = 0

# 라쿤 각도 제어 모드로 설정하기
__(`'Raccoon4*0:mode').d = 1
__(`'Raccoon4*0:angle.joints').d = [__(`'Raccoon4*0:encoder.joint_1').d, __(`'Raccoon4*0:encoder.joint_2').d, __(`'Raccoon4*0:encoder.joint_3').d, __(`'Raccoon4*0:encoder.joint_4').d]
```

### 주변 장치 설정하기

라쿤과 연결해 사용할 주변 장치를 설정합니다.

따로 설정하지 않아도, 연결된 주변 장치를 자동으로 인식하여 제어할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
peripheral	드롭다운 옵션	라쿤 주변 장치	컨베이어 (1), 슬라이더 (2)

## 자바스크립트 코드

```
// 라쿤 주변 장치를 컨베이어 (으) 로 설정하기
$('Raccoon4*0:peripheral>').d = 1;

// 라쿤 주변 장치를 슬라이더 (으) 로 설정하기
$('Raccoon4*0:peripheral>').d = 2;
```

## 파이썬 코드

```
# 라쿤 주변 장치를 컨베이어 (으) 로 설정하기
__('Raccoon4*0:peripheral>').d = 1

# 라쿤 주변 장치를 슬라이더 (으) 로 설정하기
__('Raccoon4*0:peripheral>').d = 2
```

## 관절 모터 제어 켜기 / 끄기

라쿤 각 관절 모터에 가해져 있는 제어를 풀지 유지할지 결정합니다.

설정하지 않을 경우, 모든 관절의 모터 제어가 켜진 상태로 시작합니다.



각 관절 모터에 가해져 있는 제어를 풀지 유지할지 결정합니다.  
설정하지 않을 경우, 모든 관절의 모터 제어가 켜진 상태로 시작합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	라쿤 관절 모터 번호	1(joint_1), 2(joint_2), 3(joint_3), 4(joint_4), 전 체 (joint_1, joint_2, joint_3, joint_4)
power	드롭다운 옵션	켜기/끄기	켜기 (0), 끄기 (1)

## 자바스크립트 코드

```
// 관절 1 모터 제어 켜기
$(`'Raccoon4*0:motor_off.joint_1').d = 0; // 켜기

// 관절 2 모터 제어 끄기
$(`'Raccoon4*0:motor_off.joint_2').d = 1; // 끄기

// 관절 3 모터 제어 켜기
$(`'Raccoon4*0:motor_off.joint_3').d = 0; // 켜기

// 관절 4 모터 제어 끄기
$(`'Raccoon4*0:motor_off.joint_4').d = 1; // 끄기

// 관절 전체 모터 제어 켜기
$(`'Raccoon4*0:motor_off.joint_1').d = 0; // 켜기
$(`'Raccoon4*0:motor_off.joint_2').d = 0; // 켜기
$(`'Raccoon4*0:motor_off.joint_3').d = 0; // 켜기
$(`'Raccoon4*0:motor_off.joint_4').d = 0; // 켜기
```

## 파이썬 코드

```
# 관절 1 모터 제어 켜기
__(`'Raccoon4*0:motor_off.joint_1').d = 0 # 켜기

# 관절 2 모터 제어 끄기
__(`'Raccoon4*0:motor_off.joint_2').d = 1 # 끄기
```

```

# 관절 3 모터 제어 켜기
__(`Raccoon4*0:motor_off.joint_3`).d = 0 # 켜기

# 관절 4 모터 제어 끄기
__(`Raccoon4*0:motor_off.joint_4`).d = 1 # 끄기

# 관절 전체 모터 제어 켜기
__(`Raccoon4*0:motor_off.joint_1`).d = 0 # 켜기
__(`Raccoon4*0:motor_off.joint_2`).d = 0 # 켜기
__(`Raccoon4*0:motor_off.joint_3`).d = 0 # 켜기
__(`Raccoon4*0:motor_off.joint_4`).d = 0 # 켜기

```

## 관절 속도 설정하기

선택한 관절 속도를 설정합니다.

관절 속도의 범위는 -100 ~ 100입니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	라쿤 관절 모터 번호	1(joint_1), 2(joint_2), 3(joint_3), 4(joint_4)
speed	입력 값	관절 속도	-100 ~ 100 사이 실수

## 자바스크립트 코드

```

// 라쿤 관절 1 속도를 100(으)로 정하기
$(`Raccoon4*0:speed.joint_1`).d = 100;

// 라쿤 관절 2 속도를 90(으)로 정하기
$(`Raccoon4*0:speed.joint_2`).d = 90;

// 라쿤 관절 3 속도를 80(으)로 정하기
$(`Raccoon4*0:speed.joint_3`).d = 80;

// 라쿤 관절 4 속도를 70(으)로 정하기
$(`Raccoon4*0:speed.joint_4`).d = 70;

```

## 파이썬 코드

```
# 라쿤 관절 1 속도를 100(으)로 정하기  
__('Raccoon4*0:speed.joint_1').d = 100  
  
# 라쿤 관절 2 속도를 90(으)로 정하기  
__('Raccoon4*0:speed.joint_2').d = 90  
  
# 라쿤 관절 3 속도를 80(으)로 정하기  
__('Raccoon4*0:speed.joint_3').d = 80  
  
# 라쿤 관절 4 속도를 70(으)로 정하기  
__('Raccoon4*0:speed.joint_4').d = 70
```

## 관절 속도 변경하기

선택한 관절의 속도를 입력값만큼 변경합니다.

입력값의 범위는 -200 ~ 200입니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	라쿤 관절 모터 번호	1(joint_1), 2(joint_2), 3(joint_3), 4(joint_4)
speed	입력 값	변경 속도	-200 ~ 200 사이 실수

## 자바스크립트 코드

```
// 관절 1 속도를 -200 만큼 바꾸기  
$('Raccoon4*0:speed.joint_1').d = $('Raccoon4*0:speed.joint_1').d + -200;  
  
// 관절 2 속도를 40 만큼 바꾸기  
$('Raccoon4*0:speed.joint_2').d = $('Raccoon4*0:speed.joint_2').d + 40;  
  
// 관절 3 속도를 30 만큼 바꾸기  
$('Raccoon4*0:speed.joint_3').d = $('Raccoon4*0:speed.joint_3').d + 30;
```

```
// 관절 4 속도를 20 만큼 바꾸기
$(`Raccoon4*0:speed.joint_4`).d = `(`Raccoon4*0:speed.joint_4`).d + 20;
```

## 파이썬 코드

```
# 관절 1 속도를 -200 만큼 바꾸기
__(`Raccoon4*0:speed.joint_1`).d = __(`Raccoon4*0:speed.joint_1`).d + (-200)

# 관절 2 속도를 40 만큼 바꾸기
__(`Raccoon4*0:speed.joint_2`).d = __(`Raccoon4*0:speed.joint_2`).d + 40

# 관절 3 속도를 30 만큼 바꾸기
__(`Raccoon4*0:speed.joint_3`).d = __(`Raccoon4*0:speed.joint_3`).d + 30

# 관절 4 속도를 20 만큼 바꾸기
__(`Raccoon4*0:speed.joint_4`).d = __(`Raccoon4*0:speed.joint_4`).d + 20
```

## 관절 속도 배열로 설정하기

네 관절의 속도를 한 번에 설정합니다.

각 관절 속도의 범위는 -100 ~ 100입니다.



**라쿤 :** 각 관절의 속도를 **배열 [0, 0, 0, 0]** (으)로 정하기

네 관절의 속도를 한 번에 설정합니다. 각 관절 속도의 범위는 -100 ~ 100입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
speed	입력 값	속도	-100 ~ 100 사이 실수 배열

## 자바스크립트 코드

```
// 각 관절의 속도를 [10, 20, 30, 40] (으)로 정하기
$(`Raccoon4*0:speed.joint_1`).d = 10;
$(`Raccoon4*0:speed.joint_2`).d = 20;
$(`Raccoon4*0:speed.joint_3`).d = 30;
$(`Raccoon4*0:speed.joint_4`).d = 40;
```

## 파이썬 코드

```
# 각 관절의 속도를 [10, 20, 30, 40] (으)로 정하기
__('Raccoon4*0:speed.joint_1').d = 10
__('Raccoon4*0:speed.joint_2').d = 20
__('Raccoon4*0:speed.joint_3').d = 30
__('Raccoon4*0:speed.joint_4').d = 40
```

## 관절 각도 제어 속도 설정하기

각도 제어 모드에서 관절들을 제어할 속도를 설정합니다.

속도의 범위는 0 ~ 100 입니다.



각도 제어 모드에서 관절들을 제어할 속도를 설정합니다. 속도의 범위는 0 ~ 100 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
speed	입력값	관절 제어 속도	0 ~ 100 실수

## 자바스크립트 코드

```
// 관절 각도 제어 속도를 100(으)로 정하기
$('Raccoon4*0:angle.max_speed').d = 100;
```

## 파이썬 코드

```
# 관절 각도 제어 속도를 100(으)로 정하기
__('Raccoon4*0:angle.max_speed').d = 100
```

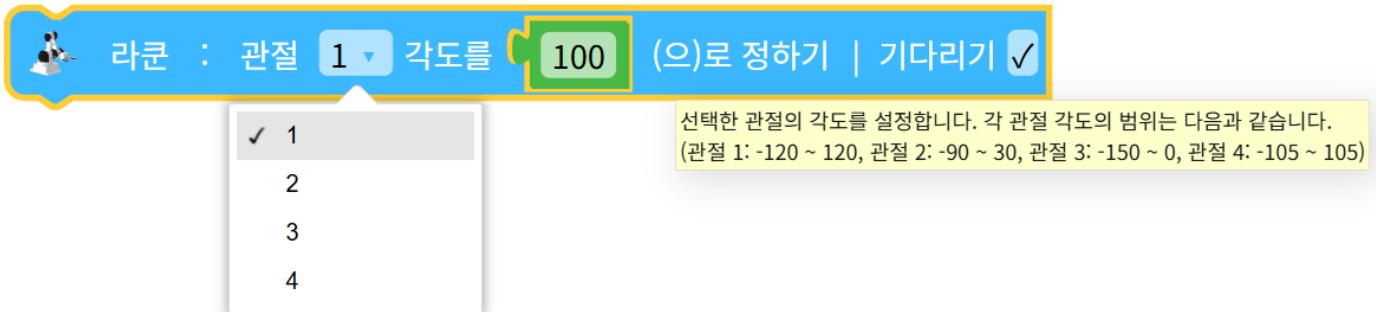
## 관절 각도 설정하기

선택한 관절의 각도를 설정합니다. 각 관절 각도의 범위는 다음과 같습니다.

관절 1 : -120 ~ 120, 관절 2 : -90 ~ 30, 관절 3 : -150 ~ 0, 관절 4 : -105 ~ 105

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	라쿤 관절 모터 번호	1(joint_1), 2(joint_2), 3(joint_3), 4(joint_4)
angle	입력 값	관절 각도	관절 1 : -120 ~ 120, 관절 2 : -90 ~ 30, 관절 3 : -150 ~ 0, 관절 4 : -105 ~ 105

## 자바스크립트 코드

```
// 관절 1 각도를 100 (으)로 정하기 | 기다리기 0
$(`'Raccoon4*0:angle.joints').d = $(`'Raccoon4*0:angle.joints').d.map((data, i) => (i + 1 === 1) ? 100 : Math.floor(data));
await $(`'Raccoon4*0:angle.!joints').w();

// 관절 2 각도를 30 (으)로 정하기 | 기다리기 0
$(`'Raccoon4*0:angle.joints').d = $(`'Raccoon4*0:angle.joints').d.map((data, i) => (i + 1 === 2) ? 30 : Math.floor(data));
await $(`'Raccoon4*0:angle.!joints').w();

// 관절 3 각도를 0 (으)로 정하기 | 기다리기 X
$(`'Raccoon4*0:angle.joints').d = $(`'Raccoon4*0:angle.joints').d.map((data, i) => (i + 1 === 3) ? 0 : Math.floor(data));

// 관절 4 각도를 100 (으)로 정하기 | 기다리기 X
$(`'Raccoon4*0:angle.joints').d = $(`'Raccoon4*0:angle.joints').d.map((data, i) => (i + 1 === 4) ? 100 : Math.floor(data));
```

## 파이썬 코드

```
# 관절 1 각도를 100 (으)로 정하기 | 기다리기 0
__(`'Raccoon4*0:angle.joints').d = [100 if i + 1 == 1 else __(`'Raccoon4*0:angle.joints').d[i] for i in range(4)]
await __(`'Raccoon4*0:angle.!joints').w()

# 관절 2 각도를 30 (으)로 정하기 | 기다리기 0
__(`'Raccoon4*0:angle.joints').d = [30 if i + 1 == 2 else __(`'Raccoon4*0:angle.joints').d[i] for i in range(4)]
await __(`'Raccoon4*0:angle.!joints').w()

# 관절 3 각도를 0 (으)로 정하기 | 기다리기 X
__(`'Raccoon4*0:angle.joints').d = [0 if i + 1 == 3 else __(`'Raccoon4*0:angle.joints').d[i] for i in range(4)]

# 관절 4 각도를 100 (으)로 정하기 | 기다리기 X
__(`'Raccoon4*0:angle.joints').d = [100 if i + 1 == 4 else __(`'Raccoon4*0:angle.joints').d[i] for i in range(4)]
```

## 관절 각도 변경하기

선택한 관절의 각도를 입력값만큼 변경합니다. 각 입력값의 범위는 다음과 같습니다.

관절 1 : -240 ~ 240, 관절 2 : -120 ~ 120, 관절 3 : -150 ~ 0, 관절 4 : -210 ~ 210

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	라쿤 관절 모터 번호	1(joint_1), 2(joint_2), 3(joint_3), 4(joint_4)
angle	입력 값	관절 변경값	관절 1 : -240 ~ 240, 관절 2 : -120 ~ 120, 관절 3 : -150 ~ 0, 관절 4 : -210 ~ 210

## 자바스크립트 코드

```
// 관절 1 각도를 50 만큼 바꾸기 | 기다리기 0
$(`Raccoon4*0:angle.joints`).d = $($(`Raccoon4*0:angle.joints`)).d.map((data, i) => (i + 1 === 1) ? data + 50 : data);
await $($(`Raccoon4*0:angle.!joints`)).w();

// 관절 2 각도를 40 만큼 바꾸기 | 기다리기 0
$(`Raccoon4*0:angle.joints`).d = $($(`Raccoon4*0:angle.joints`)).d.map((data, i) => (i + 1 === 2) ? data + 40 : data);
await $($(`Raccoon4*0:angle.!joints`)).w();

// 관절 3 각도를 30 만큼 바꾸기 | 기다리기 X
$(`Raccoon4*0:angle.joints`).d = $($(`Raccoon4*0:angle.joints`)).d.map((data, i) => (i + 1 === 3) ? data + 30 : data);

// 관절 4 각도를 20 만큼 바꾸기 | 기다리기 X
$(`Raccoon4*0:angle.joints`).d = $($(`Raccoon4*0:angle.joints`)).d.map((data, i) => (i + 1 === 4) ? data + 20 : data);
```

## 파이썬 코드

```
# 관절 1 각도를 50 만큼 바꾸기 | 기다리기 0
__('Raccoon4*0:angle.joints').d = [__(('Raccoon4*0:angle.joints').d[i] + 50 if i + 1 == 1 else __('Raccoon4*0:angle.joints').d[i]) for i in range(4)]
await __('Raccoon4*0:angle.!joints').w()

# 관절 2 각도를 40 만큼 바꾸기 | 기다리기 0
__('Raccoon4*0:angle.joints').d = [__(('Raccoon4*0:angle.joints').d[i] + 40 if i + 1 == 2 else __('Raccoon4*0:angle.joints').d[i]) for i in range(4)]
await __('Raccoon4*0:angle.!joints').w()

# 관절 3 각도를 30 만큼 바꾸기 | 기다리기 X
__('Raccoon4*0:angle.joints').d = [__(('Raccoon4*0:angle.joints').d[i] + 30 if i + 1 == 3 else __('Raccoon4*0:angle.joints').d[i]) for i in range(4)]

# 관절 4 각도를 20 만큼 바꾸기 | 기다리기 X
__('Raccoon4*0:angle.joints').d = [__(('Raccoon4*0:angle.joints').d[i] + 20 if i + 1 == 4 else __('Raccoon4*0:angle.joints').d[i]) for i in range(4)]
```

## 관절 각도 배열로 설정하기

네 관절의 각도를 한 번에 설정합니다. 각 관절 각도의 범위는 다음과 같습니다.

관절 1 : -120 ~ 120, 관절 2 : -90 ~ 30, 관절 3 : -150 ~ 0, 관절 4 : -105 ~ 105

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



라쿤 : 각 관절의 각도를 **배열 [0, 0, 0, 0]** (으)로 정하기 | 기다리기 ✓

네 관절의 각도를 한 번에 설정합니다. 각 관절 각도의 범위는 다음과 같습니다.  
(관절 1: -120 ~ 120, 관절 2: -90 ~ 30, 관절 3: -150 ~ 0, 관절 4: -105 ~ 105)

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
angle	입력 값	각도	관절 1 : -120 ~ 120, 관절 2 : -90 ~ 30, 관절 3 : -150 ~ 0, 관절 4 : -105 ~ 105

## 자바스크립트 코드

```
// 각 관절의 각도를 [0, 0, 0, 0] (으)로 정하기 | 기다리기 X
$('Raccoon4*0:angle.joints').d = [0,0,0,0];

// 각 관절의 각도를 [10, 10, -10, 10] (으)로 정하기 | 기다리기 0
$('Raccoon4*0:angle.joints').d = [10,10,-10,10];
await $('Raccoon4*0:angle.!joints').w();
```

## 파이썬 코드

```
# 각 관절의 각도를 [0, 0, 0, 0] (으)로 정하기 | 기다리기 X
__('Raccoon4*0:angle.joints').d = [0,0,0,0]

# 각 관절의 각도를 [10, 10, -10, 10] (으)로 정하기 | 기다리기 0
__('Raccoon4*0:angle.joints').d = [10,10,-10,10]
await __('Raccoon4*0:angle.!joints').w()
```

## XYZ 좌표로 이동하기

로봇 팔을 이동시킬 x, y, z 좌표를 설정합니다.

각 좌표의 범위는 다음과 같습니다.

손목 기준 => x : -200mm ~ 200mm, y : -100mm ~ 200mm, z : -20mm ~ 280mm

말단 장치 기준 => x : -280mm ~ 280mm, y : -180mm ~ 280mm, z : -100mm ~ 360mm

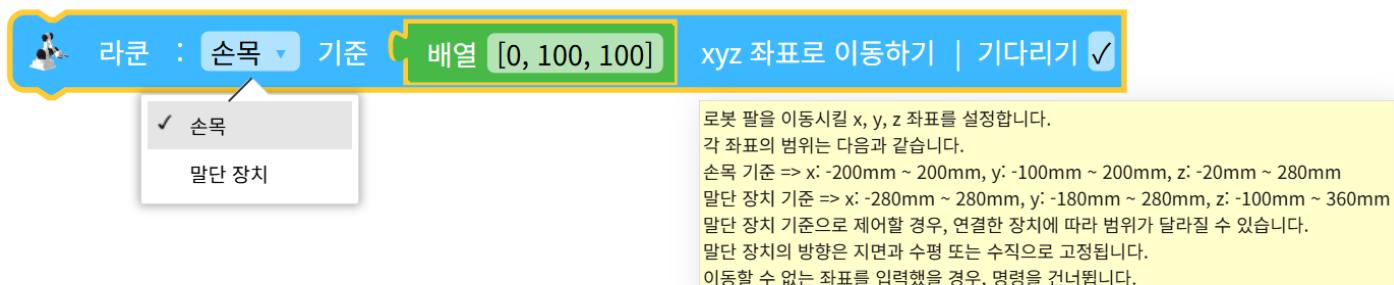
말단 장치 기준으로 제어할 경우, 연결된 장치에 따라 범위가 달라질 수 있습니다.

말단 장치의 방향은 지면과 수평 혹은 수직으로 고정됩니다.

이동할 수 없는 좌표를 입력했을 경우, 명령을 건너뜁니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pivot	드롭다운 옵션	좌표 기준	말단장치 (end_effector), 손목 (wrist)
coordinate	입력 값	이동 좌표 x,y,z	실수 배열

## 자바스크립트 코드

```
// 말단장치 기준 [0, 100, 100] x,y,z 좌표로 이동하기 | 기다리기 0
$(`Raccoon4*0:angle.joints').d = __xyz_to_angles(`Raccoon4*0`, [0,100,100], 'end_effector');
await ${`Raccoon4*0:angle.!joints`).w();

// 손목 기준 [0, 100, 100] x,y,z 좌표로 이동하기 | 기다리기 X
$(`Raccoon4*0:angle.joints').d = __xyz_to_angles(`Raccoon4*0`, [0,100,100], 'wrist');
```

## 파이썬 코드

```
# 말단장치 기준 [0, 100, 100] x,y,z 좌표로 이동하기 | 기다리기 0
__('Raccoon4*0:angle.joints').d = __xyz_to_angles('Raccoon4*0', [0,100,100], 'end_effector')
await __('Raccoon4*0:angle.!joints').w()

# 손목 기준 [0, 100, 100] x,y,z 좌표로 이동하기 | 기다리기 X
__('Raccoon4*0:angle.joints').d = __xyz_to_angles('Raccoon4*0', [0,100,100], 'wrist')
```

## 관절 상태 초기화 하기

네 관절의 각도를 선택한 기본 옵션으로 설정합니다.

기본 옵션의 각도는 다음과 같습니다.

zero : [0,0,0,0], park : [0,25,-145,-60], home : [0,-10,-140,60]

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
option	드롭다운 옵션	기본 옵션	zero, park, home

## 자바스크립트 코드

```
// 관절 상태 zero(으)로 초기화하기 | 기다리기 0
$(`'Raccoon4*0:angle.joints').d = [0, 0, 0, 0];
await $($(`'Raccoon4*0:angle.!joints')).w();

// 관절 상태 park(으)로 초기화하기 | 기다리기 X
$(`'Raccoon4*0:angle.joints').d = [0, 25, -145, -60];

// 관절 상태 home(으)로 초기화하기 | 기다리기 X
$(`'Raccoon4*0:angle.joints').d = [0, -10, -140, 60];
```

## 파이썬 코드

```
# 관절 상태 zero(으) 로 초기화하기 | 기다리기 0
__('Raccoon4*:angle.joints').d = [0, 0, 0]
await __('Raccoon4*:angle.!joints').w()

# 관절 상태 park(으) 로 초기화하기 | 기다리기 X
__('Raccoon4*:angle.joints').d = [0, 25, -145, -60]

# 관절 상태 home(으) 로 초기화하기 | 기다리기 X
__('Raccoon4*:angle.joints').d = [0, -10, -140, 60]
```

## 관절 각도 저장하기

저장하기 버튼을 눌러 지정한 변수에 현재 관절 각도 값을 저장합니다.

또한, 현재 관절 각도 값을 배열로 동시에 받습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
variable	입력값	관절 값을 받을 변수	Block composer 변수

## 자바스크립트 코드

```
// 현재 각도 값 [0,-10,-140,60] 일 때, 저장하기 클릭 시
var encoder;
encoder = [0,-10,-140,60];
```

## 파이썬 코드

```
# 현재 각도 값 [0,-10,-140,60] 일 때, 저장하기 클릭 시
encoder = None
encoder = [0,-10,-140,60]
```

## 말단 장치 고정 설정하기

말단 장치를 고정할 방향을 설정합니다.

옵션이 수평 혹은 수직으로 설정된 이후로는, 관절 4의 속도나 각도를 제어할 수 없습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
fix	드롭다운 옵션	고정 설정값	안함 (0), 수평 (1), 수직 (2)

## 자바스크립트 코드

```
// 말단 장치 고정을 수평 (으)로 설정하기
$('Raccoon4*0:end_effector.lock').d = 1;

// 말단 장치 고정을 수직 (으)로 설정하기
$('Raccoon4*0:end_effector.lock').d = 2;

// 말단 장치 고정을 안함 (으)로 설정하기
$('Raccoon4*0:end_effector.lock').d = 0;
```

## 파이썬 코드

```
# 말단 장치 고정을 수평 (으)로 설정하기
__('Raccoon4*0:end_effector.lock').d = 1

# 말단 장치 고정을 수직 (으)로 설정하기
__('Raccoon4*0:end_effector.lock').d = 2

# 말단 장치 고정을 안함 (으)로 설정하기
__('Raccoon4*0:end_effector.lock').d = 0
```

## 말단 장치로 사물 잡기 / 놓기

말단 장치를 이용하여 물건을 잡거나 놓습니다.



말단 장치(End Effector)를 이용해 물건을 잡거나 놓습니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	말단 장치 사용 여부	잡기 (1), 놓기 (0)

## 자바스크립트 코드

```
// 말단 장치로 사물 잡기  
$('Raccoon4*0:end_effector.control').d = 1;  
  
// 말단 장치로 사물 놓기  
$('Raccoon4*0:end_effector.control').d = 0;
```

## 파이썬 코드

```
# 말단 장치로 사물 잡기  
__('Raccoon4*0:end_effector.control').d = 1  
  
# 말단 장치로 사물 놓기  
__('Raccoon4*0:end_effector.control').d = 0
```

## 말단 장치 값

현재 연결되어 있는 말단 장치 정보에 따라 숫자를 반환합니다.

통합 그리퍼 일 시 1, 진동 그리퍼 일 시 2, 서보 그리퍼 일 시 3, DC 그리퍼 일 시 4 를 반환합니다.



현재 연결되어 있는 말단 장치 번호  
(1, 3, 4: 집게 그리퍼, 2: 진공 그리퍼)

## 자바스크립트 코드

```
$( 'Raccoon4*0:end_effector.device' ).d; // 말단 장치 정보 값
```

## 파이썬 코드

```
__('Raccoon4*0:end_effector.device').d # 말단 장치 정보 값
```

## 말단 장치 상태

말단 장치가 사물을 잡고 있는 상태에 따라 숫자를 반환합니다.

놓은 상태일 시 0, 잡은 상태일 시 1 을 반환합니다.



### 라쿤 : 말단 장치 상태

말단 장치가 사물을 잡고 있는 상태  
(0: 놓은 상태, 1: 잡은 상태)

## 자바스크립트 코드

```
$(('Raccoon4*0:end_effector.status').d; // 말단 장치 상태 값
```

## 파이썬 코드

```
__($('Raccoon4*0:end_effector.status').d # 말단 장치 상태 값
```

## 음계 연주하기

라쿤이 지정된 음계를 재생합니다.



라쿤 :

도 ▾

1 ▾

음을 연주하기



라쿤 :

시 ▾

4 ▾

음을 연주하기

특정 음계를 재생합니다.

- 1
- 2
- 3
- 4
- 5
- 6
- 7

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
note	드롭다운 옵션	음계	도 (Do), 도 #(Do#), 레 (Re), 레 #(Re#), 미 (Mi), 파 (Fa), 파 #(Fa#), 솔 (So), 솔 #(So#), 라 (La), 라 #(La#), 시 (Ti)
octave	드롭다운 옵션	옥타브	1 ~ 7

### 자바스크립트 코드

```
// 1 옥타브 도 (Do) 음을 연주하기
$(`'Raccoon4*0:sound.note'`).d = 4;

// 1 옥타브 레 (Re) 음을 연주하기
$(`'Raccoon4*0:sound.note'`).d = 6;
```

```
// 2 옥타브 도 (Do) 음을 연주하기
$(`'Raccoon4*0:sound.note'`).d = 16;

// 7 옥타브 시 (Ti) 음을 연주하기
$(`'Raccoon4*0:sound.note'`).d = 87;
```

## 파이썬 코드

```
# 1 옥타브 도 (Do) 음을 연주하기
__(`'Raccoon4*0:sound.note'`).d = 4

# 1 옥타브 레 (Re) 음을 연주하기
__(`'Raccoon4*0:sound.note'`).d = 6

# 2 옥타브 도 (Do) 음을 연주하기
__(`'Raccoon4*0:sound.note'`).d = 16

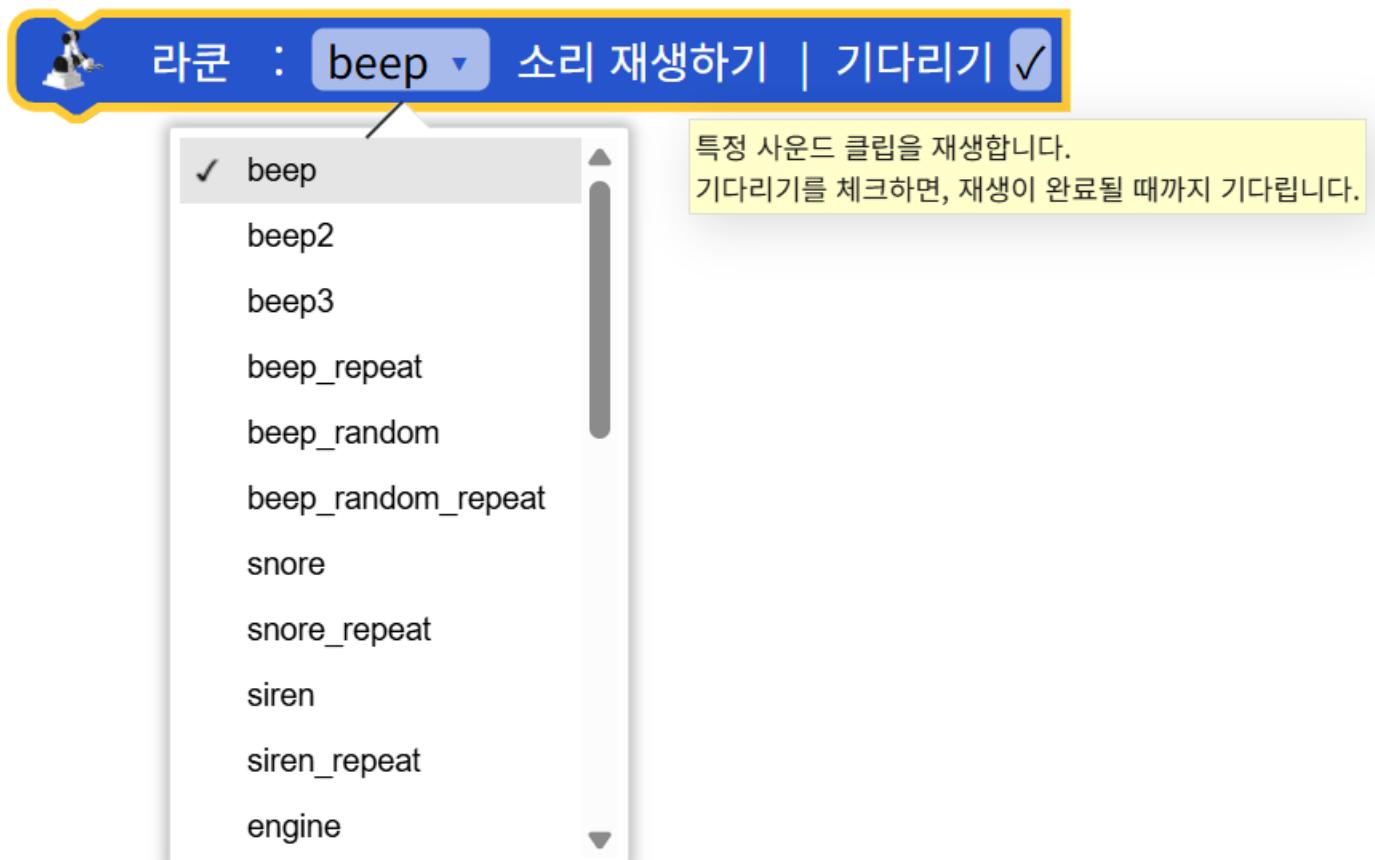
# 7 옥타브 시 (Ti) 음을 연주하기
__(`'Raccoon4*0:sound.note'`).d = 87
```

## 소리 재생하기

라쿤이 특정 사운드 클립을 재생합니다.

기다리기를 체크하면, 재생이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound_clip	드롭다운 옵션	사운드 클립	beep(1), beep2(2), beep3(3), beep_repeat(4), beep_random(5), beep_random_repeat(6), snore(7), snore_repeat(8), siren(9), siren_repeat(10), engine(11), engine_repeat(12), fart_a(13), fart_b(14), noise(15), noise_repeat(16), whistle(17), chop_chop(18), chop_chop_repeat(19), random(20), r2d2(21), connected(22), dee_bee(33), finish(34), power_on(35), start(36), power_off(37)

## 자바스크립트 코드

```
// finish(34) 소리 재생하기 | 기다리기 0
$('Raccoon4*0:sound.clip').d = 34;
await $('Raccoon4*0:sound.!clip').w();

// power_on(35) 소리 재생하기 | 기다리기 X
$('Raccoon4*0:sound.clip').d = 35;
```

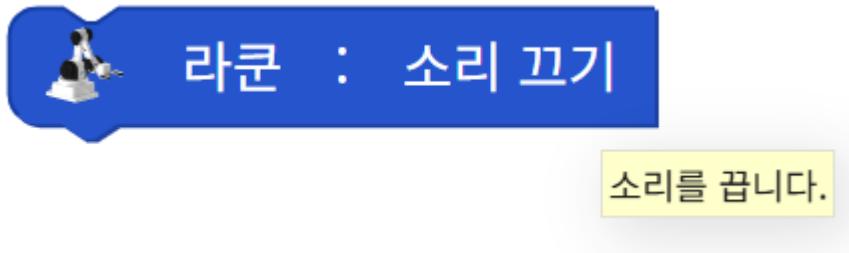
## 파이썬 코드

```
# finish(34) 소리 재생하기 | 기다리기 0
__('Raccoon4*0:sound.clip').d = 34
await __('Raccoon4*0:sound.!clip').w()

# power_on(35) 소리 재생하기 | 기다리기 X
__('Raccoon4*0:sound.clip').d = 35
```

## 소리 끄기

라쿤의 소리를 끕니다.



## 자바스크립트 코드

```
// 라쿤 소리 끄기
__stopSound('Raccoon4*0');
```

## 파이썬 코드

```
# 라쿤 소리 끄기
__stopSound('Raccoon4*0')
```

## 관절 엔코더 값

선택한 관절의 엔코더 값을 반환합니다.



## 라쿤 : 관절 1 엔코더 값

1 ▾

✓ 1

2

3

4

전체

선택한 관절의 엔코더 값

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	라쿤 관절 모터 번호	1(joint_1), 2(joint_2), 3(joint_3), 4(joint_4), 전체 (joint_1, joint_2, joint_3, joint_4)

### 자바스크립트 코드

```
// 관절 1 엔코더 값
$('Raccoon4*0:encoder.joint_1').d;

// 관절 2 엔코더 값
$('Raccoon4*0:encoder.joint_2').d;

// 관절 3 엔코더 값
$('Raccoon4*0:encoder.joint_3').d;

// 관절 4 엔코더 값
$('Raccoon4*0:encoder.joint_4').d;

// 관절 전체 엔코더 값
[$('Raccoon4*0:encoder.joint_1').d, $('Raccoon4*0:encoder.joint_2').d, $('Raccoon4*0:encoder.joint_3').d, $('Raccoon4*0:encoder.joint_4').d];
```

### 파이썬 코드

```
# 관절 1 엔코더 값
__($('Raccoon4*0:encoder.joint_1').d

# 관절 2 엔코더 값
```

```

__('Raccoon4*0:encoder.joint_2').d
# 관절 3 엔코더 값
__('Raccoon4*0:encoder.joint_3').d
# 관절 4 엔코더 값
__('Raccoon4*0:encoder.joint_4').d

# 관절 전체 엔코더 값
[__('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d]

```

## 선택한 요소의 좌표

선택한 요소의 현재 좌표를 반환합니다.

연결된 말단 장치가 없을 경우, 손목 기준 위치가 반환됩니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pivot	드롭다운 옵션	선택 요소	손목 (wrist), 말단 장치 (end_effector)
coordinate	드롭다운 옵션	좌표 축	xyz, x, y, z

## 자바스크립트 코드

```

// 손목 기준 xyz 좌표
__angles_to_xyz('Raccoon4*0', [__('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist');

// 손목 기준 x 좌표
__angles_to_xyz('Raccoon4*0', [__('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist')[0];

// 손목 기준 y 좌표
__angles_to_xyz('Raccoon4*0', [__('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist')[1];

// 손목 기준 z 좌표
__angles_to_xyz('Raccoon4*0', [__('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist')[2];

// 말단 장치 기준 xyz 좌표
__angles_to_xyz('Raccoon4*0', [__('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'end_effector');

```

```

// 말단 장치 기준 x 좌표
__angles_to_xyz('Raccoon4*0', [$(('Raccoon4*0:encoder.joint_1').d, $(('Raccoon4*0:encoder.joint_2').d, $(('Raccoon4*0:encoder.joint_3').d, $(('Raccoon4*0:encoder.joint_4').d], 'end_effector')[0];

// 말단 장치 기준 y 좌표
__angles_to_xyz('Raccoon4*0', [$(('Raccoon4*0:encoder.joint_1').d, $(('Raccoon4*0:encoder.joint_2').d, $(('Raccoon4*0:encoder.joint_3').d, $(('Raccoon4*0:encoder.joint_4').d], 'end_effector')[1];

// 말단 장치 기준 z 좌표
__angles_to_xyz('Raccoon4*0', [$(('Raccoon4*0:encoder.joint_1').d, $(('Raccoon4*0:encoder.joint_2').d, $(('Raccoon4*0:encoder.joint_3').d, $(('Raccoon4*0:encoder.joint_4').d], 'end_effector')[2];

```

## 파이썬 코드

```

# 손목 기준 xyz 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist')

# 손목 기준 y 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist')[0]

# 손목 기준 y 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist')[1]

# 손목 기준 z 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'wrist')[2]

# 말단 장치 기준 xyz 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'end_effector')

# 말단 장치 기준 x 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'end_effector')[0]

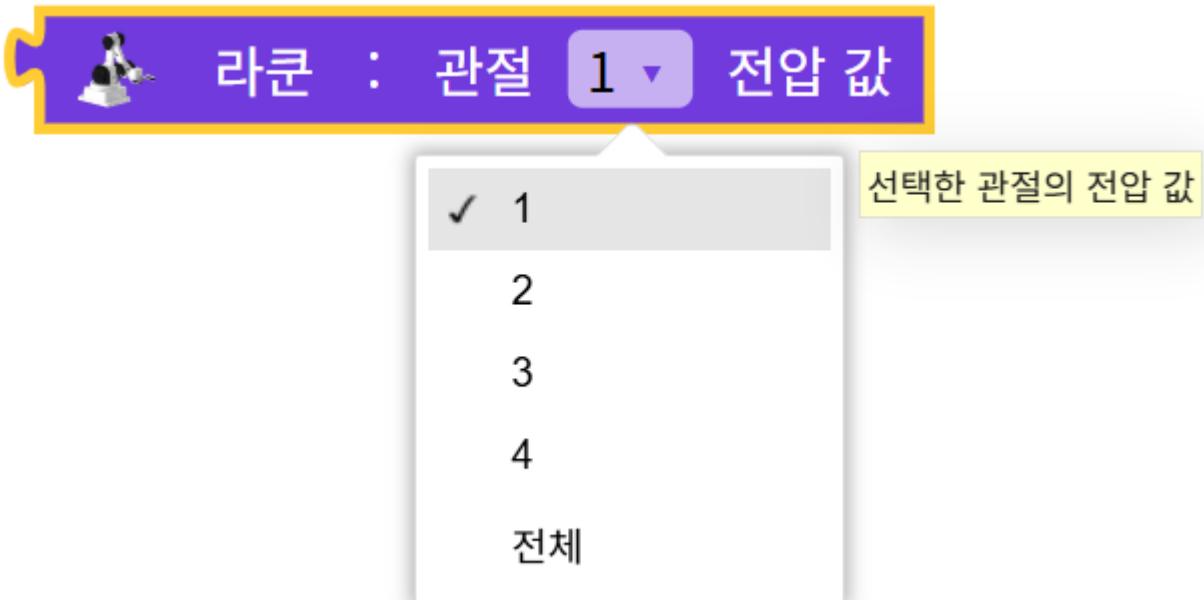
# 말단 장치 기준 y 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'end_effector')[1]

# 말단 장치 기준 z 좌표
__angles_to_xyz('Raccoon4*0', [__(('Raccoon4*0:encoder.joint_1').d, __('Raccoon4*0:encoder.joint_2').d, __('Raccoon4*0:encoder.joint_3').d, __('Raccoon4*0:encoder.joint_4').d], 'end_effector')[2]

```

## 관절의 전압 값

선택한 관절의 전압 값을 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	라쿤 관절 모터 번호	1(joint_1), 2(joint_2), 3(joint_3), 4(joint_4), 전체 (joint_1, joint_2, joint_3, joint_4)

## 자바스크립트 코드

```
// 관절 1 전압 값
$(`'Raccoon4*0:voltage.joint_1').d;

// 관절 2 전압 값
$(`'Raccoon4*0:voltage.joint_2').d;

// 관절 3 전압 값
$(`'Raccoon4*0:voltage.joint_3').d;

// 관절 4 전압 값
$(`'Raccoon4*0:voltage.joint_4').d;

// 관절 전체 전압 값
[$(`'Raccoon4*0:voltage.joint_1').d, $(`'Raccoon4*0:voltage.joint_2').d, $(`'Raccoon4*0:voltage.joint_3').d, $(`'Raccoon4*0:voltage.joint_4').d];
```

## 파이썬 코드

```
# 관절 1 전압 값
__(`'Raccoon4*0:voltage.joint_1').d

# 관절 2 전압 값
__(`'Raccoon4*0:voltage.joint_2').d

# 관절 3 전압 값
__(`'Raccoon4*0:voltage.joint_3').d

# 관절 4 전압 값
__(`'Raccoon4*0:voltage.joint_4').d

# 관절 전체 전압 값
[__(`'Raccoon4*0:voltage.joint_1').d, __(`'Raccoon4*0:voltage.joint_2').d, __(`'Raccoon4*0:voltage.joint_3').d, __(`'Raccoon4*0:voltage.joint_4').d]
```

## 신호 세기 값

라쿤의 신호 세기를 반환합니다.



## 자바스크립트 코드

```
// 라쿤 신호 세기  
$('Raccoon4*0:signal_strength').d;
```

## 파이썬 코드

```
# 라쿤 신호 세기  
__('Raccoon4*0:signal_strength').d
```

## 배터리 충전 상태 값

라쿤의 배터리 충전 상태 값을 반환합니다.



배터리의 충전 상태 값

## 자바스크립트 코드

```
// 라쿤 배터리 충전 상태 값  
$('Raccoon4*0:battery.level').d;
```

## 파이썬 코드

```
# 라쿤 배터리 충전 상태 값  
__('Raccoon4*0:battery.level').d
```

## 버튼이 눌려있는가?

선택한 버튼이 눌려있는지 또는 클릭 이벤트가 발생했는지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.

단, power 버튼을 길게 클릭시에는 전원 ON/OFF 이므로 길게 클릭했는지 여부를 확인할 수 없습니다.



✓ 이 눌려있는가

선택한 버튼이 눌려있는지 또는 클릭 이벤트가 발생했는지 여부

을 클릭했는가

을 길게 클릭했는가

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
button	드롭다운 옵션	버튼 종류	pick, play, power, erase
state	드롭다운 옵션	클릭 이벤트 눌려있는가 (pressed), 클릭했는가 (click), 길게 클릭했는가 (long_click)	

## 자바스크립트 코드

```
// pick 버튼이 눌려있는가?
$(Raccoon4*0:button.pressed.pick').d;
// pick 버튼을 클릭했는가?
$(Raccoon4*0:button.click.pick').e;
// pick 버튼을 길게 클릭했는가?
$(Raccoon4*0:button.long_click.pick').e;

// play 버튼이 눌려있는가?
$(Raccoon4*0:button.pressed.play').d;
// play 버튼을 클릭했는가?
$(Raccoon4*0:button.click.play').e;
// play 버튼을 길게 클릭했는가?
$(Raccoon4*0:button.long_click.play').e;

// power 버튼이 눌려있는가?
$(Raccoon4*0:button.pressed.power').d;
// power 버튼을 클릭했는가?
$(Raccoon4*0:button.click.power').e;
// power 버튼을 길게 클릭했는가? -> X

// erase 버튼이 눌려있는가?
$(Raccoon4*0:button.pressed.erase').d;
// erase 버튼을 클릭했는가?
$(Raccoon4*0:button.click.erase').e;
// erase 버튼을 길게 클릭했는가?
$(Raccoon4*0:button.long_click.erase').e;
```

## 파이썬 코드

```
# pick 버튼이 눌려있는가?
___(Raccoon4*0:button.pressed.pick').d
# pick 버튼을 클릭했는가?
___(Raccoon4*0:button.click.pick').e
```

```

# pick 버튼을 길게 클릭했는가?
__(`Raccoon4*0:button.long_click.pick`).e

# play 버튼이 눌려있는가?
__(`Raccoon4*0:button.pressed.play`).d
# play 버튼을 클릭했는가?
__(`Raccoon4*0:button.click.play`).e
# play 버튼을 길게 클릭했는가?
__(`Raccoon4*0:button.long_click.play`).e

# power 버튼이 눌려있는가?
__(`Raccoon4*0:button.pressed.power`).d
# power 버튼을 클릭했는가?
__(`Raccoon4*0:button.click.power`).e

# erase 버튼이 눌려있는가?
__(`Raccoon4*0:button.pressed.erase`).d
# erase 버튼을 클릭했는가?
__(`Raccoon4*0:button.click.erase`).e
# erase 버튼을 길게 클릭했는가?
__(`Raccoon4*0:button.long_click.erase`).e

```

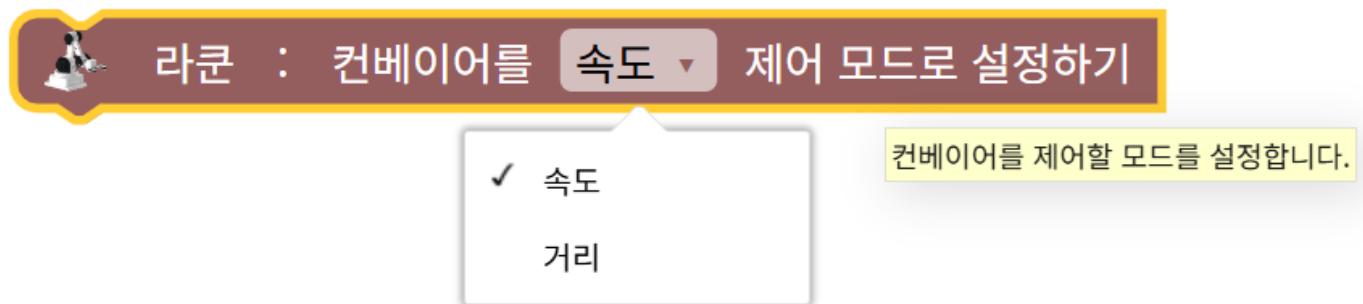
## 컨베이어 모드 설정하기

컨베이어를 제어할 모드를 설정합니다.

속도 제어 모드에서는 컨베이어의 속도만을 통해 컨베이어를 움직입니다.

거리 제어 모드에서는 컨베이어의 이동 거리만을 통해 컨베이어를 움직입니다.

두 모드는 동시에 사용할 수 없습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
mode	드롭다운 옵션	컨베이어 모드	속도 (0), 거리 (1)

## 자바스크립트 코드

```

// 컨베이어를 속도 제어 모드로 설정하기
$(`Raccoon4*0:peripheral>conveyor+mode`).d = 0;

// 컨베이어를 거리 제어 모드로 설정하기
$(`Raccoon4*0:peripheral>conveyor+mode`).d = 1;

```

## 파이썬 코드

```
# 컨베이어를 속도 제어 모드로 설정하기  
__('Raccoon4*0:peripheral>conveyor+mode').d = 0  
  
# 컨베이어를 거리 제어 모드로 설정하기  
__('Raccoon4*0:peripheral>conveyor+mode').d = 1
```

## 컨베이어 속도 설정하기

컨베이어 벨트의 속도를 설정합니다.

속도의 범위는 -100 ~ 100입니다.



컨베이어 벨트의 속도를 설정합니다. 속도의 범위는 -100 ~ 100입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
speed	입력값	컨베이어 속도	-100 ~ 100 사이 실수

## 자바스크립트 코드

```
// 컨베이어 속도를 100(으)로 정하기  
$('Raccoon4*0:peripheral>conveyor+speed').d = 100;
```

## 파이썬 코드

```
# 컨베이어 속도를 100(으)로 정하기  
__('Raccoon4*0:peripheral>conveyor+speed').d = 100
```

## 컨베이어 이동 거리 설정하기

컨베이어 벨트를 이동시킬 거리를 설정합니다.

이 함수는 거리 제어 모드에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
distance	입력값	거리	0 이상 실수
unit	드롭다운 옵션	거리 단위	cm, mm, 인치 (inch)

## 자바스크립트 코드

```
// 컨베이어를 100cm 이동시키기 | 기다리기 0
$(`'Raccoon4*0:peripheral>conveyor+distance').d = __getDistance('Conveyor', 100, 'cm');
await ${`'Raccoon4*0:peripheral>conveyor+!distance'}.w();

// 컨베이어를 100mm 이동시키기 | 기다리기 0
$(`'Raccoon4*0:peripheral>conveyor+distance').d = __getDistance('Conveyor', 100, 'mm');
await ${`'Raccoon4*0:peripheral>conveyor+!distance'}.w();

// 컨베이어를 100 인치 이동시키기 | 기다리기 X
$(`'Raccoon4*0:peripheral>conveyor+distance').d = __getDistance('Conveyor', 100, 'inch');
```

## 파이썬 코드

```
# 컨베이어를 100cm 이동시키기 | 기다리기 0
__(`'Raccoon4*0:peripheral>conveyor+distance').d = __getDistance('Conveyor', 100, 'cm')
await __(`'Raccoon4*0:peripheral>conveyor+!distance'`.w)

# 컨베이어를 100mm 이동시키기 | 기다리기 0
__(`'Raccoon4*0:peripheral>conveyor+distance').d = __getDistance('Conveyor', 100, 'mm')
await __(`'Raccoon4*0:peripheral>conveyor+!distance'`.w)

# 컨베이어를 100 인치 이동시키기 | 기다리기 X
__(`'Raccoon4*0:peripheral>conveyor+distance').d = __getDistance('Conveyor', 100, 'inch')
```

## 컨베이어 속도 변경하기

컨베이어 벨트의 속도를 입력값만큼 변경합니다.



라쿤 : 컨베이어 속도를

50

만큼 바꾸기

컨베이어 벨트의 속도를 변경합니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
speed	입력 값	변경 속도	-200 ~ 200 사이 실수

### 자바스크립트 코드

```
// 컨베이어 속도를 -200 만큼 바꾸기
$('.Raccoon4*0:peripheral>conveyor+speed').d = $('.Raccoon4*0:peripheral>conveyor+speed').d + -200;

// 컨베이어 속도를 200 만큼 바꾸기
$('.Raccoon4*0:peripheral>conveyor+speed').d = $('.Raccoon4*0:peripheral>conveyor+speed').d + 200;
```

### 파이썬 코드

```
# 컨베이어 속도를 -200 만큼 바꾸기
___.('Raccoon4*0:peripheral>conveyor+speed').d = ___.('Raccoon4*0:peripheral>conveyor+speed').d + -200

# 컨베이어 속도를 200 만큼 바꾸기
___.('Raccoon4*0:peripheral>conveyor+speed').d = ___.('Raccoon4*0:peripheral>conveyor+speed').d + 200
```

### 컨베이어가 동작 중인가?

컨베이어 벨트의 동작 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



라쿤 : 컨베이어가 동작 중인가?

컨베이어 벨트 동작 여부

## 자바스크립트 코드

```
// 컨베이어가 동작 중인가?  
$('Raccoon4*0:peripheral>conveyor+running').d;
```

## 파이썬 코드

```
# 컨베이어가 동작 중인가?  
__('Raccoon4*0:peripheral>conveyor+running').d
```

## 컨베이어 버튼이 눌려있는가?

컨베이어의 버튼이 눌려 있는지 또는 클릭 이벤트가 발생했는지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
state	드롭다운 옵션	클릭 이벤트 눌려있는가 (pressed), 클 릭했는가 (click), 길게 클릭 했는가 (long_click)	눌려있는가 (pressed), 클 릭했는가 (click), 길게 클릭 했는가 (long_click)

## 자바스크립트 코드

```
// 컨베이어 버튼이 눌려있는가?  
$('Raccoon4*0:peripheral>button.pressed').d;  
  
// 컨베이어 버튼을 클릭했는가?  
$('Raccoon4*0:peripheral>button.click').e;  
  
// 컨베이어 버튼을 길게 클릭했는가?  
$('Raccoon4*0:peripheral>button.long_click').e;
```

## 파이썬 코드

```
# 컨베이어 버튼이 눌려있는가?  
__('Raccoon4*0:peripheral>conveyor+button.pressed').d  
  
# 컨베이어 버튼을 클릭했는가?  
__('Raccoon4*0:peripheral>conveyor+button.click').e  
  
# 컨베이어 버튼을 길게 클릭했는가?  
__('Raccoon4*0:peripheral>conveyor+button.long_click').e
```

## 리스트

일상적인 말처럼, 블록 컴포저에서 리스트는 “할 일 목록”이나 “쇼핑 목록”처럼 항목들이 순서대로 나열된 집합입니다. 리스트의 항목들은 어떤 타입이라도 될 수 있으며, 동일한 값이 리스트에 여러 번 나타날 수 있습니다.

## 리스트 생성

### 빈 리스트 생성

가장 간단한 리스트는 빈 리스트로, **빈 리스트 생성** 블록을 사용하여 생성합니다:



Figure 13: Image

### Javascript 코드

```
[]; // 빈 리스트
```

### Python 코드

```
[] # 빈 리스트
```

## 리스트 만들기

### 기본 사용법

**리스트 만들기** 블록을 사용하면 새로운 리스트에 초기값을 지정할 수 있습니다. 예를 들어, 단어 리스트를 만들고 이를 **letters**라는 변수에 저장할 수 있습니다:



Figure 14: Image

이 문서에서는 이 리스트를 [“하나”, “둘”, “셋”]으로 나타냅니다. 이 블록에서 정의된 변수들은 이후 예제에서 사용될 것입니다.

### Javascript 코드

```
var letters;
letters = ['하나', '둘', '셋'];
```

### Python 코드

```
letters = None
letters = ['하나', '둘', '셋']
```

다음은 숫자 형태의 문자열 리스트를 만드는 예입니다:



Figure 15: Image

### Javascript 코드

```
var numbers;
numbers = ['1', '2', '3'];
```

### Python 코드

```
numbers = None  
  
numbers = ['1', '2', '3']
```

다음은 색상 리스트를 만드는 예입니다:



Figure 16: Image

## Javascript 코드

```
color = [[255, 0, 0], [0, 0, 255], [0, 255, 0], [255, 255, 0]];
```

## Python 코드

```
color = [[255, 0, 0], [0, 0, 255], [0, 255, 0], [255, 255, 0]]
```

덜 흔하지만, 다양한 타입의 값을 가진 리스트도 만들 수 있습니다:



Figure 17: Image

## Javascript 코드

```
['하나', 1, [255, 0, 0]];
```

## Python 코드

```
['하나', 1, [255, 0, 0]]
```

## 입력 항목 개수 변경

입력 항목의 개수를 변경하려면 기어 아이콘을 클릭하세요. 새로운 창이 열리며, 여기서 왼쪽의 **항목** 서브 블록을 오른쪽의 **리스트** 블록으로 드래그하여 새로운 입력을 추가할 수 있습니다:

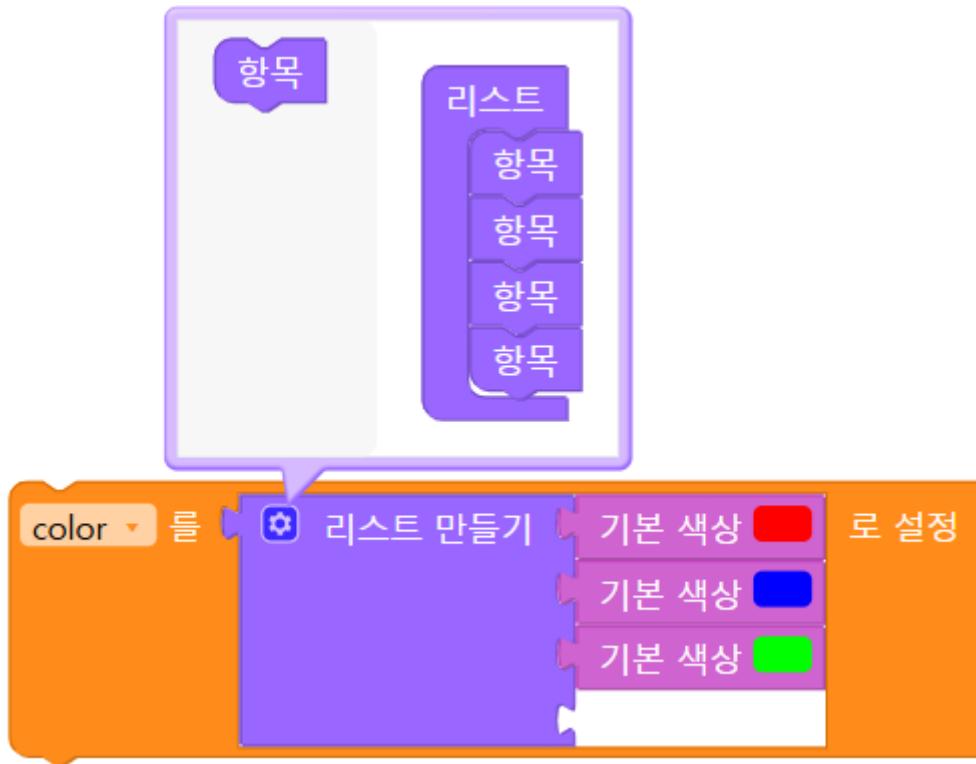


Figure 18: Image

새 항목은 원하는 위치에 추가할 수 있습니다. 마찬가지로, 원하지 않는 **항목** 서브 블록은 왼쪽으로 드래그하여 제거할 수 있습니다.

## 항목으로 리스트 설정

**항목으로 리스트 설정** 블록을 사용하면 지정된 항목을 반복하여 원하는 개수만큼 리스트를 만들 수 있습니다. 예를 들어, 아래의 블록은 변수 **words** 를 **["very", "very", "very"]** 로 설정합니다.



Figure 19: Image

## Javascript 코드

```

var words;

function listsRepeat(value, n) { // 리스트 내 동일 값 반복 생성 함수
  var array = [];
  for (var i = 0; i < n; i++) {
    array[i] = value;
  }
  return array;
}

words = listsRepeat('very', 3); // words = ['very', 'very', 'very']

```

## Python 코드

```

words = None
words = ['very'] * 3 # words = ['very', 'very', 'very']

```

## 리스트 길이

### 비어 있습니다

비어 있습니다 블록의 값은 입력이 빈 리스트일 경우 참, 그 외에는 거짓입니다 (리스트가 아니더라도 마찬가지). 아래 블록의 값은 거짓이 됩니다. 왜냐하면 **color** 변수는 비어 있지 않으며, 3 개의 항목을 가지고 있기 때문입니다.



Figure 20: Image

## Javascript 코드

```

var color;

color = [[255, 0, 0], [0, 0, 255], [0, 255, 0]];
!color.length; // 비어있지 않으므로 거짓 출력

```

## Python 코드

```

color = None

color = [[255, 0, 0], [0, 0, 255], [0, 255, 0]]
not len(color) # 비어있지 않으므로 거짓 출력

```

이것은 문자열의 “빈 문자열 확인” 블록과 유사합니다.

## 길이

길이 블록의 값은 리스트의 항목 수입니다. 예를 들어, 아래 블록에서 **color** 는 3 개의 항목을 가졌기 때문에 값은 3 입니다.



Figure 21: Image

## Javascript 코드

```
var color;  
  
color = [[255, 0, 0], [0, 0, 255], [0, 255, 0]];  
color.length; // 3 개의 항목이므로 3 의 값 반환
```

## Python 코드

```
color = None  
  
color = [[255, 0, 0], [0, 0, 255], [0, 255, 0]]  
len(color) # 3 개의 항목이므로 3 의 값 반환
```

길이 블록은 리스트에 몇 개의 항목이 있는지 알려줍니다. 리스트 내에 다른 항목이 몇 개 있는지가 아니라, 그 자체의 항목 수를 세는 것입니다. 예를 들어, 아래는 값이 3 인데, **words** 는 세 번 반복된 동일한 텍스트 ([“very”, “very”, “very”])로 구성되어 있습니다.



Figure 22: Image

이것은 문자열의“문자열 길이”블록과 유사합니다.

## Javascript 코드

```
var words;  
  
function listsRepeat(value, n) { // 리스트 내 동일 값 반복 생성 함수  
    var array = [];  
    for (var i = 0; i < n; i++) {  
        array[i] = value;  
    }  
    return array;  
}  
  
words = listsRepeat('very', 3);  
words.length; // 3 개의 항목이므로 3 의 값 반환
```

## Python 코드

```
words = None  
  
words = ['very'] * 3  
len(words) # 3 개의 항목이므로 3 의 값 반환
```

## 리스트 나타난 위치

이 블록들은 리스트에서 항목의 위치를 찾습니다. 예를 들어, 아래 블록의 값은 1입니다. 이유는 “very”라는 첫 번째 항목이 **words** 리스트 ([“very”, “very”, “very”])에서 첫 번째 위치에 있기 때문입니다.



Figure 23: Image

다음 결과는 3입니다. 왜냐하면 “very”라는 항목이 **words** 리스트에서 마지막 위치에 있기 때문입니다.

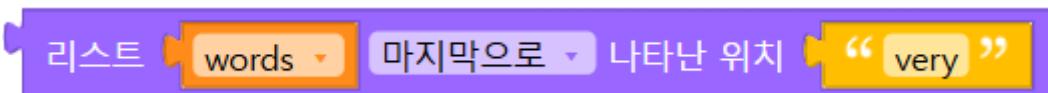


Figure 24: Image

리스트에 항목이 없는 경우, 결과는 0입니다. 예를 들어:



Figure 25: Image

이 블록들은 문자열의 “문자열 찾기” 블록과 유사합니다.

## Javascript 코드

```
var words;  
  
function listsRepeat(value, n) { // 리스트 내 동일 값 반복 생성 함수  
  var array = [];  
  for (var i = 0; i < n; i++) {  
    array[i] = value;  
  }  
  return array;  
}  
  
words = listsRepeat('very', 3);  
words.indexOf('very') + 1 // "very" 문자열이 첫 번째로 나타난 위치 1 반환  
words.lastIndexOf('very') + 1 // "very" 문자열이 마지막으로 나타난 위치 3 반환  
words.indexOf('유니콘') + 1 // "유니콘" 문자열이 첫 번째로 나타난 위치 0 반환
```

## Python 코드

```
words = None

def first_index(my_list, elem): # 문자열 첫 번째 위치 찾기 함수
    try: index = my_list.index(elem) + 1
    except: index = 0
    return index

def last_index(my_list, elem): # 문자열 마지막 위치 찾기 함수
    try: index = len(my_list) - my_list[::-1].index(elem)
    except: index = 0
    return index

words = ['very'] * 3
first_index(words, 'very') # "very" 문자열이 첫 번째로 나타난 위치 1 반환
last_index(words, 'very') # "very" 문자열이 마지막으로 나타난 위치 3 반환
first_index(words, '유니콘') # " 유니콘" 문자열이 첫 번째로 나타난 위치 0 반환
```

## 리스트에서 항목 가져오기

### 단일 항목 가져오기

**color** 리스트의 정의를 기억하세요:

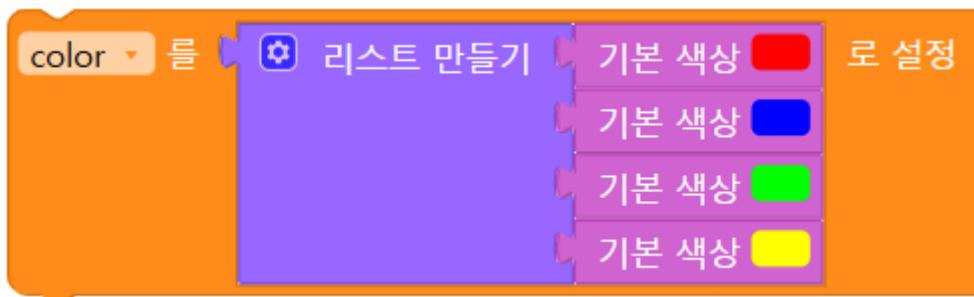


Figure 26: Image

다음 블록은 색상 **파랑**을 가져옵니다. **파랑**은 리스트의 두 번째 항목이기 때문입니다 (왼쪽에서부터 세기):

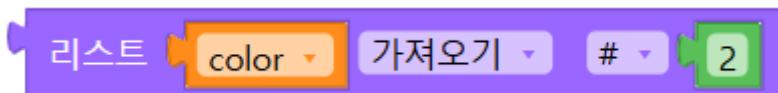


Figure 27: Image

이 블록은 **초록**을 가져옵니다. **초록**은 오른쪽 끝에서 두 번째 항목이기 때문입니다:

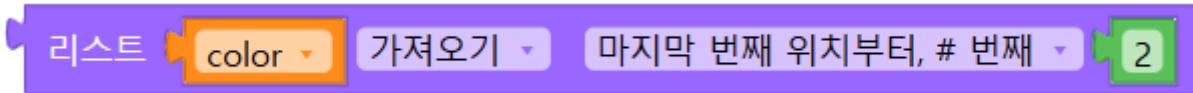


Figure 28: Image

이 블록은 첫 번째 항목인 **빨강**을 가져옵니다:



Figure 29: Image

이 블록은 마지막 항목인 노랑을 가져옵니다:



Figure 30: Image

이 블록은 리스트에서 임의로 하나의 항목을 선택합니다. 빨강, 파랑, 초록, 노랑 중 하나를 동일한 확률로 선택합니다.



Figure 31: Image

## Javascript 코드

```
var color;

function listsGetRandomItem(list, remove) { // 리스트 무작위 항목 선택 함수
  var x = Math.floor(Math.random() * list.length);
  if (remove) {
    return list.splice(x, 1)[0];
  } else {
    return list[x];
  }
}

color = [[255, 0, 0], [0, 0, 255], [0, 255, 0], [255, 255, 0]];
color[1]; // color에서 두 번째 항목 파랑 가져오기
color.slice(-2)[0]; // color에서 끝에서 두 번째 항목 초록 가져오기
color[0]; // color에서 첫 번째 항목 빨강 가져오기
color[-1][0]; // color에서 마지막 항목 노랑 가져오기
listsGetRandomItem(color, false); // color에서 무작위 항목 가져오기
```

## Python 코드

```
import random

color = None

color = [[255, 0, 0], [0, 0, 255], [0, 255, 0], [255, 255, 0]]
color[1] # color에서 두 번째 항목 파랑 가져오기
color[-2] # color에서 끝에서 두 번째 항목 초록 가져오기
color[0] # color에서 첫 번째 항목 빨강 가져오기
color[-1] # color에서 마지막 항목 노랑 가져오기
random.choice(color) # color에서 무작위 항목 가져오기
```

## 항목 가져오기 및 제거

리스트에서…가져오기 블록에서 드롭다운 메뉴를 선택하면 **리스트에서…가져오기 및 제거**로 변경되어, 항목을 가져오는 것뿐만 아니라 원본 리스트도 수정합니다:



Figure 32: Image

## Javascript 코드

```
var list;

function listsGetRandomItem(list, remove) { // 리스트 랜덤 항목 추출 함수
  var x = Math.floor(Math.random() * list.length);
  if (remove) {
    return list.splice(x, 1)[0];
  } else {
    return list[x];
  }
}

list[0]; // 리스트 첫 번째 항목 가져오기
list.slice(-1)[0]; // 리스트 마지막 항목 가져오기
listsGetRandomItem(list, false); // 리스트 랜덤 항목 가져오기
list.splice(0, 1)[0]; // 리스트 첫 번째 항목 잘라내기
list.shift(); // 리스트 앞에서 항목 제거 (첫 번째 항목 반환)
list.pop(); // 리스트 마지막 항목 잘라내기
listsGetRandomItem(list, true); // 리스트 임의 항목 잘라내기
```

## Python 코드

```
List2 = None # List 는 예약어 이기 때문에 List2 사용

def lists_remove_random_item(myList): # 리스트 임의 항목 제거
    x = int(random.random() * len(myList))
    return myList.pop(x)

List2[0] # 리스트 첫 번째 항목 가져오기
List2[-1] # 리스트 마지막 항목 가져오기
random.choice(List2) # 리스트 랜덤 항목 가져오기
List2.pop(0) # 리스트 첫 번째 항목 제거 (첫 번째 항목 반환)
List2.pop(0) # 리스트 앞에서 항목 제거 (첫 번째 항목 반환)
List2.pop() # 리스트 마지막 항목 제거
lists_remove_random_item(List2) # 리스트 임의 항목 제거
```

이 예제는 변수 **first letter**에 [“하나”]를 설정하고, **letters**는 [“둘”, “셋”]으로 남겨둡니다.



Figure 33: Image

## Javascript 코드

```
var letters, first_letter;

letters = ['하나', '둘', '셋'];
first_letter = letters.splice(0, 1)[0]; // letters 리스트 첫 번째 항목 잘라내기 first_letter = [" 하나], letters = [" 둘", " 셋"]
```

## Python 코드

```
letters = None
first_letter = None

letters = ['하나', '둘', '셋']
first_letter = letters.pop(0) # letters 리스트 첫 번째 항목 잘라내기 first_letter = [" 하나], letters = [" 둘", " 셋"]
```

## 항목 제거

드롭다운에서 “삭제”를 선택하면 블록 왼쪽의 플러그가 사라집니다:



Figure 34: Image

이렇게 하면 **letters** 에서 첫 번째 항목이 제거됩니다.

## Javascript 코드

```
var letters;  
  
letters = ['하나', '둘', '셋'];  
letters.splice(0, 1); // letters 리스트 첫 번째 항목 삭제, letters = ["둘", "셋"]
```

## Python 코드

```
letters = None  
  
letters = ['하나', '둘', '셋']  
letters.pop(0) # letters 리스트 첫 번째 항목 삭제, letters = ["둘", "셋"]
```

## 서브리스트 추출

리스트에서…서브리스트 추출 블록은 리스트에서…가져오기 블록과 유사하지만, 개별 항목 대신 서브리스트를 추출합니다. 서브리스트의 시작과 끝을 지정하는 방법은 여러 가지가 있습니다:



Figure 35: Image



Figure 36: Image

## Javascript 코드

```

var list;

list.slice(0, 1); // 첫 번째 위치부터 첫 번째 항목까지 서브리스트 추출
list.slice(0, list.length - 0); // 첫 번째 위치부터 끝에서 첫 번째까지 서브리스트 추출
list.slice(0, list.length); // 첫 번째 위치부터 마지막까지 서브리스트 추출
list.slice(list.length - 1, 1); // 마지막 첫 번째 위치부터 마지막까지 서브리스트 추출
list.slice(list.length - 1, list.length); // 마지막 첫 번째 위치부터 마지막까지 서브리스트 추출
list.slice(0, 1); // 첫 번째 위치부터 첫 번째 항목까지 서브리스트 추출
list.slice(0, list.length - 0); // 첫 번째 위치부터 끝까지 서브리스트 추출
list.slice(0); // 첫 번째 위치부터 마지막까지 서브리스트 추출

```

## Python 코드

```
List2 = None # list 는 예약어 이기 때문에 list2 사용
```

```

list2[ : 1] # 첫 번째 위치부터 첫 번째 항목까지 서브리스트 추출
list2[ : ] # 첫 번째 위치부터 끝까지 서브리스트 추출
list2[ : ] # 첫 번째 위치부터 마지막까지 서브리스트 추출
list2[-1 : 1] # 마지막 첫 번째 위치부터 끝까지 서브리스트 추출
list2[-1 : ] # 마지막 첫 번째 위치부터 끝까지 서브리스트 추출
list2[-1 : 1] # 마지막 첫 번째 위치부터 마지막까지 서브리스트 추출
list2[ : 1] # 첫 번째 위치부터 첫 번째 항목까지 서브리스트 추출
list2[ : ] # 첫 번째 위치부터 끝까지 서브리스트 추출
list2[ : ] # 첫 번째 위치부터 마지막까지 서브리스트 추출

```

이 예제에서는 새로운 리스트 **first letters** 가 생성됩니다. 이 리스트에는 두 개의 항목이 포함됩니다: [“하나”, “둘”].

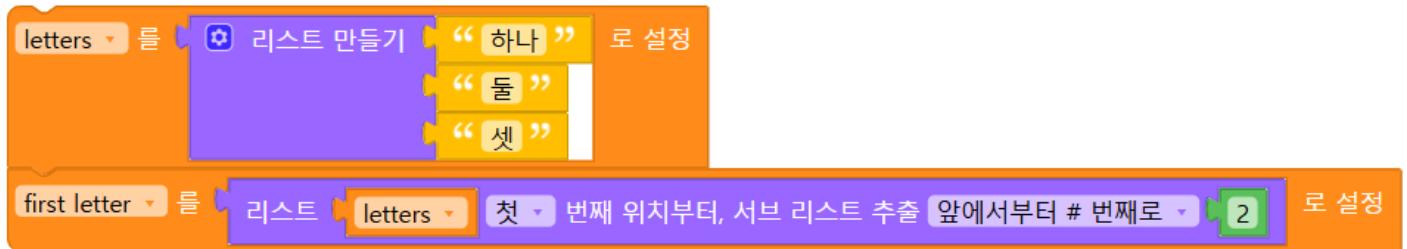


Figure 37: Image

이 블록은 원본 리스트를 수정하지 않음을 유의하세요.

## Javascript 코드

```

var letters, first_letter, list;

letters = ['하나', '둘', '셋'];
first_letter = list.slice(0, 2); // 첫 번째 위치부터 두 번째 항목까지 추출 first_letter = ["하나", "둘"]

```

## Python 코드

```

letters = None
first_letter = None
list2 = None # list 는 예약어 이기 때문에 list2 사용

letters = ['하나', '둘', '셋']
first_letter = list2[ : 2] # 첫 번째 위치부터 두 번째 항목까지 추출 first_letter = ["하나", "둘"]

```

## 리스트에 항목 추가하기

### 리스트에서...설정

**리스트에서…설정** 블록은 지정된 위치에 있는 항목을 다른 항목으로 교체합니다.

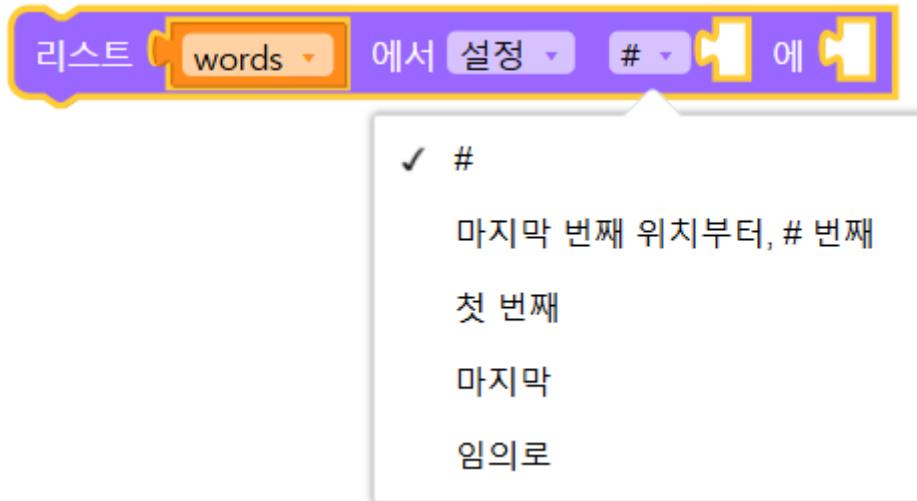


Figure 38: Image

## Javascript 코드

```
var letters, list;

list[0] = 'hello'; // 리스트에 첫 번째 위치에 "hello" 문자열 추가하기
list[list.length - 1] = 'hello'; // 리스트에 마지막 위치부터 "hello" 문자열 추가하기
list[0] = 'hello'; // 리스트 첫 번째에 "hello" 문자열 추가하기
list[list.length - 1] = 'hello'; // 리스트에 마지막 위치에 "hello" 문자열 추가하기
list[tmpX = Math.floor(Math.random() * list.length)]; // 무작위 위치 계산
list[tmpX] = 'hello'; // 리스트에 무작위 위치에 "hello" 문자열 추가하기
```

## Python 코드

```
import random

letters = None
list2 = None # list 는 예약어 이기 때문에 list2 사용

list2[0] = 'hello' # 리스트에 첫 번째 위치에 "hello" 문자열 추가하기
list2[-1] = 'hello' # 리스트에 마지막 위치부터 "hello" 문자열 추가하기
list2[0] = 'hello' # 리스트 첫 번째에 "hello" 문자열 추가하기
list2[-1] = 'hello' # 리스트의 마지막 위치에 "hello" 문자열 추가하기
tmp_x = int(random.random() * len(list2)) # 무작위 위치 계산
list2[tmp_x] = 'hello' # 리스트에 무작위 위치에 "hello" 문자열 추가하기
```

드롭다운 옵션의 의미는 이전 섹션을 참조하세요.

다음 예제는 두 가지 일을 합니다: 1. **words** 리스트를 [“very”, “very”, “very”]로 생성합니다. 2. 리스트의 세 번째 항목을“good”으로 교체합니다. 새 값은 **words** 가 [“very”, “very”, “good”] 이 됩니다.



Figure 39: Image

## Javascript 코드

```
var words;

function listsRepeat(value, n) { // 리스트 내 동일 값 반복 생성 함수
    var array = [];
    for (var i = 0; i < n; i++) {
        array[i] = value;
    }
    return array;
}

words = listsRepeat('very', 3); // words = ["very", "very", "very"]
words[2] = 'good';// words = ["very", "very", "good"]
```

## Python 코드

```
words = None

words = ['very'] * 3 # words = ['very', 'very', 'very']
words[2] = 'good'# words = ['very', 'very', 'good']
```

## 리스트에서…원하는 위치에 삽입

리스트에서…원하는 위치에 삽입 블록은 리스트에서…설정 블록에서 드롭다운 메뉴를 사용하여 얻을 수 있습니다:

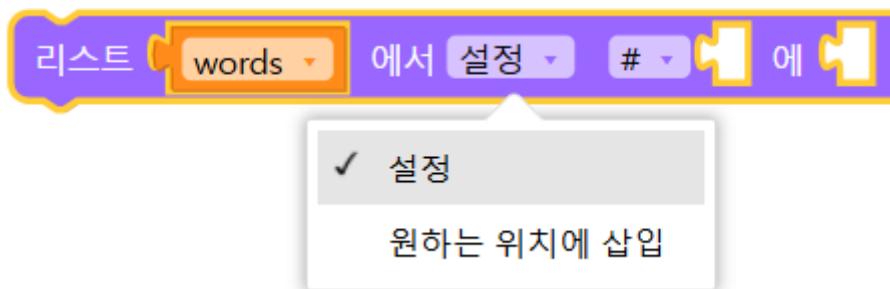


Figure 40: Image

## Javascript 코드

```
var words;

words.splice(2, 0, 'good'); // 세번째 위치 앞에 "good" 문자열 삽입
words.splice(words.length - 3, 0, 'good'); // 마지막으로부터 3 번째 위치 앞에 "good" 문자열 삽입
```

```

words.unshift('good'); // 맨 앞에 "good" 문자열 삽입
words.push('good'); // 마지막 위치에 "good" 문자열 삽입
var tmpX = Math.floor(Math.random() * words.length); // 무작위 위치 계산
words.splice(tmpX, 0, 'good'); // 무작위 위치에 "good" 문자열 삽입

```

## Python 코드

```

import random

words = None

words.insert(2, 'good') # 세번째 위치 앞에 "good" 문자열 삽입
words.insert(-3, 'good') # 마지막으로부터 3 번째 위치 앞에 "good" 문자열 삽입
words.insert(0, 'good') # 맨 앞에 "good" 문자열 삽입
words.append('good') # 마지막 위치에 "good" 문자열 삽입
tmp_x = int(random.random() * len(words)) # 무작위 위치 계산
words.insert(tmp_x, 'good') # 무작위 위치에 "good" 문자열 삽입

```

이 블록은 새 항목을 리스트의 지정된 위치에 삽입합니다. 예를 들어, 아래 예제에서는 세 가지 일이 일어납니다: 1. **words** 리스트는 [“very”, “very”, “very”]로 생성됩니다. 2. 리스트의 세 번째 항목이“good”으로 교체됩니다. 새로운 값은 [“very”, “very”, “good”]입니다. 3. “you’re”라는 단어가 리스트의 맨 앞에 삽입됩니다. 최종 값은 [“You’re”, “very”, “very”, “good”]입니다.



Figure 41: Image

## Javascript 코드

```

var words;

function listsRepeat(value, n) { // 리스트 내 동일 값 반복 생성 함수
  var array = [];
  for (var i = 0; i < n; i++) {
    array[i] = value;
  }
  return array;
}

words = listsRepeat('very', 3);
words[2] = 'good'; // words = ["very", "very", "good"]
words.splice(0, 0, 'You\'re'); // words = ["You're", "very", "very", "good"]

```

## Python 코드

```
words = None

words = ['very'] * 3
words[2] = 'good' # words = ['very', 'very', 'good']
words.insert(0, "You're") # words = ['You're', 'very', 'very', 'good']
```

## 리스트 값 치환 및 리스트 반환

리스트…항목을…(으)로 바꾼 리스트 추출 블록은 지정된 위치에 있는 항목을 다른 항목으로 치환한 뒤, 전체 리스트를 반환합니다.

## Javascript 코드

```
var words;

words.map((data, idx) => (idx === 0 ? 'good' : data)) // words 리스트의 첫번째 항목을 "good" 으로 바꾼 리스트 추출
words.map((data, idx) => (idx === words.length - 1 ? 'good' : data)) // words 리스트의 마지막으로부터 첫번째 항목을 "good" 으로 바꾼 리스트 추출
words.map((data, idx) => (idx === 0 ? 'good' : data)) // words 리스트의 첫번째 항목을 "good" 으로 바꾼 리스트 추출
words.map((data, idx) => (idx === words.length - 1 ? 'good' : data)) // words 리스트의 마지막 항목을 "good" 으로 바꾼 리스트 추출
```

## Python 코드

```
words = None

['good' if i == 0 else data for i, data in enumerate(words)] # words 리스트의 첫번째 항목을 "good" 으로 바꾼 리스트 추출
['good' if i == len(words) - 1 else data for i, data in enumerate(words)] # words 리스트의 마지막으로부터 첫번째 항목을 "good" 으로 바꾼 리스트 추출
['good' if i == 0 else data for i, data in enumerate(words)] # words 리스트의 첫번째 항목을 "good" 으로 바꾼 리스트 추출
['good' if i == len(words) - 1 else data for i, data in enumerate(words)] # words 리스트의 마지막 항목을 "good" 으로 바꾼 리스트 추출
```

드롭다운 옵션의 의미는 이전 섹션을 참조하세요.

다음 예제는 세 가지 일을 합니다:

1. **words** 리스트를 [“very”, “very”, “very”]로 생성합니다.
2. 리스트의 세 번째 항목을“good”으로 교체한 새로운 리스트를 반환하고, 해당 리스트를 새로운 변수 **list**에 할당합니다.
3. **list** 리스트는 [“very”, “very”, “good”]이 됩니다. **words** 리스트의 값은 바뀌지 않습니다.

## Javascript 코드

```
var words, list;

function listsRepeat(value, n) { // 리스트 내 동일 값 반복 생성 함수
  var array = [];
  for (var i = 0; i < n; i++) {
    array[i] = value;
  }
  return array;
```

```

}

words = listsRepeat('very', 3); // words = ["very", "very", "very"]
list = (list.map((data, idx) => (idx === 2 ? 'good' : data))); // list = ["very", "very", "good"]

```

## Python 코드

```

words = None
list2 = None # list 는 예약어 이기 때문에 list2 사용

words = ['very'] * 3 # words = ['very', 'very', 'very']
list2 = ['good' if i == 2 else data for i, data in enumerate(list2)] # list2 = ['very', 'very', 'good']

```

## 문자열 분할 및 리스트 결합

### 텍스트에서 목록 만들기

텍스트에서 목록 만들기 블록은 주어진 텍스트를 구분자를 사용하여 조각으로 나눕니다:



Figure 42: Image

### Javascript 코드

```
'311-555-2368'.split('-'); // '-' 를 기준으로 분리, ["311", "555", "2368"]
```

## Python 코드

```
'311-555-2368'.split('-') # '-' 를 기준으로 분리, ["311", "555", "2368"]
```

위 예제에서 새 리스트는 “311”, “555”, “2368”로 나뉩니다.

### 목록에서 텍스트 만들기

목록에서 텍스트 만들기 블록은 구분자를 사용하여 리스트의 항목들을 하나의 텍스트로 결합합니다:

### Javascript 코드

```
['311', '555', '2368'].join('-'); // '-'를 구분자로 사용하여 리스트를 하나의 문자열로 결합 "311-555-2368"
```



Figure 43: Image

## Python 코드

```
'-'.join(['311', '555', '2368']) # '-'를 구분자로 사용하여 리스트를 하나의 문자열로 결합 "311-555-2368"
```

위 예제에서 새 텍스트는“311-555-2368”로 반환됩니다.

## 관련 블록

### 리스트 출력하기

텍스트의“출력”블록은 리스트를 출력할 수 있습니다.

아래 프로그램의 결과는 알림 상자에 출력됩니다:



Figure 44: Image

## Javascript 코드

```
var letters;

letters = ['하나', '둘', '셋'];
window.alert(letters); // 리스트 출력하기
```

## Python 코드

```
letters = None

letters = ['하나', '둘', '셋']
print(letters) # 리스트 출력하기
```

하나, 둘, 셋이 출력됩니다.

## 리스트의 각 항목에 대해

반복의 “각 항목에 대해” 블록은 리스트의 각 항목에 대해 연산을 수행합니다.

예를 들어, 아래 블록들은 리스트의 각 항목을 개별적으로 출력합니다:



Figure 45: Image

## Javascript 코드

```
var letters;

var letters_list = ['하나', '둘', '셋'];
for (var letters_index in letters_list) { // 리스트 각 항목에 대해 출력하기
    letters = letters_list[letters_index];
    window.alert(letters);
}
```

## Python 코드

```
letters = None
for letters in ['하나', '둘', '셋']: # 리스트 각 항목에 대해 출력하기
    print(letters)
```

하나, 둘, 셋이 순차적으로 출력됩니다.

이 작업은 원본 리스트에서 항목을 제거하지 않습니다.

반복 종료 블록들의 예시도 참고하세요.

## 리스트 순서

리스트 항목의 순서를 변경할 수 있는 블록입니다.

## 정렬

리스트를 원하는 기준에 따라 정렬하는 블록입니다.

숫자 또는 알파벳을 기준으로 정렬할 수 있으며, 오름차순과 내림차순을 선택할 수 있습니다.

또한, 알파벳 정렬 시 대소문자를 무시하고 정렬할 수도 있습니다.



Figure 46: Image

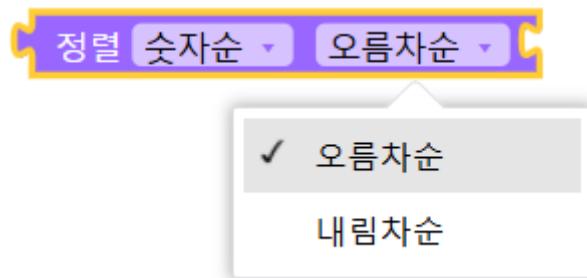


Figure 47: Image

## Javascript 코드

```
function listsGetSortCompare(type, direction) { // 문자 비교 함수
  var compareFuncs = {
    'NUMERIC': function(a, b) {
      return Number(a) - Number(b);
    },
    'TEXT': function(a, b) {
      return String(a) > String(b) ? 1 : -1;
    },
    'IGNORE_CASE': function(a, b) {
      return String(a).toLowerCase() > String(b).toLowerCase() ? 1 : -1;
    }
  };
  var compare = compareFuncs[type];
  return function(a, b) { return compare(a, b) * direction; };
}

[].slice().sort(listsGetSortCompare("NUMERIC", 1)); // 숫자, 오름차순 정렬
[].slice().sort(listsGetSortCompare("NUMERIC", -1)); // 숫자, 내림차순 정렬
[].slice().sort(listsGetSortCompare("TEXT", 1)); // 문자, 오름차순 정렬
[].slice().sort(listsGetSortCompare("TEXT", -1)) // 문자, 내림차순 정렬
[].slice().sort(listsGetSortCompare("IGNORE_CASE", 1)) // 알파벳순 (대소문자 구별 X), 오름차순 정렬
[].slice().sort(listsGetSortCompare("IGNORE_CASE", -1)) // 알파벳순 (대소문자 구별 X), 내림차순 정렬
```

## Python 코드

```
def lists_sort(my_list, type, reverse): # 정렬 함수
    def try_float(s):
        try:
            return float(s)
        except:
            return 0
    key_funcs = {
        "NUMERIC": try_float,
        "TEXT": str,
        "IGNORE_CASE": lambda s: str(s).lower()
    }
    key_func = key_funcs[type]
    list_cpy = list(my_list)
    return sorted(list_cpy, key=key_func, reverse=reverse)

lists_sort([], "NUMERIC", False) # 숫자, 오름차순 정렬
lists_sort([], "NUMERIC", True) # 숫자, 내림차순 정렬
lists_sort([], "TEXT", False) # 문자, 오름차순 정렬
lists_sort([], "TEXT", True) # 문자, 내림차순 정렬
lists_sort([], "IGNORE_CASE", False) # 알파벳순 (대소문자 구별 X), 오름차순 정렬
lists_sort([], "IGNORE_CASE", True) # 알파벳순 (대소문자 구별 X), 내림차순 정렬
```

아래 예제에서는 [gamma, beta, alpha] 리스트를 알파벳 오름차순으로 정렬하여 sortLetters 리스트를 생성하는 예제입니다.



Figure 48: Image

출력 결과는 [alpha, beta, gamma]입니다.

## Javascript 코드

```
var letters, sortLetters;

function listsGetSortCompare(type, direction) { // 문자 비교 함수
    var compareFuncs = {
        'NUMERIC': function(a, b) {
            return Number(a) - Number(b);
        },
        'TEXT': function(a, b) {
            return String(a) > String(b) ? 1 : -1;
        },
        'IGNORE_CASE': function(a, b) {
            return String(a).toLowerCase() > String(b).toLowerCase() ? 1 : -1;
        }
    };
    var compare = compareFuncs[type];
    return function(a, b) { return compare(a, b) * direction; };
}

letters = ['gamma', 'beta', 'alpha'];
sortLetters = letters.slice().sort(listsGetSortCompare("TEXT", 1)); // 알파벳 순 오름차순 정렬
window.alert(sortLetters); // ['alpha', 'beta', 'gamma']
```

## Python 코드

```
letters = None
sortLetters = None

def lists_sort(my_list, type, reverse): # 문자열 정렬 함수
    def try_float(s):
        try:
            return float(s)
        except:
            return 0
    key_funcs = {
        "NUMERIC": try_float,
        "TEXT": str,
        "IGNORE_CASE": lambda s: str(s).lower()
    }
    key_func = key_funcs[type]
    list_cpy = list(my_list)
    return sorted(list_cpy, key=key_func, reverse=reverse)

letters = ['gamma', 'beta', 'alpha']
sortLetters = lists_sort(letters, "TEXT", False) # 알파벳 순 오름차순 정렬
print(sortLetters) # ['alpha', 'beta', 'gamma']
```

## 뒤집기

리스트의 요소 순서를 역순으로 변경하는 블록입니다.



Figure 49: Image

## Javascript 코드

```
[] .slice() .reverse(); // 리스트 역순으로 변경
```

## Python 코드

```
list(reversed([])) # 리스트 역순으로 변경
```

아래 예제에서는 정렬된 `sortLetters` 리스트를 뒤집어 `[gamma, beta, alpha]` 리스트를 생성합니다.

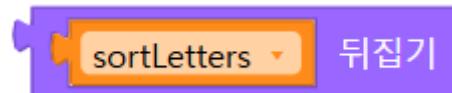


Figure 50: Image

## Javascript 코드

```
var letters, sortLetters;

function listsGetSortCompare(type, direction) { // 문자 비교 함수
  var compareFuncs = {
    'NUMERIC': function(a, b) {
      return Number(a) - Number(b);
    },
    'TEXT': function(a, b) {
      return String(a) > String(b) ? 1 : -1;
    },
    'IGNORE_CASE': function(a, b) {
      return String(a).toLowerCase() > String(b).toLowerCase() ? 1 : -1;
    }
  };
  var compare = compareFuncs[type];
  return function(a, b) { return compare(a, b) * direction; };
}

letters = ['gamma', 'beta', 'alpha'];
sortLetters = letters.slice().sort(listsGetSortCompare("TEXT", 1)); // 알파벳 기준 오름차순 정렬 ['alpha','beta','gamma']
sortLetters.slice().reverse(); // 리스트 역순으로 변경 ['gamma','beta','alpha']
```

## Python 코드

```
letters = None
sortLetters = None

def lists_sort(my_list, type, reverse): # 문자열 정렬 함수
  def try_float(s):
    try:
      return float(s)
    except:
      return 0
  key_funcs = {
    "NUMERIC": try_float,
    "TEXT": str,
    "IGNORE_CASE": lambda s: str(s).lower()
  }
  key_func = key_funcs[type]
  list_cpy = list(my_list)
  return sorted(list_cpy, key=key_func, reverse=reverse)

letters = ['gamma', 'beta', 'alpha']
sortLetters = lists_sort(letters, "TEXT", False) # 알파벳 기준 오름차순 정렬 ['alpha','beta','gamma']
list(reversed(sortLetters)) # 리스트 역순으로 변경 ['gamma','beta','alpha']
```

## 몸-찾기

## 대시보드 열기

대시보드 열기는 설치할 수 있는 블록은 아니지만, 확장 모듈에서 사용되는 모델이 어떠한 식으로 적용되는지 알 수 있는 대시보드를 열 수 있습니다. ## 대시보드 화면 대시보드 열기 클릭 시 다음과 같은 화면을 볼 수 있습니다.



## 세부 버튼

### Power

선택한 카메라를 키거나 끕니다.

### Load Model

학습된 몸 모델을 불러옵니다. ‘몸 찾기’ 확장 모듈을 사용하기 위해서 반드시 필요한 작업입니다.

### Detect

몸 찾기를 실행하거나 멈춥니다.

Once 버튼으로 한 번만 실행할지, Continuous 버튼으로 연속으로 실행할지 정할 수 있습니다.

또한, Stop 버튼을 통해 찾기를 멈출 수 있습니다.

### Show Result

몸 찾기 결과를 카메라 화면 상으로 출력합니다.

## Sensory Data

선택한 몸 부위에 따른 데이터 값을 출력합니다.

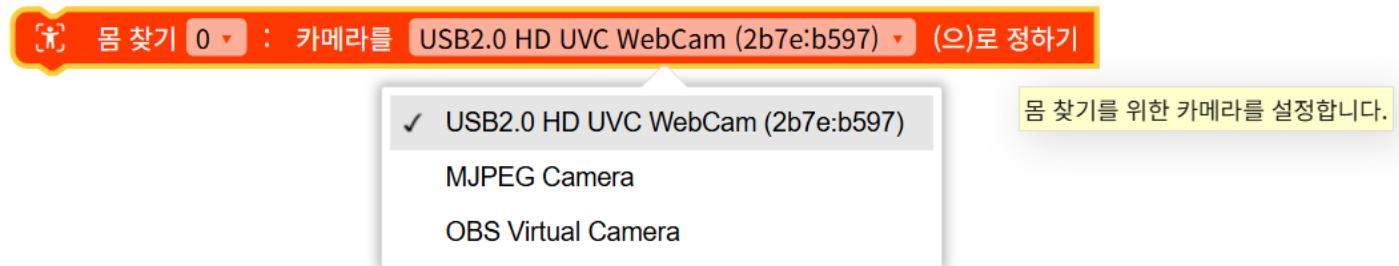
### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
parts	드롭다운 옵션	몸 부위	eye, ear, nose, mouth, neck, shoulder, elbow, wrist, hand, hip, knee, ankle, foot

## 블록

### 카메라 정하기

몸 찾기 모듈에 사용할 카메라를 선택합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
camera	드롭다운 옵션	사용할 카메라	연결한 카메라 리스트

## 자바스크립트 코드

```
// 특정 카메라를 몸 찾기를 위한 카메라로 정하기 (id는 예시)
$(`BodyDetection${0}:camera.deviceId`).d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31elae1fbca9cb';
```

## 파이썬 코드

```
# 특정 카메라를 몸 찾기를 위한 카메라로 정하기 (id는 예시)
__('BodyDetection*0:camera.deviceId').d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1aelfbca9cb'
```

## 몸 모델 불러오기

학습된 몸 모델을 불러옵니다. '몸 찾기' 모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

기다리기를 체크하면, 모델 불러오기가 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.

 몸 찾기 0 : 몸 모델 불러오기 | 기다리기 ✓

학습된 몸 모델을 불러옵니다. '몸 찾기' 모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

## 자바스크립트 코드

```
// 몸 모델 불러오기 | 기다리기 0
$('BodyDetection*0:load_model').d = 1;
await $('BodyDetection*0:!load_model').w();

// 몸 모델 불러오기 | 기다리기 X
$('BodyDetection*0:load_model').d = 1;
```

## 파이썬 코드

```
# 몸 모델 불러오기 | 기다리기 0
__('BodyDetection*0:load_model').d = 1
await __('BodyDetection*0:!load_model').w()

# 몸 모델 불러오기 | 기다리기 X
__('BodyDetection*0:load_model').d = 1
```

## 몸 한 번 찾기

현재 화면에 있는 몸을 찾아 딱 한 번 표시합니다.

 몸 찾기 0 : 몸 한 번 찾기

현재 화면에 있는 몸을 찾아 딱 한번 표시합니다.

## 자바스크립트 코드

```
// 몸 한 번 찾기  
$('BodyDetection*0:detect.once').d = 1;
```

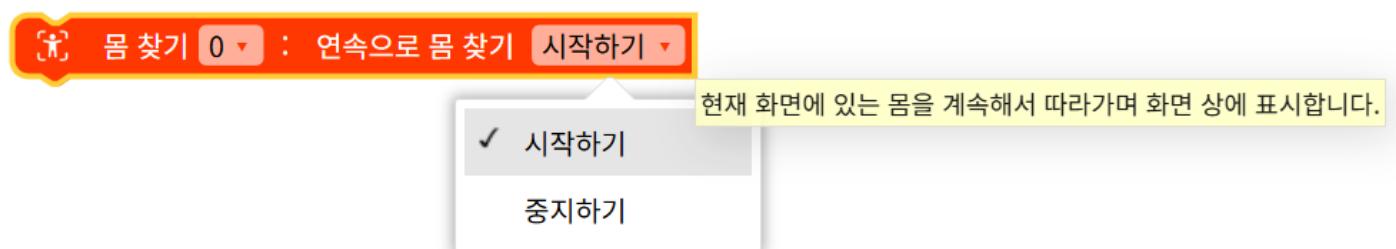
## 파이썬 코드

```
# 몸 한 번 찾기  
__('BodyDetection*0:detect.once').d = 1
```

## 몸 연속으로 찾기

몸 연속으로 찾기를 시작하거나 중지합니다.

몸 연속으로 찾기를 시작하면, 현재 화면에 있는 몸을 계속 따라가며 화면상에 표시합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	몸 찾기	시작하기 (1), 중지하기 (0)

## 자바스크립트 코드

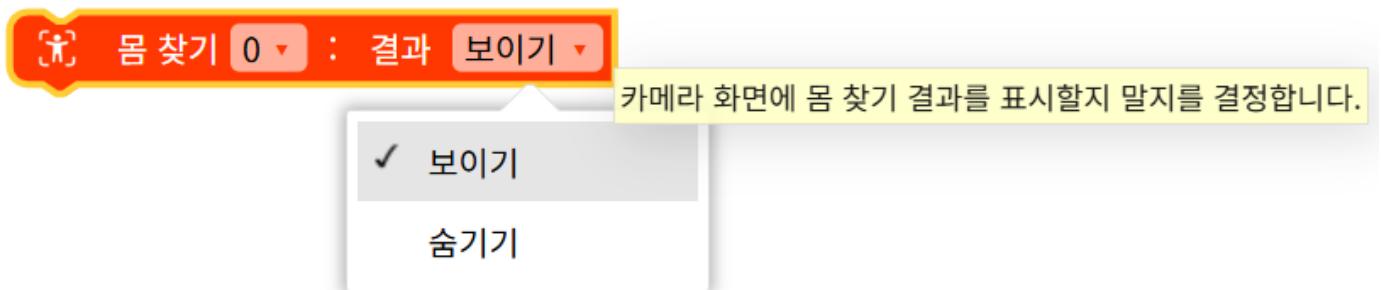
```
// 연속으로 몸 찾기 시작하기  
$('BodyDetection*0:detect.continuous').d = 1;  
  
// 연속으로 몸 찾기 중지하기  
$('BodyDetection*0:detect.continuous').d = 0;
```

## 파이썬 코드

```
# 연속으로 몸 찾기 시작하기  
__('BodyDetection*0:detect.continuous').d = 1  
  
# 연속으로 몸 찾기 중지하기  
__('BodyDetection*0:detect.continuous').d = 0
```

## 몸 찾기 결과 보이기

카메라 화면에 몸 찾기 결과를 표시할지 말지를 결정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	몸 찾기 결과	보이기 (1), 숨기기 (0)

## 자바스크립트 코드

```
// 몸 찾기 결과 보이기
$('BodyDetection*0:display').d = 1;

// 몸 찾기 결과 숨기기
$('BodyDetection*0:display').d = 0;
```

## 파이썬 코드

```
# 몸 찾기 결과 보이기
__('BodyDetection*0:display').d = 1

# 몸 찾기 결과 숨기기
__('BodyDetection*0:display').d = 0
```

## 몸 관련 데이터

선택한 몸 부위의 데이터를 반환합니다.



드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
part	드롭다운 옵션	몸 부위	왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 귀 (ear.left), 오른쪽 귀 (ear.right), 코 (nose), 입 (mouth), 목 (neck), 왼쪽 어깨 (shoulder.left), 오른쪽 어깨 (shoulder.right), 왼쪽 팔꿈치 (elbow.left), 오른쪽 팔꿈치 (elbow.right), 왼쪽 손목 (wrist.left), 오른쪽 손목 (wrist.right), 왼쪽 손 (hand.left), 오른쪽 손 (hand.right), 왼쪽 엉덩이 (hip.left), 오른쪽 엉덩이 (hip.right), 왼쪽 무릎 (knee.left), 오른쪽 무릎 (knee.right), 왼쪽 발목 (ankle.left), 오른쪽 발목 (ankle.right), 왼쪽 발 (foot.left), 오른쪽 발 (foot.right)
axis	드롭다운 옵션	좌표 방향	x, y

## 자바스크립트 코드

```
// 왼쪽 눈 x 좌표
$('BodyDetection*0:eye.left.x').d;

// 오른쪽 눈 x 좌표
$('BodyDetection*0:eye.right.x').d;

// 왼쪽 귀 x 좌표
$('BodyDetection*0:ear.left.x').d;

// 오른쪽 귀 x 좌표
$('BodyDetection*0:ear.right.x').d;

// 코 x 좌표
$('BodyDetection*0:nose.x').d;
```

```

// 입 x 좌표
$('BodyDetection*0:mouth.x').d;

// 목 x 좌표
$('BodyDetection*0:neck.x').d;

// 왼쪽 어깨 x 좌표
$('BodyDetection*0:shoulder.left.x').d;

// 오른쪽 어깨 x 좌표
$('BodyDetection*0:shoulder.right.x').d;

// 왼쪽 팔꿈치 x 좌표
$('BodyDetection*0:elbow.left.x').d;

// 오른쪽 팔꿈치 x 좌표
$('BodyDetection*0:elbow.right.x').d;

// 왼쪽 손목 x 좌표
$('BodyDetection*0:wrist.left.x').d;

// 오른쪽 손목 x 좌표
$('BodyDetection*0:wrist.right.x').d;

// 왼쪽 손 x 좌표
$('BodyDetection*0:hand.left.x').d;

// 오른쪽 손 x 좌표
$('BodyDetection*0:hand.right.x').d;

// 왼쪽 엉덩이 y 좌표
$('BodyDetection*0:hip.left.y').d;

// 오른쪽 엉덩이 y 좌표
$('BodyDetection*0:hip.right.y').d;

// 왼쪽 무릎 y 좌표
$('BodyDetection*0:knee.left.y').d;

// 오른쪽 무릎 y 좌표
$('BodyDetection*0:knee.right.y').d;

// 왼쪽 발목 y 좌표
$('BodyDetection*0:ankle.left.y').d;

// 오른쪽 발목 y 좌표
$('BodyDetection*0:ankle.right.y').d;

// 왼쪽 발 y 좌표
$('BodyDetection*0:foot.left.y').d;

// 오른쪽 발 y 좌표
$('BodyDetection*0:foot.right.y').d;

```

## 파이썬 코드

```

# 왼쪽 눈 x 좌표
__($('BodyDetection*0:eye.left.x').d

# 오른쪽 눈 x 좌표
__($('BodyDetection*0:eye.right.x').d

# 왼쪽 귀 x 좌표
__($('BodyDetection*0:ear.left.x').d

# 오른쪽 귀 x 좌표
__($('BodyDetection*0:ear.right.x').d

# 코 x 좌표
__($('BodyDetection*0:nose.x').d

# 입 x 좌표
__($('BodyDetection*0:mouth.x').d

```

```

# 목 x 좌표
__('BodyDetection*0:neck.x').d

# 왼쪽 어깨 x 좌표
__('BodyDetection*0:shoulder.left.x').d

# 오른쪽 어깨 x 좌표
__('BodyDetection*0:shoulder.right.x').d

# 왼쪽 팔꿈치 x 좌표
__('BodyDetection*0:elbow.left.x').d

# 오른쪽 팔꿈치 x 좌표
__('BodyDetection*0:elbow.right.x').d

# 왼쪽 손목 x 좌표
__('BodyDetection*0:wrist.left.x').d

# 오른쪽 손목 x 좌표
__('BodyDetection*0:wrist.right.x').d

# 왼쪽 손 x 좌표
__('BodyDetection*0:hand.left.x').d

# 오른쪽 손 x 좌표
__('BodyDetection*0:hand.right.x').d

# 왼쪽 엉덩이 y 좌표
__('BodyDetection*0:hip.left.y').d

# 오른쪽 엉덩이 y 좌표
__('BodyDetection*0:hip.right.y').d

# 왼쪽 무릎 y 좌표
__('BodyDetection*0:knee.left.y').d

# 오른쪽 무릎 y 좌표
__('BodyDetection*0:knee.right.y').d

# 왼쪽 발목 y 좌표
__('BodyDetection*0:ankle.left.y').d

# 오른쪽 발목 y 좌표
__('BodyDetection*0:ankle.right.y').d

# 왼쪽 발 y 좌표
__('BodyDetection*0:foot.left.y').d

# 오른쪽 발 y 좌표
__('BodyDetection*0:foot.right.y').d

```

## 두 몸 부위 사이의 거리

선택한 두 몸 부위 사이의 거리를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
part 1	드롭다운 옵션	몸 부위	왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 귀 (ear.left), 오른쪽 귀 (ear.right), 코 (nose), 입 (mouth), 목 (neck), 왼쪽 어깨 (shoulder.left), 오른쪽 어깨 (shoulder.right), 왼쪽 팔꿈치 (elbow.left), 오른쪽 팔꿈치 (elbow.right), 왼쪽 손목 (wrist.left), 오른쪽 손목 (wrist.right), 왼쪽 손 (hand.left), 오른쪽 손 (hand.right), 왼쪽 엉덩이 (hip.left), 오른쪽 엉덩이 (hip.right), 왼쪽 무릎 (knee.left), 오른쪽 무릎 (knee.right), 왼쪽 발목 (ankle.left), 오른쪽 발목 (ankle.right), 왼쪽 발 (foot.left), 오른쪽 발 (foot.right)

이름	구분	설명	범위 / 종류
part 2	드롭다운 옵션	몸 부위	왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 귀 (ear.left), 오른쪽 귀 (ear.right), 코 (nose), 입 (mouth), 목 (neck), 왼쪽 어깨 (shoulder.left), 오른쪽 어깨 (shoulder.right), 왼쪽 팔꿈치 (elbow.left), 오른쪽 팔꿈치 (elbow.right), 왼쪽 손목 (wrist.left), 오른쪽 손목 (wrist.right), 왼쪽 손 (hand.left), 오른쪽 손 (hand.right), 왼쪽 엉덩이 (hip.left), 오른쪽 엉덩이 (hip.right), 왼쪽 무릎 (knee.left), 오른쪽 무릎 (knee.right), 왼쪽 발목 (ankle.left), 오른쪽 발목 (ankle.right), 왼쪽 발 (foot.left), 오른쪽 발 (foot.right)
distance	드롭다운 옵션	거리	거리 (distance), 가로거리 (horizontal distance), 세로거리 (vertical distance)

## 자바스크립트 코드

```
// 왼쪽 눈에서 왼쪽 눈까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:eye.left.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:eye.left.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 왼쪽 눈에서 오른쪽 눈까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:eye.right.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:eye.right.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 왼쪽 눈에서 왼쪽 귀까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:ear.left.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:ear.left.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 왼쪽 눈에서 오른쪽 귀까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:ear.right.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:ear.right.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );
```

```

// 원쪽 눈에서 코까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:nose.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:nose.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 원쪽 눈에서 입까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:mouth.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:mouth.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 원쪽 눈에서 목까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:neck.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:neck.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 원쪽 눈에서 원쪽 어깨까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:shoulder.left.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:shoulder.left.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 원쪽 눈에서 오른쪽 어깨까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:shoulder.right.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:shoulder.right.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 원쪽 눈에서 원쪽 팔꿈치까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:elbow.left.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:elbow.left.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 원쪽 눈에서 오른쪽 팔꿈치까지 거리
Math.sqrt( Math.pow((($('BodyDetection*0:elbow.right.x').d - $('BodyDetection*0:eye.left.x').d), 2) + Math.pow((($('BodyDetection*0:elbow.right.y').d - $('BodyDetection*0:eye.left.y').d), 2) ) );

// 원쪽 눈에서 원쪽 손목까지 가로거리
Math.abs($('BodyDetection*0:wrist.left.x').d - $('BodyDetection*0:eye.left.x').d);

// 원쪽 눈에서 오른쪽 손목까지 가로거리
Math.abs($('BodyDetection*0:wrist.right.x').d - $('BodyDetection*0:eye.left.x').d);

// 원쪽 눈에서 원쪽 손까지 가로거리
Math.abs($('BodyDetection*0:hand.left.x').d - $('BodyDetection*0:eye.left.x').d);

// 원쪽 눈에서 오른쪽 손까지 가로거리
Math.abs($('BodyDetection*0:hand.right.x').d - $('BodyDetection*0:eye.left.x').d);

// 원쪽 눈에서 원쪽 엉덩이까지 가로거리
Math.abs($('BodyDetection*0:hip.left.x').d - $('BodyDetection*0:eye.left.x').d);

// 원쪽 눈에서 오른쪽 엉덩이까지 가로거리
Math.abs($('BodyDetection*0:hip.right.x').d - $('BodyDetection*0:eye.left.x').d);

// 원쪽 눈에서 원쪽 무릎까지 세로거리
Math.abs($('BodyDetection*0:knee.left.y').d - $('BodyDetection*0:eye.left.y').d);

// 원쪽 눈에서 오른쪽 무릎까지 세로거리
Math.abs($('BodyDetection*0:knee.right.y').d - $('BodyDetection*0:eye.left.y').d);

// 원쪽 눈에서 원쪽 발목까지 세로거리
Math.abs($('BodyDetection*0:ankle.left.y').d - $('BodyDetection*0:eye.left.y').d);

// 원쪽 눈에서 오른쪽 발목까지 세로거리
Math.abs($('BodyDetection*0:ankle.right.y').d - $('BodyDetection*0:eye.left.y').d);

// 원쪽 눈에서 원쪽 발까지 세로거리
Math.abs($('BodyDetection*0:foot.left.y').d - $('BodyDetection*0:eye.left.y').d);

// 원쪽 눈에서 오른쪽 발까지 세로거리
Math.abs($('BodyDetection*0:foot.right.y').d - $('BodyDetection*0:eye.left.y').d);

```

## 파이썬 코드

```

# 원쪽 눈에서 원쪽 눈까지 거리
math.sqrt( math.pow((__($('BodyDetection*0:eye.left.x').d - __($('BodyDetection*0:eye.left.x').d), 2) + math.pow((__($('BodyDetection*0:eye.left.y').d - __($('BodyDetection*0:eye.left.y').d), 2) )

# 원쪽 눈에서 오른쪽 눈까지 거리
math.sqrt( math.pow((__($('BodyDetection*0:eye.right.x').d - __($('BodyDetection*0:eye.left.x').d), 2) + math.pow((__($('BodyDetection*0:eye.right.y').d - __($('BodyDetection*0:eye.left.y').d), 2) )

# 원쪽 눈에서 원쪽 귀까지 거리
math.sqrt( math.pow((__($('BodyDetection*0:ear.left.x').d - __($('BodyDetection*0:eye.left.x').d), 2) + math.pow((__($('BodyDetection*0:ear.left.y').d - __($('BodyDetection*0:eye.left.y').d), 2) )

# 원쪽 눈에서 오른쪽 귀까지 거리
math.sqrt( math.pow((__($('BodyDetection*0:ear.right.x').d - __($('BodyDetection*0:eye.left.x').d), 2) + math.pow((__($('BodyDetection*0:ear.right.y').d - __($('BodyDetection*0:eye.left.y').d), 2) )

# 원쪽 눈에서 코까지 거리

```

```

math.sqrt( math.pow((__(('BodyDetection*0:nose.x')).d - __('BodyDetection*0:eye.left.x').d), 2) + math.pow((__(('BodyDetection*0:nose.y')).d - __('BodyDetection*0:eye.left.y').d), 2) )

# 왼쪽 눈에서 입까지 거리
math.sqrt( math.pow((__(('BodyDetection*0:mouth.x')).d - __('BodyDetection*0:eye.left.x').d), 2) + math.pow((__(('BodyDetection*0:mouth.y')).d - __('BodyDetection*0:eye.left.y').d), 2) )

# 왼쪽 눈에서 목까지 거리
math.sqrt( math.pow((__(('BodyDetection*0:neck.x')).d - __('BodyDetection*0:eye.left.x').d), 2) + math.pow((__(('BodyDetection*0:neck.y')).d - __('BodyDetection*0:eye.left.y').d), 2) )

# 왼쪽 눈에서 왼쪽 어깨까지 거리
math.sqrt( math.pow((__(('BodyDetection*0:shoulder.left.x')).d - __('BodyDetection*0:eye.left.x').d), 2) + math.pow((__(('BodyDetection*0:shoulder.left.y')).d - __('BodyDetection*0:eye.left.y').d), 2) )

# 왼쪽 눈에서 오른쪽 어깨까지 거리
math.sqrt( math.pow((__(('BodyDetection*0:shoulder.right.x')).d - __('BodyDetection*0:eye.left.x').d), 2) + math.pow((__(('BodyDetection*0:shoulder.right.y')).d - __('BodyDetection*0:eye.left.y').d), 2) )

# 왼쪽 눈에서 왼쪽 팔꿈치 까지 거리
math.sqrt( math.pow((__(('BodyDetection*0:elbow.left.x')).d - __('BodyDetection*0:eye.left.x').d), 2) + math.pow((__(('BodyDetection*0:elbow.left.y')).d - __('BodyDetection*0:eye.left.y').d), 2) )

# 왼쪽 눈에서 오른쪽 팔꿈치 까지 거리
math.sqrt( math.pow((__(('BodyDetection*0:elbow.right.x')).d - __('BodyDetection*0:eye.left.x').d), 2) + math.pow((__(('BodyDetection*0:elbow.right.y')).d - __('BodyDetection*0:eye.left.y').d), 2) )

# 왼쪽 눈에서 왼쪽 손목 까지 가로거리
math.fabs(__('BodyDetection*0:wrist.left.x').d - __('BodyDetection*0:eye.left.x').d)

# 왼쪽 눈에서 오른쪽 손목 까지 가로거리
math.fabs(__('BodyDetection*0:wrist.right.x').d - __('BodyDetection*0:eye.left.x').d)

# 왼쪽 눈에서 왼쪽 손까지 가로거리
math.fabs(__('BodyDetection*0:hand.left.x').d - __('BodyDetection*0:eye.left.x').d)

# 왼쪽 눈에서 오른쪽 손까지 가로거리
math.fabs(__('BodyDetection*0:hand.right.x').d - __('BodyDetection*0:eye.left.x').d)

# 왼쪽 눈에서 왼쪽 엉덩이까지 가로거리
math.fabs(__('BodyDetection*0:hip.left.x').d - __('BodyDetection*0:eye.left.x').d)

# 왼쪽 눈에서 오른쪽 엉덩이까지 가로거리
math.fabs(__('BodyDetection*0:hip.right.x').d - __('BodyDetection*0:eye.left.x').d)

# 왼쪽 눈에서 왼쪽 무릎까지 세로거리
math.fabs(__('BodyDetection*0:knee.left.y').d - __('BodyDetection*0:eye.left.y').d)

# 왼쪽 눈에서 오른쪽 무릎까지 세로거리
math.fabs(__('BodyDetection*0:knee.right.y').d - __('BodyDetection*0:eye.left.y').d)

# 왼쪽 눈에서 왼쪽 발까지 세로거리
math.fabs(__('BodyDetection*0:ankle.left.y').d - __('BodyDetection*0:eye.left.y').d)

# 왼쪽 눈에서 오른쪽 발까지 세로거리
math.fabs(__('BodyDetection*0:ankle.right.y').d - __('BodyDetection*0:eye.left.y').d)

# 왼쪽 눈에서 왼쪽 발까지 세로거리
math.fabs(__('BodyDetection*0:foot.left.y').d - __('BodyDetection*0:eye.left.y').d)

# 왼쪽 눈에서 오른쪽 발까지 세로거리
math.fabs(__('BodyDetection*0:foot.right.y').d - __('BodyDetection*0:eye.left.y').d)

```

## 몸 모델 로딩 상태값

몸 모델 로딩 상태를 반환합니다.

아직 불러오지 않았다면 0, 불러오는 중이면 1, 불러오기를 완료했다면 2 를 반환합니다.

## 몸 찾기 0 : 몸 모델 로딩 상태

몸 모델 로딩 상태를 반환합니다.

아직 불러오지 않았으면 0, 불러오는 중이면 1, 불러오기를 완료했으면 2를 반환합니다.

### 자바스크립트 코드

```
// 몸 모델 로딩 상태 값  
$('BodyDetection*0:model_state').d;
```

### 파이썬 코드

```
# 몸 모델 로딩 상태 값  
__('BodyDetection*0:model_state').d
```

### 몸을 찾았는가?

몸 찾기 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.

## 몸 찾기 0 : 몸을 찾았는가?

몸을 찾았는지 여부

### 자바스크립트 코드

```
// 몸을 찾았는가?  
$('BodyDetection*0:detectected').d;
```

### 파이썬 코드

```
# 몸을 찾았는가?  
__('BodyDetection*0:detectected').d
```

### 문자열

문자열 예시는 다음과 같습니다: - “thing #1”- “March 12, 2010”- “”(빈 문자열)

문자열에는 대문자 또는 소문자로 된 문자, 숫자, 구두점, 기타 기호 및 단어 사이의 공백이 포함될 수 있습니다.

## 블록

### 문자열 만들기

다음 블록은 “hello”라는 문자열을 생성하고 이를 `greeting`이라는 변수에 저장합니다.



Figure 51: Image

### Javascript 코드

```
var greeting;  
  
greeting = 'hello';
```

### Python 코드

```
greeting = None  
  
greeting = 'hello'
```

**문자열 만들기** 블록은 `greeting` 변수의 값과 새로운 문자열 “world”를 결합 (연결) 하여 “helloworld”라는 문자열을 생성합니다. 원래 문자열 중 어느 것도 공백을 포함하지 않았기 때문에 공백이 없습니다.



Figure 52: Image

### Javascript 코드

```
var greeting;  
  
greeting = 'hello';  
String(greeting) + 'world'; // "helloworld"
```

## Python 코드

```
greeting = None  
  
greeting = 'hello'  
str(greeting) + 'world' # "helloworld"
```

입력 문자열의 개수를 늘리려면 텁니바퀴 아이콘을 클릭하면 다음과 같이 보기가 변경됩니다:



Figure 53: Image

추가 입력을 추가하려면 왼쪽의 회색 도구 상자에서 **항목** 블록을 끌어와 **가입** 블록에 삽입하면 됩니다.

## 문자열 수정

…내용 덧붙이기 블록은 지정된 변수에 주어진 문자열을 추가합니다. 이 경우, `greeting` 변수의 값이 “hello”에서 “hello, there!”로 변경됩니다.



Figure 54: Image

## Javascript 코드

```
var greeting;  
  
greeting = 'hello';  
greeting += ' there!'; // greeting = "hello, there!"
```

## Python 코드

```
greeting = None  
  
greeting = 'hello'  
greeting = str(greeting) + ', there!' # greeting = "hello, there!"
```

## 문자열 길이

길이 블록은 각 문자열에서 문자, 숫자 등을 세어 총 길이를 반환합니다. “We're #1!”의 길이는 9이며, 빈 문자열의 길이는 0입니다.



Figure 55: Image



Figure 56: Image

## Javascript 코드

```
'We're #1!'.length; // 'We're #1!'의 문자열 길이 값 9 반환  
''.length // ''의 문자열 길이 값 0 반환
```

## Python 코드

```
len("We're #1!") # "We're #1!" 의 문자열 길이 값 9 반환  
len("") # ''의 문자열 길이 값 0 반환
```

## 빈 문자열 확인

비어 있습니다 블록은 주어진 문자열가 비었는지 (길이가 0인지) 확인합니다. 첫 번째 경우에는 참이 반환되고, 두 번째 경우에는 거짓이 반환됩니다.



Figure 57: Image



Figure 58: Image

## Javascript 코드

```
var greeting;

greeting = 'hello';
!''.length; // 문자열이 비어 있으므로 참 반환
!greeting.length; // 문자열이 비어있지 않으므로 거짓 반환
```

## Python 코드

```
greeting = None

greeting = 'hello'
not len('') # 문자열이 비어 있으므로 참 반환
not len(greeting) # 문자열이 비어있지 않으므로 거짓 반환
```

## 문자열 찾기

이 블록들은 특정 문자열가 다른 문자열 안에 있는지 확인하고, 존재하는 경우 위치를 반환합니다. 예를 들어, 아래 블록은 “hello”에서 “e”的 첫 번째 등장 위치를 요청하며 결과는 2입니다.



Figure 59: Image

아래 블록은 “hello”에서 “e”的 마지막 등장 위치를 요청하며 결과는 동일하게 2입니다.



Figure 60: Image

**첫 번째 또는 마지막 옵션이 선택되더라도 “hello”에는 “z”가 포함되지 않으므로 결과는 0이 됩니다.**

## Javascript 코드

```
'hello'.indexOf('e') + 1; // "hello" 문자열에서 처음으로 'e'가 나타난 위치 값 2 반환
'hello'.lastIndexOf('e') + 1; // "hello" 문자열에서 마지막으로 'e'가 나타난 위치 값 2 반환
'hello'.indexOf('z') + 1; // "hello" 문자열에서 처음으로 'z'가 나타난 위치 값 0 반환
```



Figure 61: Image

## Python 코드

```
'hello'.find('e') + 1 # "hello" 문자열에서 처음으로 'e'가 나타난 위치 값 2 반환
'hello'.rfind('e') + 1 # "hello" 문자열에서 마지막으로 'e'가 나타난 위치 값 2 반환
'hello'.find('z') + 1 # "hello" 문자열에서 처음으로 'z'가 나타난 위치 값 0 반환
```

## 문자열 추출

### 단일 문자 추출

이 블록은 “`abcde`”에서 두 번째 문자 “`b`”를 가져옵니다:



Figure 62: Image

## Javascript 코드

```
'abcde'.charAt(1); // "abcde"에서 두 번째 문자 'b' 반환
```

## Python 코드

```
'abcde'[1] # "abcde"에서 두 번째 문자 'b' 반환
```

## Javascript 코드

```
function textRandomLetter(text) { // 무작위 문자 반환 함수
  var x = Math.floor(Math.random() * text.length);
  return text[x];
}

'abcde'.charAt(1); // 앞에서부터 두 번째 위치의 문자 'b' 얻기
'abcde'.slice(-2).charAt(0); // 마지막부터 두 번째 위치의 문자 'd' 얻기
'abcde'.charAt(0); // 첫 번째 문자 'a' 얻기
```



Figure 63: Image

```
'abcde'.slice(-1); // 마지막 문자 'e' 얻기
textRandomLetter('abcde'); // 랜덤하게 한 문자 얻기
```

## Python 코드

```
def text_random_letter(text): # 무작위 문자 반환 함수
    x = int(random.random() * len(text))
    return text[x]

'abcde'[1] # 앞에서부터 두 번째 위치의 문자 'b' 얻기
'abcde'[-2] # 마지막부터 두 번째 위치의 문자 'd' 얻기
'abcde'[0] # 첫 번째 문자 'a' 얻기
'abcde'[-1] # 마지막 문자 'e' 얻기
text_random_letter('abcde') # 랜덤하게 한 문자 얻기
```

## 문자열 일부 추출

**문자열에서…부분 문자열 가져오기** 블록을 사용하면 특정 범위의 문자열을 추출할 수 있습니다.

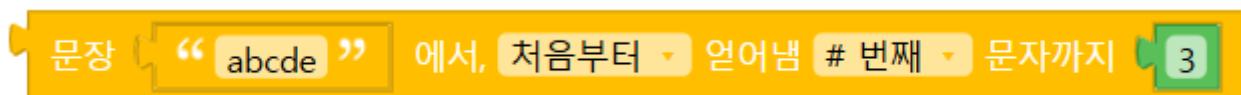


Figure 64: Image

위 블록은 “abc”를 추출합니다.

## Javascript 코드

```
'abcde'.slice(0, 3); // "abcde"에서 처음 ~ 세 번째 문자까지의 문자열 "abc" 반환
```

## Python 코드

```
'abcde'[ : 3] # "abcde"에서 처음 ~ 세 번째 문자까지의 문자열 "abc" 반환
```

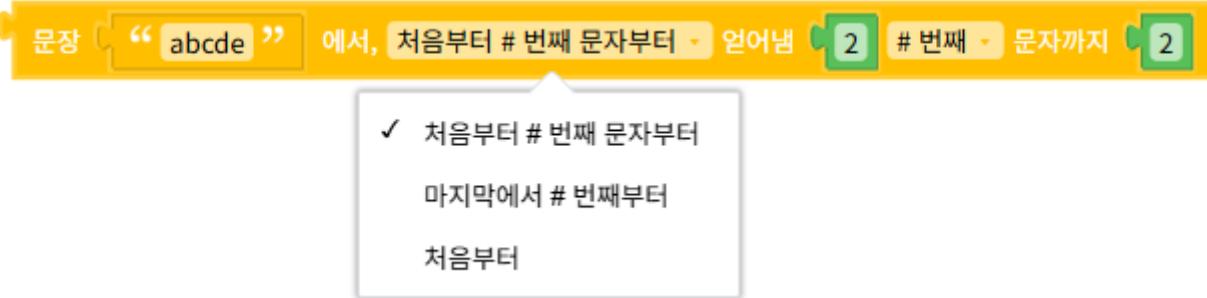


Figure 65: Image



Figure 66: Image

## Javascript 코드

```
'abcde'.slice(1, 4); // 'abcde' 처음부터 2 번째 문자부터 4 번째 문자까지의 문자열 'bcd' 추출  
'abcde'.slice(1, 'abcde'.length - 1); // 'abcde' 처음부터 2 번째 문자부터 끝에서부터 2 번째 문자까지의 문자열 'bcd' 추출  
'abcde'.slice(1, 'abcde'.length); // 'abcde' 처음부터 2 번째 문자부터 마지막 문자까지 문자열 'bcde' 추출  
'abcde'.slice('abcde'.length - 4, 4); // 'abcde' 마지막에서 4 번째 문자부터 4 번째 문자까지의 문자열 'bcd' 추출  
'abcde'.slice('abcde'.length - 4, 'abcde'.length - 1); // 'abcde' 마지막에서 4 번째 문자부터 끝에서부터 2 번째 문자까지의 문자열 'bcd' 추출  
'abcde'.slice('abcde'.length - 4, 'abcde'.length); // 'abcde' 마지막에서 4 번째 문자부터 마지막 문자까지의 문자열 'bcde' 추출  
'abcde'.slice(0, 4); // 'abcde' 처음부터 4 번째 문자까지의 문자열 'abcd' 추출  
'abcde'.slice(0, 'abcde'.length - 1); // 'abcde' 처음부터 끝에서부터 2 번째 문자까지의 문자열 'abcd' 추출  
'abcde'; // 'abcde' 처음부터 마지막 문자까지의 문자열 'abcde' 추출
```

## Python 코드

```
'abcde'[1 : 4] # 'abcde' 처음부터 2 번째 문자부터 4 번째 문자까지의 문자열 'bcd' 추출  
'abcde'[1 : -1] # 'abcde' 처음부터 2 번째 문자부터 끝에서부터 2 번째 문자까지의 문자열 'bcd' 추출  
'abcde'[1 : ] # 'abcde' 처음부터 2 번째 문자부터 마지막 문자까지 문자열 'bcde' 추출  
'abcde'[-4 : 4] # 'abcde' 마지막에서 4 번째 문자부터 4 번째 문자까지의 문자열 'bcd' 추출  
'abcde'[-4 : -1] # 'abcde' 마지막에서 4 번째 문자부터 끝에서부터 2 번째 문자까지의 문자열 'bcd' 추출  
'abcde'[-4 : ] # 'abcde' 마지막에서 4 번째 문자부터 마지막 문자까지의 문자열 'bcde' 추출  
'abcde'[1 : 4] # 'abcde' 처음부터 4 번째 문자까지의 문자열 'abcd' 추출  
'abcde'[1 : -1] # 'abcde' 처음부터 끝에서부터 2 번째 문자까지의 문자열 'abcd' 추출  
'abcde'[1 : ] # 'abcde' 처음부터 마지막 문자까지의 문자열 'abcde' 추출
```

## 문자열 대소문자 변경

이 블록은 입력 문자열을 다음 형식 중 하나로 변환합니다:

- **대문자** (모든 문자 대문자로 변환)
- **소문자**
- **첫 글자만 대문자** (각 단어의 첫 글자만 대문자로 변환)

아래 블록의 결과는 “`HELLO`”입니다.



Figure 67: Image

## Javascript 코드

```
function textToTitleCase(str) { // 첫 문자만 대문자로 변환하는 함수
    return str.replace(/\b[a-z]/g,
        function(txt) {return txt[0].toUpperCase() + txt.substring(1).toLowerCase();});
}

'hello'.toUpperCase(); // 문자열의 모든 문자를 대문자로 변환, 'HELLO' 반환
'hello'.toLowerCase(); // 문자열의 모든 문자를 소문자로 변환, 'hello' 반환
textToTitleCase('hello'); // 문자열의 각 단어를 대문자로 시작하고 나머지는 소문자로 변환, 'Hello' 반환
```

## Python 코드

```
'hello'.upper() # 문자열의 모든 문자를 대문자로 변환, 'HELLO' 반환
'hello'.lower() # 문자열의 모든 문자를 소문자로 변환, 'hello' 반환
'hello'.title() # 문자열의 각 단어를 대문자로 시작하고 나머지는 소문자로 변환, 'Hello' 반환
```

## 공백 제거

다음 블록은 문자열에서 다음 위치의 공백을 제거합니다: - 문자열의 앞쪽 - 문자열의 끝쪽 - 양쪽 모두

아래 블록의 결과는 “`hi there`”입니다. (중간의 공백은 그대로 유지됨)

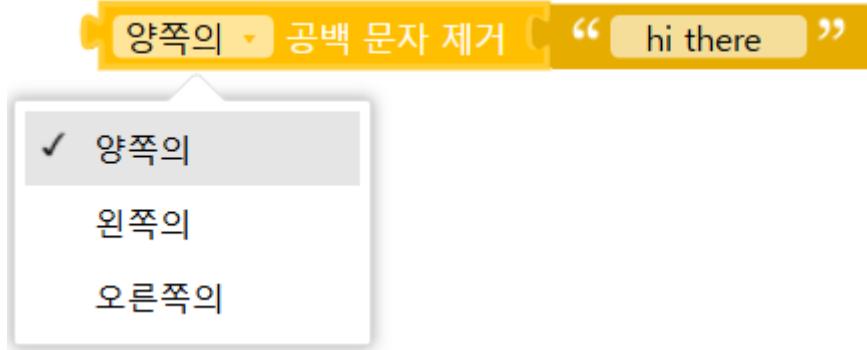


Figure 68: Image

## Javascript 코드

```
'hi there'.trim(); // 양쪽의 공백 문자 제거, 'hi there' 반환
'hi there'.replace(/^\s*\xa0*/,''); // 왼쪽의 공백 문자 제거, 'hi there' 반환
'hi there'.replace(/[\s\xA0]+$/,''); // 오른쪽의 공백 문자 제거, 'hi there' 반환
```

## Python 코드

```
'hi there'.strip() # 양쪽의 공백 문자 제거, 'hi there' 반환
'hi there'.lstrip() # 왼쪽의 공백 문자 제거, 'hi there' 반환
'hi there'.rstrip() # 오른쪽의 공백 문자 제거, 'hi there' 반환
```

## 문자열에서 특정 문자열 숫자 세기

주어진 문자열에서 특정한 단어 (부분 문자열) 가 등장하는 횟수를 세어 반환하는 기능입니다.



Figure 69: Image

## Javascript 코드

```
function textCount(haystack, needle) { // 특정 문자열 집계 함수
  if (needle.length === 0) {
    return haystack.length + 1;
  } else {
    return haystack.split(needle).length - 1;
  }
}

textCount('abc, abc, ab', 'abc'); // "abc, abc, ab" 문자열에서 "abc" 의 갯수 2 반환
```

## Python 코드

```
'abc, abc, ab'.count('abc') # "abc, abc, ab" 문자열에서 "abc" 의 갯수 2 반환
```

"abc, abc, ab"에서 "abc" 와 일치하는 경우 2를 반환합니다.

## 문자열에서 특정 문자 변경

문자열 내에서 특정 문자를 다른 문자로 변경하는 기능입니다. 원하는 문자를 선택하여 새로운 문자로 대체할 수 있습니다.

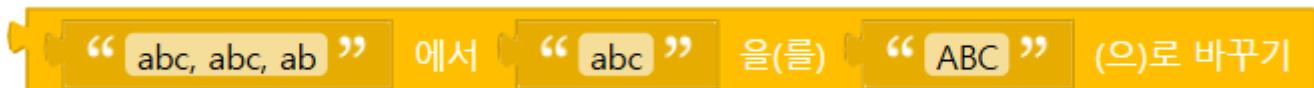


Figure 70: Image

문자열 "abc, abc, ab"에서 "abc"를 "ABC"로 변경합니다. 해당 블록의 실행 결과는 "ABC, ABC, ab"입니다.

## Javascript 코드

```
function textReplace(haystack, needle, replacement) { // 문자열에서의 특정 문자 변경 함수  
    needle = needle.replace(/([^(0)\{\}]+?[^\\{}],?:<\\{}))/g, '\\$1').replace(/\x08/g, '\\x08');  
    return haystack.replace(new RegExp(needle, 'g'), replacement);  
}  
  
textReplace('abc, abc, ab', 'abc', 'ABC'); // "abc, abc, ab" 문자열에서 "abc"를 "ABC"로 변환시 문자열 "ABC, ABC, ab" 반환
```

## Python 코드

```
'abc, abc, ab'.replace('abc', 'ABC') # "abc, abc, ab" 문자열에서 "abc"를 "ABC"로 변환시 문자열 "ABC, ABC, ab" 반환
```

## 문자열 뒤집기

문자열의 순서를 거꾸로 바꾼 새로운 문자열을 반환하는 기능입니다.



Figure 71: Image

“hello”를 뒤집은 “olleh”를 반환합니다.

## Javascript 코드

```
'hello'.split('').reverse().join('') // hello 문자열 뒤집은 결과인 "olleh" 반환
```

## Python 코드

```
'hello'[::-1] # hello 문자열 뒤집은 결과인 "olleh" 반환
```

## 문자열 출력

출력 블록은 입력 값을 팝업 창에 표시합니다.



Figure 72: Image

robomationlab.com 내용:

Hello!

확인

Figure 73: Image

## Javascript 코드

```
window.alert('Hello!'); // 팝업창에 "HELLO" 출력
```

## Python 코드

```
print('Hello!') # 팝업창에 "HELLO" 출력
```

## 사용자 입력 받기

다음 블록은 사용자에게 이름 입력을 요청하는 팝업 창을 생성하며, 입력된 값은 `name` 변수에 저장됩니다.



Figure 74: Image

## Javascript 코드

```
var name2;  
  
name2 = window.prompt('이름을 입력하세요:');
```

## Python 코드

```
name = None  
  
def text_prompt(msg): # 입력받는 함수  
    try:  
        return raw_input(msg)  
    except NameError:  
        return input(msg)  
  
name = text_prompt('이름을 입력하세요:')
```

또한 사용자로부터 숫자를 입력받는 버전도 있습니다.



Figure 75: Image

## Javascript 코드

```
var age;  
  
age = Number(window.prompt('나이를 입력하세요:'));
```

## Python 코드

```
age = None  
  
def text_prompt(msg): # 입력받는 함수  
    try:  
        return raw_input(msg)  
    except NameError:  
        return input(msg)  
age = float(text_prompt('나이를 입력하세요:'))
```

## 반복

제어 블록에는 두 가지 유형이 있습니다: 조건문과 **반복문** (변수들의 값에 따라 본문을 몇 번 실행할지 제어하는 것들)

## 반복 생성 블록

### 반복

가장 간단한 **반복** 블록은 본문의 코드를 지정된 횟수만큼 실행합니다. 예를 들어, 아래의 블록은 “Hello!”를 열 번 출력합니다.



Figure 76: Image

### Javascript 코드

```
for (var count = 0; count < 10; count++) { // 10 회 반복
    window.alert('Hello');
}
```

### Python 코드

```
for count in range(10): # 10 회 반복
    print('Hello')
```

### 동안 반복하기

플레이어가 주사위를 굴리고 총합이 30 보다 작을 때까지 값을 더하는 게임을 상상해 보세요. 아래 블록들은 그 게임을 구현한 것입니다:

1. **total**이라는 변수가 0으로 초기화됩니다.
2. 반복문은 **total**이 30 보다 작은지 확인하는 것으로 시작합니다. 그렇다면 본문의 블록들이 실행됩니다.
3. 1에서 6 까지의 랜덤 숫자가 생성되어 (주사위 굴리기) **roll**이라는 변수에 저장됩니다.
4. 나온 숫자가 출력됩니다.
5. **total** 변수는 **roll** 만큼 증가합니다.
6. 반복문이 끝나면 제어가 2 단계로 돌아갑니다.

반복문이 완료되면 그 이후의 블록들이 실행됩니다 (생략된 부분). 이 예시에서는 1에서 6 까지의 랜덤 숫자가 몇 번 출력된 후 반복문은 종료되고, **total** 변수에는 그 합이 저장됩니다. 이 합은 최소한 30 이 됩니다.

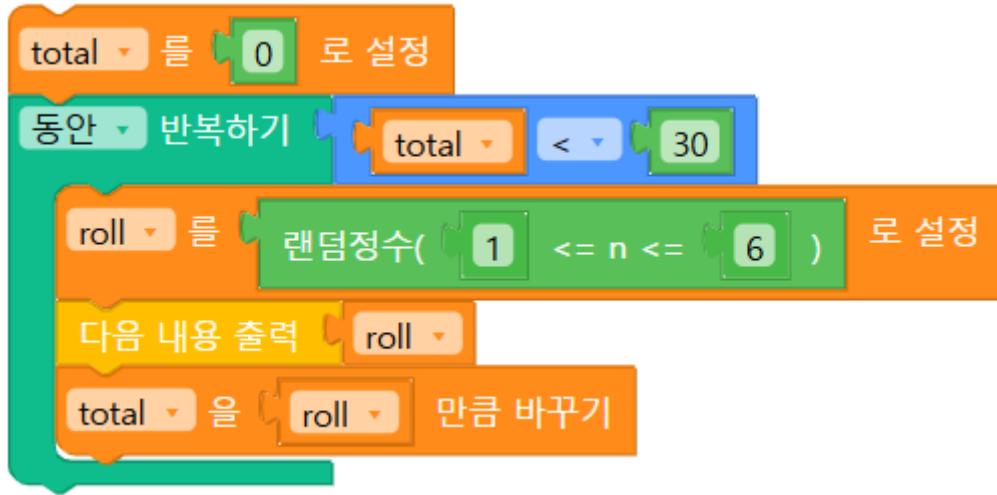


Figure 77: Image

## Javascript 코드

```

var roll, total;

function mathRandomInt(a, b) { // 무작위 수 함수
  if (a > b) {
    var c = a;
    a = b;
    b = c;
  }
  return Math.floor(Math.random() * (b - a + 1) + a);
}

total = 0;
while(total < 30) { // 30 이하인 동안 반복하기
  roll = mathRandomInt(1, 6);
  window.alert(roll);
  total = (typeof total === 'number' ? total : 0) + roll;
  await __wait(10);
}

```

## Python 코드

```

import asyncio
import random
from numbers import Number

roll = None
total = None

total = 0
while total < 30: # 30 이하인 동안 반복하기
  roll = random.randint(1, 6)
  print(roll)
  total = (total if isinstance(total, Number) else 0) + roll
  await asyncio.sleep(0.01)

```

## 까지 반복하기

**까지 반복하기** 반복문은 조건이 거짓인 동안 본문을 반복하고, 조건이 참이 되는 순간 반복문을 탈출합니다. 아래 블록들은 이전 예시와 동일한 결과를 도출합니다. 반복문은 **total** 이 30 이상일 때까지 반복됩니다.



Figure 78: Image

## Javascript 코드

```
var total, roll;

function mathRandomInt(a, b) {
    if (a > b) { // 무작위 수 함수
        var c = a;
        a = b;
        b = c;
    }
    return Math.floor(Math.random() * (b - a + 1) + a);
}

total = 0;
while(!(total >= 30)) { // 30 이상이 될 때까지 반복하기
    roll = mathRandomInt(1, 6);
    window.alert(roll);
    total = (typeof total === 'number' ? total : 0) + roll;
    await __wait(10);
}
```

## Python 코드

```
import asyncio
import random
from numbers import Number

total = None
roll = None

total = 0
while not (total >= 30): # 30 이상이 될 때까지 반복하기
    roll = random.randint(1, 6)
    print(roll)
    total = (total if isinstance(total, Number) else 0) + roll
    await asyncio.sleep(0.01)
```

## 으로 계산

으로 계산 블록 (대부분의 프로그래밍 언어에서 **for loop**라고 부름)은 변수를 첫 번째 값에서 두 번째 값까지 증가량 (두 번째 값) 만큼 증가시키면서 본문을 각 값에 대해 한 번씩 실행합니다. 예를 들어, 아래 프로그램은 1, 3, 5 를 출력합니다.



Figure 79: Image

## Javascript 코드

```
var index;  
  
for (index = 1; index <= 5; index += 2) {  
    window.alert(index);  
}
```

## Python 코드

```
index = None  
  
for index in range(1, 6, 2):  
    print(index)
```

다음 두 반복문은 첫 번째 입력이 두 번째 입력보다 클 수도 있습니다. 세 번째 값 (증가량) 이 양수인지 음수인지에 관계없이 동일한 동작을 합니다.



Figure 80: Image

## Javascript 코드

```
var index;  
  
for (index = 5; index >= 1; index -= 2) {  
    window.alert(index);  
}
```

## Python 코드

```
index = None  
  
for index in range(5, 0, -2):  
    print(index)
```

## 각 항목에 대해

각 항목에 대해 블록은 숫자 순서 대신 목록의 값을 차례대로 사용하는 점에서 유사합니다. 아래 프로그램은 “첫 번째”, “두 번째”, “세 번째” 목록의 각 항목을 출력합니다.

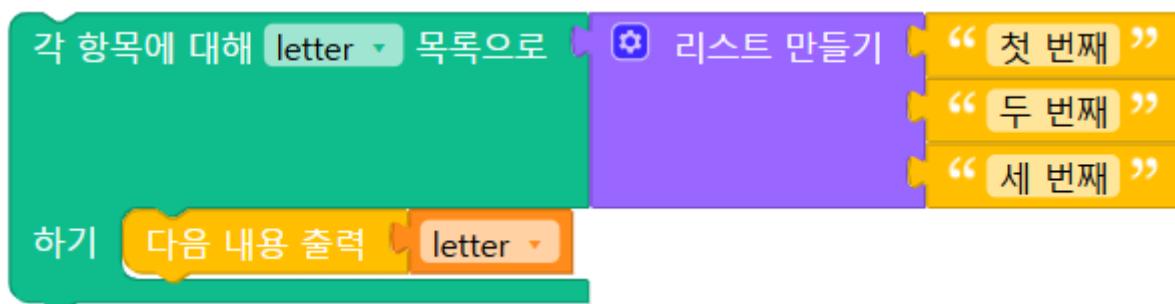


Figure 81: Image

## Javascript 코드

```
var letter;  
  
var letter_list = ['첫 번째', '두 번째', '세 번째'];  
for (var letter_index in letter_list) { // 각 항목에 대해  
    letter = letter_list[letter_index];  
    window.alert(letter);  
}
```

## Python 코드

```
letter = None  
  
for letter in ['첫 번째', '두 번째', '세 번째']: # 각 항목에 대해  
    print(letter)
```

## 반복문 종료 블록

대부분의 반복문은 **종료 조건이 충족될 때까지** (반복 블록의 경우) 또는 **반복문 변수에 의해 값이 모두 처리될 때까지** 실행됩니다 (**으로 계산과 각 항목에 대해** 블록의 경우). 드물게 사용되지만 유용할 수 있는 두 가지 블록은 반복문 동작을 제어하는 추가적인 방법을 제공합니다. 아래 예시는 **각 항목에 대해** 반복문에 해당하지만, 모든 유형의 반복문에 사용할 수 있습니다.

### 다음 반복

**다음 반복** (대부분의 프로그래밍 언어에서 **continue**)은 본문의 남은 코드를 건너뛰고 다음 반복 (패스) 을 시작하게 만듭니다.

다음 프로그램은 반복문은 첫 번째 반복에서 “첫 번째”를 출력합니다. 두 번째 반복에서는 **다음 반복** 블록이 실행되어 “두 번째” 출력이 건너뛰어 마지막 반복에서 “세 번째”가 출력됩니다.



Figure 82: Image

### Javascript 코드

```
var letter;

var letter_list = ['첫 번째', '두 번째', '세 번째'];
for (var letter_index in letter_list) {
  letter = letter_list[letter_index];
  if (letter == '두 번째') {
    continue; // 다음 반복
  }
  window.alert(letter);
}
```

## Python 코드

```
letter = None

for letter in ['첫 번째', '두 번째', '세 번째']:
    if letter == '두 번째':
        continue # 다음 반복
    print(letter)
```

## 반복 중단

반복 중단 블록은 반복문에서 일찍 빠져나올 수 있게 해줍니다. 아래 프로그램은 첫 번째 반복에서 “첫 번째”를 출력하고, 두 번째 반복에서 반복문은 변수 값이 “두 번째”일 때 반복 중단 블록이 실행되어 반복문에서 빠져나옵니다. 리스트의 “세 번째” 항목은 출력되지 않습니다.

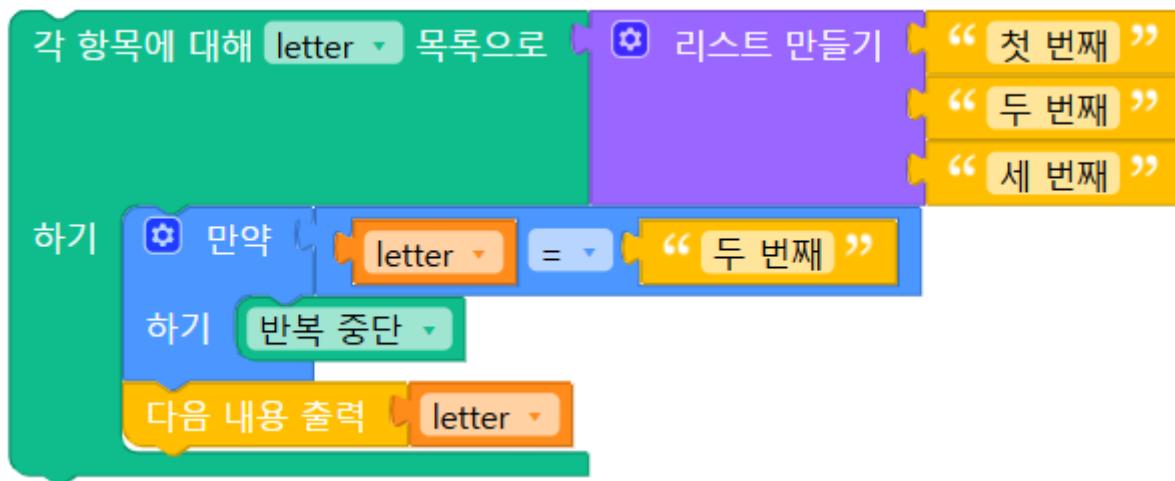


Figure 83: Image

## Javascript 코드

```
var letter;

var letter_list = ['첫 번째', '두 번째', '세 번째'];
for (var letter_index in letter_list) {
    letter = letter_list[letter_index];
    if (letter == '두 번째') {
        break; // 반복 중단
    }
    window.alert(letter);
}
```

## Python 코드

```
letter = None

for letter in ['첫 번째', '두 번째', '세 번째']:
    if letter == '두 번째':
        break # 반복 중단
    print(letter)
```

## 변수

---

우리는 변수라는 용어를 수학이나 다른 프로그래밍 언어에서 사용되는 것과 동일한 의미로 사용합니다. 즉, **값을 저장하며 변경할 수 있는 이름을 가진 요소**를 의미합니다. 변수는 여러 가지 방법으로 생성할 수 있습니다.

- 으로 계산 및 각 항목에 대해와 같은 일부 블록은 변수를 사용하며, 해당 변수의 값을 정의합니다. 이러한 변수는 전통적인 컴퓨터 과학 용어로 **반복 변수 (loop variables)**라고 합니다.
- **사용자 정의 함수**는 입력값을 정의할 수 있으며, 이는 함수 내에서만 사용할 수 있는 변수 (매개변수 또는 인자)를 생성합니다.
- 사용자는 **설정** 블록을 통해 언제든지 변수를 생성할 수 있습니다. 이는 전통적으로 “**전역 변수 (global variables)**”라고 불립니다.
- 블록 컴포저는 **지역 변수 (local variables)**를 지원하지 않습니다.

## 드롭다운 메뉴

변수의 드롭다운 기호 (삼각형)를 클릭하면 다음과 같은 메뉴가 나타납니다.



Figure 84: Image

이 메뉴에서 다음 옵션을 선택할 수 있습니다.

- 프로그램에서 정의된 모든 기존 변수 이름이 표시됩니다.
- “**변수 이름 바꾸기**”: 프로그램 전체에서 해당 변수의 이름을 변경합니다. 이 옵션을 선택하면 새 이름을 입력할 수 있는 창이 나타납니다.

- “변수 삭제”: 프로그램에서 이 변수를 참조하는 모든 블록을 삭제합니다.

## 블록

### 설정

설정 블록은 변수에 값을 할당하며, 해당 변수가 존재하지 않을 경우 새 변수를 생성합니다. 예를 들어, 아래 블록은 “age”라는 변수를 만들고 값 12를 할당합니다.



Figure 85: Image

### Javascript 코드

```
var age;
age = 12; // age 변수에 12 할당
```

### Python 코드

```
age = None
age = 12 # age 변수에 12 할당
```

## 값 가져오기

아래 블록은 변수에 저장된 값을 제공하지만, 해당 값을 변경하지는 않습니다.



Figure 86: 이미지

설정 블록 없이 블록을 사용하는 것도 가능하지만, 이는 올바른 프로그래밍 방식이 아닙니다.

### Javascript 코드

```
age; // 변수 값 가져오기
```

## Python 코드

```
age # 변수 값 가져오기
```

## 바꾸기

바꾸기 블록은 변수의 값에 숫자를 더하는 역할을 합니다.



Figure 87: Image

위 블록은 다음 코드와 동일한 기능을 수행하는 단축 표현입니다.



Figure 88: Image

## Javascript 코드

```
var age;  
age = (typeof age === 'number' ? age : 0) + 1; // age 1 만큼 더하기
```

## Python 코드

```
from numbers import Number  
  
age = None  
age = (age if isinstance(age, Number) else 0) + 1 # age 1 만큼 더하기
```

## 예제

다음은 변수 사용 예제입니다.

첫 번째 줄의 블록은 `"age"`라는 변수를 만들고 초기 값을 숫자 `12`로 설정합니다.

두 번째 줄의 블록은 값 `12`를 가져와서 `1`을 더한 후, 그 합 (`13`)을 변수에 저장합니다.

마지막 줄은 다음 메시지를 표시합니다: “`13` 살 생일축하해!”



Figure 89: Image

## Javascript 코드

```
var age;
age = 12;
age = (typeof age === 'number' ? age : 0) + 1; // age 1 만큼 더하기
window.alert(String(age) + '살 생일축하해!');
```

## Python 코드

```
from numbers import Number

age = None

age = 12
age = (age if isinstance(age, Number) else 0) + 1 # age 1 만큼 더하기
print(str(age) + '살 생일축하해!')
```

## 비글

---

## 블록

### 바퀴 속도 설정하기

비글의 바퀴 속도를 설정합니다.

바퀴 속도가 양수이면 앞쪽 방향으로 회전하고, 바퀴 속도가 음수이면 뒤쪽 방향으로 회전합니다.

예를 들어, 바퀴 속도가 100 이라면, 앞쪽 방향으로 100 의 속도로 회전하고,

바퀴 속도가 -100 이라면, 뒤쪽 방향으로 100 의 속도로 회전합니다.

한번 바퀴 속도를 설정하면, 다시 바퀴 속도를 설정하기 전까지 해당 속도로 비글이 이동합니다.



비글 : 왼쪽 ▾ 바퀴 속도를 50 (으)로 정하기

50

바퀴 속도를 결정합니다. 속도의 범위는 -128 ~ 127 입니다.

왼쪽

오른쪽

양쪽

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	바퀴 속도	-100 ~ 100 정수, 0: 정지

## 자바스크립트 코드

```
// 비글 왼쪽 바퀴 속도를 100 으로 정하기
if($('Beagle*0:wheel.move').d != 0) {
  $('Beagle*0:wheel.move').d = 0;
}
$('Beagle*0:wheel.speed.left').d = __getSpeed('Beagle*0', 100);

// 비글 오른쪽 바퀴 속도를 -100 으로 정하기
if($('Beagle*0:wheel.move').d != 0) {
  $('Beagle*0:wheel.move').d = 0;
}
$('Beagle*0:wheel.speed.right').d = __getSpeed('Beagle*0', -100);

// 비글 양쪽 바퀴 속도를 100 으로 정하기
if($('Beagle*0:wheel.move').d != 0) {
  $('Beagle*0:wheel.move').d = 0;
}
$('Beagle*0:wheel.speed.left').d = __getSpeed('Beagle*0', 100);
$('Beagle*0:wheel.speed.right').d = __getSpeed('Beagle*0', 100);
```

## 파이썬 코드

```
# 비글 왼쪽 바퀴 속도를 100 으로 정하기
if __('Beagle*0:wheel.move').d != 0:
  __('Beagle*0:wheel.move').d = 0
__('Beagle*0:wheel.speed.left').d = __getSpeed('Beagle*0', 100)

# 비글 오른쪽 바퀴 속도를 -100 으로 정하기
if __('Beagle*0:wheel.move').d != 0:
  __('Beagle*0:wheel.move').d = 0
__('Beagle*0:wheel.speed.right').d = __getSpeed('Beagle*0', -100)
```

```
# 비글 양쪽 바퀴 속도를 100 으로 정하기
if __('Beagle*0:wheel.move').d != 0:
    __('Beagle*0:wheel.move').d = 0
__('Beagle*0:wheel.speed.left').d = __getSpeed('Beagle*0', 100)
__('Beagle*0:wheel.speed.right').d = __getSpeed('Beagle*0', 100)
```

## 거리 이동하기

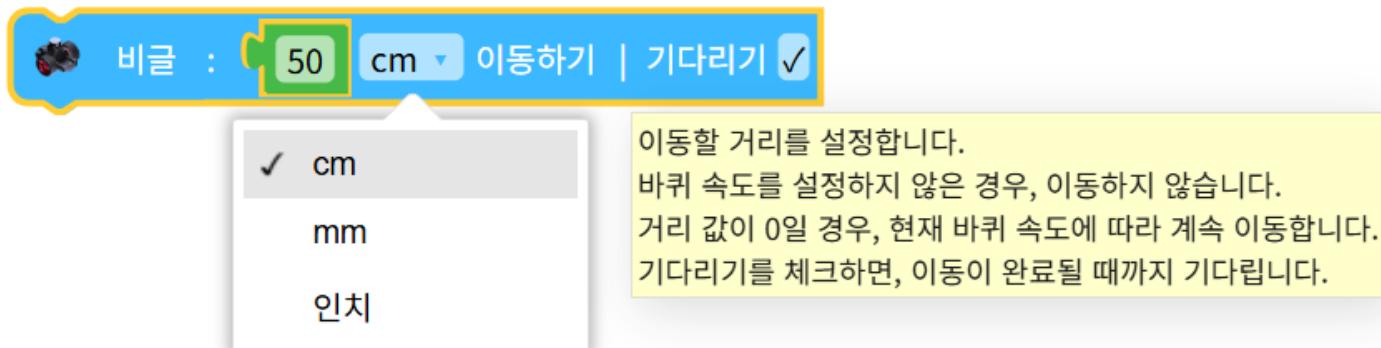
비글이 이동할 거리를 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

거리 값이 0 일 경우에는, 현재 설정되어 있는 바퀴 속도대로 멈추지 않고 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
distance	입력값	거리 값	0 이상 실수
unit	드롭다운 옵션	거리 단위	cm, mm, 인치 (inch)

## 자바스크립트 코드

```
// 50cm 이동하기 | 기다리기 0
$('.Beagle*0:wheel.move').d = __getDistance('Beagle*0', 50, 'cm');
await $('.Beagle*0:wheel.!move').w();

// 50mm 이동하기 | 기다리기 X
$('.Beagle*0:wheel.move').d = __getDistance('Beagle*0', 50, 'mm');

// 50inch 이동하기 | 기다리기 X
$('.Beagle*0:wheel.move').d = __getDistance('Beagle*0', 50, 'inch');
```

## 파이썬 코드

```
# 50cm 이동하기 | 기다리기 0
__('Beagle*0:wheel.move').d = __getDistance('Beagle*0', 50, 'cm')
await __('Beagle*0:wheel.!move').w()

# 50mm 이동하기 | 기다리기 X
__('Beagle*0:wheel.move').d = __getDistance('Beagle*0', 50, 'mm')

# 50inch 이동하기 | 기다리기 X
__('Beagle*0:wheel.move').d = __getDistance('Beagle*0', 50, 'inch')
```

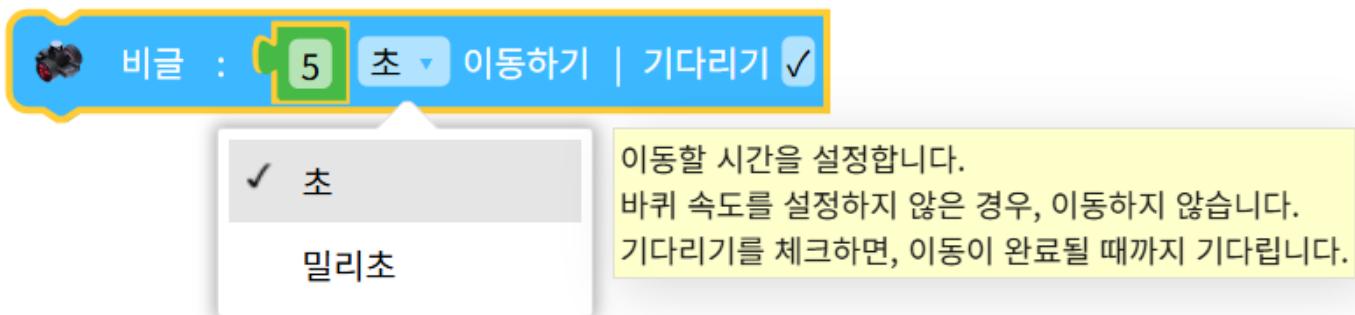
## 시간 이동하기

비글이 이동할 시간을 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
time	입력값	시간 값	0 이상 실수
unit	드롭다운 옵션	시간 단위	초 (seconds), 밀리초 (milliseconds)

옵션을 밀리초 (milliseconds)로 설정한 경우에는, `time` 값을 1000으로 나눈 값이 입력됩니다.

## 자바스크립트 코드

```
// 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Beagle*0', 5, true);

// 5 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Beagle*0', 0.005, false);
```

## 파이썬 코드

```
# 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Beagle*0', 5, True)

# 5 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Beagle*0', 0.005, False)
```

## 제자리 돌기

비글이 제자리에서 회전할 방향과 각도를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	회전 방향	왼쪽 (left), 오른쪽 (right)
degree	입력값	회전 각도	0 이상 정수

## 자바스크립트 코드

```
// 왼쪽으로 90 도 제자리 돌기 | 기다리기 0
await __turn_degree_left('Beagle*0', 90, true);

// 오른쪽으로 270 도 제자리 돌기 | 기다리기 X
__turn_degree_right('Beagle*0', 270, false);
```

## 파이썬 코드

```
# 왼쪽으로 90 도 제자리 돌기 | 기다리기 0
await __turn_degree_left('Beagle*0', 90, True)

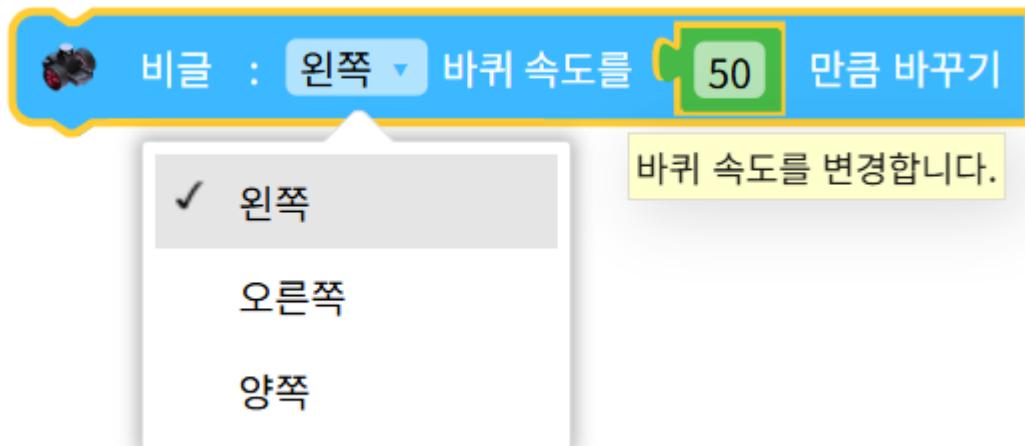
# 오른쪽으로 270 도 제자리 돌기 | 기다리기 X
__turn_degree_right('Beagle*0', 270, False)
```

## 바퀴 속도 변경하기

비글의 바퀴 속도를 변경합니다.

현재의 바퀴 속도에 입력한 속도를 더한 값이 새로운 바퀴 속도가 됩니다.

새롭게 설정된 바퀴 속도의 범위는 -100 ~ 100 으로 설정됩니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	현재 바퀴 속도에 더할 속도 값	-200 ~ 200 정수

## 자바스크립트 코드

```
// 왼쪽 바퀴 속도를 50 만큼 바꾸기
if($('Beagle*0:wheel.move').d != 0) {
    $('Beagle*0:wheel.move').d = 0;
}
$('Beagle*0:wheel.speed.left').d = $('Beagle*0:wheel.speed.left').d + __getSpeed('Beagle*0', 50);

// 오른쪽 바퀴 속도를 40 만큼 바꾸기
if($('Beagle*0:wheel.move').d != 0) {
    $('Beagle*0:wheel.move').d = 0;
}
$('Beagle*0:wheel.speed.right').d = $('Beagle*0:wheel.speed.right').d + __getSpeed('Beagle*0', 40);

// 양쪽 바퀴 속도를 30 만큼 바꾸기
if($('Beagle*0:wheel.move').d != 0) {
    $('Beagle*0:wheel.move').d = 0;
}
$('Beagle*0:wheel.speed.left').d = $('Beagle*0:wheel.speed.left').d + __getSpeed('Beagle*0', 30);
$('Beagle*0:wheel.speed.right').d = $('Beagle*0:wheel.speed.right').d + __getSpeed('Beagle*0', 30);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도를 50 만큼 바꾸기
if __('Beagle*0:wheel.move').d != 0:
    __('Beagle*0:wheel.move').d = 0
    __('Beagle*0:wheel.speed.left').d = __('Beagle*0:wheel.speed.left').d + __getSpeed('Beagle*0', 50)

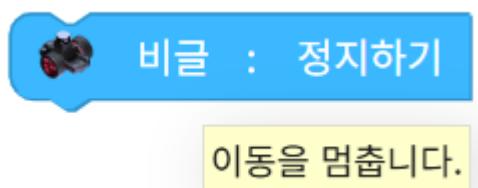
# 오른쪽 바퀴 속도를 40 만큼 바꾸기
if __('Beagle*0:wheel.move').d != 0:
    __('Beagle*0:wheel.move').d = 0
    __('Beagle*0:wheel.speed.right').d = __('Beagle*0:wheel.speed.right').d + __getSpeed('Beagle*0', 40)

# 양쪽 바퀴 속도를 30 만큼 바꾸기
if __('Beagle*0:wheel.move').d != 0:
    __('Beagle*0:wheel.move').d = 0
    __('Beagle*0:wheel.speed.left').d = __('Beagle*0:wheel.speed.left').d + __getSpeed('Beagle*0', 30)
    __('Beagle*0:wheel.speed.right').d = __('Beagle*0:wheel.speed.right').d + __getSpeed('Beagle*0', 30)
```

## 정지하기

비글의 이동을 멈춥니다.

비글의 양쪽 바퀴 속도가 모두 0으로 초기화됩니다.



## 자바스크립트 코드

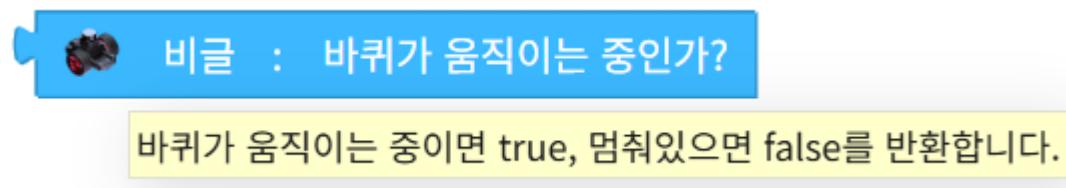
```
__stopMove('Beagle*0');
```

## 파이썬 코드

```
__stopMove('Beagle*0')
```

## 바퀴가 움직이는 중인가?

비글의 바퀴가 움직이고 있는지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 자바스크립트 코드

```
$(`'Beagle*0:wheel.speed.left').d != 0 || $(`'Beagle*0:wheel.speed.right').d != 0
```

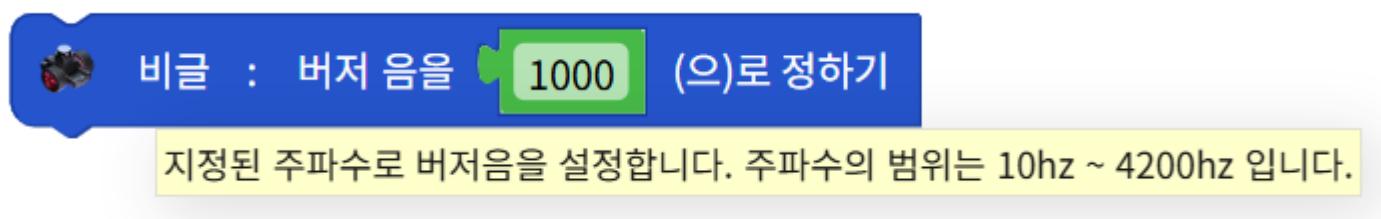
## 파이썬 코드

```
__('Beagle*0:wheel.speed.left').d != 0 or __('Beagle*0:wheel.speed.right').d != 0
```

## 버저음 설정하기

지정된 주파수로 비글의 버저음을 설정합니다.

주파수의 범위는 10hz ~ 4200hz 입니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound	입력값	버저음 주파수	10 ~ 4200(hz)

## 자바스크립트 코드

```
// 버저음 주파수 4200hz 로 설정하기  
__stopSound('Beagle*0');  
$('Beagle*0:sound.buzz').d = 4200;
```

## 파이썬 코드

```
# 버저음 주파수 4200hz 로 설정하기  
__stopSound('Beagle*0')  
__('Beagle*0:sound.buzz').d = 4200
```

## 음계 연주하기

비글이 지정된 음계를 재생합니다.



비글 : 시 7 음을 연주하기

도#(레 ↞ )      특정 음계를 재생합니다.

레

레#(미 ↞ )

미

파

파#(솔 ↞ )

솔

솔#(라 ↞ )

라

라#(시 ↞ )

#### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
note	드롭다운 옵션	음계	도 (Do), 도 #(Do#), 레 (Re), 레 #(Re#), 미 (Mi), 파 (Fa), 파 #(Fa#), 솔 (So), 솔 #(So#), 라 (La), 라 #(La#), 시 (Ti)
octave	드롭다운 옵션	옥타브	1 ~ 7

## 자바스크립트 코드

```
// 1 옥타브 도 (Do) 음을 연주하기
__stopSound('Beagle*0');
$('Beagle*0:sound.note').d = 4;

// 1 옥타브 레 (Re) 음을 연주하기
__stopSound('Beagle*0');
$('Beagle*0:sound.note').d = 6;

// 2 옥타브 도 (Do) 음을 연주하기
__stopSound('Beagle*0');
$('Beagle*0:sound.note').d = 16;

// 7 옥타브 시 (Ti) 음을 연주하기
__stopSound('Beagle*0');
$('Beagle*0:sound.note').d = 87;
```

## 파이썬 코드

```
# 1 옥타브 도 (Do) 음을 연주하기
__stopSound('Beagle*0')
__('Beagle*0:sound.note').d = 4

# 1 옥타브 레 (Re) 음을 연주하기
__stopSound('Beagle*0')
__('Beagle*0:sound.note').d = 6

# 2 옥타브 도 (Do) 음을 연주하기
__stopSound('Beagle*0')
__('Beagle*0:sound.note').d = 16

# 7 옥타브 시 (Ti) 음을 연주하기
__stopSound('Beagle*0')
__('Beagle*0:sound.note').d = 87
```

## 소리 재생하기

비글이 특정 사운드 클립을 재생합니다.

기다리기를 체크하면, 재생이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound_clip	드롭다운 옵션	사운드 클립	beep(1), beep2(2), beep3(3), beep_repeat(4), beep_random(5), beep_random_repeat(6), snore(7), snore_repeat(8), siren(9), siren_repeat(10), engine(11), engine_repeat(12), fart_a(13), fart_b(14), noise(15), noise_repeat(16), whistle(17), chop_chop(18), chop_chop_repeat(19), r2d2(32), dibidibidip(33), simple_melody(35), happy(48), angry(49), sad(50), sleep(51), march(52), birthday(53)

## 자바스크립트 코드

```
// dibidibidip 소리 재생하기 | 기다리기 0
__stopSound('Beagle*0');
$('Beagle*0:sound.clip').d = 33;
await $('Beagle*0:sound.!clip').w();

// happy 소리 재생하기 | 기다리기 X
__stopSound('Beagle*0');
$('Beagle*0:sound.clip').d = 48;
```

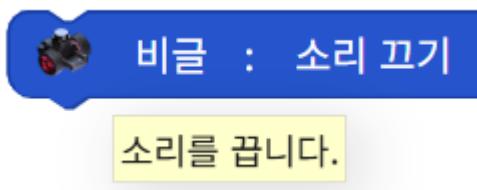
## 파이썬 코드

```
# dibidibidip 소리 재생하기 | 기다리기 0
__stopSound('Beagle*0')
__('Beagle*0:sound.clip').d = 33
await __('Beagle*0:sound.!clip').w()

# happy 소리 재생하기 | 기다리기 X
__stopSound('Beagle*0')
__('Beagle*0:sound.clip').d = 48
```

## 소리 끄기

비글의 소리를 끕니다.



## 자바스크립트 코드

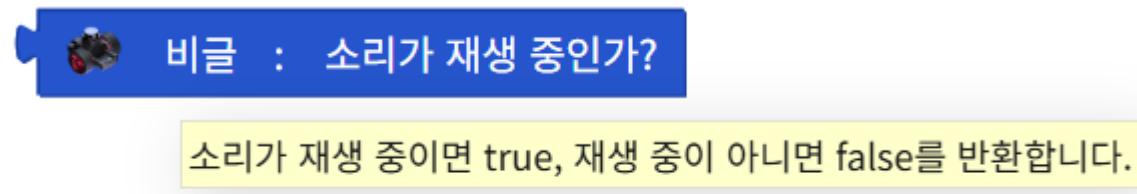
```
// 비글 소리 끄기
__stopSound('Beagle*0');
```

## 파이썬 코드

```
# 비글 소리 끄기
__stopSound('Beagle*0')
```

## 소리가 재생 중인가?

비글의 소리가 재생중인지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 자바스크립트 코드

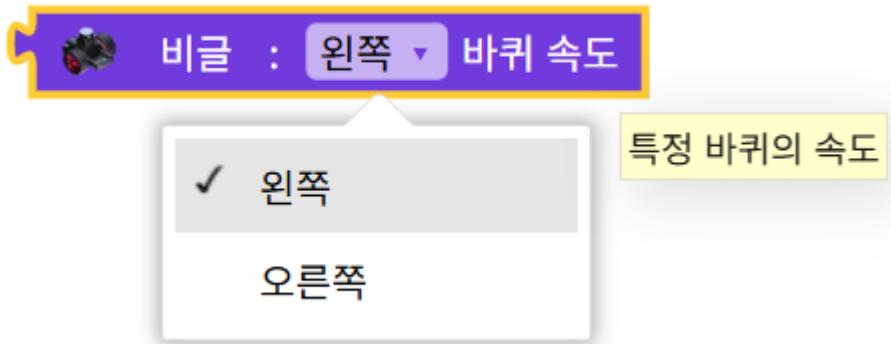
```
// 비글의 소리가 재생 중인가? - 재생 시 true, 아닐시 false  
$(`Beagle*0:sound.playing`).d;
```

## 파이썬 코드

```
# 비글의 소리가 재생 중인가? - 재생 시 True, 아닐시 False  
__(`Beagle*0:sound.playing').d
```

## 바퀴 속도 값

비글의 지정한 바퀴 속도 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

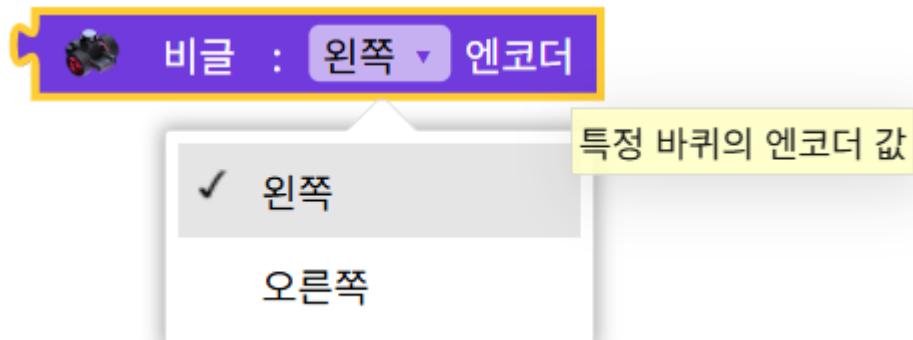
```
// 왼쪽 바퀴 속도  
__getSpeedInput('Beagle*0', $('Beagle*0:wheel.speed.left').d);  
  
// 오른쪽 바퀴 속도  
__getSpeedInput('Beagle*0', $('Beagle*0:wheel.speed.right').d);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도  
__getSpeedInput('Beagle*0', __(`Beagle*0:wheel.speed.left`).d)  
  
# 오른쪽 바퀴 속도  
__getSpeedInput('Beagle*0', __(`Beagle*0:wheel.speed.right`).d)
```

## 엔코더 값

비글의 지정한 바퀴의 엔코더 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
// 원쪽 바퀴 엔코더 값  
$(`'Beagle*0:encoder.left').d;  
  
// 오른쪽 바퀴 엔코더 값  
$(`'Beagle*0:encoder.right').d;
```

## 파이썬 코드

```
# 원쪽 바퀴 엔코더 값  
__(`'Beagle*0:encoder.left').d  
  
# 오른쪽 바퀴 엔코더 값  
__(`'Beagle*0:encoder.right').d
```

## 자이로 센서 값

비글의 특정 축의 자이로 센서의 값을 가져옵니다.



비글 : x축 ▾ 자이로 센서 ( $^{\circ}/s$ )

✓ x축

특정 축의 자이로 센서의 값

y축

z축

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	축 기준	x 축 (x), y 축 (y), z 축 (z)

## 자바스크립트 코드

```
// x 축 자이로 센서
$('Beagle*0:gyroscope.x').d;

// y 축 자이로 센서
$('Beagle*0:gyroscope.y').d;

// z 축 자이로 센서
$('Beagle*0:gyroscope.z').d;
```

## 파이썬 코드

```
# x 축 자이로 센서
__('Beagle*0:gyroscope.x').d

# y 축 자이로 센서
__('Beagle*0:gyroscope.y').d

# z 축 자이로 센서
__('Beagle*0:gyroscope.z').d
```

## 가속도 센서 값

비글의 특정 축의 가속도 센서의 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	축 기준	x 축 (x), y 축 (y), z 축 (z)

## 자바스크립트 코드

```
// x 축 가속도 센서
$(`Beagle*0:accelerometer.x`).d;

// y 축 가속도 센서
$(`Beagle*0:accelerometer.y`).d;

// z 축 가속도 센서
$(`Beagle*0:accelerometer.z`).d;
```

## 파이썬 코드

```
# x 축 가속도 센서
__(`Beagle*0:accelerometer.x`).d

# y 축 가속도 센서
__(`Beagle*0:accelerometer.y`).d

# z 축 가속도 센서
__(`Beagle*0:accelerometer.z`).d
```

## 지자계 센서 값

비글의 특정 축의 지자계 센서의 값을 가져옵니다.



✓ x축

특정 축의 지자계 센서의 값

y축

z축

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	축 기준	x 축 (x), y 축 (y), z 축 (z)

## 자바스크립트 코드

```
// x 축 지자계 센서
$('Beagle*0:magnetometer.x').d;

// y 축 지자계 센서
$('Beagle*0:magnetometer.y').d;

// z 축 지자계 센서
$('Beagle*0:magnetometer.z').d;
```

## 파이썬 코드

```
# x 축 지자계 센서
__('Beagle*0:magnetometer.x').d

# y 축 지자계 센서
__('Beagle*0:magnetometer.y').d

# z 축 지자계 센서
__('Beagle*0:magnetometer.z').d
```

## 온도 센서 값

비글의 온도 센서 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
unit	드롭다운 옵션	온도 단위	섭씨 (°C), 화씨 (°F)

## 자바스크립트 코드

```
// 섭씨 기준 온도센서 값
__getTemperature($('Beagle*0:temperature').d, '°C');

// 화씨 기준 온도센서 값
__getTemperature($('Beagle*0:temperature').d, '°F');
```

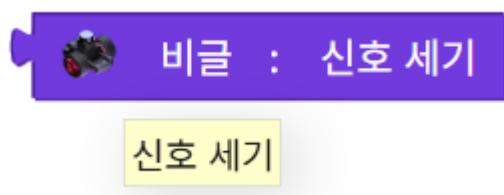
## 파이썬 코드

```
# 섭씨 기준 온도센서 값
__getTemperature(__('Beagle*0:temperature').d, '°C')

# 화씨 기준 온도센서 값
__getTemperature(__('Beagle*0:temperature').d, '°F')
```

## 신호 세기 값

비글의 신호 세기 값을 가져옵니다.



## 자바스크립트 코드

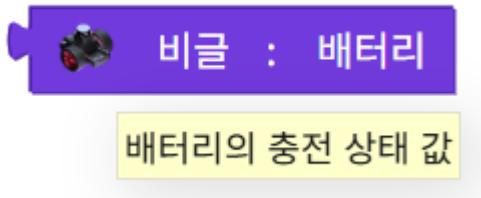
```
// 신호 세기 값  
$('Beagle*0:signal_strength').d;
```

## 파이썬 코드

```
# 신호 세기 값  
__('Beagle*0:signal_strength').d
```

## 배터리 충전 상태 값

비글의 배터리 충전 상태 값을 가져옵니다.



## 자바스크립트 코드

```
// 배터리 충전 상태 값  
$('Beagle*0:battery.level').d;
```

## 파이썬 코드

```
# 배터리 충전 상태 값  
__('Beagle*0:battery.level').d
```

## 상태 변경 여부

비글의 상태 변경 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



비글 : 앞으로 기울였는가 ?

로봇의 상태가 변했는지 여부

앞으로 기울였는가

뒤로 기울였는가

왼쪽으로 기울였는가

오른쪽으로 기울였는가

거꾸로 뒤집어졌는가

뒤집어지지 않았는가

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
condition	드롭다운 옵션	위치 조건	앞으로 기울였는가, 뒤로 기울였는가, 왼쪽으로 기울였는가, 오른쪽으로 기울였는가, 거꾸로 뒤집어졌는가, 뒤집어지지 않았는가

## 자바스크립트 코드

```
// 비글이 앞으로 기울였는가?
$(`Beagle*0:accelerometer.x`).d > 0.8;

// 비글이 뒤로 기울였는가?
$(`Beagle*0:accelerometer.x`).d < -0.8;

// 비글이 왼쪽으로 기울였는가?
$(`Beagle*0:accelerometer.y`).d > 0.8;

// 비글이 오른쪽으로 기울였는가?
$(`Beagle*0:accelerometer.y`).d < -0.8;

// 비글이 거꾸로 뒤집어졌는가?
$(`Beagle*0:accelerometer.z`).d > 0;

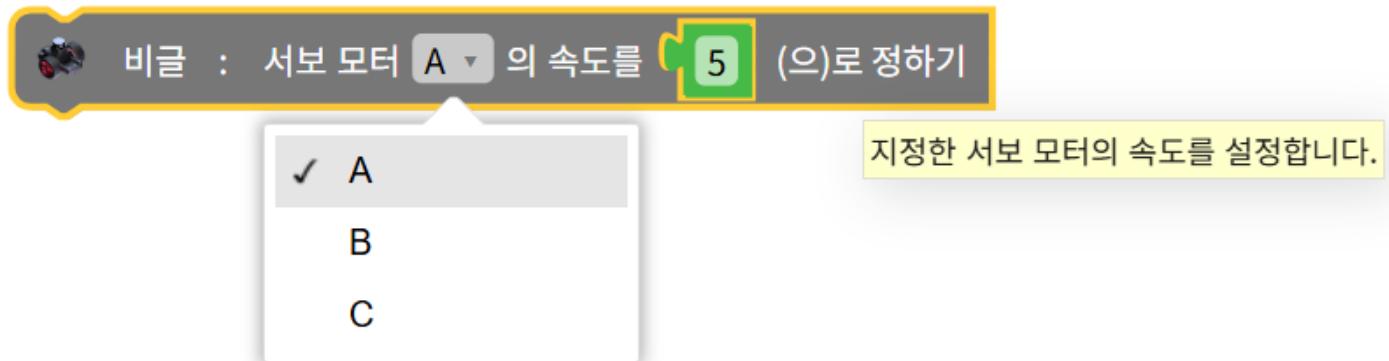
// 비글이 뒤집어지지 않았는가?
$(`Beagle*0:accelerometer.z`).d < 0;
```

## 파이썬 코드

```
# 비글이 앞으로 기울였는가?  
__('Beagle*0:accelerometer.x').d > 0.8  
  
# 비글이 뒤로 기울였는가?  
__('Beagle*0:accelerometer.x').d < -0.8  
  
# 비글이 원쪽으로 기울였는가?  
__('Beagle*0:accelerometer.y').d > 0.8  
  
# 비글이 오른쪽으로 기울였는가?  
__('Beagle*0:accelerometer.y').d < -0.8  
  
# 비글이 거꾸로 뒤집어졌는가?  
__('Beagle*0:accelerometer.z').d > 0  
  
# 비글이 뒤집어지지 않았는가?  
__('Beagle*0:accelerometer.z').d < 0
```

## 서보모터 속도 설정하기

지정한 서보모터의 속도를 설정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	서보모터 종류	A(a), B(b), C(c)
speed	입력값	속도	0 ~ 10 사이 실수

## 자바스크립트 코드

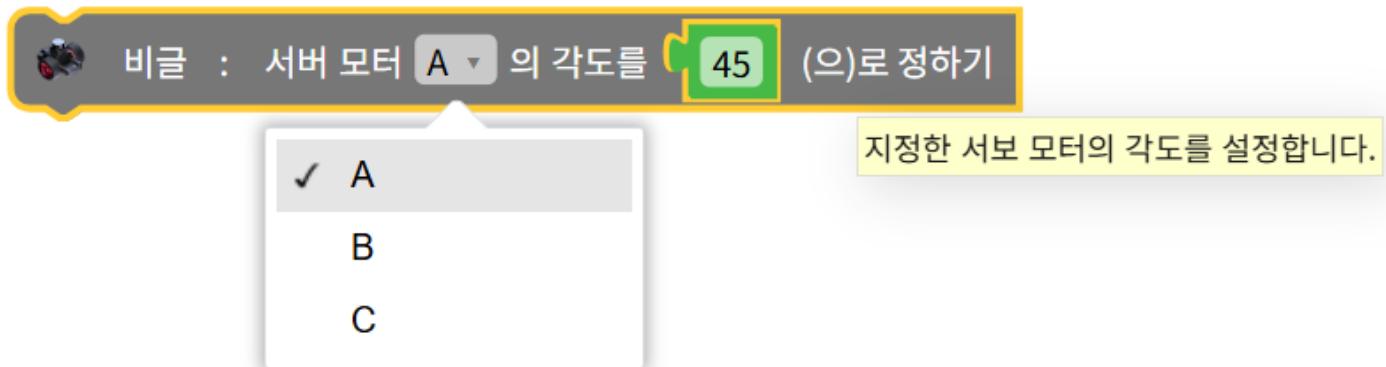
```
// 서보 모터 A 의 속도를 1(으)로 정하기  
$('Beagle*0:servo.a.speed').d = 1;  
  
// 서보 모터 B 의 속도를 10(으)로 정하기  
$('Beagle*0:servo.b.speed').d = 10;  
  
// 서보 모터 C 의 속도를 5.5(으)로 정하기  
$('Beagle*0:servo.c.speed').d = 5.5;
```

## 파이썬 코드

```
# 서보 모터 A 의 속도를 1(으)로 정하기  
__('Beagle*0:servo.a.speed').d = 1  
  
# 서보 모터 B 의 속도를 10(으)로 정하기  
__('Beagle*0:servo.b.speed').d = 10  
  
# 서보 모터 C 의 속도를 5.5(으)로 정하기  
__('Beagle*0:servo.c.speed').d = 5.5
```

## 서보모터 각도 설정하기

지정한 서보모터의 각도를 설정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	서보모터 종류	A(a), B(b), C(c)
angle	입력값	각도	0 ~ 180 사이 실수

## 자바스크립트 코드

```
// 서보 모터 A 의 각도를 0(으)로 정하기  
$(('Beagle*0:servo.a.angle').d = 0;  
  
// 서보 모터 B 의 각도를 180(으)로 정하기  
$(('Beagle*0:servo.b.angle').d = 180;  
  
// 서보 모터 C 의 각도를 90(으)로 정하기  
$(('Beagle*0:servo.c.angle').d = 90;
```

## 파이썬 코드

```
# 서보 모터 A 의 각도를 0(으)로 정하기  
__('Beagle*0:servo.a.angle').d = 0
```

```
# 서보 모터 B 의 각도를 180(으)로 정하기  
__('Beagle*0:servo.b.angle').d = 180
```

```
# 서보 모터 C 의 각도를 90(으)로 정하기  
__('Beagle*0:servo.c.angle').d = 90
```

## 서보모터 각도 바꾸기

지정한 서보모터의 각도를 변경합니다.

현재 서보모터의 각도에 입력한 각도를 더한 값이 새로운 서보모터 각도가 됩니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	서보모터 종류	A(a), B(b), C(c)
angle	입력값	현재 모터 각도에 더할 각도 값	-180 ~ 180 사이 실수

## 자바스크립트 코드

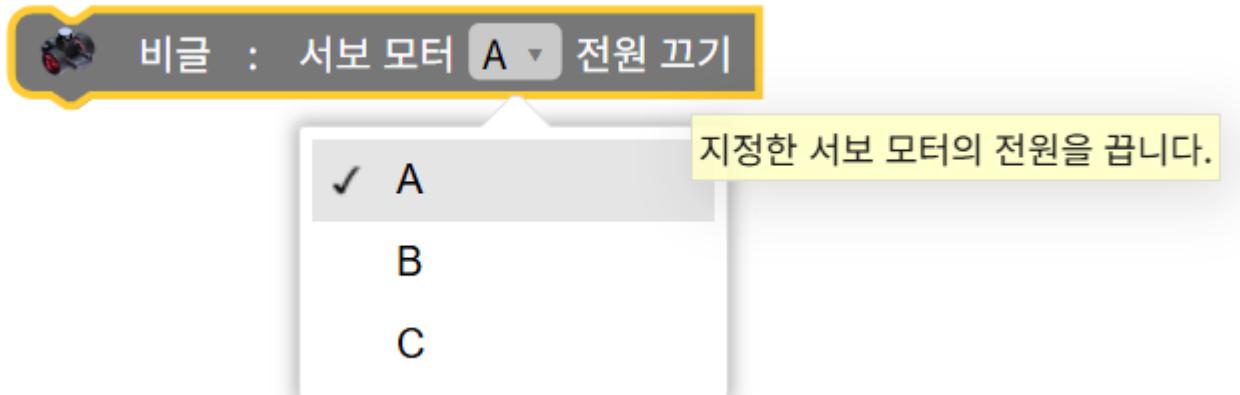
```
// 서보 모터 A 의 각도를 -10 만큼 바꾸기  
$('Beagle*0:servo.a.angle').d = $('Beagle*0:servo.a.angle').d + -10;  
  
// 서보 모터 B 의 각도를 10 만큼 바꾸기  
$('Beagle*0:servo.b.angle').d = $('Beagle*0:servo.b.angle').d + 10;  
  
// 서보 모터 C 의 각도를 20 만큼 바꾸기  
$('Beagle*0:servo.c.angle').d = $('Beagle*0:servo.c.angle').d + 20;
```

## 파이썬 코드

```
# 서보 모터 A 의 각도를 -10 만큼 바꾸기  
__('Beagle*0:servo.a.angle').d = __('Beagle*0:servo.a.angle').d + -10  
  
# 서보 모터 B 의 각도를 10 만큼 바꾸기  
__('Beagle*0:servo.b.angle').d = __('Beagle*0:servo.b.angle').d + 10  
  
# 서보 모터 C 의 각도를 20 만큼 바꾸기  
__('Beagle*0:servo.c.angle').d = __('Beagle*0:servo.c.angle').d + 20
```

## 서보모터 끄기

지정한 서보모터의 전원을 끕니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	서보모터 종류	A(a), B(b), C(c)

## 자바스크립트 코드

```
// 서보 모터 A 전원 끄기
$(`Beagle*0:servo.a.angle`).d = 0;

// 서보 모터 B 전원 끄기
$(`Beagle*0:servo.b.angle`).d = 0;

// 서보 모터 C 전원 끄기
$(`Beagle*0:servo.c.angle`).d = 0;
```

## 파이썬 코드

```
# 서보 모터 A 전원 끄기
__(`Beagle*0:servo.a.angle`).d = 0

# 서보 모터 B 전원 끄기
__(`Beagle*0:servo.b.angle`).d = 0

# 서보 모터 C 전원 끄기
__(`Beagle*0:servo.c.angle`).d = 0
```

## 서보모터 각도 값

지정한 서보모터의 현재 각도 값을 반환합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
motor	드롭다운 옵션	서보모터 종류	A(a), B(b), C(c)

### 자바스크립트 코드

```
// 서보 모터 A 의 각도
$('Beagle*0:servo.a.angle').d;

// 서보 모터 B 의 각도
$('Beagle*0:servo.b.angle').d;

// 서보 모터 C 의 각도
$('Beagle*0:servo.c.angle').d;
```

### 파이썬 코드

```
# 서보 모터 A 의 각도
__('Beagle*0:servo.a.angle').d

# 서보 모터 B 의 각도
__('Beagle*0:servo.b.angle').d

# 서보 모터 C 의 각도
__('Beagle*0:servo.c.angle').d
```

### 라이다 켜기 / 끄기

라이다 센서를 활성화하거나 비활성화 합니다.



비글 : 라이다 켜기 ▾

라이다 센서를 활성화하거나 비활성화합니다.

✓ 켜기

끄기

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	라이다 전원	켜기 (1), 끄기 (0)

## 자바스크립트 코드

```
// 라이다 센서 켜기
$(`'Beagle*0:Lidar+connect'`).d = 1;

// 라이다 센서 끄기
$(`'Beagle*0:Lidar+connect'`).d = 0;
```

## 파이썬 코드

```
# 라이다 센서 켜기
__(`'Beagle*0:Lidar+connect'`).d = 1

# 라이다 센서 끄기
__(`'Beagle*0:Lidar+connect'`).d = 0
```

## 라이다 ~ 번째 사물의 거리 값

라이다 센서를 사용하여 주변 사물과의 거리를 측정합니다 (mm).

비글 로봇 앞에 있는 사물을 0 번으로 시작하여, 반시계 방향으로 순차적으로 번호를 붙입니다.

이후 특정 번호에 해당하는 사물과 비글 사이의 거리를 반환합니다.



비글 : 라이다 0 번째 값(mm)

라이다 센서는 주변 360도 사물과의 거리를 측정할 수 있습니다.

비글의 앞쪽(0번째 값)을 기준으로, 반시계 방향으로 갈수록 번호가 1씩 커집니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
number	입력값	사물 번호	0 ~ 359 사이 정수

## 자바스크립트 코드

```
// 라이다와 0 번째 물건 사이 거리 (mm)
$('Beagle*0:Lidar+array').d[0];

// 라이다와 1 번째 물건 사이 거리 (mm)
$('Beagle*0:Lidar+array').d[1];

// 라이다와 359 번째 물건 사이 거리 (mm)
$('Beagle*0:Lidar+array').d[359];
```

## 파이썬 코드

```
# 라이다와 0 번째 물건 사이 거리 (mm)
__('Beagle*0:Lidar+array').d[0]

# 라이다와 1 번째 물건 사이 거리 (mm)
__('Beagle*0:Lidar+array').d[1]

# 라이다와 359 번째 물건 사이 거리 (mm)
__('Beagle*0:Lidar+array').d[359]
```

## 라이다 센서 거리 값

라이다 센서를 사용하여 측정한 앞, 뒤, 양 옆, 대각선 방향의 거리를 나타냅니다 (mm).

해당 방향의 좌우 45 도 거리 값들의 평균 값을 출력합니다.



라이다 센서가 측정한 앞, 뒤, 양 옆, 대각선 방향의 거리를 나타냅니다.  
해당 방향의 좌우 45도 거리 값들의 평균값을 출력합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	특정 방향	앞쪽 (0), 左前 (1), 左 (2), 左后 (3), 后 (4), 右后 (5), 右 (6), 右前 (7)

## 자바스크립트 코드

```
// 라이다 앞쪽 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[0];

// 라이다 左前 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[1];

// 라이다 左 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[2];

// 라이다 左后 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[3];

// 라이다 后 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[4];

// 라이다 右后 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[5];

// 라이다 右 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[6];
```

```
// 라이다 오른쪽 앞 거리 값 (mm)
$('Beagle*0:Lidar+directions').d[7];
```

## 파이썬 코드

```
# 라이다 앞쪽 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[0]

# 라이다 왼쪽 앞 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[1]

# 라이다 왼쪽 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[2]

# 라이다 왼쪽 뒤 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[3]

# 라이다 뒤쪽 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[4]

# 라이다 오른쪽 뒤 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[5]

# 라이다 오른쪽 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[6]

# 라이다 오른쪽 앞 거리 값 (mm)
__('Beagle*0:Lidar+directions').d[7]
```

## 라이다가 켜져 있는가?

라이다가 켜져있는지 여부에 따라 참 (1) / 거짓 (0) (으)로 반환합니다.



## 자바스크립트 코드

```
// 라이다가 켜져 있는가?
$('Beagle*0:Lidar+ready').d;
```

## 파이썬 코드

```
# 라이다가 켜져 있는가?
__('Beagle*0:Lidar+ready').d
```

빼오

## 블록

### 바퀴 속도 설정하기

빼오의 바퀴 속도를 설정합니다.

바퀴 속도가 양수이면 앞쪽 방향으로 회전하고, 바퀴 속도가 음수이면 뒤쪽 방향으로 회전합니다.

예를 들어, 바퀴 속도가 100 이라면, 앞쪽 방향으로 100 의 속도로 회전하고,

바퀴 속도가 -100 이라면, 뒤쪽 방향으로 100 의 속도로 회전합니다.

한번 바퀴 속도를 설정하면, 다시 바퀴 속도를 설정하기 전까지 해당 속도로 빼오가 이동합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	바퀴 속도	-100 ~ 100 정수, 0: 정지

### 자바스크립트 코드

```
// 왼쪽 바퀴 속도를 50(으)로 정하기
if($('#Pio*0:wheel.move').d != 0) {
  $('#Pio*0:wheel.move').d = 0;
}
$('#Pio*0:wheel.speed.left').d = __getSpeed('Pio*0', 50);

// 오른쪽 바퀴 속도를 -50(으)로 정하기
if($('#Pio*0:wheel.move').d != 0) {
  $('#Pio*0:wheel.move').d = 0;
}
$('#Pio*0:wheel.speed.right').d = __getSpeed('Pio*0', -50);

// 양쪽 바퀴 속도를 0(으)로 정하기
if($('#Pio*0:wheel.move').d != 0) {
  $('#Pio*0:wheel.move').d = 0;
}
```

```
$('Pio*0:wheel.speed.left').d = __getSpeed('Pio*0', 0);
$('Pio*0:wheel.speed.right').d = __getSpeed('Pio*0', 0);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도를 50(으)로 정하기
if ___('Pio*0:wheel.move').d != 0:
    ___('Pio*0:wheel.move').d = 0
___('Pio*0:wheel.speed.left').d = __getSpeed('Pio*0', 50)

# 오른쪽 바퀴 속도를 -50(으)로 정하기
if ___('Pio*0:wheel.move').d != 0:
    ___('Pio*0:wheel.move').d = 0
___('Pio*0:wheel.speed.right').d = __getSpeed('Pio*0', -50)

# 양쪽 바퀴 속도를 0(으)로 정하기
if ___('Pio*0:wheel.move').d != 0:
    ___('Pio*0:wheel.move').d = 0
___('Pio*0:wheel.speed.left').d = __getSpeed('Pio*0', 0)
___('Pio*0:wheel.speed.right').d = __getSpeed('Pio*0', 0)
```

## 거리 이동하기

빼오가 이동할 거리를 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

거리 값이 0 일 경우에는, 현재 설정되어 있는 바퀴 속도대로 멈추지 않고 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
distance	입력값	거리 값	0 이상 실수
unit	드롭다운 옵션	거리 단위	cm, mm, 인치 (inch)

## 자바스크립트 코드

```
// 50 cm 이동하기 | 기다리기 0
$(`'Pio*0:wheel.move}`).d = __getDistance('Pio*0', 50, 'cm'); // (robot, distance, unit)
await $(`'Pio*0:wheel.!move}`).w();

// 50 mm 이동하기 | 기다리기 X
$(`'Pio*0:wheel.move}`).d = __getDistance('Pio*0', 50, 'mm'); // (robot, distance, unit)

// 50 inch 이동하기 | 기다리기 X
$(`'Pio*0:wheel.move}`).d = __getDistance('Pio*0', 50, 'inch'); // (robot, distance, unit)
```

## 파이썬 코드

```
# 50 cm 이동하기 | 기다리기 0
__(`'Pio*0:wheel.move}`).d = __getDistance('Pio*0', 50, 'cm') # (robot, distance, unit)
await __(`'Pio*0:wheel.!move}`).w()

# 50 mm 이동하기 | 기다리기 X
__(`'Pio*0:wheel.move}`).d = __getDistance('Pio*0', 50, 'mm') # (robot, distance, unit)

# 50 inch 이동하기 | 기다리기 X
__(`'Pio*0:wheel.move}`).d = __getDistance('Pio*0', 50, 'inch') # (robot, distance, unit)
```

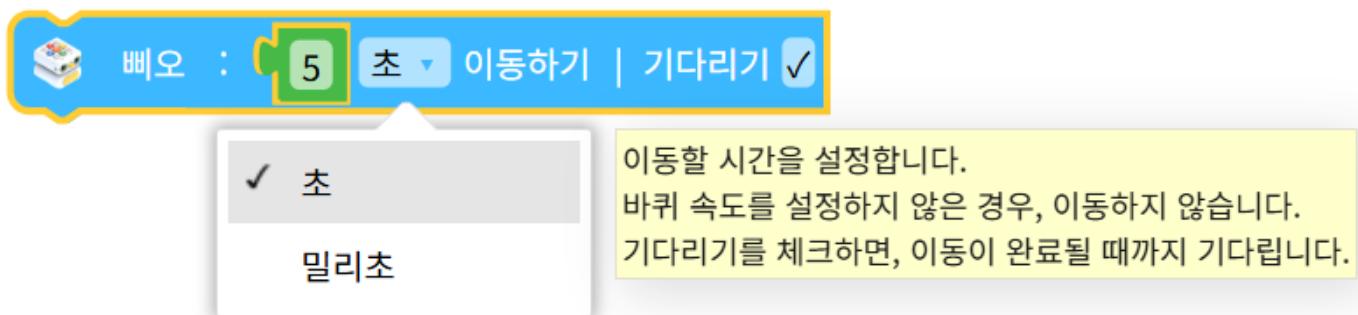
## 시간 이동하기

빼오가 이동할 시간을 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
time	입력값	시간 값	0 이상 실수
unit	드롭다운 옵션	시간 단위	초 (seconds), 밀리초 (milliseconds)

옵션을 밀리초 (milliseconds)로 설정한 경우에는, time 값을 1000으로 나눈 값이 입력됩니다.

## 자바스크립트 코드

```
// 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Pio*0', 5, true); // (robot, time, wait_w)

// 5 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Pio*0', 0.005, false); // (robot, time, wait_w)
```

## 파이썬 코드

```
# 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Pio*0', 5, True) # (robot, time, wait_w)

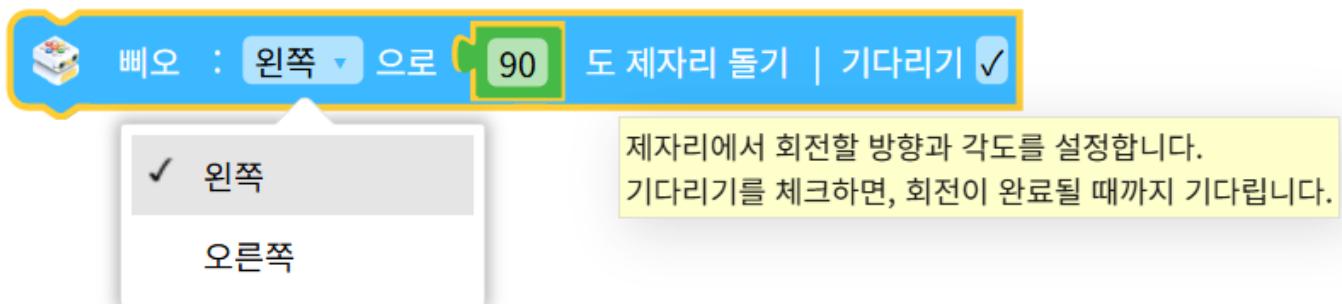
# 5 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Pio*0', 0.005, False) # (robot, time, wait_w)
```

## 제자리 돌기

빼오가 제자리에서 회전할 방향과 각도를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	회전 방향	왼쪽 (left), 오른쪽 (right)
angle	입력값	회전 각도	0 이상 정수

## 자바스크립트 코드

```
// 왼쪽으로 90 도 제자리 돌기 | 기다리기 0  
await __turn_degree_left('Pio*0', 90, true); // (robot, degree, wait_w)  
  
// 오른쪽으로 90 도 제자리 돌기 | 기다리기 X  
__turn_degree_right('Pio*0', 90, false); // (robot, degree, wait_w)
```

## 파이썬 코드

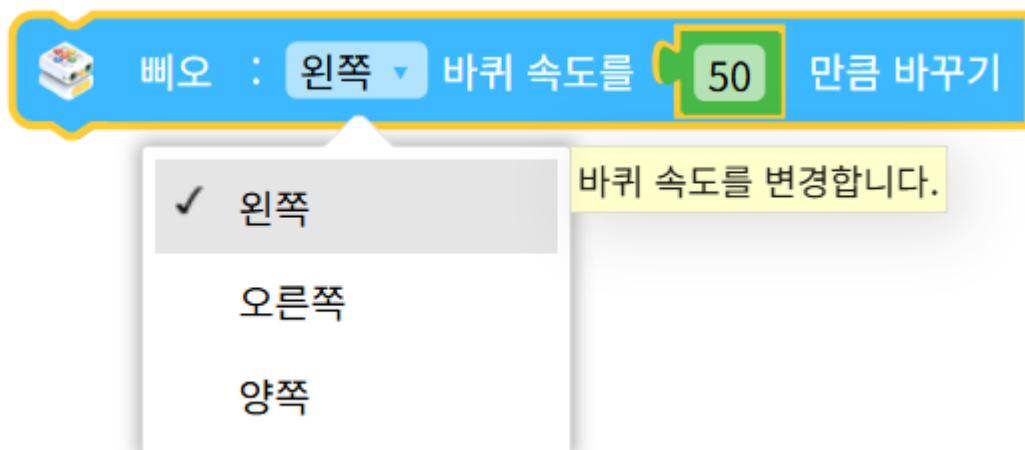
```
# 왼쪽으로 90 도 제자리 돌기 | 기다리기 0  
await __turn_degree_left('Pio*0', 90, True) # (robot, degree, wait_w)  
  
# 오른쪽으로 90 도 제자리 돌기 | 기다리기 X  
__turn_degree_right('Pio*0', 90, False) # (robot, degree, wait_w)
```

## 바퀴 속도 변경하기

빼오의 바퀴 속도를 변경합니다.

현재의 바퀴 속도에 입력한 속도를 더한 값이 새로운 바퀴 속도가 됩니다.

새롭게 설정된 바퀴 속도의 범위는 -100 ~ 100 으로 설정됩니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	현재 바퀴 속도에 더할 속도 값	-200 ~ 200 정수, 0: 정지

## 자바스크립트 코드

```
// 左側 바퀴 속도를 50 만큼 바꾸기
if($('Pio*0:wheel.move').d != 0) {
    $('Pio*0:wheel.move').d = 0;
}
$('Pio*0:wheel.speed.left').d = $('Pio*0:wheel.speed.left').d + __getSpeed('Pio*0', 50); // (robot, value)

// 오른쪽 바퀴 속도를 50 만큼 바꾸기
if($('Pio*0:wheel.move').d != 0) {
    $('Pio*0:wheel.move').d = 0;
}
$('Pio*0:wheel.speed.right').d = $('Pio*0:wheel.speed.right').d + __getSpeed('Pio*0', 50); // (robot, value)

// 양쪽 바퀴 속도를 50 만큼 바꾸기
if($('Pio*0:wheel.move').d != 0) {
    $('Pio*0:wheel.move').d = 0;
}
$('Pio*0:wheel.speed.left').d = $('Pio*0:wheel.speed.left').d + __getSpeed('Pio*0', 50); // (robot, value)
$('Pio*0:wheel.speed.right').d = $('Pio*0:wheel.speed.right').d + __getSpeed('Pio*0', 50); // (robot, value)
```

## 파이썬 코드

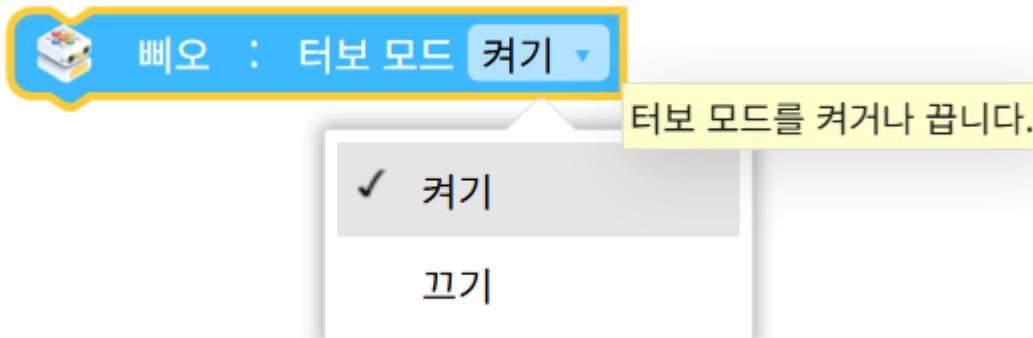
```
# 左側 바퀴 속도를 50 만큼 바꾸기
if __('Pio*0:wheel.move').d != 0:
    __('Pio*0:wheel.move').d = 0
__('Pio*0:wheel.speed.left').d = __('Pio*0:wheel.speed.left').d + __getSpeed('Pio*0', 50) # (robot, value)

# 오른쪽 바퀴 속도를 50 만큼 바꾸기
if __('Pio*0:wheel.move').d != 0:
    __('Pio*0:wheel.move').d = 0
__('Pio*0:wheel.speed.right').d = __('Pio*0:wheel.speed.right').d + __getSpeed('Pio*0', 50) # (robot, value)

# 양쪽 바퀴 속도를 50 만큼 바꾸기
if __('Pio*0:wheel.move').d != 0:
    __('Pio*0:wheel.move').d = 0
__('Pio*0:wheel.speed.left').d = __('Pio*0:wheel.speed.left').d + __getSpeed('Pio*0', 50) # (robot, value)
__('Pio*0:wheel.speed.right').d = __('Pio*0:wheel.speed.right').d + __getSpeed('Pio*0', 50) # (robot, value)
```

## 터보 모드 켜기 / 끄기

빼오의 터보모드를 켜거나 끕니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	터보 토글	켜기 (true), 끄기 (false)

## 자바스크립트 코드

```
// 터보 모드 켜기
__turbo('Pio*0', true);

// 터보 모드 끄기
__turbo('Pio*0', false);
```

## 파이썬 코드

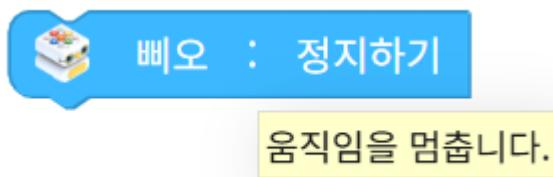
```
# 터보 모드 켜기
__turbo('Pio*0', True)

# 터보 모드 끄기
__turbo('Pio*0', False)
```

## 정지하기

빼오의 이동을 멈춥니다.

빼오의 양쪽 바퀴 속도가 모두 0으로 초기화됩니다.



## 자바스크립트 코드

```
// 빼오 정지하기
__stopMove('Pio*0');
```

## 파이썬 코드

```
# 빼오 정지하기
__stopMove('Pio*0')
```

## 바퀴가 움직이는 중인가?

빼오의 바퀴가 움직이고 있는지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



빼오 : 바퀴가 움직이는 중인가?

바퀴가 움직이는 중이면 true, 멈춰있으면 false를 반환한다.

## 자바스크립트 코드

```
$('Pio*0:wheel.moving').d;
```

## 파이썬 코드

```
--('Pio*0:wheel.moving').d
```

## 말판 앞으로 한 칸 이동하기

빼오가 말판 위에서 한 칸 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



빼오 : 말판 앞으로 ▾ 한 칸 이동하기 | 기다리기 ✓

말판 위에서 정해진 대로 한 칸씩 움직입니다.

✓ 앞으로

뒤로

왼쪽으로

오른쪽으로

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	이동 방향	앞으로 (forward), 뒤로 (backward), 왼쪽으로 (left), 오른쪽으로 (right)

## 자바스크립트 코드

```
// 말판 앞으로 한 칸 이동하기 | 기다리기 0
await __grid_move_forward('Pio*0', true); // (robot, wait_w)

// 말판 뒤로 한 칸 이동하기 | 기다리기 0
await __grid_move_backward('Pio*0', true); // (robot, wait_w)

// 말판 왼쪽으로 한 칸 이동하기 | 기다리기 X
__grid_move_left('Pio*0', false); // (robot, wait_w)

// 말판 오른쪽으로 한 칸 이동하기 | 기다리기 X
__grid_move_right('Pio*0', false); // (robot, wait_w)
```

## 파이썬 코드

```
# 말판 앞으로 한 칸 이동하기 | 기다리기 0
await __grid_move_forward('Pio*0', True) # (robot, wait_w)

# 말판 뒤로 한 칸 이동하기 | 기다리기 0
await __grid_move_backward('Pio*0', True) # (robot, wait_w)

# 말판 왼쪽으로 한 칸 이동하기 | 기다리기 X
__grid_move_left('Pio*0', False) # (robot, wait_w)

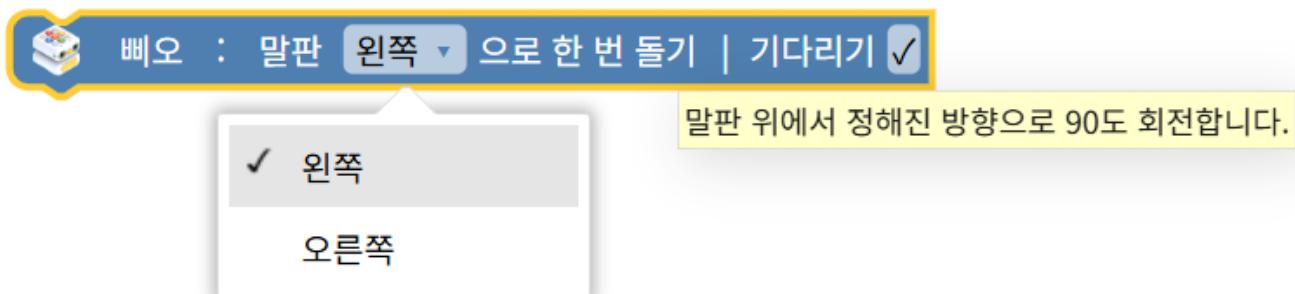
# 말판 오른쪽으로 한 칸 이동하기 | 기다리기 X
__grid_move_right('Pio*0', False) # (robot, wait_w)
```

## 말판에서 한번 돌기

말판 위 빠오가 입력받은 방향으로 90 도 회전합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	회전 방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
// 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_left('Pio*0', true); // (robot, wait_w)

// 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 X
__grid_turn_right('Pio*0', false); // (robot, wait_w)
```

## 파이썬 코드

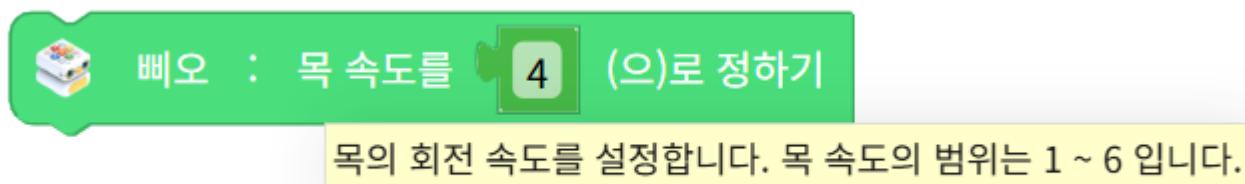
```
# 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_left('Pio*0', True) # (robot, wait_w)

# 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 X
__grid_turn_right('Pio*0', False) # (robot, wait_w)
```

## 목 회전속도 설정하기

빼오 목의 회전속도를 설정합니다.

목 속도의 범위는 1 ~ 6 입니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
speed	입력값	목 회전속도	1 ~ 6 사이 실수

## 자바스크립트 코드

```
// 빠오 목 회전속도를 4로 정하기  
$('Pio*0:neck.speed').d = 4;
```

## 파이썬 코드

```
# 빠오 목 회전속도를 4로 정하기  
__('Pio*0:neck.speed').d = 4
```

## 목 각도 설정하기

빠오 목을 회전하여 도착할 각도를 설정합니다.

목 각도의 범위는 -45 ~ 45 도입니다.



목을 회전하여 도착할 각도를 설정합니다. 목 각도의 범위는 -45 ~ 45입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
angle	입력값	목 각도	-45 ~ 45 사이 실수

## 자바스크립트 코드

```
// 빠오 목 각도를 10 도로 바꾸기 | 기다리기 0  
$('Pio*0:neck.angle').d = 10;  
await __('Pio*0:neck.!angle').w();  
  
// 빠오 목 각도를 -10 도로 바꾸기 | 기다리기 X  
$('Pio*0:neck.angle').d = -10;
```

## 파이썬 코드

```
# 빠오 목 각도를 10 도로 바꾸기 | 기다리기 0  
__('Pio*0:neck.angle').d = 10  
await __('Pio*0:neck.!angle').w()  
  
# 빠오 목 각도를 -10 도로 바꾸기 | 기다리기 X  
__('Pio*0:neck.angle').d = -10
```

## 빼오의 움직이는 중인가?

빼오의 목이 움직이고 있는지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



빼오 : 목이 움직이는 중인가?

목이 움직이는 중이면 true, 멈춰있으면 false를 반환한다.

## 자바스크립트 코드

```
$(('Pio*0:neck.moving').d;
```

## 파이썬 코드

```
__('Pio*0:neck.moving').d
```

## 눈 색상 설정하기

빼오의 눈 색을 변경합니다.

왼쪽, 오른쪽 또는 양쪽의 눈 색을 변경할 수 있습니다.



빼오 : 왼쪽 ▾ 눈을 검정색 ▾ 으로 정하기



빼오 : 오른쪽 ▾ 눈을 검정색 ▾ 으로 정하기



빼오 : 양쪽 ▾ 눈을 검정색 ▾ 으로 정하기

눈의 색을 결정합니다.

✓ 검정색

빨간색

노란색

초록색

청록색

파란색

자홍색

흰색

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	눈 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	색상	검정색 ([0, 0, 0]), 빨간색 ([255, 0, 0]), 노란색 ([255, 255, 0]), 초록색 ([0, 255, 0]), 청록색 ([0, 255, 255]), 파란색 ([0, 0, 255]), 자홍색 ([255, 0, 255]), 흰색 ([255, 255, 255])

## 자바스크립트 코드

```
// 뼈오 왼쪽눈을 검정색으로 정하기
$('Pio*0:eye.left.rgb').d = [0, 0, 0];

// 뼈오 오른쪽 눈을 빨간색으로 정하기
$('Pio*0:eye.right.rgb').d = [255, 0, 0];

// 뼈오 양쪽 눈을 노란색으로 정하기
$('Pio*0:eye.left.rgb').d = [255, 255, 0];
$('Pio*0:eye.right.rgb').d = [255, 255, 0];
```

## 파이썬 코드

```
# 뼈오 왼쪽 눈을 검정색으로 정하기
__($('Pio*0:eye.left.rgb').d = [0, 0, 0]

# 뼈오 오른쪽 눈을 빨간색으로 정하기
__($('Pio*0:eye.right.rgb').d = [255, 0, 0]

# 뼈오 양쪽 눈을 노란색으로 정하기
__($('Pio*0:eye.left.rgb').d = [255, 255, 0]
__($('Pio*0:eye.right.rgb').d = [255, 255, 0]
```

## 눈 색상 색상 카테고리 블록들로 설정하기

뼈오의 눈 색을 색상 카테고리 블록으로 설정합니다.

왼쪽, 오른쪽 또는 양쪽의 눈 색을 변경할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
color	입력값	눈 색상	RGB 배열 ([255,255,255])

## 자바스크립트 코드

```
// 빠오 왼쪽 눈을 빨간색으로 정하기
$('#Pio*0:eye.left.rgb').d = [255, 0, 0];

// 빠오 오른쪽 눈을 R:255, G:255, B:255 로 정하기
$('#Pio*0:eye.right.rgb').d = [255, 255, 255];

// 빠오 양쪽 눈을 무작위 색상으로 정하기
$('#Pio*0:eye.left.rgb').d = __randomColor();
$('#Pio*0:eye.right.rgb').d = __randomColor();
```

## 파이썬 코드

```
# 빠오 왼쪽 눈을 빨간색으로 정하기
__('Pio*0:eye.left.rgb').d = [255, 0, 0]

# 빠오 오른쪽 눈을 R:255, G:255, B:255 로 정하기
__('Pio*0:eye.right.rgb').d = [255, 255, 255]

# 빠오 양쪽 눈을 무작위 색상으로 정하기
__('Pio*0:eye.left.rgb').d = __randomColor()
__('Pio*0:eye.right.rgb').d = __randomColor()
```

## 눈 색상 지정 RGB 만큼 변경하기

지정한 R,G,B 값만큼 빠오의 눈 색을 변경합니다.  
왼쪽, 오른쪽 또는 양쪽의 색을 설정할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
color	입력값	변경 R,G,B 값	R,G,B 각각 -255~255 사이 정수

## 자바스크립트 코드

```
// 빠오의 왼쪽 눈을 R : 1, G : 2, B : 3 만큼 바꾸기
$(`'Pio*0:eye.left.rgb').d = [$(`'Pio*0:eye.left.rgb').d[0] + 1, $(`'Pio*0:eye.left.rgb').d[1] + 2, $(`'Pio*0:eye.left.rgb').d[2] + 3];

// 빠오의 오른쪽 눈을 R : -1, G : -2, B : -3 만큼 바꾸기
$(`'Pio*0:eye.right.rgb').d = [$(`'Pio*0:eye.right.rgb').d[0] + -1, $(`'Pio*0:eye.right.rgb').d[1] + -2, $(`'Pio*0:eye.right.rgb').d[2] + -3];

// 빠오의 양쪽 눈을 R : 10, G : 20, B : 30 만큼 바꾸기
$(`'Pio*0:eye.left.rgb').d = [$(`'Pio*0:eye.left.rgb').d[0] + 10, $(`'Pio*0:eye.left.rgb').d[1] + 20, $(`'Pio*0:eye.left.rgb').d[2] + 30];
$(`'Pio*0:eye.right.rgb').d = [$(`'Pio*0:eye.right.rgb').d[0] + 10, $(`'Pio*0:eye.right.rgb').d[1] + 20, $(`'Pio*0:eye.right.rgb').d[2] + 30];
```

## 파이썬 코드

```
# 빠오의 왼쪽 눈을 R : 1, G : 2, B : 3 만큼 바꾸기
__(`'Pio*0:eye.left.rgb').d = [__(`'Pio*0:eye.left.rgb').d[0] + 1, __(`'Pio*0:eye.left.rgb').d[1] + 2, __(`'Pio*0:eye.left.rgb').d[2] + 3]

# 빠오의 오른쪽 눈을 R : -1, G : -2, B : -3 만큼 바꾸기
__(`'Pio*0:eye.right.rgb').d = [__(`'Pio*0:eye.right.rgb').d[0] + -1, __(`'Pio*0:eye.right.rgb').d[1] + -2, __(`'Pio*0:eye.right.rgb').d[2] + -3]

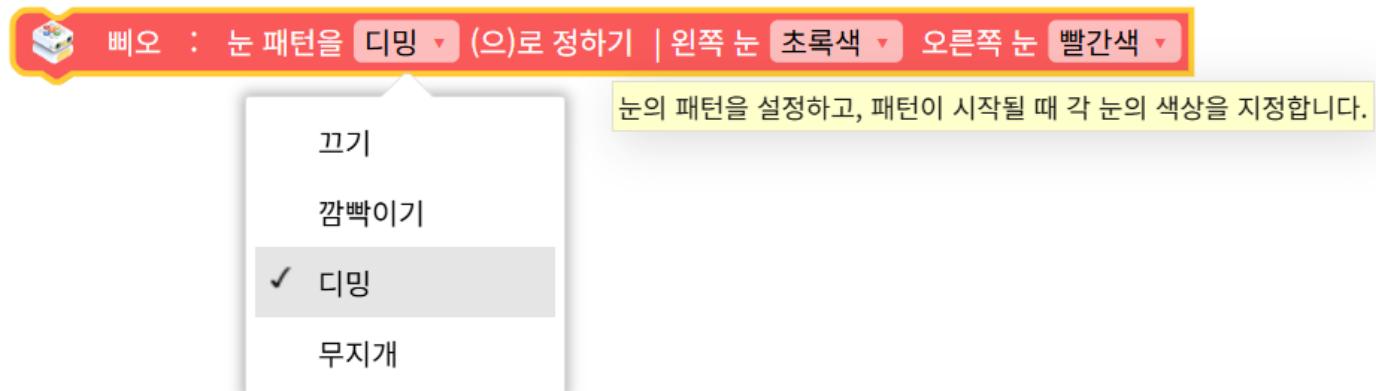
# 빠오의 양쪽 눈을 R : 10, G : 20, B : 30 만큼 바꾸기
__(`'Pio*0:eye.left.rgb').d = [__(`'Pio*0:eye.left.rgb').d[0] + 10, __(`'Pio*0:eye.left.rgb').d[1] + 20, __(`'Pio*0:eye.left.rgb').d[2] + 30]
__(`'Pio*0:eye.right.rgb').d = [__(`'Pio*0:eye.right.rgb').d[0] + 10, __(`'Pio*0:eye.right.rgb').d[1] + 20, __(`'Pio*0:eye.right.rgb').d[2] + 30]
```

## 눈 패턴 변경하기

빠오의 눈 패턴을 변경합니다.

패턴을 정하고 패턴 시작시 눈의 색상을 각각 정할 수 있습니다.

단, 눈 패턴을 무지개로 선택시엔 눈의 색상을 기본, 노란색, 청록색, 자홍색 중 하나로만 선택 가능합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pattern	드롭다운 옵션	눈 패턴	끄기 (0), 깜빡이기 (1), 디밍 (2), 무지개 (3)
color	드롭다운 옵션	왼쪽, 오른쪽 눈 색상	빨간색 (1), 노란색 (2), 초록색 (3), 청록색 (4), 파란색 (5), 자홍색 (6), 흰색 (7)

## 자바스크립트 코드

```
// 빠오의 눈 패턴을 깜빡이기로 정하기 | 왼쪽 눈 : 빨간색 오른쪽 눈 : 노란색
$(`'Pio*0:eye.pattern.mode').d = 1;
$(`'Pio*0:eye.pattern.parameter').d = { index: 0, value: 1 };
$(`'Pio*0:eye.pattern.parameter').d = { index: 3, value: 2 };

// 빠오의 눈 패턴을 디밍으로 정하기 | 왼쪽 눈 : 초록색 오른쪽 눈 : 파란색
$(`'Pio*0:eye.pattern.mode').d = 2;
$(`'Pio*0:eye.pattern.parameter').d = { index: 0, value: 3 };
$(`'Pio*0:eye.pattern.parameter').d = { index: 3, value: 5 };

// 빠오의 눈 패턴을 무지개로 정하기 | 왼쪽 눈 : 청록색 오른쪽 눈 : 자홍색
$(`'Pio*0:eye.pattern.mode').d = 3;
$(`'Pio*0:eye.pattern.parameter').d = { index: 0, value: 3 };
$(`'Pio*0:eye.pattern.parameter').d = { index: 3, value: 5 };
```

## 파이썬 코드

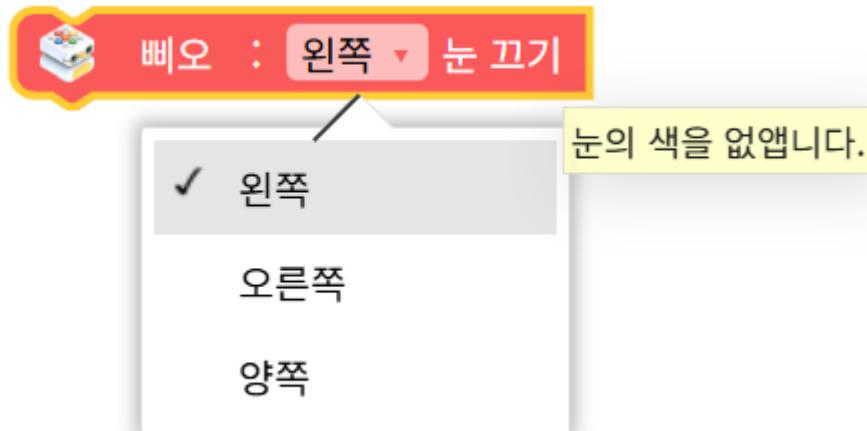
```
# 빠오의 눈 패턴을 깜빡이기로 정하기 | 왼쪽 눈 : 빨간색 오른쪽 눈 : 노란색
__(`'Pio*:eye.pattern.parameter').d = [1,0,0,2,0,0]
__(`'Pio*:eye.pattern.mode').d = 1
# 빠오의 눈 패턴을 디밍으로 정하기 | 왼쪽 눈 : 초록색 오른쪽 눈 : 파란색
__(`'Pio*:eye.pattern.parameter').d = [3,0,0,5,0,0]
__(`'Pio*:eye.pattern.mode').d = 2
# 빠오의 눈 패턴을 무지개로 정하기 | 왼쪽 눈 : 청록색 오른쪽 눈 : 자홍색
```

```
--('Pio*0:eye.pattern.parameter').d = [3,0,0,5,0,0]
--('Pio*0:eye.pattern.mode').d = 3
```

## LED 끄기

빼오의 눈색을 없앱니다.

왼쪽, 오른쪽 또는 양쪽의 눈 색을 없앨 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)

## 자바스크립트 코드

```
// 빼오 왼쪽 눈 끄기
$(`'Pio*0:eye.left.rgb'`).d = [0, 0, 0];

// 빼오 오른쪽 눈 끄기
$(`'Pio*0:eye.right.rgb'`).d = [0, 0, 0];

// 빼오 양쪽 눈 끄기
$(`'Pio*0:eye.left.rgb'`).d = [0, 0, 0];
$(`'Pio*0:eye.right.rgb'`).d = [0, 0, 0];
```

## 파이썬 코드

```
# 빼오 왼쪽 눈 끄기
--('Pio*0:eye.left.rgb').d = [0, 0, 0]

# 빼오 오른쪽 눈 끄기
```

```
__('Pio*0:eye.right.rgb').d = [0, 0, 0]
# 빠오 양쪽 눈 끄기
__('Pio*0:eye.left.rgb').d = [0, 0, 0]
__('Pio*0:eye.right.rgb').d = [0, 0, 0]
```

## 버저음 설정하기

지정된 주파수로 빠오의 버저음을 설정합니다.

주파수의 범위는 10hz ~ 4200hz 입니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound	입력값	버저음 주파수	10 ~ 4200(hz)

## 자바스크립트 코드

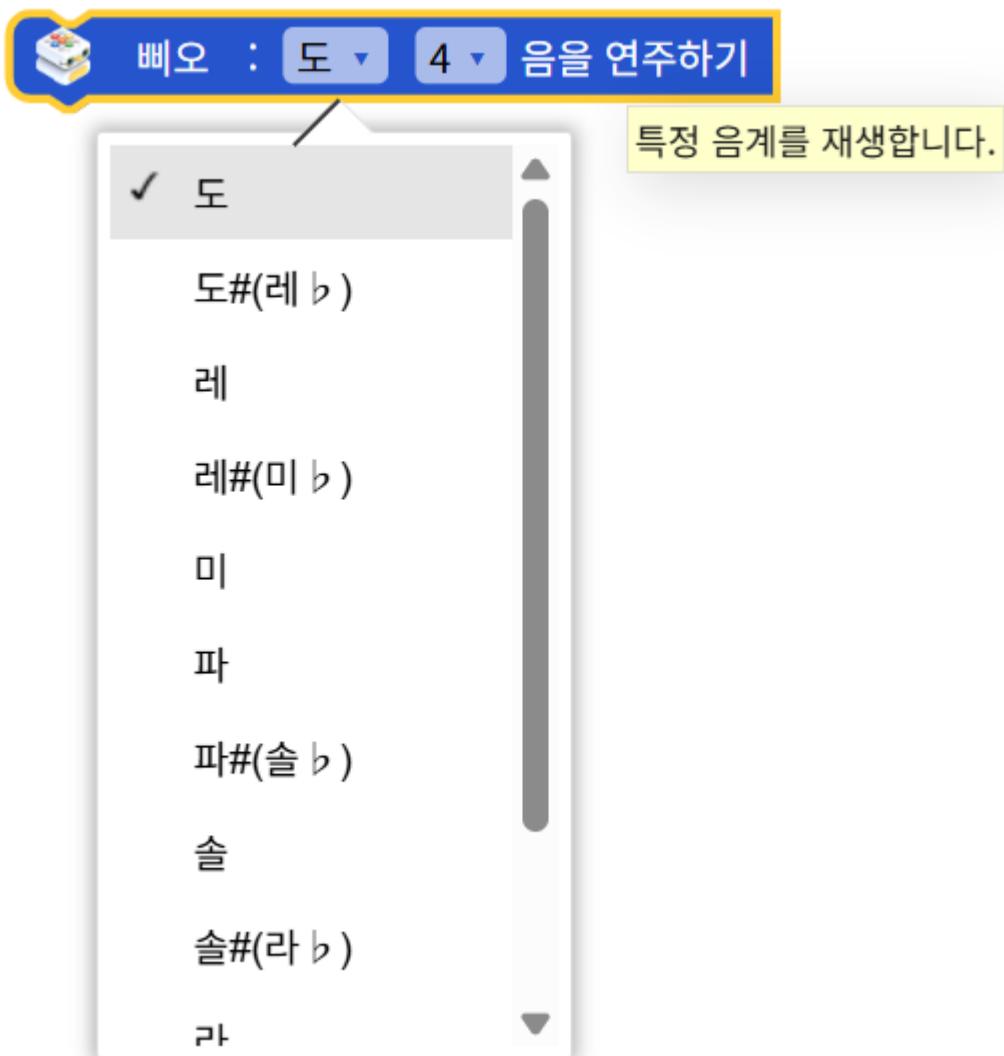
```
// 버저음 주파수 4200hz 로 설정하기
__stopSound('Pio*0');
$(Pio*0:sound.buzz').d = 4200;
```

## 파이썬 코드

```
# 버저음 주파수 4200hz 로 설정하기
__stopSound('Pio*0')
__('Pio*0:sound.buzz').d = 4200
```

## 음계 연주하기

빠오가 지정된 음계를 재생합니다.



빼오 : 도 4 음을 연주하기

특정 음계를 재생합니다.

- 도  
도#(레 ↞ )
- 레
- 레#(미 ↞ )
- 미
- 파
- 파#(솔 ↞ )
- 솔
- 솔#(라 ↞ )
- 라

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
note	드롭다운 옵션	음계	도 (Do), 도 #(Do#), 레 (Re), 레 #(Re#), 미 (Mi), 파 (Fa), 파 #(Fa#), 솔 (So), 솔 #(So#), 라 (La), 라 #(La#), 시 (Ti)
octave	드롭다운 옵션	옥타브	1 ~ 7

## 자바스크립트 코드

```
// 1 옥타브 도 (Do) 음을 연주하기
__stopSound('Pio*0');
$('Pio*0:sound.note').d = 4;

// 1 옥타브 레 (Re) 음을 연주하기
__stopSound('Pio*0');
$('Pio*0:sound.note').d = 6;

// 2 옥타브 도 (Do) 음을 연주하기
__stopSound('Pio*0');
$('Pio*0:sound.note').d = 16;

// 7 옥타브 시 (Ti) 음을 연주하기
__stopSound('Pio*0');
$('Pio*0:sound.note').d = 87;
```

## 파이썬 코드

```
# 1 옥타브 도 (Do) 음을 연주하기
__stopSound('Pio*0')
__('Pio*0:sound.note').d = 4

# 1 옥타브 레 (Re) 음을 연주하기
__stopSound('Pio*0')
__('Pio*0:sound.note').d = 6

# 2 옥타브 도 (Do) 음을 연주하기
__stopSound('Pio*0')
__('Pio*0:sound.note').d = 16

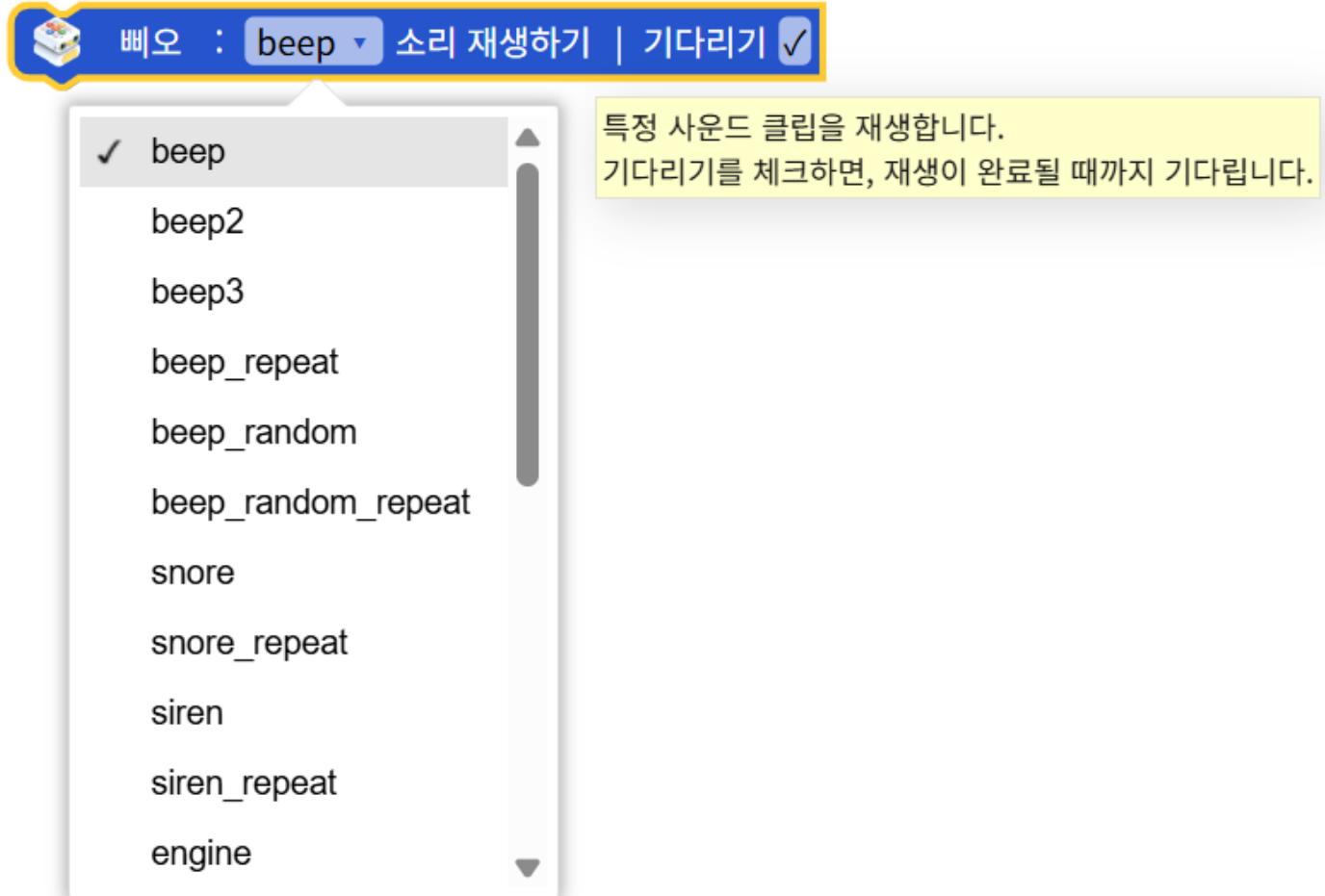
# 7 옥타브 시 (Ti) 음을 연주하기
__stopSound('Pio*0')
__('Pio*0:sound.note').d = 87
```

## 소리 재생하기

빼오가 특정 사운드 클립을 재생합니다.

기다리기를 체크하면, 재생이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound_clip	드롭다운 옵션	사운드 클립	beep(1), beep2(2), beep3(3), beep_repeat(4), beep_random(5), beep_random_repeat(6), snore(7), snore_repeat(8), siren(9), siren_repeat(10), engine(11), engine_repeat(12), fart_a(13), fart_b(14), noise(15), noise_repeat(16), whistle(17), chop(18), chop_repeat(19), melody(20), robot_r2d2(21), connect(22), happy(81)

## 자바스크립트 코드

```
// beep(1) 소리 재생하기 | 기다리기 0
__stopSound('Pio*0');
$('Pio*0:sound.clip').d = 1;
await $('Pio*0:sound.!clip').w();

// happy(81) 소리 재생하기 | 기다리기 X
__stopSound('Pio*0');
$('Pio*0:sound.clip').d = 81;
```

## 파이썬 코드

```
# beep(1) 소리 재생하기 | 기다리기 0
__stopSound('Pio*0')
__(Pio*0:sound.clip).d = 1
await __(Pio*0:sound.!clip).w()

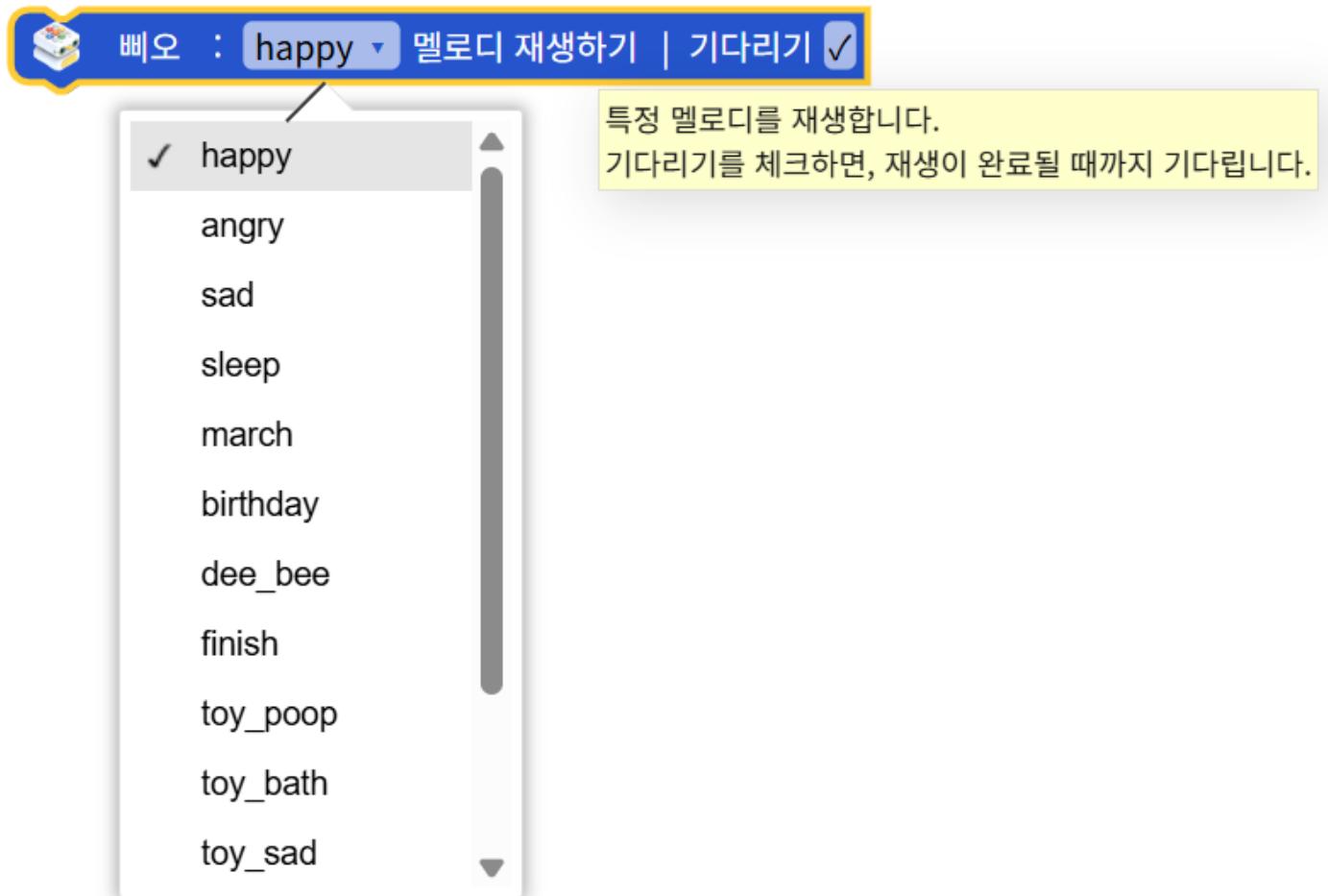
# happy(81) 소리 재생하기 | 기다리기 X
__stopSound('Pio*0')
__(Pio*0:sound.clip).d = 81
```

## 멜로디 재생하기

빼오가 특정 멜로디를 재생합니다.

기다리기를 체크하면, 재생이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
melody	드롭다운 옵션	멜로디	happy(1), angry(2), sad(3), sleep(4), march(5), birthday(6), dee_bee(7), finish(8), toy_poop(9), toy_bath(10), toy_sad(11), toy_happy(12), toy_angry(13), toy_sleep(14)

## 자바스크립트 코드

```
// toy_happy(12) 멜로디 재생하기 | 기다리기 0
__stopSound('Pio*0');
$('Pio*0:sound.melody').d = 12;
await $('Pio*0:sound.!melody').w();

// sleep(4) 멜로디 재생하기 | 기다리기 X
__stopSound('Pio*0');
$('Pio*0:sound.melody').d = 4;
```

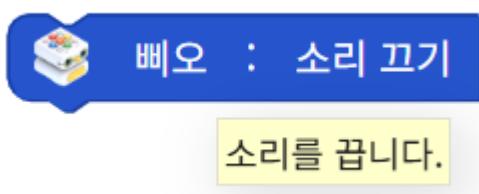
## 파이썬 코드

```
# toy_happy(12) 멜로디 재생하기 | 기다리기 0
__stopSound('Pio*0')
__('Pio*0:sound.melody').d = 12
await ____('Pio*0:sound.!melody').w()

# sleep(4) 멜로디 재생하기 | 기다리기 X
__stopSound('Pio*0')
__('Pio*0:sound.melody').d = 4
```

## 소리 끄기

빼오의 소리를 끕니다.



## 자바스크립트 코드

```
// 빼오 소리 고기  
__stopSound('Pio*0');
```

## 파이썬 코드

```
# 빼오 소리 고기  
__stopSound('Pio*0')
```

## 소리가 재생 중인가?

빼오의 소리가 재생중인지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



소리가 재생 중이면 true, 재생 중이 아니면 false를 반환합니다.

## 자바스크립트 코드

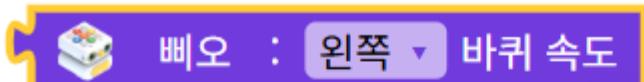
```
// 빼오의 소리가 재생 중인가? - 재생시 true, 아닐시 false  
$(('Pio*0:sound.playing')).d;
```

## 파이썬 코드

```
# 빼오의 소리가 재생 중인가? - 재생시 True, 아닐시 False  
__('Pio*0:sound.playing').d
```

## 바퀴 속도 값

빼오의 지정한 바퀴 속도 값을 가져옵니다.



✓ 왼쪽

특정 바퀴의 속도

오른쪽

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
//왼쪽 바퀴 속도  
__getSpeedInput('Pio*0', $('Pio*0:wheel.speed.left').d);  
  
//오른쪽 바퀴 속도  
__getSpeedInput('Pio*0', $('Pio*0:wheel.speed.right').d);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도  
__getSpeedInput('Pio*0', __('Pio*0:wheel.speed.left').d)  
  
# 오른쪽 바퀴 속도  
__getSpeedInput('Pio*0', __('Pio*0:wheel.speed.right').d)
```

## 신호 세기 값

빼오의 신호 세기 값을 가져옵니다.



## 자바스크립트 코드

```
// 신호 세기 값  
$('Pio*0:signal_strength').d;
```

## 파이썬 코드

```
# 신호 세기 값  
__('Pio*0:signal_strength').d
```

## 배터리 충전 상태 값

빼오의 배터리 충전 상태 값을 가져옵니다.



빼오 : 배터리

배터리

## 자바스크립트 코드

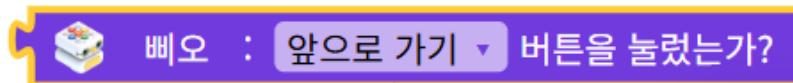
```
// 배터리 충전 상태 값  
$(`'Pio*0:battery.level').d;
```

## 파이썬 코드

```
# 배터리 충전 상태 값  
__(`'Pio*0:battery.level').d
```

## 마지막으로 누른 버튼이 ~인가?

사용자가 마지막으로 누른 버튼에 따라 참 (1) / 거짓 (0) (으)로 반환합니다.



빼오 : 앞으로 가기 ▾ 버튼을 눌렀는가?

✓ 앞으로 가기

뒤로 가기

왼쪽으로 가기

오른쪽으로 가기

실행하기

행동하기

반복하기

삭제하기

사용자가 마지막으로 누른 빼오 버튼을 감지합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
condition	드롭다운 옵션	마지막으로 누른 버튼	앞으로 가기 (2), 뒤로 가기 (4), 왼쪽으로 가기 (8), 오른쪽으로 가기 (16), 실행하기 (1), 행동하기 (32), 반복하기 (64), 삭제하기 (128)

## 자바스크립트 코드

```
// 빠오 앞으로 가기 (2) 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 2

// 빠오 뒤로 가기 (4) 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 4

// 빠오 왼쪽으로 (8) 가기 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 8

// 빠오 오른쪽으로 (16) 가기 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 16

// 빠오 실행하기 (1) 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 1

// 빠오 행동하기 (32) 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 32

// 빠오 반복하기 (64) 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 64

// 빠오 삭제하기 (128) 버튼을 눌렀는가?
$(`'Pio*0:keypad'`).e && `(``'Pio*0:keypad'``).d == 128
```

## 파이썬 코드

```
# 빠오 앞으로 가기 (2) 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 2

# 빠오 뒤로 가기 (4) 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 4

# 빠오 왼쪽으로 (8) 가기 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 8

# 빠오 오른쪽으로 (16) 가기 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 16

# 빠오 실행하기 (1) 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 1

# 빠오 행동하기 (32) 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 32

# 빠오 반복하기 (64) 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 64

# 빠오 삭제하기 (128) 버튼을 눌렀는가?
__(`'Pio*0:keypad'`).e and __(`'Pio*0:keypad'`).d == 128
```

### 대시보드 열기

대시보드 열기는 설치할 수 있는 블록은 아니지만, 확장 모듈에서 사용되는 모델이 어떠한 식으로 적용되는지 알 수 있는 대시보드를 열 수 있습니다. ## 대시보드 화면 대시보드 열기 클릭 시 다음과 같은 화면을 볼 수 있습니다.



### 세부 버튼

#### Power

선택한 카메라를 키거나 끕니다.

#### Load Model

학습된 사물 찾기 모델을 불러옵니다. ‘사물 찾기’ 확장 모듈을 사용하기 위해서 반드시 필요한 작업입니다.

## **Detect**

사물 찾기를 실행하거나 멈춥니다.

Once 버튼으로 한번만 실행할지, Continuous 버튼으로 연속으로 실행할지 정할 수 있습니다.

또한, Stop 버튼을 통해 찾기를 멈출 수 있습니다.

## **Show Result**

사물 찾기 결과를 카메라 화면 상으로 출력합니다.

## **Object Data**

선택한 사물에 따른 데이터 값을 출력합니다.

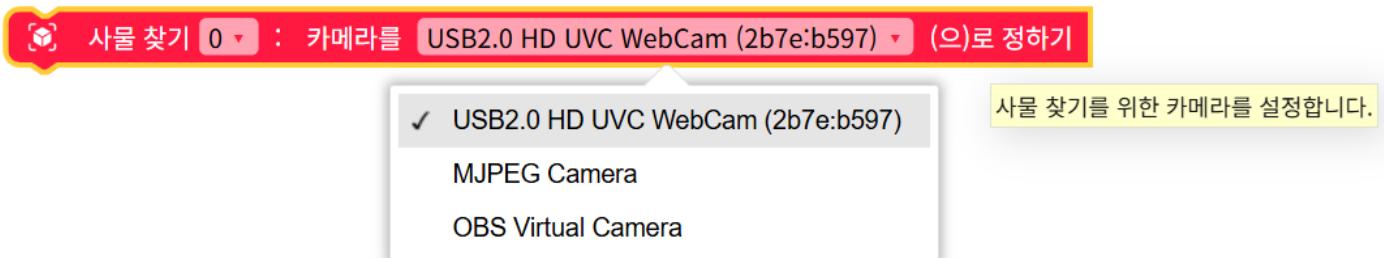
## **드롭다운 옵션 및 입력값**

이름	구분	설명	범위 / 종류
object	드롭다운 옵션	사물	person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed, dining table, toilet, tv, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush

## 블록

### 카메라 정하기

사물 찾기 모듈에 사용할 카메라를 선택합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
camera	드롭다운 옵션	사용할 카메라	연결한 카메라 리스트

### 자바스크립트 코드

```
// 특정 카메라를 사물 찾기를 위한 카메라로 정하기 (id 는 예시)
$(`'ObjectDetection*0:camera.deviceId'`).d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb';
```

### 파이썬 코드

```
# 특정 카메라를 사물 찾기를 위한 카메라로 정하기 (id 는 예시)
__(`'ObjectDetection*0:camera.deviceId'`).d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb'
```

### 사물 모델 불러오기

학습된 사물 모델을 불러옵니다. ‘사물 찾기’모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

기다리기를 체크하면, 모델 불러오기가 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 자바스크립트 코드

```
// 사물 모델 불러오기 | 기다리기 0
$(`'ObjectDetection*0:load_model}`).d = 1;
await $(`'ObjectDetection*0:!load_model}`).w();

// 사물 모델 불러오기 | 기다리기 X
$(`'ObjectDetection*0:load_model}`).d = 1;
```

## 파이썬 코드

```
# 사물 모델 불러오기 | 기다리기 0
__(`'ObjectDetection*0:load_model}`).d = 1
await __(`'ObjectDetection*0:!load_model}`).w()

# 사물 모델 불러오기 | 기다리기 X
__(`'ObjectDetection*0:load_model}`).d = 1
```

## 찾는 사물 개수 정하기

최대로 찾을 수 있는 사물의 개수를 설정합니다. 사물 개수의 범위는 0 ~ 10 입니다.



최대로 찾을 수 있는 사물의 개수를 설정합니다. 사물 개수의 범위는 0 ~ 10 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pivot	입력값	찾는 사물 개수	0 ~ 10 사이 정수

## 자바스크립트 코드

```
// 최대 사물 개수를 5(으)로 정하기
$(`'ObjectDetection*0:maxObjects}`).d = 5;
```

## 파이썬 코드

```
# 최대 사물 개수를 5(으)로 정하기
__(`'ObjectDetection*0:maxObjects}`).d = 5
```

## 사물 찾기 최소 확률 정하기

사물 찾기의 최소 확률(신뢰도)를 설정합니다. 확률(신뢰도) 가 이 이상인 경우에만 화면에 표시됩니다.  
확률(신뢰도)의 범위는 0 ~ 1 입니다.



사물 찾기 0 ▾ : 사물 찾기 확률(신뢰도) > 0.5 (으)로 정하기

사물 찾기의 최소 확률(신뢰도)를 설정합니다. 확률(신뢰도)가 이 이상인 경우에만 화면에 표시됩니다. 확률(신뢰도)의 범위는 0 ~ 1 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
reliability	입력값	사물찾기 신뢰도	0 ~ 1 사이 실수

## 자바스크립트 코드

```
// 최소 사물찾기 신뢰도를 0.5(으)로 정하기  
$('ObjectDetection*0:confidenceThreshold').d = 0.5;
```

## 파이썬 코드

```
# 최소 사물찾기 신뢰도를 0.5(으)로 정하기  
__('ObjectDetection*0:confidenceThreshold').d = 0.5
```

## 사물 한번 찾기

현재 화면에 있는 사물을 찾아 딱 한번 표시합니다.



사물 찾기 0 ▾ : 사물 한 번 찾기

현재 화면에 있는 사물을 찾아 딱 한번 표시합니다.

## 자바스크립트 코드

```
// 사물 한 번 찾기  
$('ObjectDetection*0:detect.once').d = 1;
```

## 파이썬 코드

```
# 사물 한 번 찾기  
__('ObjectDetection*0:detect.once').d = 1
```

## 사물 연속으로 찾기

사물 연속으로 찾기를 시작하거나 중지합니다.

사물 연속으로 찾기를 시작하면, 현재 화면에 있는 사물을 계속 따라가며 화면상에 표시합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	사물 찾기	시작하기 (1), 중지하기 (0)

## 자바스크립트 코드

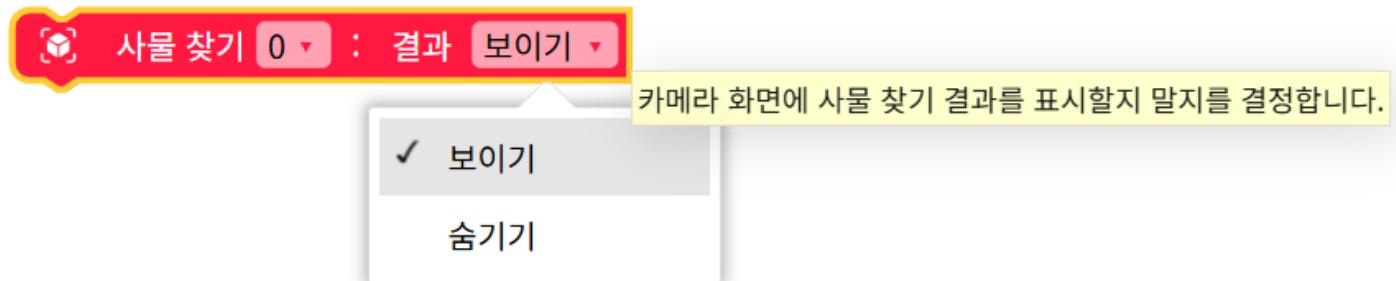
```
// 연속으로 사물 찾기 시작하기  
$('ObjectDetection*0:detect.continuous').d = 1;  
  
// 연속으로 사물 찾기 중지하기  
$('ObjectDetection*0:detect.continuous').d = 0;
```

## 파이썬 코드

```
# 연속으로 사물 찾기 시작하기  
__('ObjectDetection*0:detect.continuous').d = 1  
  
# 연속으로 사물 찾기 중지하기  
__('ObjectDetection*0:detect.continuous').d = 0
```

## 사물 찾기 결과 보이기

카메라 화면에 사물 찾기 결과를 표시할지 말지를 결정합니다.



이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	사물 찾기 결과	보이기 (1), 숨기기 (0)

## 자바스크립트 코드

```
// 사물 찾기 결과 보이기
$('ObjectDetection*0:display').d = 1;

// 사물 찾기 결과 숨기기
$('ObjectDetection*0:display').d = 0;
```

## 파이썬 코드

```
# 사물 찾기 결과 보이기
__('ObjectDetection*0:display').d = 1

# 사물 찾기 결과 숨기기
__('ObjectDetection*0:display').d = 0
```

## 사물 관련 데이터

선택한 사물의 데이터를 반환합니다.



드롭다운 옵션 및 입력값



이름	구분	설명	범위 / 종류	
이름	구분	설명	범위 / 종류	
object	드롭다운 옵션	사물	사람 (0), 자전거 (1), 자동차 (2), 오토바이 (3), 비행기 (4), 버스 (5), 기차 (6), 트럭 (7), 배 (8), 신호등 (9), 소화전 (10), 정지 신호 (11), 주차 미터기 (12), 벤치 (13), 새 (14), 고양이 (15), 개 (16), 말 (17), 양 (18), 소 (19), 코끼리 (20), 곰 (21), 얼룩말 (22), 기린 (23), 배낭 (24), 우산 (25), 핸드백 (26), 넥타이 (27), 여행가방 (28), 원반 (29), 스키 (30), 스노보드 (31), 공 (32), 연 (33), 야구 방망이 (34), 야구 글러브 (35), 스케이트보드 (36), 서프보드 (37), 테니스 채 (38), 병 (39), 포도주 잔 (40), 컵 (41), 포크 (42), 칼 (43), 숟가락 (44), 그릇 (45), 바나나 (46), 사과 (47), 샌드위치 (48), 오렌지 (49), 브로콜리 (50), 당근 (51), 핫도그 (52), 피자 (53), 도넛 (54), 케이크 (55), 의자 (56), 소파 (57), 화분 (58), 침대 (59), 식탁 (60), 변기 (61), 텔레비전 (62), 노트북 (63), 마우스 (64), 리모컨 (65), 키보드 (66), 휴대전화 (67), 전자레인지 (68), 오븐 (69), 토스터 (70), 싱크대 (71), 냉장고 (72), 책 (73), 시계 (74),	

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	좌표 방향	x, y

## 자바스크립트 코드

```
// 사람 (0) x 좌표
$('ObjectDetection*0:object.x').d[0];

// 칫솔 (79) y 좌표
$('ObjectDetection*0:object.y').d[79];
```

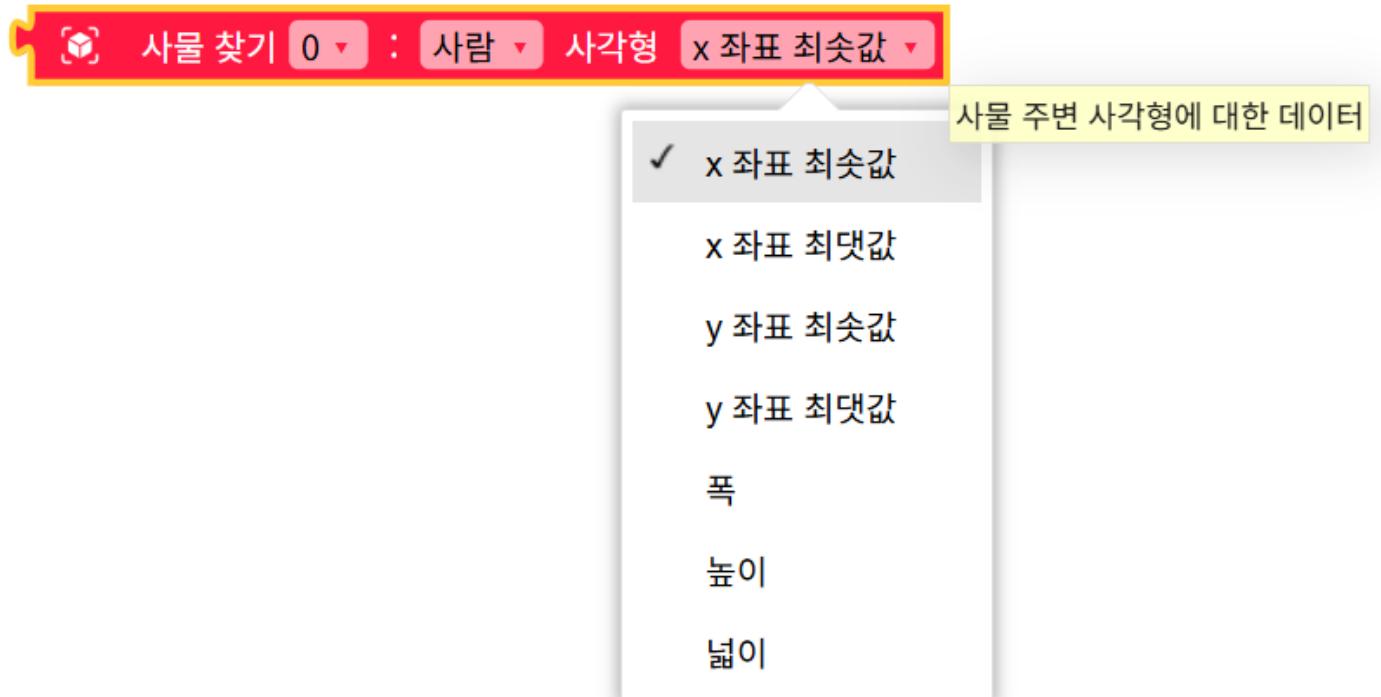
## 파이썬 코드

```
# 사람 (0) x 좌표
__('ObjectDetection*0:object.x').d[0]

# 칫솔 (79) y 좌표
__('ObjectDetection*0:object.y').d[79]
```

## 사물 주변 사각형 데이터

사물 찾기로 찾은 사물의 주변을 사각형으로 정의하여, 그 사각형의 데이터를 반환합니다.



## 드롭다운 옵션 및 입력값



이름	구분	설명	범위 / 종류	
이름	구분	설명	범위 / 종류	
object	드롭다운 옵션	사물	사람 (0), 자전거 (1), 자동차 (2), 오토바이 (3), 비행기 (4), 버스 (5), 기차 (6), 트럭 (7), 배 (8), 신호등 (9), 소화전 (10), 정지 신호 (11), 주차 미터기 (12), 벤치 (13), 새 (14), 고양이 (15), 개 (16), 말 (17), 양 (18), 소 (19), 코끼리 (20), 곰 (21), 얼룩말 (22), 기린 (23), 배낭 (24), 우산 (25), 핸드백 (26), 넥타이 (27), 여행가방 (28), 원반 (29), 스키 (30), 스노보드 (31), 공 (32), 연 (33), 야구 방망이 (34), 야구 글러브 (35), 스케이트보드 (36), 서프보드 (37), 테니스 채 (38), 병 (39), 포도주 잔 (40), 컵 (41), 포크 (42), 칼 (43), 숟가락 (44), 그릇 (45), 바나나 (46), 사과 (47), 샌드위치 (48), 오렌지 (49), 브로콜리 (50), 당근 (51), 핫도그 (52), 피자 (53), 도넛 (54), 케이크 (55), 의자 (56), 소파 (57), 화분 (58), 침대 (59), 식탁 (60), 변기 (61), 텔레비전 (62), 노트북 (63), 마우스 (64), 리모컨 (65), 키보드 (66), 휴대전화 (67), 전자레인지 (68), 오븐 (69), 토스터 (70), 싱크대 (71), 냉장고 (72), 책 (73), 시계 (74),	

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	좌표 방향	x 좌표 최솟값 (min_x), x 좌표 최댓값 (max_x), y 좌표 최솟값 (min_y), y 좌표 최댓값 (max_y), 폭 (width), 높이 (height), 넓이 (area)

## 자바스크립트 코드

```
// 버스 (5) 사각형 x 좌표 최솟값
$('ObjectDetection*0:object.face.min_x').d[5];

// 버스 (5) 사각형 x 좌표 최댓값
$('ObjectDetection*0:object.face.max_x').d[5];

// 버스 (5) 사각형 y 좌표 최솟값
$('ObjectDetection*0:object.face.min_y').d[5];

// 버스 (5) 사각형 y 좌표 최댓값
$('ObjectDetection*0:object.face.max_y').d[5];

// 버스 (5) 사각형 폭
$('ObjectDetection*0:object.face.width').d[5];

// 버스 (5) 사각형 높이
$('ObjectDetection*0:object.face.height').d[5];

// 버스 (5) 사각형 넓이
$('ObjectDetection*0:object.face.area').d[5];
```

## 파이썬 코드

```
# 버스 (5) 사각형 x 좌표 최솟값
__('ObjectDetection*0:object.face.min_x').d[5]

# 버스 (5) 사각형 x 좌표 최댓값
__('ObjectDetection*0:object.face.max_x').d[5]

# 버스 (5) 사각형 y 좌표 최솟값
__('ObjectDetection*0:object.face.min_y').d[5]

# 버스 (5) 사각형 y 좌표 최댓값
__('ObjectDetection*0:object.face.max_y').d[5]

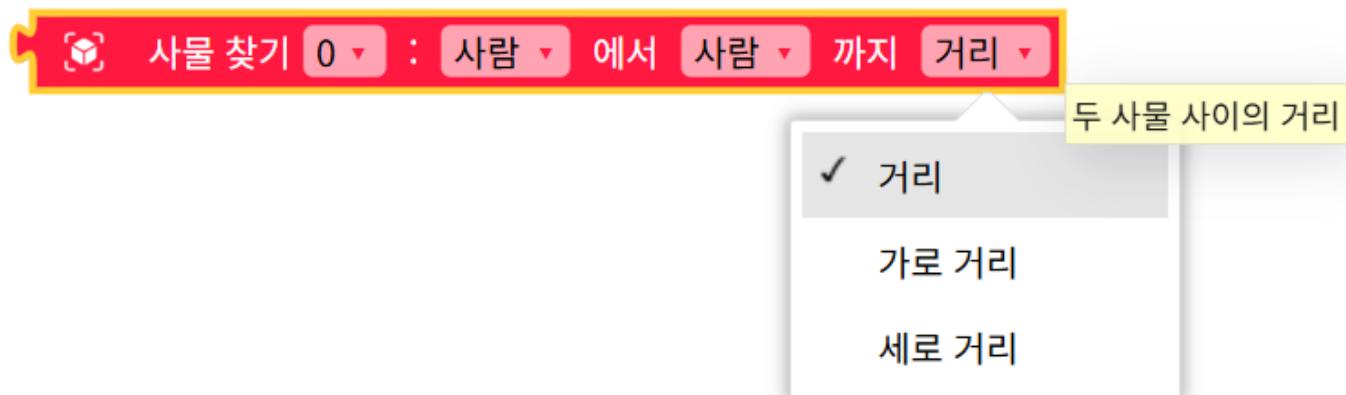
# 버스 (5) 사각형 폭
__('ObjectDetection*0:object.face.width').d[5]

# 버스 (5) 사각형 높이
__('ObjectDetection*0:object.face.height').d[5]

# 버스 (5) 사각형 넓이
__('ObjectDetection*0:object.face.area').d[5]
```

## 두 사물 사이의 거리

선택한 두 사물 사이의 거리를 반환합니다.



드롭다운 옵션 및 입력값



이름	구분	설명	범위 / 종류
이름	구분	설명	범위 / 종류
object 1	드롭다운 옵션	사물	사람 (0), 자전거 (1), 자동차 (2), 오토바이 (3), 비행기 (4), 버스 (5), 기차 (6), 트럭 (7), 배 (8), 신호등 (9), 소화전 (10), 정지 신호 (11), 주차 미터기 (12), 벤치 (13), 새 (14), 고양이 (15), 개 (16), 말 (17), 양 (18), 소 (19), 코끼리 (20), 곰 (21), 얼룩말 (22), 기린 (23), 배낭 (24), 우산 (25), 핸드백 (26), 넥타이 (27), 여행가방 (28), 원반 (29), 스키 (30), 스노보드 (31), 공 (32), 연 (33), 야구 방망이 (34), 야구 글러브 (35), 스케이트보드 (36), 서프보드 (37), 테니스 채 (38), 병 (39), 포도주 잔 (40), 컵 (41), 포크 (42), 칼 (43), 숟가락 (44), 그릇 (45), 바나나 (46), 사과 (47), 샌드위치 (48), 오렌지 (49), 브로콜리 (50), 당근 (51), 핫도그 (52), 피자 (53), 도넛 (54), 케이크 (55), 의자 (56), 소파 (57), 화분 (58), 침대 (59), 식탁 (60), 변기 (61), 텔레비전 (62), 노트북 (63), 마우스 (64), 리모컨 (65), 키보드 (66), 휴대전화 (67), 전자레인지 (68), 오븐 (69), 토스터 (70), 싱크대 (71), 냉장고 (72), 책 (73), 시계 (74),

이름	구분	설명	범위 / 종류
object 2	드롭다운 옵션	사물	사람 (0), 자전거 (1), 자동차 (2), 오토바이 (3), 비행기 (4), 버스 (5), 기차 (6), 트럭 (7), 배 (8), 신호등 (9), 소화전 (10), 정지 신호 (11), 주차 미터기 (12), 벤치 (13), 새 (14), 고양이 (15), 개 (16), 말 (17), 양 (18), 소 (19), 코끼리 (20), 곰 (21), 얼룩말 (22), 기린 (23), 배낭 (24), 우산 (25), 핸드백 (26), 넥타이 (27), 여행가방 (28), 원반 (29), 스키 (30), 스노보드 (31), 공 (32), 연 (33), 야구 방망이 (34), 야구 글러브 (35), 스케이트보드 (36), 서프보드 (37), 테니스 채 (38), 병 (39), 포도주 잔 (40), 컵 (41), 포크 (42), 칼 (43), 숟가락 (44), 그릇 (45), 바나나 (46), 사과 (47), 샌드위치 (48), 오렌지 (49), 브로콜리 (50), 당근 (51), 핫도그 (52), 피자 (53), 도넛 (54), 케이크 (55), 의자 (56), 소파 (57), 화분 (58), 침대 (59), 식탁 (60), 변기 (61), 텔레비전 (62), 노트북 (63), 마우스 (64), 리모컨 (65), 키보드 (66), 휴대전화 (67), 전자레인지 (68), 오븐 (69), 토스터 (70), 싱크대 (71), 냉장고 (72), 책 (73), 시계 (74), 꽃병 (75), 가위 (76), 곰 이

이름	구분	설명	범위 / 종류
distance	드롭다운 옵션	거리	거리 (distance), 가로거리 (horizontal distance), 세로거리 (vertical distance)

## 자바스크립트 코드

```
// 사람 (0) 에서 사람 (0) 까지 거리
Math.sqrt( Math.pow($('ObjectDetection*0:object.x').d[0] - $('ObjectDetection*0:object.x').d[0], 2) + Math.pow($('ObjectDetection*0:object.y').d[0] - $('ObjectDetection*0:object.y').d[0], 2) );

// 사람 (0) 에서 자전거 (1) 까지 가로 거리
Math.abs($('ObjectDetection*0:object.x').d[1] - $('ObjectDetection*0:object.x').d[0]);

// 자동차 (2) 에서 자전거 (1) 까지 세로 거리
Math.abs($('ObjectDetection*0:object.y').d[1] - $('ObjectDetection*0:object.y').d[2]);
```

## 파이썬 코드

```
# 사람 (0) 에서 사람 (0) 까지 거리
math.sqrt( math.pow(_('_('ObjectDetection*0:object.x').d[0] - _('_('ObjectDetection*0:object.x').d[0]), 2) + math.pow(_('_('ObjectDetection*0:object.y').d[0] - _('_('ObjectDetection*0:object.y').d[0]), 2) )

# 사람 (0) 에서 자전거 (1) 까지 가로 거리
math.fabs(_('_('ObjectDetection*0:object.x').d[1] - _('_('ObjectDetection*0:object.x').d[0])

# 자동차 (2) 에서 자전거 (1) 까지 세로 거리
math.fabs(_('_('ObjectDetection*0:object.y').d[1] - _('_('ObjectDetection*0:object.y').d[2]))
```

## 사물이 ~ 일 확률 (신뢰도)

선택한 사물이 맞을 확률 (신뢰도) 를 반환합니다.

반환값은 0 ~ 1 사이 실수 입니다.



선택한 사물이 맞을 확률(신뢰도)

## 드롭다운 옵션 및 입력값



이름	구분	설명	범위 / 종류	
이름	구분	설명	범위 / 종류	
object	드롭다운 옵션	사물	사람 (0), 자전거 (1), 자동차 (2), 오토바이 (3), 비행기 (4), 버스 (5), 기차 (6), 트럭 (7), 배 (8), 신호등 (9), 소화전 (10), 정지 신호 (11), 주차 미터기 (12), 벤치 (13), 새 (14), 고양이 (15), 개 (16), 말 (17), 양 (18), 소 (19), 코끼리 (20), 곰 (21), 얼룩말 (22), 기린 (23), 배낭 (24), 우산 (25), 핸드백 (26), 넥타이 (27), 여행가방 (28), 원반 (29), 스키 (30), 스노보드 (31), 공 (32), 연 (33), 야구 방망이 (34), 야구 글러브 (35), 스케이트보드 (36), 서프보드 (37), 테니스 채 (38), 병 (39), 포도주 잔 (40), 컵 (41), 포크 (42), 칼 (43), 숟가락 (44), 그릇 (45), 바나나 (46), 사과 (47), 샌드위치 (48), 오렌지 (49), 브로콜리 (50), 당근 (51), 핫도그 (52), 피자 (53), 도넛 (54), 케이크 (55), 의자 (56), 소파 (57), 화분 (58), 침대 (59), 식탁 (60), 변기 (61), 텔레비전 (62), 노트북 (63), 마우스 (64), 리모컨 (65), 키보드 (66), 휴대전화 (67), 전자레인지 (68), 오븐 (69), 토스터 (70), 싱크대 (71), 냉장고 (72), 책 (73), 시계 (74),	

## 자바스크립트 코드

```
// 예측 사물이 사람 (0) 일 확률  
$('ObjectDetection*0:object.confidence').d[0];  
  
// 예측 사물이 자전거 (1) 일 확률  
$('ObjectDetection*0:object.confidence').d[1];
```

## 파이썬 코드

```
# 예측 사물이 사람 (0) 일 확률  
__('ObjectDetection*0:object.confidence').d[0]  
  
# 예측 사물이 자전거 (1) 일 확률  
__('ObjectDetection*0:object.confidence').d[1]
```

## 사물 모델 로딩 상태값

사물 모델 로딩 상태를 반환합니다.

아직 불러오지 않았다면 0, 불러오는 중이면 1, 불러오기를 완료했다면 2 를 반환합니다.



사물 모델 로딩 상태를 반환합니다.  
아직 불러오지 않았으면 0, 불러오는 중이면 1, 불러오기를 완료했으면 2를 반환합니다.

## 자바스크립트 코드

```
// 사물 모델 로딩 상태 값  
$('ObjectDetection*0:model_state').d;
```

## 파이썬 코드

```
# 사물 모델 로딩 상태 값  
__('ObjectDetection*0:model_state').d
```

## 사물을 찾았는가?

사물 찾기 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



사물을 찾았는지 여부

## 자바스크립트 코드

```
// 사물을 찾았는가?  
$('ObjectDetection*0:detected').d;
```

## 파이썬 코드

```
# 사물을 찾았는가?  
__('ObjectDetection*0:detected').d;
```

## ~ 을 찾았는가?

선택한 사물을 찾았는지를 참 (1) / 거짓 (0) (으)로 반환합니다.



선택한 사물을 찾았는지 여부

## 드롭다운 옵션 및 입력값



이름	구분	설명	범위 / 종류	
이름	구분	설명	범위 / 종류	
object	드롭다운 옵션	사물	사람 (0), 자전거 (1), 자동차 (2), 오토바이 (3), 비행기 (4), 버스 (5), 기차 (6), 트럭 (7), 배 (8), 신호등 (9), 소화전 (10), 정지 신호 (11), 주차 미터기 (12), 벤치 (13), 새 (14), 고양이 (15), 개 (16), 말 (17), 양 (18), 소 (19), 코끼리 (20), 곰 (21), 얼룩말 (22), 기린 (23), 배낭 (24), 우산 (25), 핸드백 (26), 넥타이 (27), 여행가방 (28), 원반 (29), 스키 (30), 스노보드 (31), 공 (32), 연 (33), 야구 방망이 (34), 야구 글러브 (35), 스케이트보드 (36), 서프보드 (37), 테니스 채 (38), 병 (39), 포도주 잔 (40), 컵 (41), 포크 (42), 칼 (43), 숟가락 (44), 그릇 (45), 바나나 (46), 사과 (47), 샌드위치 (48), 오렌지 (49), 브로콜리 (50), 당근 (51), 핫도그 (52), 피자 (53), 도넛 (54), 케이크 (55), 의자 (56), 소파 (57), 화분 (58), 침대 (59), 식탁 (60), 변기 (61), 텔레비전 (62), 노트북 (63), 마우스 (64), 리모컨 (65), 키보드 (66), 휴대전화 (67), 전자레인지 (68), 오븐 (69), 토스터 (70), 싱크대 (71), 냉장고 (72), 책 (73), 시계 (74),	

## 자바스크립트 코드

```
// 사람 (0) 을 찾았는가?  
$('ObjectDetection*0:object.confidence').d[0] >= $('ObjectDetection*0:confidenceThreshold').d
```

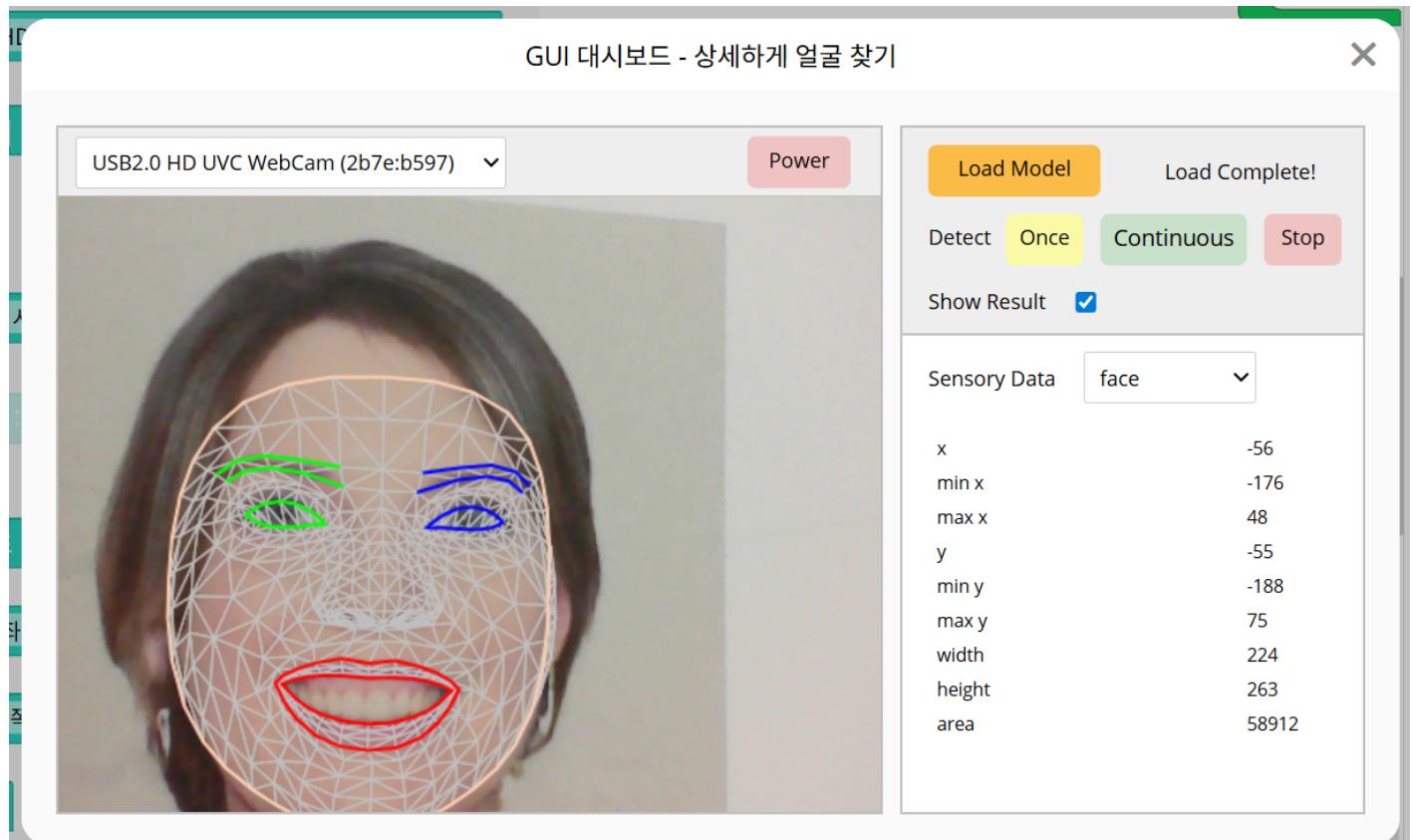
## 파이썬 코드

```
# 사람 (0) 을 찾았는가?  
__('ObjectDetection*0:object.confidence').d[0] >= __('ObjectDetection*0:confidenceThreshold').d
```

## 상세하게-얼굴-찾기

## 대시보드 열기

대시보드 열기는 설치할 수 있는 블록은 아니지만, 확장 모듈에서 사용되는 모델이 어떠한 식으로 적용되는지 알 수 있는 대시보드를 열 수 있습니다. ## 대시보드 화면 대시보드 열기 클릭 시 다음과 같은 화면을 볼 수 있습니다.



## 세부 버튼

### Power

선택한 카메라를 키거나 끕니다.

### Load Model

학습된 상세하게 얼굴 찾기 모델을 불러옵니다. ‘상세하게 얼굴 찾기’확장 모듈을 사용하기 위해서 반드시 필요한 작업입니다.

### Detect

상세하게 얼굴 찾기를 실행하거나 멈춥니다.

Once 버튼으로 한번만 실행할 지, Continuous 버튼으로 연속으로 실행할지 정할 수 있습니다.

또한, Stop 버튼을 통해 찾기를 멈출 수 있습니다.

### Show Result

상세하게 얼굴 찾기 결과를 카메라 화면 상으로 출력합니다.

### Sensory Data

얼굴 찾기에서 찾은 얼굴 데이터 값을 출력합니다.

선택한 얼굴 부위의 x,y 좌표를 확인할 수 있습니다.

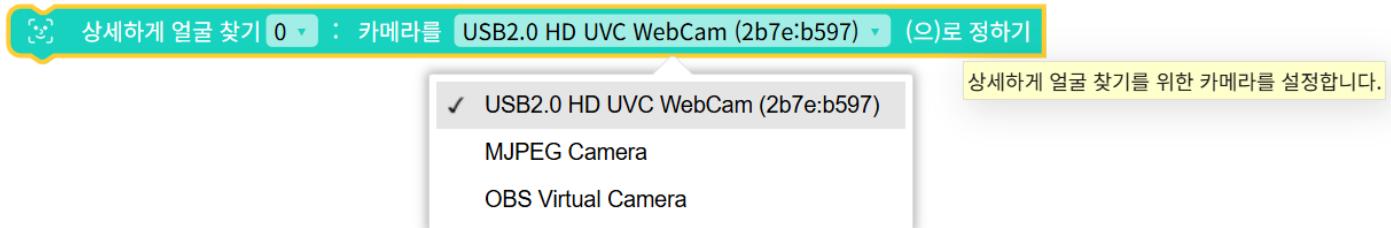
### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
part	드롭다운 옵션	얼굴 부위	face, left eye, right eye, left pupil, right pupil, nose, mouth, left lip, right lip, upper lip, lower lip

# 블럭

## 카메라 정하기

상세하게 얼굴 찾기 모듈에 사용할 카메라를 선택합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
camera	드롭다운 옵션	사용할 카메라	연결한 카메라 리스트

## 자바스크립트 코드

```
// 특정 카메라를 상세하게 얼굴찾기를 위한 카메라로 정하기 (id 는 예시)
$(`'DetailedFaceDetection*0:camera.deviceId'`).d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb';
```

## 파이썬 코드

```
# 특정 카메라를 상세하게 얼굴찾기를 위한 카메라로 정하기 (id 는 예시)
__(`'DetailedFaceDetection*0:camera.deviceId'`).d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb'
```

## 얼굴 모델 불러오기

학습된 얼굴 모델을 불러옵니다. ‘상세하게 얼굴 찾기’모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

기다리기를 체크하면, 모델 불러오기가 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 자바스크립트 코드

```
// 얼굴 모델 불러오기 | 기다리기 0
$(`DetailedFaceDetection*0:load_model`).d = 1;
await $(`DetailedFaceDetection*0:!load_model`).w();

// 얼굴 모델 불러오기 | 기다리기 X
$(`DetailedFaceDetection*0:load_model`).d = 1;
```

## 파이썬 코드

```
# 얼굴 모델 불러오기 | 기다리기 0
__(`DetailedFaceDetection*0:load_model`).d = 1
await __(`DetailedFaceDetection*0:!load_model`).w()

# 얼굴 모델 불러오기 | 기다리기 X
__(`DetailedFaceDetection*0:load_model`).d = 1
```

## 얼굴 한 번 찾기

현재 화면에 있는 얼굴을 찾아 딱 한번 표시합니다.



## 자바스크립트 코드

```
// 얼굴 한 번 찾기
$(`DetailedFaceDetection*0:detect.once`).d = 1;
```

## 파이썬 코드

```
# 얼굴 한 번 찾기
__(`DetailedFaceDetection*0:detect.once`).d = 1
```

## 얼굴 연속으로 찾기

얼굴 연속으로 찾기를 시작하거나 중지합니다.

얼굴 연속으로 찾기를 시작하면, 현재 화면에 있는 얼굴을 계속 따라가며 화면상에 표시합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	얼굴 찾기	시작하기 (1), 중지하기 (0)

## 자바스크립트 코드

```
// 연속으로 얼굴 찾기 시작하기
$(`'DetailedFaceDetection#0:detect.continuous'`).d = 1;

// 연속으로 얼굴 찾기 중지하기
$(`'DetailedFaceDetection#0:detect.continuous'`).d = 0;
```

## 파이썬 코드

```
# 연속으로 얼굴 찾기 시작하기
__(`'DetailedFaceDetection#0:detect.continuous'`).d = 1

# 연속으로 얼굴 찾기 중지하기
__(`'DetailedFaceDetection#0:detect.continuous'`).d = 0
```

## 상세하게 얼굴찾기 결과 보이기

카메라 화면에 상세하게 얼굴 찾기 결과를 표시할지 말지를 결정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	얼굴 찾기 결과	보이기 (1), 숨기기 (0)

## 자바스크립트 코드

```
// 상세하게 얼굴찾기 결과 보이기
$('#DetailedFaceDetection*0:display').d = 1;

// 상세하게 얼굴찾기 결과 숨기기
$('#DetailedFaceDetection*0:display').d = 0;
```

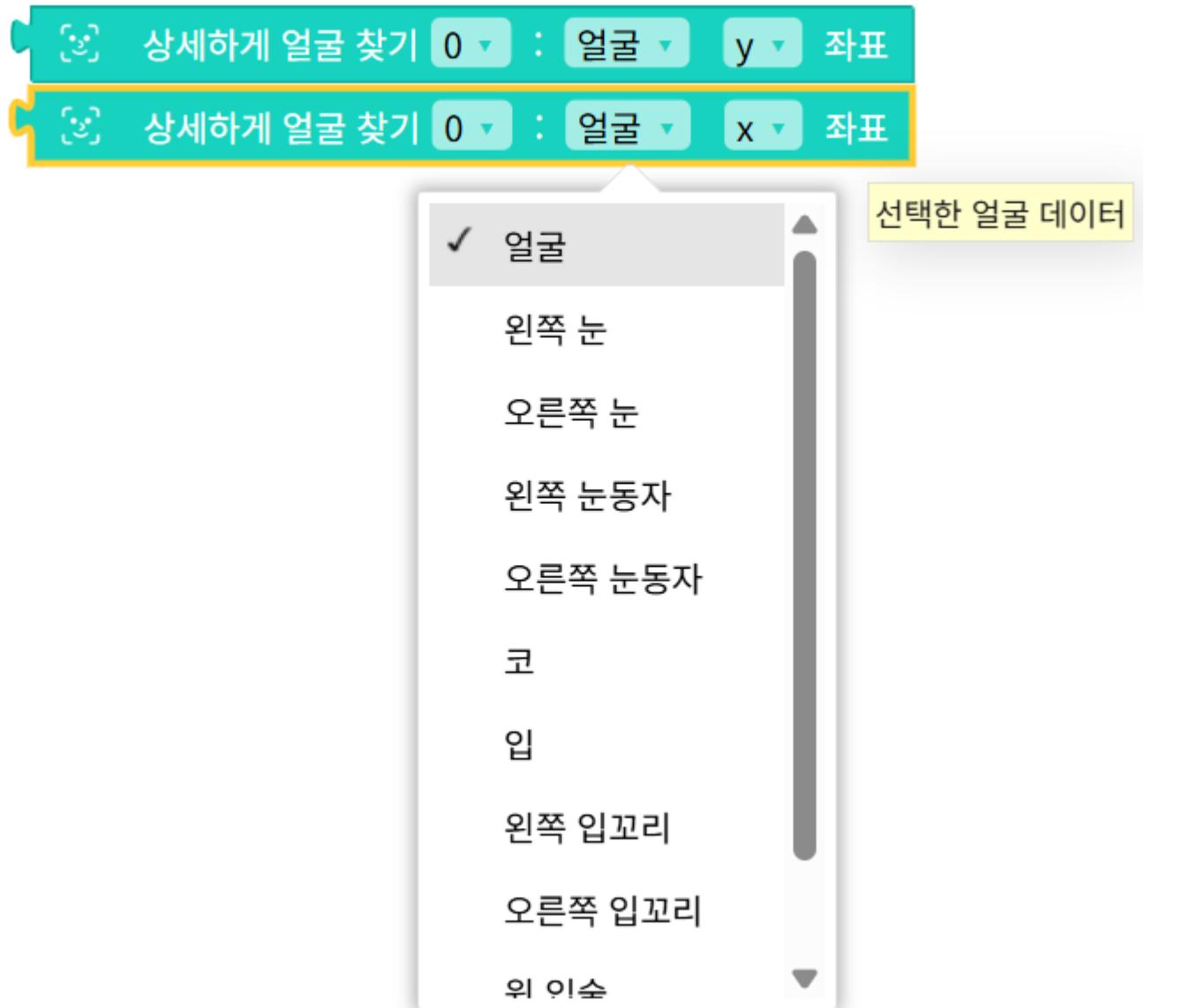
## 파이썬 코드

```
# 상세하게 얼굴찾기 결과 보이기
__('DetailedFaceDetection*0:display').d = 1

# 상세하게 얼굴찾기 결과 숨기기
__('DetailedFaceDetection*0:display').d = 0
```

## 얼굴 데이터

얼굴찾기로 얻은 얼굴 데이터를 반환합니다.



이름	구분	설명	범위 / 종류
part	드롭다운 옵션	얼굴 부위	얼굴 (face), 왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 눈동자 (pupil.left), 오른쪽 눈동자 (pupil.right), 코 (nose), 입 (mouth), 왼쪽 입꼬리 (lip.left), 오른쪽 입꼬리 (lip.right), 윗 입술 (lip.up), 아랫 입술 (lip.down)

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	좌표	x, y

## 자바스크립트 코드

```
// 얼굴 x 좌표
$(`DetailedFaceDetection*0:face.x`).d;

// 얼굴 y 좌표
$(`DetailedFaceDetection*0:face.y`).d;

// 왼쪽 눈 x 좌표
$(`DetailedFaceDetection*0:eye.left.x`).d;

// 왼쪽 눈 y 좌표
$(`DetailedFaceDetection*0:eye.left.y`).d;

// 오른쪽 눈 x 좌표
$(`DetailedFaceDetection*0:eye.right.x`).d;

// 오른쪽 눈 y 좌표
$(`DetailedFaceDetection*0:eye.right.y`).d;

// 왼쪽 눈동자 x 좌표
$(`DetailedFaceDetection*0:pupil.left.x`).d;

// 왼쪽 눈동자 y 좌표
$(`DetailedFaceDetection*0:pupil.left.y`).d;

// 오른쪽 눈동자 x 좌표
$(`DetailedFaceDetection*0:pupil.right.x`).d;

// 오른쪽 눈동자 y 좌표
$(`DetailedFaceDetection*0:pupil.right.y`).d;

// 코 x 좌표
$(`DetailedFaceDetection*0:nose.x`).d;

// 코 y 좌표
$(`DetailedFaceDetection*0:nose.y`).d;

// 입 x 좌표
$(`DetailedFaceDetection*0:mouth.x`).d;

// 입 y 좌표
$(`DetailedFaceDetection*0:mouth.y`).d;

// 왼쪽 입꼬리 x 좌표
$(`DetailedFaceDetection*0:lip.left.x`).d;

// 왼쪽 입꼬리 y 좌표
$(`DetailedFaceDetection*0:lip.left.y`).d;

// 오른쪽 입꼬리 x 좌표
$(`DetailedFaceDetection*0:lip.right.x`).d;

// 오른쪽 입꼬리 y 좌표
$(`DetailedFaceDetection*0:lip.right.y`).d;

// 윗 입술 x 좌표
$(`DetailedFaceDetection*0:lip.up.x`).d;

// 윗 입술 y 좌표
$(`DetailedFaceDetection*0:lip.up.y`).d;

// 아랫 입술 x 좌표
$(`DetailedFaceDetection*0:lip.down.x`).d;

// 아랫 입술 y 좌표
$(`DetailedFaceDetection*0:lip.down.y`).d;
```

## 파이썬 코드

```
# 얼굴 x 좌표
__('DetailedFaceDetection*0:face.x').d

# 얼굴 y 좌표
__('DetailedFaceDetection*0:face.y').d

# 왼쪽 눈 x 좌표
__('DetailedFaceDetection*0:eye.left.x').d

# 왼쪽 눈 y 좌표
__('DetailedFaceDetection*0:eye.left.y').d

# 오른쪽 눈 x 좌표
__('DetailedFaceDetection*0:eye.right.x').d

# 오른쪽 눈 y 좌표
__('DetailedFaceDetection*0:eye.right.y').d

# 왼쪽 눈동자 x 좌표
__('DetailedFaceDetection*0:pupil.left.x').d

# 왼쪽 눈동자 y 좌표
__('DetailedFaceDetection*0:pupil.left.y').d

# 오른쪽 눈동자 x 좌표
__('DetailedFaceDetection*0:pupil.right.x').d

# 오른쪽 눈동자 y 좌표
__('DetailedFaceDetection*0:pupil.right.y').d

# 코 x 좌표
__('DetailedFaceDetection*0:nose.x').d

# 코 y 좌표
__('DetailedFaceDetection*0:nose.y').d

# 입 x 좌표
__('DetailedFaceDetection*0:mouth.x').d

# 입 y 좌표
__('DetailedFaceDetection*0:mouth.y').d

# 왼쪽 입꼬리 x 좌표
__('DetailedFaceDetection*0:lip.left.x').d

# 왼쪽 입꼬리 y 좌표
__('DetailedFaceDetection*0:lip.left.y').d

# 오른쪽 입꼬리 x 좌표
__('DetailedFaceDetection*0:lip.right.x').d

# 오른쪽 입꼬리 y 좌표
__('DetailedFaceDetection*0:lip.right.y').d

# 윗 입술 x 좌표
__('DetailedFaceDetection*0:lip.up.x').d

# 윗 입술 y 좌표
__('DetailedFaceDetection*0:lip.up.y').d

# 아랫 입술 x 좌표
__('DetailedFaceDetection*0:lip.down.x').d

# 아랫 입술 y 좌표
__('DetailedFaceDetection*0:lip.down.y').d
```

## 얼굴 주변 사각형 데이터

얼굴찾기로 찾은 얼굴의 주변을 사각형으로 정의하여, 그 사각형의 데이터를 반환합니다.

상세하게 얼굴 찾기 0 : 얼굴 사각형 x 좌표 최솟값 얼굴 주변 사각형에 대한 데이터

- x 좌표 최솟값
- x 좌표 최댓값
- y 좌표 최솟값
- y 좌표 최댓값
- 폭
- 높이
- 넓이

이름	구분	설명	범위 / 종류
part	드롭다운 옵션	얼굴 부위	얼굴 (face), 왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 입 (mouth)
data	드롭다운 옵션	얼굴 데이터	x 좌표 최솟값 (min_x), x 좌표 최댓값 (max_x), y 좌표 최솟값 (min_y), y 좌표 최댓값 (max_y), 폭 (width), 높이 (height), 넓이 (area)

## 자바스크립트 코드

```
// 얼굴 사각형 x 좌표 최솟값
$('DetailedFaceDetection*0:face.min_x').d;

// 얼굴 사각형 x 좌표 최댓값
$('DetailedFaceDetection*0:face.max_x').d;

// 얼굴 사각형 y 좌표 최솟값
$('DetailedFaceDetection*0:face.min_y').d;

// 얼굴 사각형 y 좌표 최댓값
$('DetailedFaceDetection*0:face.max_y').d;

// 얼굴 사각형 폭
$('DetailedFaceDetection*0:face.width').d;

// 얼굴 사각형 높이
$('DetailedFaceDetection*0:face.height').d;
```

```

// 얼굴 사각형 넓이
$(`'DetailedFaceDetection*0:face.area').d;

// 왼쪽 눈 사각형 x 좌표 최솟값
$(`'DetailedFaceDetection*0:eye.left.min_x').d;

// 오른쪽 눈 사각형 x 좌표 최솟값
$(`'DetailedFaceDetection*0:eye.right.min_x').d;

// 입 사각형 x 좌표 최솟값
$(`'DetailedFaceDetection*0:mouth.min_x').d;

```

## 파이썬 코드

```

# 얼굴 사각형 x 좌표 최솟값
__(`'DetailedFaceDetection*0:face.min_x').d

# 얼굴 사각형 x 좌표 최댓값
__(`'DetailedFaceDetection*0:face.max_x').d

# 얼굴 사각형 y 좌표 최솟값
__(`'DetailedFaceDetection*0:face.min_y').d

# 얼굴 사각형 y 좌표 최댓값
__(`'DetailedFaceDetection*0:face.max_y').d

# 얼굴 사각형 폭
__(`'DetailedFaceDetection*0:face.width').d

# 얼굴 사각형 높이
__(`'DetailedFaceDetection*0:face.height').d

# 얼굴 사각형 넓이
__(`'DetailedFaceDetection*0:face.area').d

# 왼쪽 눈 사각형 x 좌표 최솟값
__(`'DetailedFaceDetection*0:eye.left.min_x').d

# 오른쪽 눈 사각형 x 좌표 최솟값
__(`'DetailedFaceDetection*0:eye.right.min_x').d

# 입 사각형 x 좌표 최솟값
__(`'DetailedFaceDetection*0:mouth.min_x').d

```

## 얼굴 요소 사이 거리

얼굴찾기로 찾은 얼굴의 데이터를 이용하여 두 얼굴 요소 사이의 거리를 반환합니다.



이름	구분	설명	범위 / 종류
part 1	드롭다운 옵션	얼굴 부위	얼굴 (face), 왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 눈동자 (pupil.left), 오른쪽 눈동자 (pupil.right), 코 (nose), 입 (mouth), 왼쪽 입꼬리 (lip.left), 오른쪽 입꼬리 (lip.right), 윗 입술 (lip.up), 아랫 입술 (lip.down)

이름	구분	설명	범위 / 종류
part 2	드롭다운 옵션	얼굴 부위	얼굴 (face), 왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 눈동자 (pupil.left), 오른쪽 눈동자 (pupil.right), 코 (nose), 입 (mouth), 왼쪽 입꼬리 (lip.left), 오른쪽 입꼬리 (lip.right), 윗 입술 (lip.up), 아랫 입술 (lip.down)
distance	드롭다운 옵션	거리	거리 (distance), 가로거리 (horizontal distance), 세 로거리 (vertical distance)

## 자바스크립트 코드

```
// 원쪽 눈에서 원쪽 눈 까지 거리
Math.sqrt( Math.pow($('DetailedFaceDetection*0:eye.left.x').d - $('DetailedFaceDetection*0:eye.left.x').d, 2) + Math.pow($('DetailedFaceDetection*0:eye.left.y').d - $('DetailedFaceDetection*0:eye.left.y').d, 2) );

// 원쪽 눈에서 오른쪽 눈 까지 거리
Math.sqrt( Math.pow($('DetailedFaceDetection*0:eye.right.x').d - $('DetailedFaceDetection*0:eye.left.x').d, 2) + Math.pow($('DetailedFaceDetection*0:eye.right.y').d - $('DetailedFaceDetection*0:eye.left.y').d, 2) );

// 원쪽 눈에서 원쪽 눈동자 까지 가로 거리
Math.abs($('DetailedFaceDetection*0:pupil.left.x').d - $('DetailedFaceDetection*0:eye.left.x').d);

// 원쪽 눈에서 오른쪽 눈동자 까지 가로 거리
Math.abs($('DetailedFaceDetection*0:pupil.right.x').d - $('DetailedFaceDetection*0:eye.left.x').d);

// 원쪽 눈에서 코 까지 세로 거리
Math.abs($('DetailedFaceDetection*0:nose.y').d - $('DetailedFaceDetection*0:eye.left.y').d);

// 원쪽 눈에서 입 까지 세로 거리
Math.abs($('DetailedFaceDetection*0:mouth.y').d - $('DetailedFaceDetection*0:eye.left.y').d);

// 원쪽 눈에서 왼쪽 입꼬리 까지 거리
Math.sqrt( Math.pow($('DetailedFaceDetection*0:lip.left.x').d - $('DetailedFaceDetection*0:eye.left.x').d, 2) + Math.pow($('DetailedFaceDetection*0:lip.left.y').d - $('DetailedFaceDetection*0:eye.left.y').d, 2) );

// 원쪽 눈에서 오른쪽 입꼬리 까지 거리
Math.sqrt( Math.pow($('DetailedFaceDetection*0:lip.right.x').d - $('DetailedFaceDetection*0:eye.left.x').d, 2) + Math.pow($('DetailedFaceDetection*0:lip.right.y').d - $('DetailedFaceDetection*0:eye.left.y').d, 2) );

// 원쪽 눈에서 윗 입술 까지 거리
Math.sqrt( Math.pow($('DetailedFaceDetection*0:lip.up.x').d - $('DetailedFaceDetection*0:eye.left.x').d, 2) + Math.pow($('DetailedFaceDetection*0:lip.up.y').d - $('DetailedFaceDetection*0:eye.left.y').d, 2) );

// 원쪽 눈에서 아랫 입술 까지 거리
Math.sqrt( Math.pow($('DetailedFaceDetection*0:lip.down.x').d - $('DetailedFaceDetection*0:eye.left.x').d, 2) + Math.pow($('DetailedFaceDetection*0:lip.down.y').d - $('DetailedFaceDetection*0:eye.left.y').d, 2) );
```

## 파이썬 코드

```
# 왼쪽 눈에서 왼쪽 눈 까지 거리
math.sqrt( math.pow(_('_('DetailedFaceDetection*0:eye.left.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('DetailedFaceDetection*0:eye.left.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d), 2) ) )

# 왼쪽 눈에서 오른쪽 눈 까지 거리
math.sqrt( math.pow(_('_('DetailedFaceDetection*0:eye.right.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('DetailedFaceDetection*0:eye.right.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d), 2) ) )

# 왼쪽 눈에서 왼쪽 눈동자 까지 가로 거리
math.fabs(_('_('DetailedFaceDetection*0:pupil.left.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d)

# 왼쪽 눈에서 오른쪽 눈동자 까지 가로 거리
math.fabs(_('_('DetailedFaceDetection*0:pupil.right.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d)

# 왼쪽 눈에서 코 까지 세로 거리
math.fabs(_('_('DetailedFaceDetection*0:nose.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d)

# 왼쪽 눈에서 입 까지 세로 거리
math.fabs(_('_('DetailedFaceDetection*0:mouth.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d)

# 왼쪽 눈에서 왼쪽 입꼬리 까지 거리
math.sqrt( math.pow(_('_('DetailedFaceDetection*0:lip.left.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('DetailedFaceDetection*0:lip.left.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d), 2) ) )

# 왼쪽 눈에서 오른쪽 입꼬리 까지 거리
math.sqrt( math.pow(_('_('DetailedFaceDetection*0:lip.right.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('DetailedFaceDetection*0:lip.right.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d), 2) ) )

# 왼쪽 눈에서 윗 입술 까지 거리
math.sqrt( math.pow(_('_('DetailedFaceDetection*0:lip.up.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('DetailedFaceDetection*0:lip.up.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d), 2) ) )

# 왼쪽 눈에서 아랫 입술 까지 거리
math.sqrt( math.pow(_('_('DetailedFaceDetection*0:lip.down.x').d - _('_('DetailedFaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('DetailedFaceDetection*0:lip.down.y').d - _('_('DetailedFaceDetection*0:eye.left.y').d), 2) ) )
```

## 얼굴 모델 로딩 상태값

얼굴 모델 로딩 상태를 반환합니다.

아직 불러오지 않았다면 0, 불러오는 중이면 1, 불러오기를 완료했다면 2를 반환합니다.



## 자바스크립트 코드

```
// 얼굴 모델 로딩 상태 값
$('DetailedFaceDetection*0:model_state').d;
```

## 파이썬 코드

```
# 얼굴 모델 로딩 상태 값  
__("DetailedFaceDetection*0:model_state").d
```

## 얼굴을 찾았는가?

얼굴 찾기 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 자바스크립트 코드

```
// 얼굴을 찾았는가?  
__("DetailedFaceDetection*0:detectected").d;
```

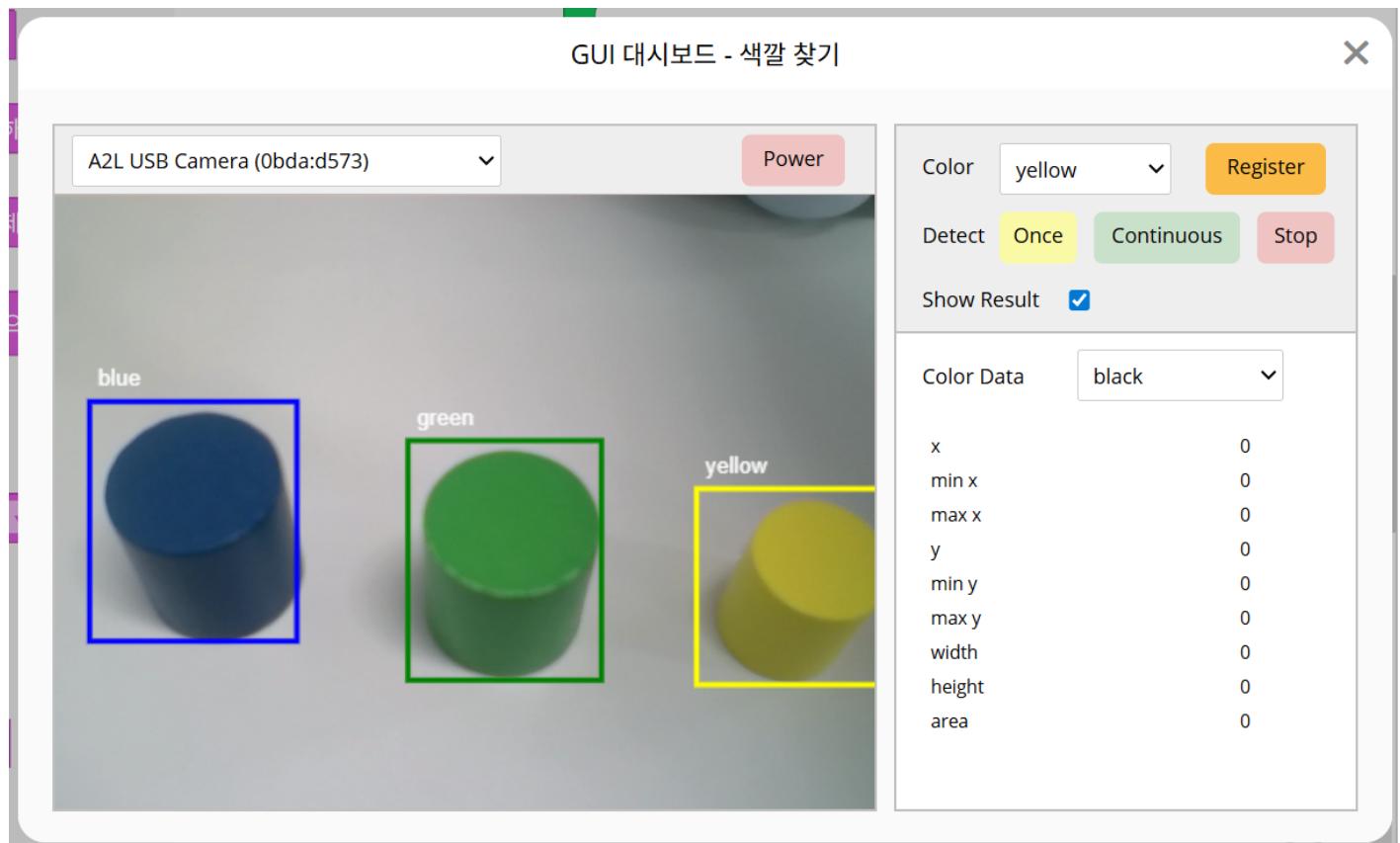
## 파이썬 코드

```
# 얼굴을 찾았는가?  
__("DetailedFaceDetection*0:detectected").d
```

## 색깔-찾기

## 대시보드 열기

대시보드 열기는 설치할 수 있는 블록은 아니지만, 확장 모듈에서 사용되는 모델이 어떠한 식으로 적용되는지 알 수 있는 대시보드를 열 수 있습니다. ## 대시보드 화면 대시보드 열기 클릭 시 다음과 같은 화면을 볼 수 있습니다.



## 세부 버튼

### Power

선택한 카메라를 키거나 끕니다.

### Register

드롭다운 옵션으로 선택한 색을 찾을 색깔에 추가합니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	선택한 색	black, red, yellow, green, cyan, blue, magenta, white

## Detect

색깔 찾기를 실행하거나 멈춥니다.

Once 버튼으로 한번만 실행할지, Continuous 버튼으로 연속으로 실행할지 정할 수 있습니다.

## Show Result

색깔 찾기 결과를 카메라 화면 상으로 출력합니다.

## Color Data

선택한 색깔에 따른 데이터 값을 출력합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	선택한 색	black, red, yellow, green, cyan, blue, magenta, white

## 블럭

### 카메라 정하기

색깔 찾기 모듈에 사용할 카메라를 선택합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
camera	드롭다운 옵션	사용할 카메라	연결한 카메라 리스트

## 자바스크립트 코드

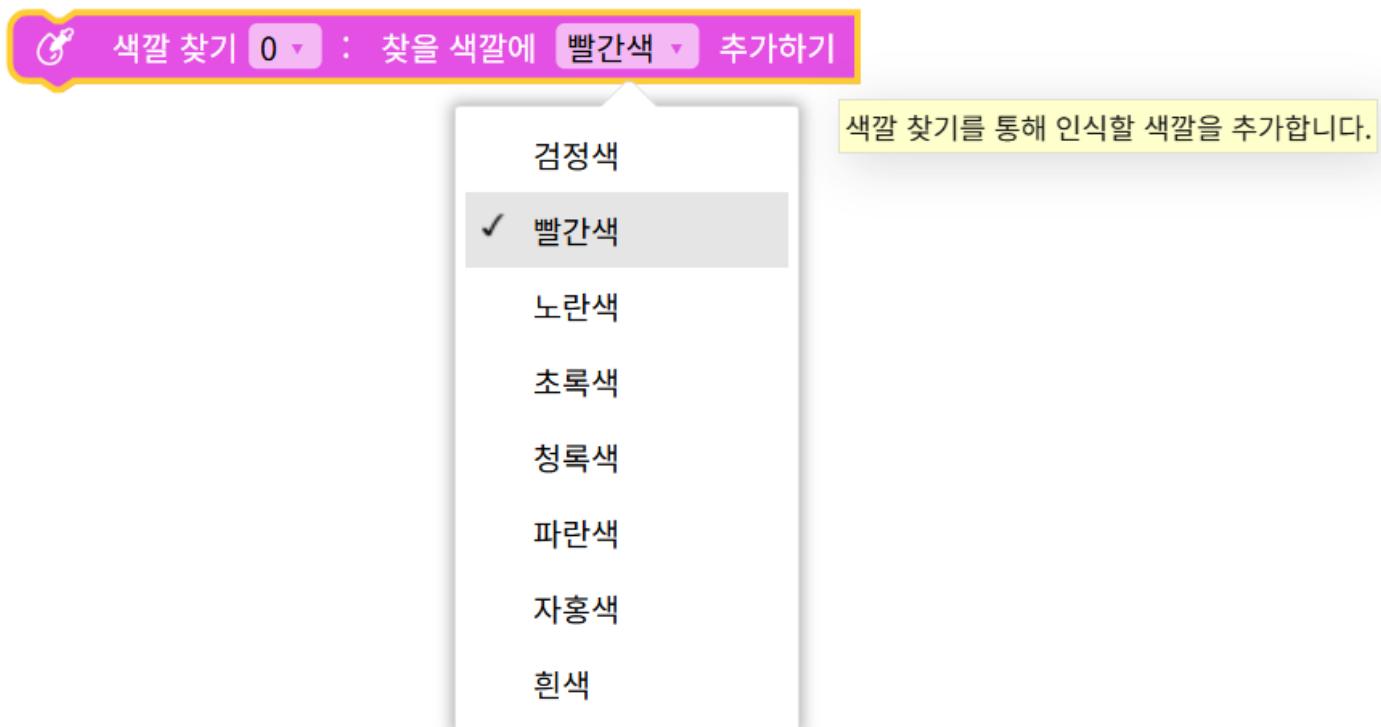
```
// 특정 카메라를 색깔 찾기를 위한 카메라로 정하기 (id 는 예시)
$(`ColorDetection*0:camera.deviceId`).d = 'feed7de06e4339e1a37c0982453051d5a706fa8c06064bfcf215e77b2dc8818';
```

## 파이썬 코드

```
# 특정 카메라를 색깔 찾기를 위한 카메라로 정하기 (id 는 예시)
__(`ColorDetection*0:camera.deviceId').d = 'feed7de06e4339e1a37c0982453051d5a706fa8c06064bfcf215e77b2dc8818'
```

## 인식 색깔 추가하기

색깔 찾기를 통해 인식할 색깔을 추가합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	인식 색깔	검정색 (0), 빨간색 (1), 노란색 (2), 초록색 (3), 청록색 (4), 파란색 (5), 자홍색 (6), 흰색 (7)

## 자바스크립트 코드

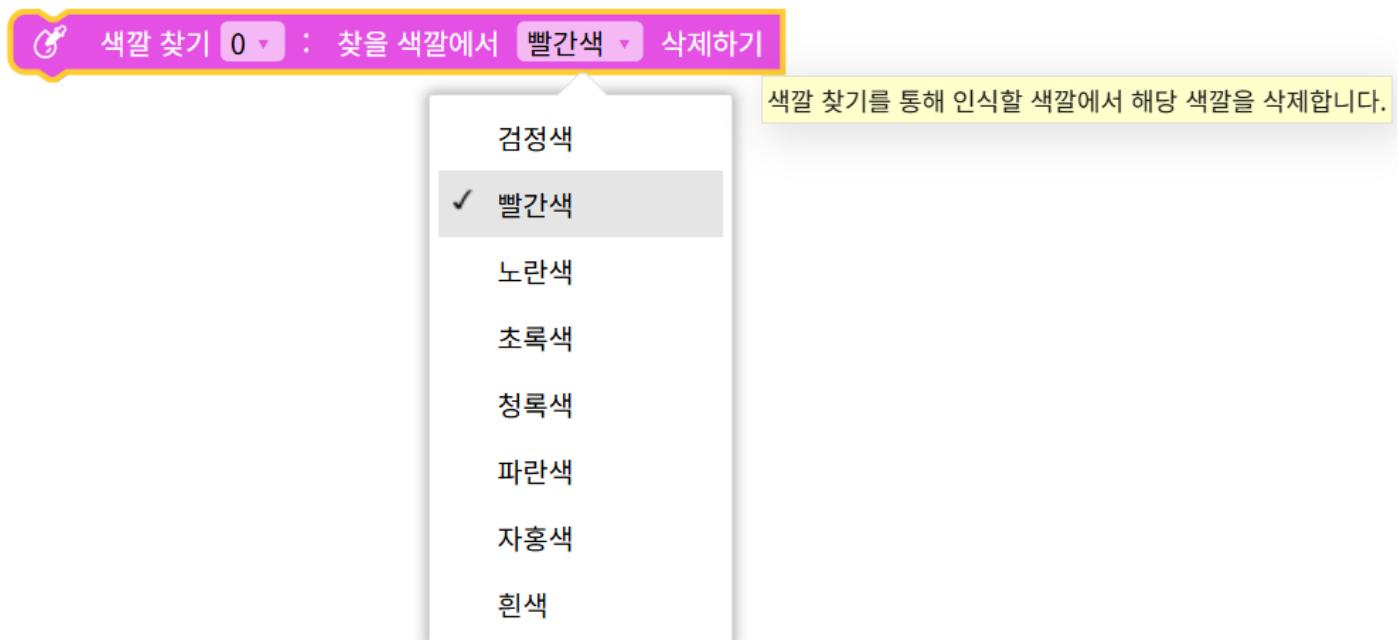
```
// 찾을 색깔에 검정색 (0) 추가하기  
$('ColorDetection*0:color.register').d = 0;  
await $('ColorDetection*0:color.!register').w();  
  
// 찾을 색깔에 초록색 (3) 추가하기  
$('ColorDetection*0:color.register').d = 3;  
await $('ColorDetection*0:color.!register').w();  
  
// 찾을 색깔에 흰색 (7) 추가하기  
$('ColorDetection*0:color.register').d = 7;  
await $('ColorDetection*0:color.!register').w();
```

## 파이썬 코드

```
# 찾을 색깔에 검정색 (0) 추가하기  
__('ColorDetection*0:color.register').d = 0  
await __('ColorDetection*0:color.!register').w()  
  
# 찾을 색깔에 초록색 (3) 추가하기  
__('ColorDetection*0:color.register').d = 3  
await __('ColorDetection*0:color.!register').w()  
  
# 찾을 색깔에 흰색 (7) 추가하기  
__('ColorDetection*0:color.register').d = 7  
await __('ColorDetection*0:color.!register').w()
```

## 인식 색깔 삭제하기

색깔 찾기를 통해 인식할 색깔을 삭제합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	인식 색깔	검정색 (0), 빨간색 (1), 노란색 (2), 초록색 (3), 청록색 (4), 파란색 (5), 자홍색 (6), 흰색 (7)

## 자바스크립트 코드

```
// 찾을 색깔에 검정색 (0) 삭제하기
$(`'ColorDetection*0:color.delete').d = 0;
await $(`'ColorDetection*0:color.!delete').w();

// 찾을 색깔에 빨간색 (1) 삭제하기
$(`'ColorDetection*0:color.delete').d = 1;
await $(`'ColorDetection*0:color.!delete').w();

// 찾을 색깔에 청록색 (4) 삭제하기
$(`'ColorDetection*0:color.delete').d = 4;
await $(`'ColorDetection*0:color.!delete').w();
```

## 파이썬 코드

```
# 찾을 색깔에 검정색 (0) 삭제하기
__('ColorDetection*0:color.delete').d = 0
await __('ColorDetection*0:color.!delete').w()

# 찾을 색깔에 빨간색 (1) 삭제하기
__('ColorDetection*0:color.delete').d = 1
await __('ColorDetection*0:color.!delete').w()

# 찾을 색깔에 청록색 (4) 삭제하기
__('ColorDetection*0:color.delete').d = 4
await __('ColorDetection*0:color.!delete').w()
```

## 색깔 찾기 최소 넓이 정하기

색깔 찾기의 최소 넓이를 설정합니다. 넓이가 이 이상인 경우에만 화면에 표시됩니다.

☞ 색깔 찾기 0 : 찾기 조건을 넓이 > 50 (으)로 정하기

인식할 색깔 영역 넓이의 최소 크기를 정합니다. 영역의 넓이가 이 이상인 경우에만 화면에 표시됩니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
area	입력값	색깔 찾기 최소 넓이	0 이상 실수

## 자바스크립트 코드

```
// 색깔 찾기 최소 넓이 조건을 50 으로 정하기  
$('ColorDetection*0:color.area_condition').d = 50;
```

## 파이썬 코드

```
# 색깔 찾기 최소 넓이 조건을 50 으로 정하기  
__('ColorDetection*0:color.area_condition').d = 50
```

### 색깔 한번 찾기

인식 가능한 색깔 중, 현재 화면에 있는 색깔들을 찾아 딱 한번 영역을 표시합니다.



색깔 찾기 0 : 색깔을 한 번 찾기

인식 가능한 색깔 중, 현재 화면에 있는 색깔들을 찾아 딱 한번 영역을 표시합니다.

## 자바스크립트 코드

```
// 색깔 한 번 찾기  
$('ColorDetection*0:detect.once').d = 1;
```

## 파이썬 코드

```
# 색깔 한 번 찾기  
__('ColorDetection*0:detect.once').d = 1
```

### 색깔 연속으로 찾기

색깔 연속으로 찾기를 시작하거나 중지합니다.

색깔 연속으로 찾기를 시작하면, 인식 가능한 색 중 현재 화면에 있는 색깔들을 계속해서 따라가며 화면 상에 영역을 표시합니다.



색깔 찾기 0 : 연속으로 색깔 찾기

✓ 시작하기

종지하기

인식 가능한 색깔 중, 현재 화면에 있는 색깔들을 계속해서 따라가며 화면 상에 영역을 표시합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	색깔 찾기 결과	보이기 (1), 숨기기 (0)

## 자바스크립트 코드

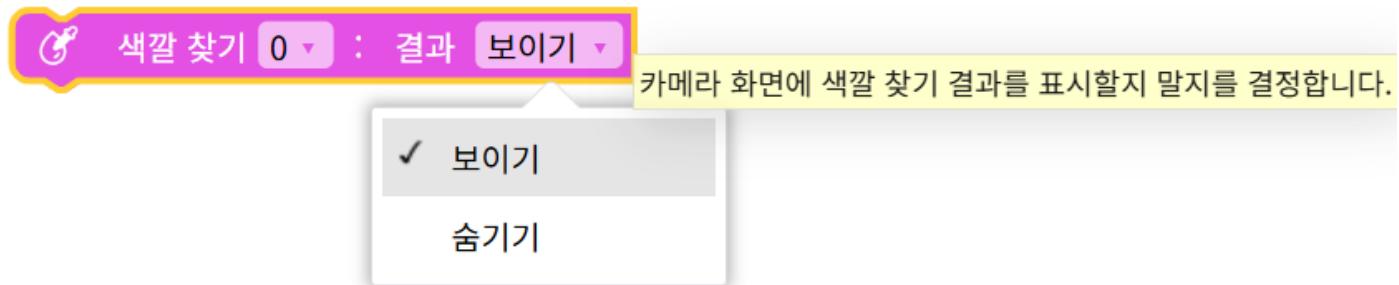
```
// 연속으로 색깔 찾기 시작하기  
$('ColorDetection*0:detect.continuous').d = 1;  
  
// 연속으로 색깔 찾기 중지하기  
$('ColorDetection*0:detect.continuous').d = 0;
```

## 파이썬 코드

```
# 연속으로 색깔 찾기 시작하기  
__('ColorDetection*0:detect.continuous').d = 1  
  
# 연속으로 색깔 찾기 중지하기  
__('ColorDetection*0:detect.continuous').d = 0
```

## 색깔 찾기 결과 보이기

카메라 화면에 색깔 찾기 결과를 표시할지 말지를 결정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	색깔 찾기 결과	보이기 (1), 숨기기 (0)

## 자바스크립트 코드

```
// 색깔 찾기 결과 보이기  
$('ColorDetection*0:display').d = 1;  
  
// 색깔 찾기 결과 숨기기  
$('ColorDetection*0:display').d = 0;
```

## 파이썬 코드

```
# 색깔 찾기 결과 보이기  
__('ColorDetection*0:display').d = 1  
  
# 색깔 찾기 결과 숨기기  
__('ColorDetection*0:display').d = 0
```

## 색깔 관련 데이터

선택한 색깔 영역의 데이터를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	인식 색깔	검정색 (0), 빨간색 (1), 노란색 (2), 초록색 (3), 청록색 (4), 파란색 (5), 자홍색 (6), 흰색 (7)
axis	드롭다운 옵션	좌표 방향	x 좌표 최솟값 (min_x), x 좌표 최댓값 (max_x), y 좌표 최솟값 (min_y), y 좌표 최댓값 (max_y), 폭 (width), 높이 (height), 넓이 (area)

## 자바스크립트 코드

```
// 빨간색 (1) 영역의 x 좌표
$('ColorDetection*0:color.x').d[1];

// 빨간색 (1) 영역의 y 좌표
$('ColorDetection*0:color.y').d[1];

// 빨간색 (1) 영역의 x 좌표 최솟값
$('ColorDetection*0:color.min_x').d[1];

// 빨간색 (1) 영역의 x 좌표 최댓값
$('ColorDetection*0:color.max_x').d[1];

// 빨간색 (1) 영역의 y 좌표 최솟값
$('ColorDetection*0:color.min_y').d[1];

// 빨간색 (1) 영역의 y 좌표 최댓값
$('ColorDetection*0:color.max_y').d[1];

// 빨간색 (1) 영역의 너비
$('ColorDetection*0:color.width').d[1];

// 빨간색 (1) 영역의 높이
$('ColorDetection*0:color.height').d[1];

// 빨간색 (1) 영역의 넓이
$('ColorDetection*0:color.area').d[1];
```

## 파이썬 코드

```
# 빨간색 (1) 영역의 x 좌표
__('ColorDetection*0:color.x').d[1]

# 빨간색 (1) 영역의 y 좌표
__('ColorDetection*0:color.y').d[1]

# 빨간색 (1) 영역의 x 좌표 최솟값
__('ColorDetection*0:color.min_x').d[1]

# 빨간색 (1) 영역의 x 좌표 최댓값
__('ColorDetection*0:color.max_x').d[1]
```

```

# 빨간색 (1) 영역의 y 좌표 최솟값
__('ColorDetection*0:color.min_y').d[1]

# 빨간색 (1) 영역의 y 좌표 최댓값
__('ColorDetection*0:color.max_y').d[1]

# 빨간색 (1) 영역의 너비
__('ColorDetection*0:color.width').d[1]

# 빨간색 (1) 영역의 높이
__('ColorDetection*0:color.height').d[1]

# 빨간색 (1) 영역의 넓이
__('ColorDetection*0:color.area').d[1]

```

## ~ 색을 찾았는가?

선택한 색깔을 찾았는지를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	인식 색깔	검정색 (0), 빨간색 (1), 노란색 (2), 초록색 (3), 청록색 (4), 파란색 (5), 자홍색 (6), 흰색 (7)

## 자바스크립트 코드

```
// 빨간색 (1) 을 찾았는가?
$('ColorDetection*0:color.area').d[1] >= $('ColorDetection*0:color.area_condition').d;

// 파란색 (5) 을 찾았는가?
$('ColorDetection*0:color.area').d[5] >= $('ColorDetection*0:color.area_condition').d;
```

## 파이썬 코드

```
# 빨간색 (1) 을 찾았는가?
__('ColorDetection*0:color.area').d[1] >= __('ColorDetection*0:color.area_condition').d

# 파란색 (5) 을 찾았는가?
__('ColorDetection*0:color.area').d[5] >= __('ColorDetection*0:color.area_condition').d
```

## 색상

색상은 다양한 그래픽 프로그램에서 사용됩니다.

## 블록

### 팔레트에서 색상 선택

가장 간단한 색상을 얻는 방법은 **색상 선택기**를 사용하는 것입니다. 이 블록은 흰색의 둥근 사각형으로 표시됩니다. 클릭하면 색상 팔레트가 나타나며, 원하는 색상을 선택할 수 있습니다.



Figure 90: Image

## Javascript 코드

```
[255, 0, 0]; // 기본 색상 검은색  
[255, 0, 0]; // 기본 색상 빨간색  
[255, 255, 0]; // 기본 색상 노란색  
[0, 255, 0]; // 기본 색상 초록색  
[0, 255, 255]; // 기본 색상 청록색  
[0, 0, 255]; // 기본 색상 파란색  
[255, 0, 255]; // 기본 색상 분홍색  
[255, 255, 255]; // 기본 색상 하얀색
```

## Python 코드

```
[255, 0, 0] # 기본 색상 검은색  
[255, 0, 0] # 기본 색상 빨간색  
[255, 255, 0] # 기본 색상 노란색  
[0, 255, 0] # 기본 색상 초록색  
[0, 255, 255] # 기본 색상 청록색  
[0, 0, 255] # 기본 색상 파란색  
[255, 0, 255] # 기본 색상 분홍색  
[255, 255, 255] # 기본 색상 하얀색
```

## 빨강, 초록, 파랑 (RGB) 값으로 색상 만들기

색상 설정 블록을 사용하면 원하는 빨강, 초록, 파랑의 비율을 지정할 수 있습니다. 아래 예제에서는 빨강과 파랑을 최대로 설정하고 초록을 0으로 설정하여 보라색을 만듭니다.



Figure 91: Image

각 색상 요소의 범위는 0에서 255(포함) 까지입니다.

## Javascript 코드

```
[255, 0, 255]; // R : 255, G : 0, B : 255
```

## Python 코드

```
[255, 0, 255] # R : 255, G : 0, B : 255
```

## 슬라이더를 사용한 색상 생성

이 블록은 **슬라이더**를 이용해 색상을 선택하는 기능을 제공합니다. 사용자는 슬라이더를 조절하여 원하는 색상을 직접 조합할 수 있습니다. 각 슬라이더는 빨강 (R), 초록 (G), 파랑 (B) 값 조정을 담당하며, 오른쪽 버튼을 이용해 **명도** (밝기)를 조절할 수 있습니다.



Figure 92: Image

슬라이더 값을 변경하면 즉시 반영되어, 선택한 색상이 R, G, B 영역에 실시간으로 표시됩니다.

### Javascript 코드

```
[90, 85, 224]; // R : 90, G : 85, B : 224
```

### Python 코드

```
[90, 85, 224] # R : 90, G : 85, B : 224
```

### 랜덤 색상 생성

랜덤 색상 블록은 호출될 때마다 무작위 색상을 생성합니다.

무작위 색상

Figure 93: Image

이 블록은 빨강, 초록, 파랑 값을 각각 0~255(포함) 사이의 랜덤한 값으로 설정합니다.

### Javascript 코드

```
--randomColor(); // 무작위 색상 생성
```

## Python 코드

```
__randomColor() # 무작위 색상 생성
```

## 기술적 세부 사항

블록 컴포저에서 색상은 “rr,gg,bb” 형식의 텍스트로 표현됩니다. 여기서 “rr”, “gg”, “bb”는 각각 빨강, 초록, 파랑 값 (0 ~ 255)을 나타냅니다. 일반적으로 이 형식은 사용자가 직접 볼 일이 없지만, 아래 프로그램을 실행하면 확인할 수 있습니다.



Figure 94: Image

위 프로그램은 “255,255,255” 을 출력합니다.

## Javascript 코드

```
window.alert([255, 255, 255]); // "255,255,255" 출력
```

## Python 코드

```
print([255, 255, 255]) # "255,255,255" 출력
```

여담으로, 빛을 혼합하는 방식은 페인트를 혼합하는 방식과 다릅니다. 빨강, 초록, 파랑 빛을 동일한 비율로 혼합하면 흰색이 되지만, 페인트를 섞으면 탁한 색이 나옵니다. ## 소리

---

소리 블록을 활용하면 다양한 효과음과 음성을 재생할 수 있습니다.

## 블록

### 소리 재생하기

**소리 재생하기** 블록은 원하는 소리를 지정한 **볼륨**으로 재생하는 기능을 수행합니다. - **볼륨 조절**: 소리 크기를 조절할 수 있습니다. - **반복 재생**: 반복 체크박스를 활성화하면 선택한 소리를 계속해서 반복 재생할 수 있습니다.

소리 ▾ 를 볼륨 100 (으)로 반복 재생하기

Figure 95: image

### Javascript 코드

```
--playSound(' ', 100, false);
```

### Python 코드

```
--playSound(' ', 100, False)
```

### 소리 추가

상단의 **소리** 메뉴에서 원하는 소리를 선택할 수 있습니다.



Figure 96: Image

목록에서 소리를 선택하기 전에 먼저 확인하고 직접 들어볼 수 있습니다.

소리 추가 기능을 활용하면 프로젝트에 원하는 효과음을 직접 추가할 수 있습니다. - 추가된 소리는 왼쪽 목록에서 확인할 수 있습니다. - 목록에서 선택한 소리를 재생하여 미리 들어볼 수 있으며, 필요 없는 소리는 삭제할 수도 있습니다.

### 소리 선택

블록 아래 화살표를 클릭하면 추가한 소리들을 확인하고 선택하여 사용할 수 있습니다.

### Javascript 코드

```
--playSound('Notes/A Elec Bass', 100, false); // Notes/A Elec Bass 소리 선택
```

### Python 코드

```
--playSound('Notes/A Elec Bass', 100, False) # Notes/A Elec Bass 소리 선택
```

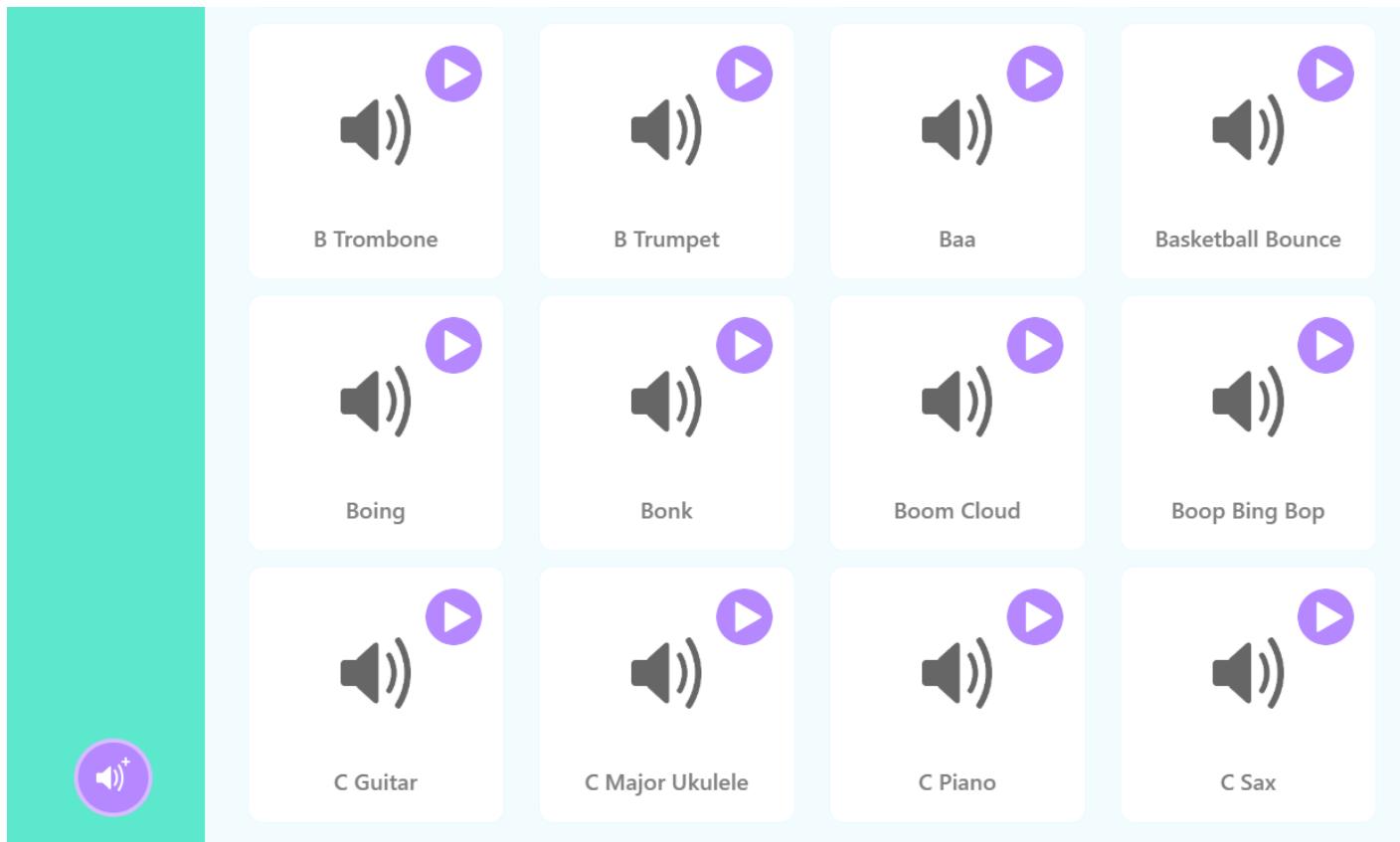


Figure 97: image

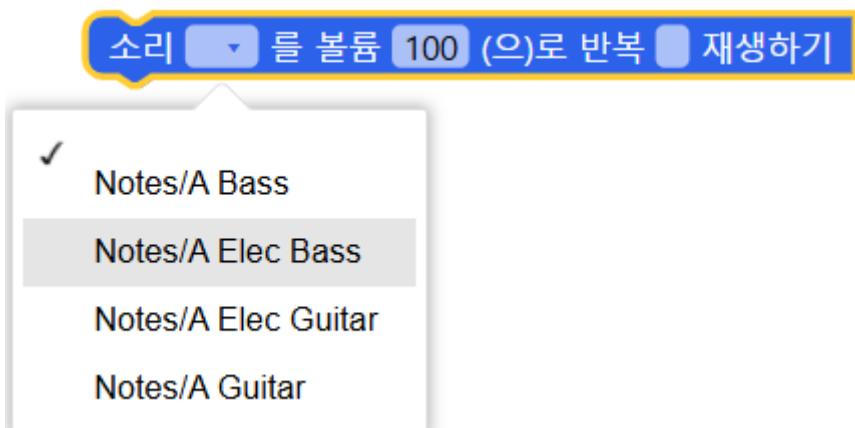


Figure 98: image

## 언어 음성 정하기

소리를 재생할 언어와 음성을 설정하는 블록입니다.

다양한 언어 및 목소리를 선택하여 더욱 자연스러운 음성을 출력할 수 있습니다.

언어 en-US (English) ▾ 음성 Microsoft Mark - English (United States) ▾ 로 정하기

Figure 99: image

## Javascript 코드

```
--setTTSOption('en-US', 'Microsoft Mark - English (United States)'); // 영어, Microsoft Mark 음성 사용하기
```

## Python 코드

```
--setTTSOption('en-US', 'Microsoft Mark - English (United States)') # 영어, Microsoft Mark 음성 사용하기
```

- 언어 선택: 원하는 언어를 선택할 수 있습니다.
- 목소리 선택: 다양한 목소리 중에서 원하는 음성을 선택할 수 있습니다.

언어 en-US (English) ▾ 음성 Microsoft Mark - English (United States) ▾ 로 정하기

✓ en-US (English)  
ko-KR (한국어)

Figure 100: image

언어 ko-KR (한국어) ▾ 음성 Microsoft Heami - Korean (Korean) ▾ 로 정하기

✓ Microsoft Heami - Korean (Korean)

Google 한국의

Figure 101: image

## Javascript 코드

```
--setTTSOption('ko-KR', 'Microsoft Heami - Korean (Korean)'); // 한국어, Heami 음성 사용하기
```

## Python 코드

```
--setTTSOption('ko-KR', 'Microsoft Heami - Korean (Korean)') # 한국어, Heami 음성 사용하기
```

## 다음을 말하기

입력된 텍스트를 소리로 변환하여 말하는 블록입니다.

이 블록을 활용하면 프로젝트에서 원하는 문장을 음성으로 출력할 수 있습니다.



Figure 102: image

- 텍스트 입력: 원하는 문장을 입력하면 해당 문장을 음성으로 변환하여 재생합니다.

## Javascript 코드

```
--speak(''); // '' 문장 음성 출력
```

## Python 코드

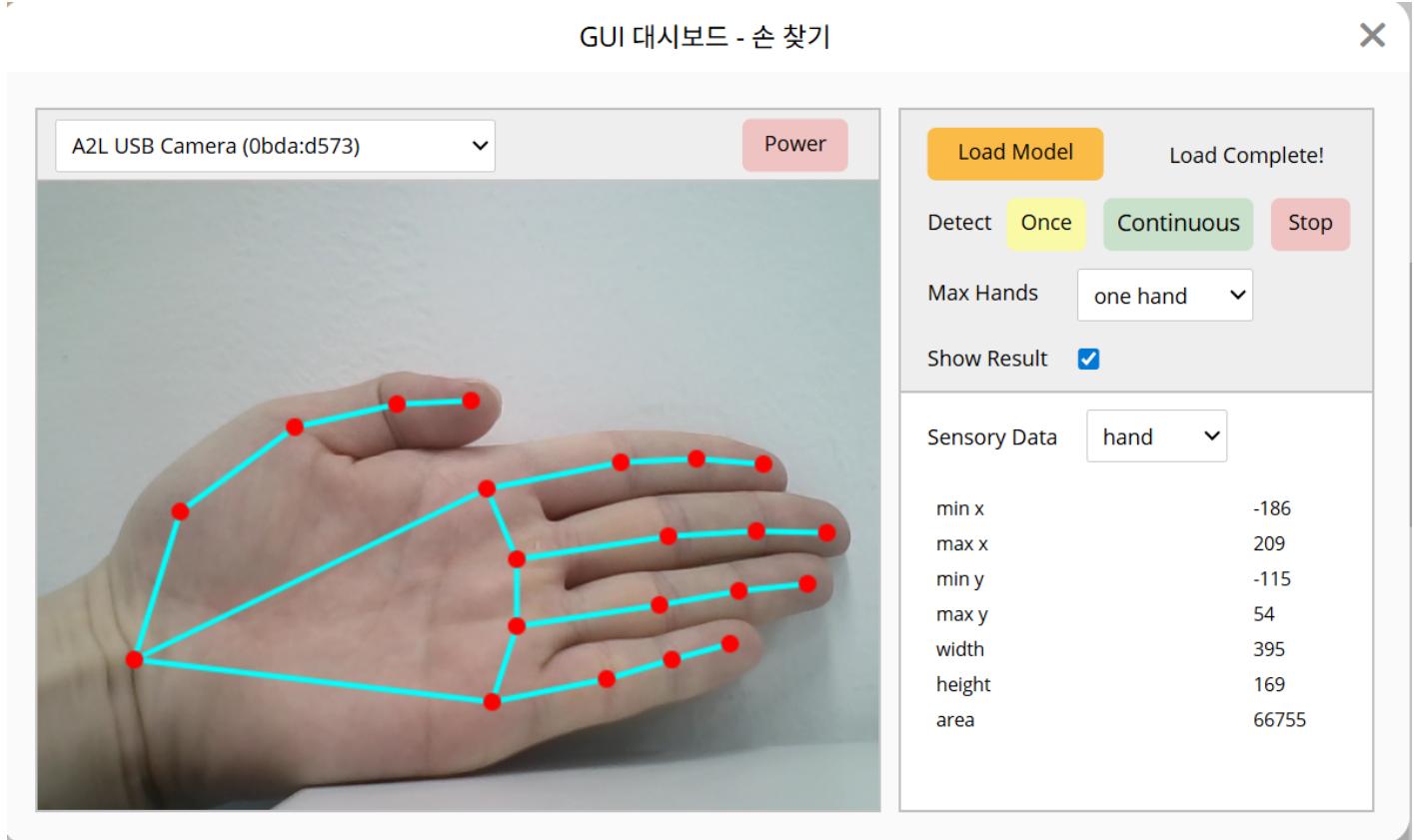
```
--speak('') # '' 문장 음성 출력
```

## 손-찾기

## 대시보드 열기

대시보드 열기는 설치할 수 있는 블록은 아니지만, 확장 모듈에서 사용되는 모델이 어떠한 식으로 적용되는지 알 수 있는 대시보드를 열 수 있습니다. ## 대시보드 화면 대시보드 열기 클릭 시 다음과 같은 화면을 볼 수 있습니다.

## GUI 대시보드 - 손 찾기



### 세부 버튼

#### Power

선택한 카메라를 키거나 끕니다.

#### Load Model

학습된 손 모델을 불러옵니다. ‘손 찾기’ 확장 모듈을 사용하기 위해서 반드시 필요한 작업입니다.

#### Detect

손 찾기를 실행하거나 멈춥니다.

Once 버튼으로 한 번만 실행할지, Continuous 버튼으로 연속으로 실행할지 정할 수 있습니다.

또한, Stop 버튼을 통해 찾기를 멈출 수 있습니다.

#### Show Result

손 찾기 결과를 카메라 화면 상으로 출력합니다.

## Max Hands

감지할 손의 최대 개수를 선택합니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
hand	드롭다운 옵션	감지 개수	one hand, both hands

## Sensory Data

선택한 손 부위에 따른 데이터 값을 출력합니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
parts	드롭다운 옵션	손 부위	hand, palm, wrist, thumbs, index, middle, ring, pinky

## 블럭

### 카메라 정하기

손 찾기 모듈에 사용할 카메라를 선택합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
camera	드롭다운 옵션	사용할 카메라	연결한 카메라 리스트

## 자바스크립트 코드

```
// 특정 카메라를 손 찾기를 위한 카메라로 정하기 (id는 예시)
$('HandDetection*0:camera.deviceId').d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb';
```

## 파이썬 코드

```
# 특정 카메라를 손 찾기를 위한 카메라로 정하기 (id는 예시)
__('HandDetection*0:camera.deviceId').d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb'
```

## 손 모델 불러오기

학습된 손 모델을 불러옵니다. ‘손 찾기’모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

기다리기를 체크하면, 모델 불러오기가 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



손 찾기 0 : 손 모델 불러오기 | 기다리기 ✓

학습된 손 모델을 불러옵니다. ‘손 찾기’ 모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

## 자바스크립트 코드

```
// 손 모델 불러오기 | 기다리기 0
$('HandDetection*0:load_model').d = 1;
await $('HandDetection*0:!load_model').w();

// 손 모델 불러오기 | 기다리기 X
$('HandDetection*0:load_model').d = 1;
```

## 파이썬 코드

```
# 손 모델 불러오기 | 기다리기 0
__('HandDetection*0:load_model').d = 1
await __('HandDetection*0:!load_model').w()

# 손 모델 불러오기 | 기다리기 X
__('HandDetection*0:load_model').d = 1
```

## 찾는 대상 기준 정하기

손을 찾을 때, 한 손을 기준으로 찾을지, 양손을 기준으로 찾을지 결정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pivot	드롭다운 옵션	찾는 기준	한 손 (1), 양손 (2)

## 자바스크립트 코드

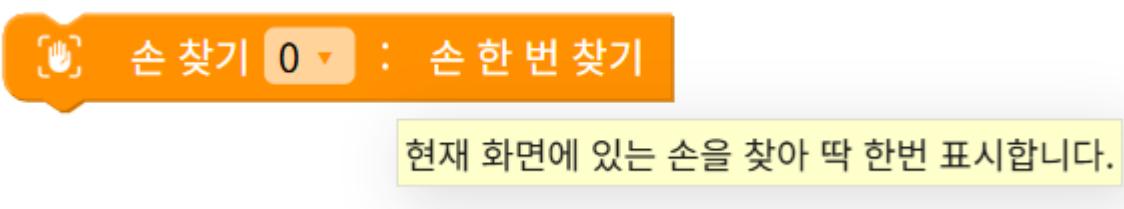
```
// 찾는 대상을 한 손 (으) 로 정하기  
$('HandDetection*0:maxHands').d = 1;  
  
// 찾는 대상을 양손 (으) 로 정하기  
$('HandDetection*0:maxHands').d = 2;
```

## 파이썬 코드

```
# 찾는 대상을 한 손 (으) 로 정하기  
__('HandDetection*0:maxHands').d = 1  
  
# 찾는 대상을 양손 (으) 로 정하기  
__('HandDetection*0:maxHands').d = 2
```

## 손 한 번 찾기

현재 화면에 있는 손을 찾아 딱 한 번 표시합니다.



## 자바스크립트 코드

```
// 손 한 번 찾기  
$('HandDetection*0:detect.once').d = 1;
```

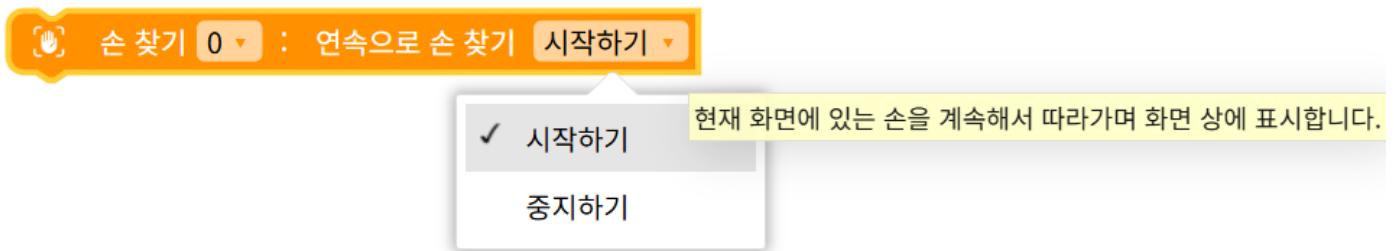
## 파이썬 코드

```
# 손 한 번 찾기  
__('HandDetection*0:detect.once').d = 1
```

## 손 연속으로 찾기

손 연속으로 찾기를 시작하거나 중지합니다.

손 연속으로 찾기를 시작하면, 현재 화면에 있는 손을 계속 따라가며 화면상에 표시합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	손 찾기	시작하기 (1), 중지하기 (0)

## 자바스크립트 코드

```
// 연속으로 손 찾기 시작하기  
$('HandDetection*0:detect.continuous').d = 1;  
  
// 연속으로 손 찾기 중지하기  
$('HandDetection*0:detect.continuous').d = 0;
```

## 파이썬 코드

```
# 연속으로 손 찾기 시작하기  
__('HandDetection*0:detect.continuous').d = 1  
  
# 연속으로 손 찾기 중지하기  
__('HandDetection*0:detect.continuous').d = 0
```

## 손 찾기 결과 보이기

카메라 화면에 손 찾기 결과를 표시할지 말지를 결정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	손 찾기 결과 보이기 (1), 숨기기 (0)	

## 자바스크립트 코드

```
// 손 찾기 결과 보이기
$('.HandDetection*0:display').d = 1;

// 손 찾기 결과 숨기기
$('.HandDetection*0:display').d = 0;
```

## 파이썬 코드

```
# 손 찾기 결과 보이기
__('HandDetection*0:display').d = 1

# 손 찾기 결과 숨기기
__('HandDetection*0:display').d = 0
```

## 손 관련 데이터

선택한 손 부위의 데이터를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	손 방향	왼쪽 (left), 오른쪽 (right)
part	드롭다운 옵션	손 부위	손바닥 (palm), 손목 (wrist)
axis	드롭다운 옵션	좌표 방향	x, y

## 자바스크립트 코드

```
// 오른쪽 손바닥 x 좌표
$('HandDetection*0:right.palm.x').d;

// 오른쪽 손바닥 y 좌표
$('HandDetection*0:right.palm.y').d;

// 오른쪽 손목 x 좌표
$('HandDetection*0:right.wrist.x').d;

// 오른쪽 손목 y 좌표
$('HandDetection*0:right.wrist.y').d;

// 왼쪽 손바닥 x 좌표
$('HandDetection*0:left.palm.x').d;

// 왼쪽 손바닥 y 좌표
$('HandDetection*0:left.palm.y').d;

// 왼쪽 손목 x 좌표
$('HandDetection*0:left.wrist.x').d;

// 왼쪽 손목 y 좌표
$('HandDetection*0:left.wrist.y').d;
```

## 파이썬 코드

```
# 오른쪽 손바닥 x 좌표
__($('HandDetection*0:right.palm.x').d

# 오른쪽 손바닥 y 좌표
__($('HandDetection*0:right.palm.y').d

# 오른쪽 손목 x 좌표
__($('HandDetection*0:right.wrist.x').d
```

```

# 오른쪽 손목 y 좌표
__('HandDetection*0:right.wrist.y').d

# 왼쪽 손바닥 x 좌표
__('HandDetection*0:left.palm.x').d

# 왼쪽 손바닥 y 좌표
__('HandDetection*0:left.palm.y').d

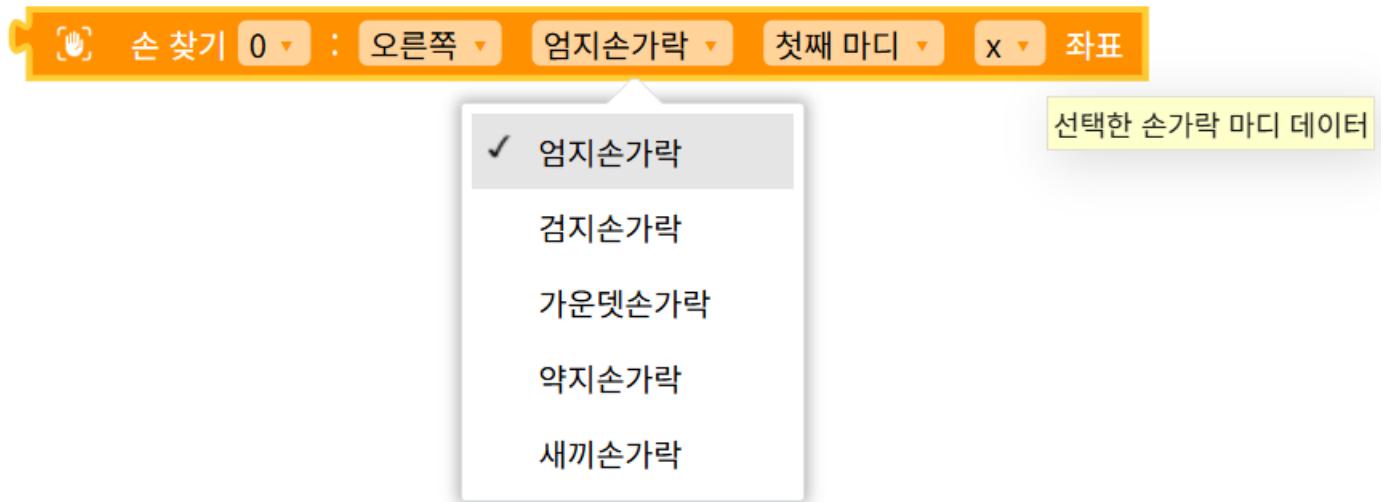
# 왼쪽 손목 x 좌표
__('HandDetection*0:left.wrist.x').d

# 왼쪽 손목 y 좌표
__('HandDetection*0:left.wrist.y').d

```

## 손가락 관련 데이터

선택한 손가락 마디 데이터를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	손 방향	왼쪽 (left), 오른쪽 (right)
finger	드롭다운 옵션	손가락 부위	엄지손가락 (thumb), 검지손가락 (index), 가운데손가락 (middle), 약지손가락 (ring), 새끼손가락 (pinky)
joint	드롭다운 옵션	손가락 마디	첫째 마디 (first), 둘째 마디 (second), 셋째 마디 (third), 끝 (last)
axis	드롭다운 옵션	좌표 방향	x, y

## 자바스크립트 코드

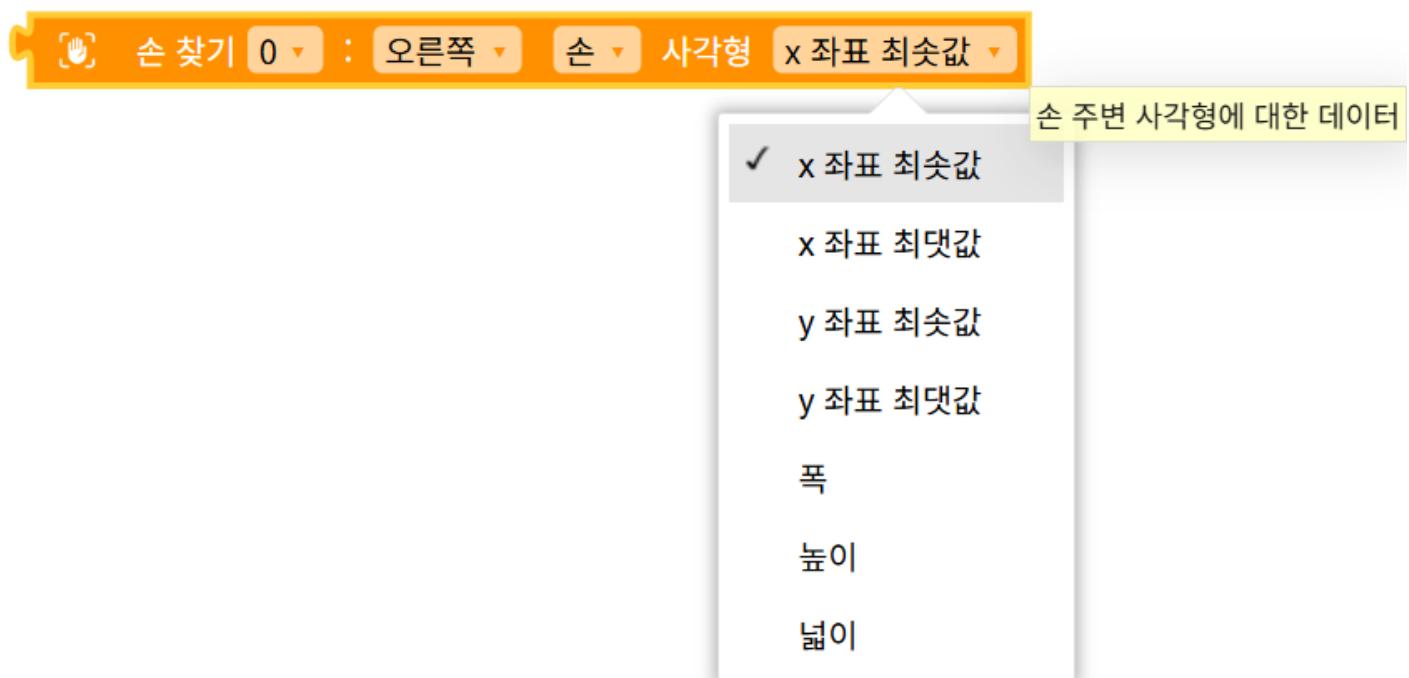
```
// 오른쪽 엄지손가락 첫째 마디 x 좌표  
$('HandDetection*0:right.thumb.first.x').d;  
  
// 오른쪽 검지손가락 둘째 마디 x 좌표  
$('HandDetection*0:right.index.second.x').d;  
  
// 오른쪽 가운뎃손가락 셋째 마디 x 좌표  
$('HandDetection*0:right.middle.third.x').d;  
  
// 오른쪽 약지손가락 첫째 마디 y 좌표  
$('HandDetection*0:right.ring.first.y').d;  
  
// 오른쪽 새끼손가락 첫째 마디 y 좌표  
$('HandDetection*0:right.pinky.first.y').d;
```

## 파이썬 코드

```
# 오른쪽 엄지손가락 첫째 마디 x 좌표  
__('HandDetection*0:right.thumb.first.x').d  
  
# 오른쪽 검지손가락 둘째 마디 x 좌표  
__('HandDetection*0:right.index.second.x').d  
  
# 오른쪽 가운뎃손가락 셋째 마디 x 좌표  
__('HandDetection*0:right.middle.third.x').d  
  
# 오른쪽 약지손가락 첫째 마디 y 좌표  
__('HandDetection*0:right.ring.first.y').d  
  
# 오른쪽 새끼손가락 첫째 마디 y 좌표  
__('HandDetection*0:right.pinky.first.y').d
```

## 손 주변 사각형 데이터

손 찾기로 찾은 손의 주변을 사각형으로 정의하여, 그 사각형의 데이터를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	손 방향	왼쪽 (left), 오른쪽 (right)
part	드롭다운 옵션	손 부위	손 (hand), 손바닥 (palm)
axis	드롭다운 옵션	좌표 방향	x 좌표 최솟값 (min_x), x 좌표 최댓값 (max_x), y 좌 표 최솟값 (min_y), y 좌표 최댓값 (max_y), 폭 (width), 높이 (height), 넓 이 (area)

## 자바스크립트 코드

```
// 오른쪽 손 사각형 x 좌표 최솟값
$('HandDetection*0:right.hand.min_x').d;

// 오른쪽 손 사각형 x 좌표 최댓값
$('HandDetection*0:right.hand.max_x').d;

// 오른쪽 손 사각형 y 좌표 최솟값
$('HandDetection*0:right.hand.min_y').d;

// 오른쪽 손 사각형 y 좌표 최댓값
$('HandDetection*0:right.hand.max_y').d;

// 오른쪽 손 사각형 폭
$('HandDetection*0:right.hand.width').d;

// 오른쪽 손 사각형 높이
$('HandDetection*0:right.hand.height').d;

// 오른쪽 손 사각형 넓이
$('HandDetection*0:right.hand.area').d;

// 왼쪽 손 바닥 사각형 넓이
$('HandDetection*0:left.palm.area').d;
```

## 파이썬 코드

```
# 오른쪽 손 사각형 x 좌표 최솟값
__('HandDetection*0:right.hand.min_x').d

# 오른쪽 손 사각형 x 좌표 최댓값
__('HandDetection*0:right.hand.max_x').d

# 오른쪽 손 사각형 y 좌표 최솟값
__('HandDetection*0:right.hand.min_y').d

# 오른쪽 손 사각형 y 좌표 최댓값
__('HandDetection*0:right.hand.max_y').d

# 오른쪽 손 사각형 폭
__('HandDetection*0:right.hand.width').d

# 오른쪽 손 사각형 높이
```

```

__('HandDetection*0:right.hand.height').d
# 오른쪽 손 사각형 넓이
__('HandDetection*0:right.hand.area').d

# 왼쪽 손바닥 사각형 넓이
__('HandDetection*0:left.palm.area').d

```

## 두 손 사이의 거리

두 손 사이의 거리를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction 1	드롭다운 옵션	손 방향	오른쪽 (right), 왼쪽 (left)
part 1	드롭다운 옵션	손 부위	손 (hand), 손바닥 (palm)
direction 2	드롭다운 옵션	손 방향	오른쪽 (right), 왼쪽 (left)
part 2	드롭다운 옵션	손 부위	손 (hand), 손바닥 (palm)
distance	드롭다운 옵션	거리	거리 (distance), 가로거리 (horizontal distance), 세 로거리 (vertical distance)

## 자바스크립트 코드

```

// 오른쪽 손에서 왼쪽 손 까지 거리
Math.sqrt( Math.pow((($('HandDetection*0:left.hand.x').d - $('HandDetection*0:right.hand.x').d), 2) + Math.pow((($('HandDetection*0:left.hand.y').d - $('HandDetection*0:right.hand.y').d), 2) );

// 오른쪽 손바닥에서 왼쪽 손바닥 까지 가로거리
Math.abs($('HandDetection*0:left.palm.x').d - $('HandDetection*0:right.palm.x').d);

// 오른쪽 손에서 왼쪽 손바닥 까지 세로거리
Math.abs($('HandDetection*0:left.palm.y').d - $('HandDetection*0:right.hand.y').d);

```

## 파이썬 코드

```
# 오른쪽 손에서 왼쪽 손 까지 거리
math.sqrt( math.pow(_('_('HandDetection*0:left.hand.x').d - _('_('HandDetection*0:right.hand.x').d), 2) + math.pow(_('_('HandDetection*0:left.hand.y').d - _('_('HandDetection*0:right.hand.y').d), 2) )

# 오른쪽 손바닥에서 왼쪽 손바닥 까지 가로거리
math.fabs(_('_('HandDetection*0:left.palm.x').d - _('_('HandDetection*0:right.palm.x').d)

# 오른쪽 손에서 왼쪽 손바닥 까지 세로거리
math.fabs(_('_('HandDetection*0:left.palm.y').d - _('_('HandDetection*0:right.hand.y').d)
```

## 손과 손가락 마디 사이 거리

손과 손가락 마디 사이의 거리를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction 1	드롭다운 옵션	손 방향	오른쪽 (right), 왼쪽 (left)
part 1	드롭다운 옵션	손 부위	손 (hand), 손바닥 (palm)
direction 2	드롭다운 옵션	손 방향	오른쪽 (right), 왼쪽 (left)
finger 2	드롭다운 옵션	손가락 부위	엄지손가락 (thumb), 검지손가락 (index), 가운뎃손가락 (middle), 약지손가락 (ring), 새끼손가락 (pinky)
joint2	드롭다운 옵션	손가락 마디	첫째 마디 (first), 둘째 마디 (second), 셋째 마디 (third), 끝 (last)
distance	드롭다운 옵션	거리	거리 (distance), 가로거리 (horizontal distance), 세로거리 (vertical distance)

## 자바스크립트 코드

```
// 오른쪽 손바닥에서 오른쪽 엄지손가락 첫째 마디 까지 거리
Math.sqrt( Math.pow($('HandDetection*0:right.thumb.first.x').d - $('HandDetection*0:right.palm.x').d, 2) + Math.pow($('HandDetection*0:right.thumb.first.y').d - $('HandDetection*0:right.palm.y').d, 2) );

// 오른쪽 손바닥에서 오른쪽 검지손가락 둘째 마디 까지 가로 거리
Math.abs($('HandDetection*0:right.index.second.x').d - $('HandDetection*0:right.palm.x').d);

// 오른쪽 손바닥에서 오른쪽 가운데손가락 세째 마디 까지 세로 거리
Math.abs($('HandDetection*0:right.middle.third.y').d - $('HandDetection*0:right.palm.y').d);

// 오른쪽 손바닥에서 오른쪽 약지손가락 끝 까지 거리
Math.sqrt( Math.pow($('HandDetection*0:right.ring.last.x').d - $('HandDetection*0:right.palm.x').d, 2) + Math.pow($('HandDetection*0:right.ring.last.y').d - $('HandDetection*0:right.palm.y').d, 2) );

// 오른쪽 손바닥에서 오른쪽 새끼손가락 끝 까지 거리
Math.sqrt( Math.pow($('HandDetection*0:right.pinky.last.x').d - $('HandDetection*0:right.palm.x').d, 2) + Math.pow($('HandDetection*0:right.pinky.last.y').d - $('HandDetection*0:right.palm.y').d, 2) );
```

## 파이썬 코드

```
# 오른쪽 손바닥에서 오른쪽 엄지손가락 첫째 마디 까지 거리
math.sqrt( math.pow(___('HandDetection*0:right.thumb.first.x').d - ___('HandDetection*0:right.palm.x').d, 2) + math.pow(___('HandDetection*0:right.thumb.first.y').d - ___('HandDetection*0:right.palm.y').d, 2) )

# 오른쪽 손바닥에서 오른쪽 검지손가락 둘째 마디 까지 가로 거리
math.fabs(___('HandDetection*0:right.index.second.x').d - ___('HandDetection*0:right.palm.x').d)

# 오른쪽 손바닥에서 오른쪽 가운데손가락 세째 마디 까지 세로 거리
math.fabs(___('HandDetection*0:right.middle.third.y').d - ___('HandDetection*0:right.palm.y').d)

# 오른쪽 손바닥에서 오른쪽 약지손가락 끝 까지 거리
math.sqrt( math.pow(___('HandDetection*0:right.ring.last.x').d - ___('HandDetection*0:right.palm.x').d, 2) + math.pow(___('HandDetection*0:right.ring.last.y').d - ___('HandDetection*0:right.palm.y').d, 2) )

# 오른쪽 손바닥에서 오른쪽 새끼손가락 끝 까지 거리
math.sqrt( math.pow(___('HandDetection*0:right.pinky.last.x').d - ___('HandDetection*0:right.palm.x').d, 2) + math.pow(___('HandDetection*0:right.pinky.last.y').d - ___('HandDetection*0:right.palm.y').d, 2) )
```

## 두 손가락 마디 사이의 거리

두 손가락 마디 사이의 거리를 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction 1	드롭다운 옵션	손 방향	오른쪽 (right), 왼쪽 (left)
finger 1	드롭다운 옵션	손가락 부위	엄지손가락 (thumb), 검지손가락 (index), 가운데손가락 (middle), 약지손가락 (ring), 새끼손가락 (pinky)

이름	구분	설명	범위 / 종류
joint 1	드롭다운 옵션	손가락 마디	첫째 마디 (first), 둘째 마디 (second), 셋째 마디 (third), 끝 (last)
direction 2	드롭다운 옵션	손 방향	오른쪽 (right), 왼쪽 (left)
finger 2	드롭다운 옵션	손가락 부위	엄지손가락 (thumb), 검지손가락 (index), 가운데손가락 (middle), 약지손가락 (ring), 새끼손가락 (pinky)
joint 2	드롭다운 옵션	손가락 마디	첫째 마디 (first), 둘째 마디 (second), 셋째 마디 (third), 끝 (last)
distance	드롭다운 옵션	거리	거리 (distance), 가로거리 (horizontal distance), 세로거리 (vertical distance)

## 자바스크립트 코드

```
// 오른쪽 검지손가락 첫째 마디에서 왼쪽 엄지손가락 첫째 마디 까지 거리
Math.sqrt( Math.pow($('HandDetection*0:left.thumb.first.x').d - $('HandDetection*0:right.thumb.first.x').d, 2) + Math.pow($('HandDetection*0:left.thumb.first.y').d - $('HandDetection*0:right.thumb.first.y').d, 2) );

// 오른쪽 엄지손가락 첫째 마디에서 왼쪽 검지손가락 둘째 마디 까지 거리
Math.sqrt( Math.pow($('HandDetection*0:left.index.second.x').d - $('HandDetection*0:right.thumb.first.x').d, 2) + Math.pow($('HandDetection*0:left.index.second.y').d - $('HandDetection*0:right.thumb.first.y').d, 2) );

// 오른쪽 엄지손가락 첫째 마디에서 왼쪽 가운데손가락 셋째 마디 까지 거리
Math.sqrt( Math.pow($('HandDetection*0:left.middle.third.x').d - $('HandDetection*0:right.thumb.first.x').d, 2) + Math.pow($('HandDetection*0:left.middle.third.y').d - $('HandDetection*0:right.thumb.first.y').d, 2) );

// 오른쪽 엄지손가락 첫째 마디에서 왼쪽 약지손가락 둘째 마디 까지 가로 거리
Math.abs($('HandDetection*0:left.ring.second.x').d - $('HandDetection*0:right.thumb.first.x').d);

// 오른쪽 엄지손가락 첫째 마디에서 왼쪽 새끼손가락 둘째 마디 까지 세로 거리
Math.abs($('HandDetection*0:left.pinky.second.y').d - $('HandDetection*0:right.thumb.first.y').d);
```

## 파이썬 코드

```
# 오른쪽 검지손가락 첫째 마디에서 왼쪽 엄지손가락 첫째 마디 까지 거리
math.sqrt( math.pow(_('_('HandDetection*0:left.thumb.first.x').d - _('_('HandDetection*0:right.thumb.first.x').d, 2) + math.pow(_('_('HandDetection*0:left.thumb.first.y').d - _('_('HandDetection*0:right.thumb.first.y').d, 2) )

# 오른쪽 엄지손가락 첫째 마디에서 왼쪽 검지손가락 둘째 마디 까지 거리
math.sqrt( math.pow(_('_('HandDetection*0:left.index.second.x').d - _('_('HandDetection*0:right.thumb.first.x').d, 2) + math.pow(_('_('HandDetection*0:left.index.second.y').d - _('_('HandDetection*0:right.thumb.first.y').d, 2) )

# 오른쪽 엄지손가락 첫째 마디에서 왼쪽 가운데손가락 셋째 마디 까지 거리
math.sqrt( math.pow(_('_('HandDetection*0:left.middle.third.x').d - _('_('HandDetection*0:right.thumb.first.x').d, 2) + math.pow(_('_('HandDetection*0:left.middle.third.y').d - _('_('HandDetection*0:right.thumb.first.y').d, 2) )
```

```
# 오른쪽 엄지손가락 첫째 마디에서 원쪽 약지손가락 둘째 마디 까지 가로 거리  
math.abs(__('HandDetection*0:left.ring.second.x').d - __('HandDetection*0:right.thumb.first.x').d)  
  
# 오른쪽 엄지손가락 첫째 마디에서 원쪽 새끼손가락 둘째 마디 까지 세로 거리  
math.abs(__('HandDetection*0:left.pinky.second.y').d - __('HandDetection*0:right.thumb.first.y').d)
```

## 손 모델 로딩 상태값

손 모델 로딩 상태를 반환합니다.

아직 불러오지 않았다면 0, 불러오는 중이면 1, 불러오기를 완료했다면 2를 반환합니다.



손 모델 로딩 상태를 반환합니다.  
아직 불러오지 않았으면 0, 불러오는 중이면 1, 불러오기를 완료했으면 2를 반환합니다.

## 자바스크립트 코드

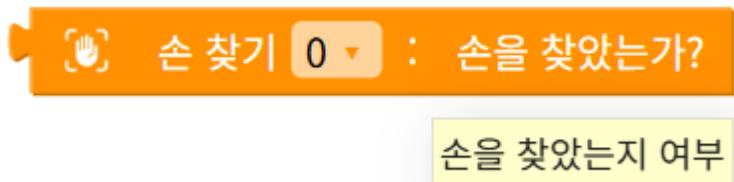
```
// 손 모델 로딩 상태 값  
$('HandDetection*0:model_state').d;
```

## 파이썬 코드

```
# 손 모델 로딩 상태 값  
__('HandDetection*0:model_state').d
```

## 손을 찾았는가?

손 찾기 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 자바스크립트 코드

```
// 손을 찾았는가?  
$('HandDetection*0:detected').d;
```

## 파이썬 코드

```
# 손을 찾았는가?  
_<('HandDetection*0:detected').d
```

## 얼굴-찾기

## 대시보드 열기

대시보드 열기는 설치할 수 있는 블록은 아니지만, 확장 모듈에서 사용되는 모델이 어떠한 식으로 적용되는지 알 수 있는 대시보드를 열 수 있습니다. ## 대시보드 화면 대시보드 열기 클릭 시 다음과 같은 화면을 볼 수 있습니다.



## 세부 버튼

### Power

선택한 카메라를 키거나 끕니다.

## Load Model

학습된 얼굴 모델을 불러옵니다. ‘얼굴 찾기’ 확장 모듈을 사용하기 위해서 반드시 필요한 작업입니다.

## Detect

얼굴 찾기를 실행하거나 멈춥니다.

Once 버튼으로 한번만 실행할지, Continuous 버튼으로 연속으로 실행할지 정할 수 있습니다.

또한, Stop 버튼을 통해 찾기를 멈출 수 있습니다.

## Show Result

얼굴 찾기 결과를 카메라 화면 상으로 출력합니다.

## Sensory Data

얼굴 찾기에서 찾은 얼굴 데이터 값을 출력합니다.

선택한 얼굴 부위의 x,y 좌표를 확인할 수 있습니다. ##### 드롭다운 옵션 및 입력값 | 이름 | 구분 | 설명 | 범위 / 종류 |  
| — | — | — | — | part | 드롭다운 옵션 | 얼굴 부위 | face, ear, eye, nose, mouth |

## 블록

### 카메라 정하기

얼굴 찾기 모듈에 사용할 카메라를 선택합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
camera	드롭다운 옵션	사용할 카메라	연결한 카메라 리스트

## 자바스크립트 코드

```
// 특정 카메라를 얼굴 찾기를 위한 카메라로 정하기 (id 는 예시)
$(`FaceDetection*0:camera.deviceId`).d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb';
```

## 파이썬 코드

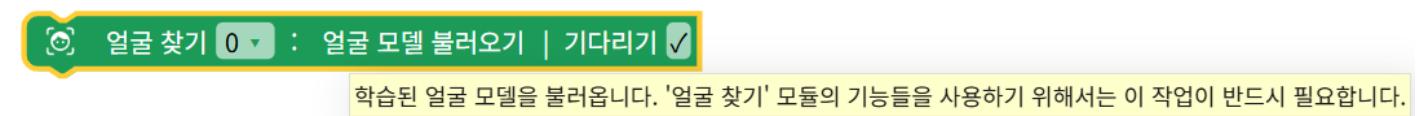
```
# 특정 카메라를 얼굴 찾기를 위한 카메라로 정하기 (id 는 예시)
__(`FaceDetection*0:camera.deviceId`).d = '035658da47183882a695a82c45b8f3e9ae50cef47945ccdc3f31e1ae1fbca9cb'
```

## 얼굴 모델 불러오기

학습된 얼굴 모델을 불러옵니다. ‘얼굴 찾기’모듈의 기능들을 사용하기 위해서는 이 작업이 반드시 필요합니다.

기다리기를 체크하면, 모델 불러오기가 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 자바스크립트 코드

```
// 얼굴 모델 불러오기 | 기다리기 0
$(`FaceDetection*0:load_model`).d = 1;
await $(`FaceDetection*0:!load_model`).w();

// 얼굴 모델 불러오기 | 기다리기 X
$(`FaceDetection*0:load_model`).d = 1;
```

## 파이썬 코드

```
# 얼굴 모델 불러오기 | 기다리기 0
__(`FaceDetection*0:load_model`).d = 1
await __(`FaceDetection*0:!load_model`).w()

# 얼굴 모델 불러오기 | 기다리기 X
__(`FaceDetection*0:load_model`).d = 1
```

## 얼굴 한 번 찾기

현재 화면에 있는 얼굴을 찾아 딱 한번 표시합니다.



얼굴 찾기 0 : 얼굴 한 번 찾기

현재 화면에 있는 얼굴을 찾아 딱 한번 표시합니다.

## 자바스크립트 코드

```
// 얼굴 한 번 찾기
$('FaceDetection*0:detect.once').d = 1;
```

## 파이썬 코드

```
# 얼굴 한 번 찾기
__('FaceDetection*0:detect.once').d = 1
```

## 얼굴 연속으로 찾기

얼굴 연속으로 찾기를 시작하거나 중지합니다.

얼굴 연속으로 찾기를 시작하면, 현재 화면에 있는 얼굴을 계속 따라가며 화면상에 표시합니다.



얼굴 찾기 0 : 연속으로 얼굴 찾기

현재 화면에 있는 얼굴을 계속해서 따라가며 화면 상에 표시합니다.

<input checked="" type="checkbox"/> 시작하기
중지하기

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	얼굴 찾기	시작하기 (1), 중지하기 (0)

## 자바스크립트 코드

```
// 연속으로 얼굴 찾기 시작하기
$('FaceDetection*0:detect.continuous').d = 1;

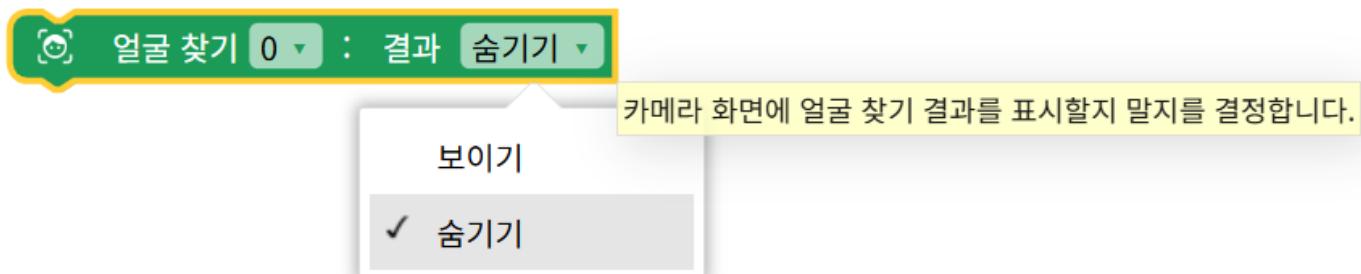
// 연속으로 얼굴 찾기 중지하기
$('FaceDetection*0:detect.continuous').d = 0;
```

## 파이썬 코드

```
# 연속으로 얼굴 찾기 시작하기  
_('_FaceDetection*0:detect_continuous').d = 1  
  
# 연속으로 얼굴 찾기 중지하기  
_('_FaceDetection*0:detect_continuous').d = 0
```

## 얼굴 찾기 결과 보이기

카메라 화면에 얼굴 찾기 결과를 표시할지 말지를 결정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	얼굴 찾기 결과	보이기 (1), 숨기기 (0)

## 자바스크립트 코드

```
// 얼굴 찾기 결과 보이기  
$('FaceDetection*0:display').d = 1;  
  
// 얼굴 찾기 결과 숨기기  
$('FaceDetection*0:display').d = 0;
```

## 파이썬 코드

```
# 얼굴 찾기 결과 보이기  
_('_FaceDetection*0:display').d = 1  
  
# 얼굴 찾기 결과 숨기기  
_('_FaceDetection*0:display').d = 0
```

## 얼굴 데이터

얼굴 찾기로 얻은 얼굴 데이터를 반환합니다.



이름	구분	설명	범위 / 종류
part	드롭다운 옵션	얼굴 부위	얼굴 (face), 왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 귀 (ear.left), 오른쪽 귀 (ear.right), 코 (nose), 입 (mouth)
axis	드롭다운 옵션	좌표	x, y

## 자바스크립트 코드

```
// 얼굴의 x 좌표
$('FaceDetection*0:face.x').d;

// 얼굴의 y 좌표
$('FaceDetection*0:face.y').d;

// 왼쪽 눈의 x 좌표
$('FaceDetection*0:eye.left.x').d;

// 왼쪽 눈의 y 좌표
$('FaceDetection*0:eye.left.y').d;

// 오른쪽 눈의 x 좌표
$('FaceDetection*0:eye.right.x').d;
```

```

// 오른쪽 눈의 x 좌표
$('FaceDetection*0:eye.right.x').d;

// 왼쪽 귀의 x 좌표
$('FaceDetection*0:ear.left.x').d;

// 왼쪽 귀의 y 좌표
$('FaceDetection*0:ear.left.y').d;

// 오른쪽 귀의 x 좌표
$('FaceDetection*0:ear.right.x').d;

// 오른쪽 귀의 y 좌표
$('FaceDetection*0:ear.right.y').d;

// 코의 x 좌표
$('FaceDetection*0:nose.x').d;

// 코의 y 좌표
$('FaceDetection*0:nose.y').d;

// 입의 x 좌표
$('FaceDetection*0:mouth.x').d;

// 입의 y 좌표
$('FaceDetection*0:mouth.y').d;

```

## 파이썬 코드

```

# 얼굴의 x 좌표
__($('FaceDetection*0:face.x')).d

# 얼굴의 y 좌표
__($('FaceDetection*0:face.y')).d

# 왼쪽 눈의 x 좌표
__($('FaceDetection*0:eye.left.x')).d

# 왼쪽 눈의 y 좌표
__($('FaceDetection*0:eye.left.y')).d

# 오른쪽 눈의 x 좌표
__($('FaceDetection*0:eye.right.x')).d

# 오른쪽 눈의 y 좌표
__($('FaceDetection*0:eye.right.y')).d

# 왼쪽 귀의 x 좌표
__($('FaceDetection*0:ear.left.x')).d

# 왼쪽 귀의 y 좌표
__($('FaceDetection*0:ear.left.y')).d

# 오른쪽 귀의 x 좌표
__($('FaceDetection*0:ear.right.x')).d

# 오른쪽 귀의 y 좌표
__($('FaceDetection*0:ear.right.y')).d

# 코의 x 좌표
__($('FaceDetection*0:nose.x')).d

# 코의 y 좌표
__($('FaceDetection*0:nose.y')).d

# 입의 x 좌표
__($('FaceDetection*0:mouth.x')).d

# 입의 y 좌표
__($('FaceDetection*0:mouth.y')).d

```

## 얼굴 주변 사각형 데이터

얼굴 찾기로 찾은 얼굴의 주변을 사각형으로 정의하여, 그 사각형의 데이터를 반환합니다.



이름	구분	설명	범위 / 종류
data	드롭다운 옵션	얼굴 데이터	x 좌표 최솟값 (min_x), x 좌표 최댓값 (max_x), y 좌표 최솟값 (min_y), y 좌표 최댓값 (max_y), 폭 (width), 높이 (height), 넓이 (area)

## 자바스크립트 코드

```
// 얼굴 사각형 x 좌표 최솟값
$('FaceDetection*0:face.min_x').d;

// 얼굴 사각형 x 좌표 최댓값
$('FaceDetection*0:face.max_x').d;

// 얼굴 사각형 y 좌표 최솟값
$('FaceDetection*0:face.min_y').d;

// 얼굴 사각형 y 좌표 최댓값
$('FaceDetection*0:face.max_y').d;

// 얼굴 사각형 폭
```

```

$(‘FaceDetection*0:face.width’).d;
// 얼굴 사각형 높이
$(‘FaceDetection*0:face.height’).d;
// 얼굴 사각형 넓이
$(‘FaceDetection*0:face.area’).d;

```

## 파이썬 코드

```

# 얼굴 사각형 x 좌표 최솟값
___(‘FaceDetection*0:face.min_x’).d

# 얼굴 사각형 x 좌표 최댓값
___(‘FaceDetection*0:face.max_x’).d

# 얼굴 사각형 y 좌표 최솟값
___(‘FaceDetection*0:face.min_y’).d

# 얼굴 사각형 y 좌표 최댓값
___(‘FaceDetection*0:face.max_y’).d

# 얼굴 사각형 폭
___(‘FaceDetection*0:face.width’).d

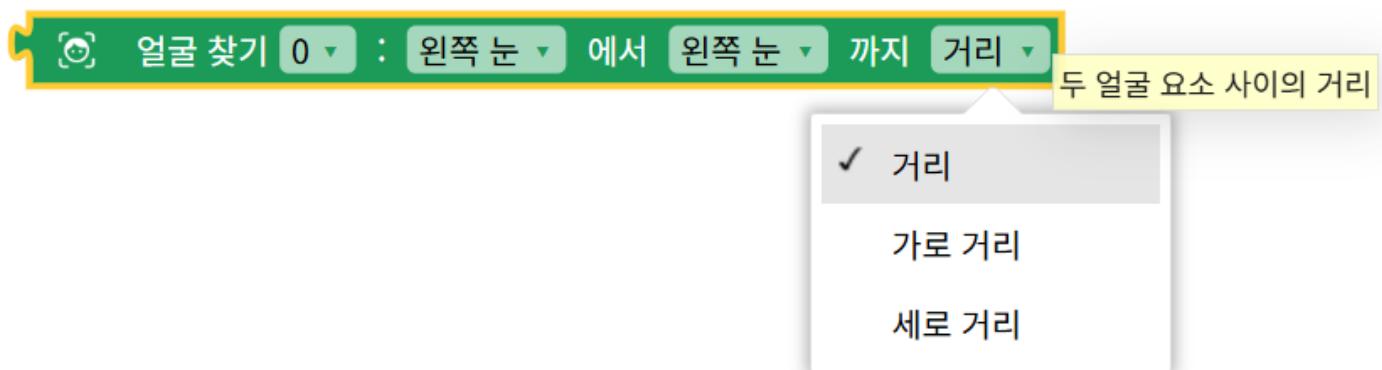
# 얼굴 사각형 높이
___(‘FaceDetection*0:face.height’).d

# 얼굴 사각형 넓이
___(‘FaceDetection*0:face.area’).d

```

## 얼굴 요소 사이 거리

얼굴 찾기로 찾은 얼굴의 데이터를 이용하여 두 얼굴 요소 사이의 거리를 반환합니다.



이름	구분	설명	범위 / 종류
part1	드롭다운 옵션	얼굴 부위	얼굴 (face), 왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 귀 (ear.left), 오른쪽 귀 (ear.right), 코 (nose), 입 (mouth)
part2	드롭다운 옵션	얼굴 부위	얼굴 (face), 왼쪽 눈 (eye.left), 오른쪽 눈 (eye.right), 왼쪽 귀 (ear.left), 오른쪽 귀 (ear.right), 코 (nose), 입 (mouth)
distance	드롭다운 옵션	거리	거리 (distance), 가로거리 (horizontal distance), 세로거리 (vertical distance)

## 자바스크립트 코드

```
// 왼쪽 눈에서 왼쪽 눈 까지 거리
Math.sqrt( Math.pow($('FaceDetection*0:eye.left.x').d - $('FaceDetection*0:eye.left.x').d, 2) + Math.pow($('FaceDetection*0:eye.left.y').d - $('FaceDetection*0:eye.left.y').d, 2) );

// 왼쪽 눈에서 오른쪽 눈 까지 가로거리
Math.abs($('FaceDetection*0:eye.right.x').d - $('FaceDetection*0:eye.left.x').d);

// 왼쪽 눈에서 왼쪽 귀까지 세로거리
Math.abs($('FaceDetection*0:ear.left.y').d - $('FaceDetection*0:eye.left.y').d);

// 왼쪽 눈에서 오른쪽 귀까지 거리
Math.sqrt( Math.pow($('FaceDetection*0:ear.right.x').d - $('FaceDetection*0:eye.left.x').d, 2) + Math.pow($('FaceDetection*0:ear.right.y').d - $('FaceDetection*0:eye.left.y').d, 2) );

// 왼쪽 눈에서 코까지 거리
Math.sqrt( Math.pow($('FaceDetection*0:nose.x').d - $('FaceDetection*0:eye.left.x').d, 2) + Math.pow($('FaceDetection*0:nose.y').d - $('FaceDetection*0:eye.left.y').d, 2) );

// 왼쪽 눈에서 입까지 거리
Math.sqrt( Math.pow($('FaceDetection*0:mouth.x').d - $('FaceDetection*0:eye.left.x').d, 2) + Math.pow($('FaceDetection*0:mouth.y').d - $('FaceDetection*0:eye.left.y').d, 2) );
```

## 파이썬 코드

```
# 왼쪽 눈에서 왼쪽 눈 까지 거리
math.sqrt( math.pow(___('FaceDetection*0:eye.left.x').d - ___('FaceDetection*0:eye.left.x').d, 2) + math.pow(___('FaceDetection*0:eye.left.y').d - ___('FaceDetection*0:eye.left.y').d, 2) )

# 왼쪽 눈에서 오른쪽 눈 까지 가로거리
math.fabs(___('FaceDetection*0:eye.right.x').d - ___('FaceDetection*0:eye.left.x').d)

# 왼쪽 눈에서 왼쪽 귀까지 세로거리
math.fabs(___('FaceDetection*0:ear.left.y').d - ___('FaceDetection*0:eye.left.y').d)
```

```

# 원쪽 눈에서 오른쪽 귀까지 거리
math.sqrt( math.pow(_('_('FaceDetection*0:ear.right.x').d - _('_('FaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('FaceDetection*0:ear.right.y').d - _('_('FaceDetection*0:eye.left.y').d), 2) )

# 원쪽 눈에서 코까지 거리
math.sqrt( math.pow(_('_('FaceDetection*0:nose.x').d - _('_('FaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('FaceDetection*0:nose.y').d - _('_('FaceDetection*0:eye.left.y').d), 2) )

# 원쪽 눈에서 입까지 거리
math.sqrt( math.pow(_('_('FaceDetection*0:mouth.x').d - _('_('FaceDetection*0:eye.left.x').d), 2) + math.pow(_('_('FaceDetection*0:mouth.y').d - _('_('FaceDetection*0:eye.left.y').d), 2) )

```

## 얼굴 모델 로딩 상태값

얼굴 모델 로딩 상태를 반환합니다.

아직 불러오지 않았다면 0, 불러오는 중이면 1, 불러오기를 완료했다면 2를 반환합니다.



얼굴 모델 로딩 상태를 반환합니다.  
아직 불러오지 않았으면 0, 불러오는 중이면 1, 불러오기를 완료했으면 2를 반환합니다.

## 자바스크립트 코드

```
// 얼굴 모델 로딩 상태 값
$('_('FaceDetection*0:model_state').d;
```

## 파이썬 코드

```
# 얼굴 모델 로딩 상태 값
_('_('FaceDetection*0:model_state').d
```

## 얼굴을 찾았는가?

얼굴 찾기 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



얼굴을 찾았는지 여부

## 자바스크립트 코드

```
// 얼굴을 찾았는가?  
$('FaceDetection*0:detected').d;
```

## 파이썬 코드

```
# 얼굴을 찾았는가?  
__($('FaceDetection*0:detected').d
```

## 연산

이 문서는 다양한 연산 블록의 기능과 사용법을 설명합니다. 숫자 연산, 리스트 처리, 확률 및 각도 연산 등 다양한 수학적 연산을 수행하는 블록을 소개합니다.

## 블록

### 숫자 값

입력된 **숫자 값을** 그대로 반환하는 블록입니다. 이 블록을 사용하면 특정 숫자를 변수에 저장하거나 다른 연산에 활용할 수 있습니다.



Figure 103: Image

### Javascript 코드

```
10; // 숫자 값 10
```

### Python 코드

```
10 # 숫자 값 10
```

## 배열 생성 및 연산

**배열을** 생성하는 블록입니다. [] 안에 입력한 값들을 요소로 가지는 배열을 반환합니다.

[] 안에 원하는 값을 입력하여 리스트를 만들 수 있으며, 문자열은”“로 감싸야 합니다.

위의 블록은 [“하나”, “둘”, 3] 을 생성하며, 하나, 둘, 3 을 순서대로 출력합니다.



Figure 104: Image

### Javascript 코드

```
var item;

var item_list = ["하나", "둘", 3]; // 배열 생성
for (var item_index in item_list) {
    item = item_list[item_index];
    window.alert(item);
}
```

### Python 코드

```
item = None

for item in ["하나", "둘", 3]: # 배열 생성
    print(item)
```

### 기본 산술 연산

두 개의 숫자 값을 사용하여 **산술 연산** (덧셈, 뺄셈, 곱셈, 나눗셈, 거듭제곱) 을 수행하는 블록입니다.

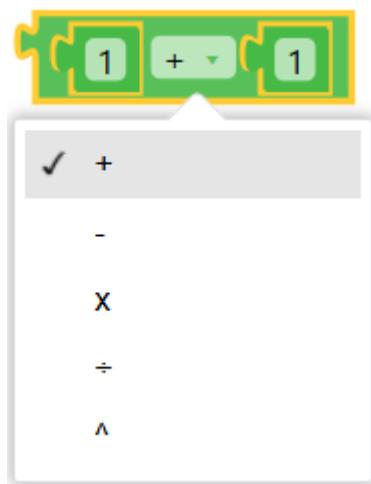


Figure 105: Image

위의 블록은  $1 + 1$  을 계산하여 2를 반환합니다.

## Javascript 코드

```
1 + 1; // 1 + 1 연산  
1 - 1; // 1 - 1 연산  
1 * 1; // 1 * 1 연산  
1 / 1; // 1 / 1 연산  
Math.pow(1, 1); // 1^1 연산
```

## Python 코드

```
1 + 1 # 1 + 1 연산  
1 - 1 # 1 - 1 연산  
1 * 1 # 1 * 1 연산  
1 / 1 # 1 / 1 연산  
1 ** 1 # 1^1 연산
```

## 고급 연산

### 제곱근

입력된 숫자의 제곱근을 반환합니다.



Figure 106: Image

위 블록은  $\sqrt{9}$ 의 제곱근을 구해  $3$ 을 반환합니다.

## Javascript 코드

```
Math.sqrt(9); // 9 의 제곱근
```

## Python 코드

```
import math  
  
math.sqrt(9) # 9 의 제곱근
```

## 절댓값

입력된 숫자의 절댓값을 반환합니다.



Figure 107: Image

다음 블록의 값은  $-10$  입니다.

### Javascript 코드

```
Math.abs(-10); // -10 의 절댓값
```

### Python 코드

```
import math  
  
math.fabs(-10) # -10 의 절댓값
```

### 부호

입력된 숫자의 **부호를 반대로 변경합니다.**



Figure 108: Image

다음 블록의 값은  $-10$  입니다.

### Javascript 코드

```
-10; // 10 의 반대 부호 값
```

### Python 코드

```
import math  
  
-10 # 10 의 반대 부호 값
```

### 지수, 로그 함수

**지수와 로그** 값을 계산하는 블록입니다. 특정 밑수의 거듭제곱이나 로그 값을 구할 때 사용됩니다.

### Javascript 코드

```
Math.log(10); // 자연 로그 10 의 값  
Math.log(10)/Math.log(10); // 밑이 10, 진수가 10 인 로그 값  
Math.exp(2); // e 의 제곱 값  
Math.pow(10,2); // 10 의 제곱 값
```



Figure 109: Image

## Python 코드

```
import math

math.log(10) # 자연 로그 10 의 값
math.log10(10) # 밑이 10, 진수가 10 인 로그 값
math.exp(2) # e 의 제곱 값
math.pow(10,2) # 10 의 제곱 값
```

## 나머지

두 숫자의 나눗셈에서 **나머지를** 구하는 블록입니다.



Figure 110: Image

## Javascript 코드

```
64 % 10; // 64 ÷ 10 의 나머지 값
```

## Python 코드

```
64 % 10 # 64 ÷ 10 의 나머지 값
```

## 삼각 함수

사인, 코사인, 탄젠트 등 **삼각 함수** 값을 계산하는 블록입니다.

## Javascript 코드

```
Math.sin(45 / 180 * Math.PI); // sin 45° 값
Math.cos(45 / 180 * Math.PI); // cos 45° 값
Math.tan(45 / 180 * Math.PI); // tan 45° 값
Math.asin(45) / Math.PI * 180; // asin 45° 값
Math.acos(45) / Math.PI * 180; // acos 45° 값
Math.atan(45) / Math.PI * 180; // atan 45° 값
```

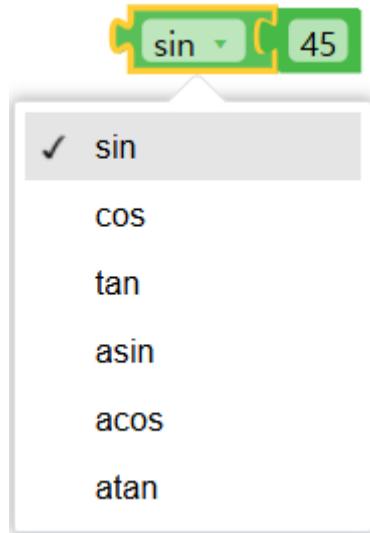


Figure 111: Image

## Python 코드

```
import math
math.sin(45 / 180.0 * math.pi) # sin 45° 값
math.cos(45 / 180.0 * math.pi) # cos 45° 값
math.tan(45 / 180.0 * math.pi) # tan 45° 값
math.asin(45) / math.pi * 180 # asin 45° 값
math.acos(45) / math.pi * 180 # acos 45° 값
math.atan(45) / math.pi * 180 # atan 45° 값
```

## 올림, 버림, 반올림

입력된 숫자를 올림, 버림, 반올림을 수행하여 값을 반환하는 블록입니다.

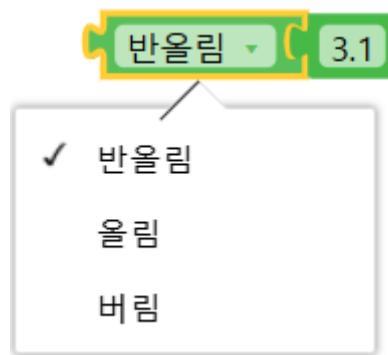


Figure 112: Image

## Javascript 코드

```
Math.round(3.1); // 3.1 반올림 값 3
Math.ceil(3.1); // 3.1 올림 값 4
Math.floor(3.1); // 3.1 버림 값 3
```

## Python 코드

```
import math

round(3.1) # 3.1 반올림 값 3
math.ceil(3.1) # 3.1 올림 값 4
math.floor(3.1) # 3.1 버림 값 3
```

## 특수 숫자 값

연산에 필요한 특수한 수학 상수 ( $\pi$ ,  $e$  등) 와 기호를 제공하는 블록입니다.

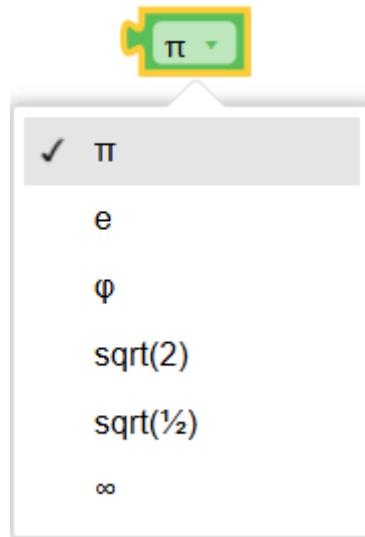


Figure 113: Image

## Javascript 코드

```
Math.PI; // π(원주율) 값
Math.E; // e(자연상수) 값
(1 + Math.sqrt(5)) / 2; // 황금비 값
Math.SQRT2; // 2 의 제곱근 값
Math.SQRT1_2; // 1/2 의 제곱근 값
Infinity; // 무한대 값
```

## Python 코드

```
import math

math.pi # π(원주율) 값
math.e # e(자연상수) 값
(1 + math.sqrt(5)) / 2 # 황금비 값
math.sqrt(2) # 2 의 제곱근 값
math.sqrt(1.0 / 2) # 1/2 의 제곱근 값
float('inf') # 무한대 값
```

## 숫자 조건

입력된 숫자 값이 특정 조건을 만족하는지 검사하는 블록입니다. 숫자가 짝수인지, 홀수인지, 소수인지 등을 판별할 수 있습니다.

조건의 성립 여부를 검사하고 성립하면 참을 성립하지 않으면 거짓을 나타냅니다.



Figure 114: Image

## Javascript 코드

```
var item;

function mathIsPrime(n) { // 소수 판단 함수
    if (n == 2 || n == 3) {
        return true;
    }
    if (isNaN(n) || n <= 1 || n % 1 != 0 || n % 2 === 0 || n % 3 === 0) {
        return false;
    }
    for (var x = 6; x <= Math.sqrt(n) + 1; x += 6) {
        if (n % (x - 1) === 0 || n % (x + 1) === 0) {
            return false;
        }
    }
    return true;
}

item % 2 === 0; // item 이 짝수인지 조건 성립 여부
item % 2 === 1; // item 이 홀수인지 조건 성립 여부
mathIsPrime(item); // item 이 소수인지 조건 성립 여부
item % 1 === 0; // item 이 정수인지 조건 성립 여부
item > 0; // item 이 양수인지 조건 성립 여부
item < 0; // item 이 음수인지 조건 성립 여부
item % 0 === 0 // item 이 0 으로 나누어 떨어지는지 조건 성립 여부
```

## Python 코드

```
import math
from numbers import Number

item = None

def math.isPrime(n): # 소수 판단 함수
    if not isinstance(n, Number):
        try:
            n = float(n)
        except:
            return False
    if n == 2 or n == 3:
        return True
    if n <= 1 or n % 1 != 0 or n % 2 == 0 or n % 3 == 0:
        return False
    for x in range(6, int(math.sqrt(n)) + 2, 6):
        if n % (x - 1) == 0 or n % (x + 1) == 0:
            return False
    return True

item % 2 == 0 # item 이 짝수인지 조건 성립 여부
item % 2 == 1 # item 이 홀수인지 조건 성립 여부
math.isPrime(item) # item 이 소수인지 조건 성립 여부
item % 1 == 0 # item 이 정수인지 조건 성립 여부
item > 0 # item 이 양수인지 조건 성립 여부
item < 0 # item 이 음수인지 조건 성립 여부
item % 0 == 0 # item 이 0 으로 나누어 떨어지는지 조건 성립 여부
```

아래 예제는 `item`이 짝수이므로, “짝수입니다.”라는 메세지가 출력됩니다.



Figure 115: Image

## Javascript 코드

```
var item;

item = 2;
if (item % 2 === 0) {
    window.alert('짝수입니다.');
} else {
    window.alert('홀수입니다.');
}
```

## Python 코드

```
item = None  
  
item = 2  
if item % 2 == 0:  
    print('짝수입니다.')  
else:  
    print('홀수입니다.')
```

## 리스트 연산

리스트를 대상으로 다양한 연산을 수행하는 블록입니다.

합계, 최솟값, 최댓값, 평균값, 중간값, 가장 여러 개 있는 값, 표준 편차, 임의의 항목을 구할 수 있습니다.

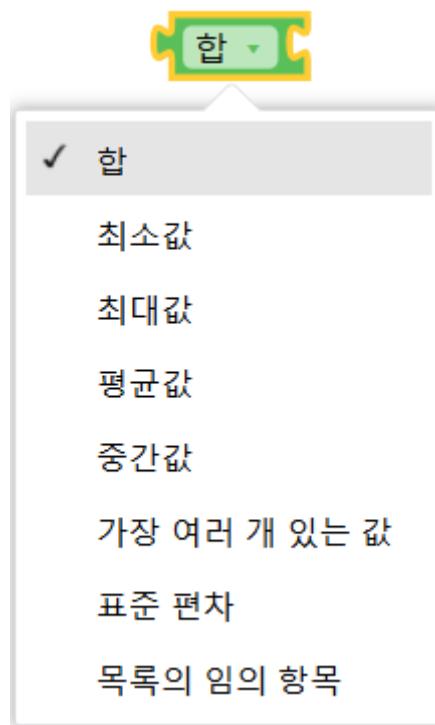


Figure 116: Image

## Javascript 코드

```
function mathMean(myList) { // 평균 계산 함수  
    return myList.reduce(function(x, y) {return x + y;}, 0) / myList.length;  
}  
  
function mathMedian(myList) { // 중간값 계산 함수  
    var localList = myList.filter(function (x) {return typeof x === 'number';});  
    if (!localList.length) return null;  
    localList.sort(function(a, b) {return b - a;});  
    if (localList.length % 2 === 0) {  
        return (localList[localList.length / 2 - 1] + localList[localList.length / 2]) / 2;  
    } else {  
        return localList[(localList.length - 1) / 2];  
    }  
}
```

```

function mathModes(values) { // 최대 빈도수 값 계산 함수
  var modes = [];
  var counts = [];
  var maxCount = 0;
  for (var i = 0; i < values.length; i++) {
    var value = values[i];
    var found = false;
    var thisCount;
    for (var j = 0; j < counts.length; j++) {
      if (counts[j][0] === value) {
        thisCount = ++counts[j][1];
        found = true;
        break;
      }
    }
    if (!found) {
      counts.push([value, 1]);
      thisCount = 1;
    }
    maxCount = Math.max(thisCount, maxCount);
  }
  for (var j = 0; j < counts.length; j++) {
    if (counts[j][1] === maxCount) {
      modes.push(counts[j][0]);
    }
  }
  return modes;
}

function mathStandardDeviation(numbers) { // 표준편차 계산 함수
  var n = numbers.length;
  if (!n) return null;
  var mean = numbers.reduce(function(x, y) {return x + y;}) / n;
  var variance = 0;
  for (var j = 0; j < n; j++) {
    variance += Math.pow(numbers[j] - mean, 2);
  }
  variance /= n;
  return Math.sqrt(variance);
}

function mathRandomList(list) { // 무작위 수 계산 함수
  var x = Math.floor(Math.random() * list.length);
  return list[x];
}

[] .reduce(function(x, y) {return x + y;}, 0); // 리스트 합
Math.min.apply(null, []); // 리스트 최소 값
Math.max.apply(null, []); // 리스트 최대 값
mathMean([]); // 리스트 평균 값
mathMedian([]); // 리스트 중간 값
mathModes([]); // 리스트 최대 빈도수 값
mathStandardDeviation([]); // 리스트 표준편차 값
mathRandomList([]); // 리스트 무작위 수 값

```

## Python 코드

```

from numbers import Number
import math
import random

item = None

def math_mean(myList): # 평균 계산 함수
  localList = [e for e in myList if isinstance(e, Number)]
  if not localList: return
  return float(sum(localList)) / len(localList)

def math_median(myList): # 중간값 계산 함수
  localList = sorted([e for e in myList if isinstance(e, Number)])
  if not localList: return
  if len(localList) % 2 == 0:

```

```

    return (localList[len(localList) // 2 - 1] + localList[len(localList) // 2]) / 2.0
else:
    return localList[(len(localList) - 1) // 2]

def math_modes(some_list): # 최대 빈도수 값 계산 함수
    modes = []
    counts = []
    maxCount = 1
    for item in some_list:
        found = False
        for count in counts:
            if count[0] == item:
                count[1] += 1
                maxCount = max(maxCount, count[1])
                found = True
        if not found:
            counts.append([item, 1])
    for counted_item, item_count in counts:
        if item_count == maxCount:
            modes.append(counted_item)
    return modes

def math_standard_deviation(numbers): # 표준편차 계산 함수
    n = len(numbers)
    if n == 0: return
    mean = float(sum(numbers)) / n
    variance = sum((x - mean) ** 2 for x in numbers) / n
    return math.sqrt(variance)

sum([]) # 리스트 합
min([]) # 리스트 최소 값
max([]) # 리스트 최대 값
math_mean([]) # 리스트 평균 값
math_median([]) # 리스트 중간 값
math_modes([]) # 리스트 최대 빈도수 값
math_standard_deviation([]) # 리스트 표준편차 값
random.choice([]) # 리스트 무작위 수 값

```

아래 블록은 [10, 20, 30] 리스트의 합을 계산하는 블록으로,  $10 + 20 + 30 = 60$  이 됩니다.



Figure 117: Image

## Javascript 코드

```
[10, 20, 30].reduce(function(x, y) {return x + y;}, 0); // [10, 20, 30] 리스트 합의 결과값 60
```

## Python 코드

```
sum([10, 20, 30]) # [10, 20, 30] 리스트 합의 결과값 60
```

## 값의 범위 설정

입력된 값이 특정 범위를 벗어나지 않도록 제한하는 블록입니다. - 최솟값보다 작으면 최솟값으로 조정됩니다. - 최댓값보다 크면 최댓값으로 조정됩니다.



Figure 118: Image

`item`의 값이 1 보다 작으면 1을, 100 보다 크면 100을 반환합니다.

1과 100 사이의 같은 동일한 값을 반환합니다.

## Javascript 코드

```
var item;  
  
Math.min(Math.max(item, 1), 100); // item 을 1 ~ 100 사이로 조정
```

## Python 코드

```
item = None  
  
min(max(item, 1), 100) # item 을 1 ~ 100 사이로 조정
```

## 랜덤 정수

지정한 범위 내에서 랜덤한 정수를 생성하는 블록입니다.



Figure 119: Image

1 이상 100 이하의 정수를 생성합니다.

## Javascript 코드

```
function mathRandomInt(a, b) { // 랜덤 정수 생성 함수  
    if (a > b) {  
        var c = a;  
        a = b;  
        b = c;  
    }  
    return Math.floor(Math.random() * (b - a + 1)) + a;  
}  
  
mathRandomInt(1, 100); // 1 ~ 100 사이 랜덤 정수 생성
```

## Python 코드

```
import random

random.randint(1, 100) # 1 ~ 100 사이 랜덤 정수 생성
```

## 임의 분수

0 과 1 사이에서 무작위의 분수 값을 생성하는 블록입니다.

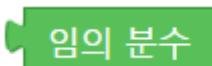


Figure 120: Image

## Javascript 코드

```
Math.random(); // 무작위의 분수 값
```

## Python 코드

```
import random

random.random() # 무작위의 분수 값
```

## 각도

주어진 (x, y) 좌표가 원점 (0,0) 과 이루는 각도를 계산하는 블록입니다. 좌표 위치를 기반으로 방향을 판별하는 데 사용할 수 있습니다.



Figure 121: Image

## Javascript 코드

```
Math.atan2(1, 1) / Math.PI * 180; // 각도 계산 값
```

## Python 코드

```
import math  
  
math.atan2(1, 1) / math.pi * 180 # 각도 계산 값
```

## 음성-인식

## 블록

### 언어 정하기

음성 인식 언어를 설정합니다.

언어를 설정하지 않아도 자동으로 설정됩니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
language	드롭다운 옵션	인식 언어	en-US(English), ko-KR(한국어)

### 자바스크립트 코드

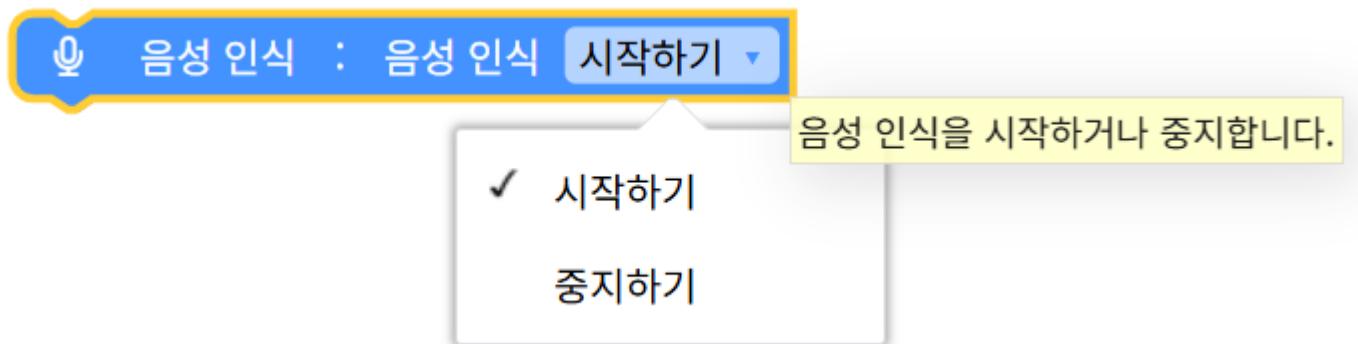
```
// 음성 인식 언어를 English로 설정하기  
$('ASR*0:Lang').d = 'en-US';  
  
// 음성 인식 언어를 한국어로 설정하기  
$('ASR*0:Lang').d = 'ko-KR';
```

### 파이썬 코드

```
# 음성 인식 언어를 English로 설정하기  
__('ASR*0:Lang').d = 'en-US'  
  
# 음성 인식 언어를 한국어로 설정하기  
__('ASR*0:Lang').d = 'ko-KR'
```

## 음성 인식 시작하기 / 중지하기

음성 인식을 시작하거나 중지합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	음성 인식 여부	시작하기 (1), 중지하기 (0)

## 자바스크립트 코드

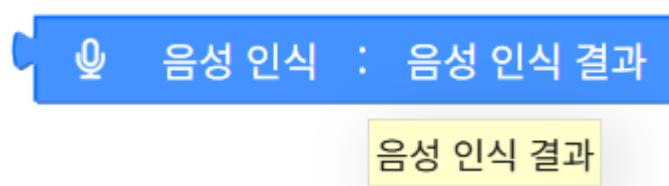
```
// 음성 인식 시작하기  
$('ASR*0:listen').d = 1;  
  
// 음성 인식 중지하기  
$('ASR*0:listen').d = 0;
```

## 파이썬 코드

```
# 음성 인식 시작하기  
__('ASR*0:listen').d = 1  
  
# 음성 인식 중지하기  
__('ASR*0:listen').d = 0
```

## 음성 인식 결과

음성 인식 결과를 반환합니다.



## 자바스크립트 코드

```
// 음성 인식 결과  
$('ASR*0:result').d;
```

## 파이썬 코드

```
# 음성 인식 결과  
__('ASR*0:result').d
```

## 음성 인식 중인가?

음성 인식 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.

 음성 인식 : 음성 인식 중인가?

음성 인식을 하고 있는지 여부

## 자바스크립트 코드

```
// 음성 인식 중인가?  
$('ASR*0:state').d;
```

## 파이썬 코드

```
# 음성 인식 중인가?  
__('ASR*0:state').d
```

## 제어

---

블록 코딩에서 제어 블록은 프로그램의 흐름을 조작하는 역할을 합니다.

일정 시간 대기하거나, 키보드 입력 감지, 로그 출력 등의 기능을 수행할 수 있습니다.

## 블록

### 기다리기

기다리기 블록은 이전 명령을 수행한 후 일정 시간 동안 대기한 후 다음 명령을 실행하는 기능을 합니다. 이 블록을 사용하면 특정 동작을 일정한 간격으로 실행하거나, 시간 차이를 두고 실행할 수 있습니다.



Figure 122: Image

이 블록이 실행되면, x 초 동안 멈춘 후 다음 명령을 실행합니다.

### Javascript 코드

```
await __wait(x * 1000);
```

### Python 코드

```
import asyncio  
  
await asyncio.sleep(x)
```

### 1 프레임 기다리기

1 프레임 기다리기 블록은 프로그램의 실행을 한 프레임 (약 0.001 초) 동안 멈춘 후 다음 명령을 실행합니다. 프레임 단위로 프로그램을 제어할 때 유용합니다.



Figure 123: Image

이 블록이 실행되면, 한 프레임 (약 0.001 초) 동안 멈춘 후 다음 명령을 실행합니다.

### Javascript 코드

```
await __wait(1);
```

## Python 코드

```
import asyncio  
  
await asyncio.sleep(0.001)
```

## 키 다운 / 키 업

- 키 다운: 특정 키를 눌렀을 때 동작을 수행
- 키 업: 특정 키에서 손을 뗄 때 동작을 수행



Figure 124: Image

키 다운, 키 업 코드는 아래와 같습니다.

## Javascript 코드

```
--keydown(32); // 스페이스바 키 다운  
--keyup(32); // 스페이스바 키 업
```

## Python 코드

```
--keydown(32) # 스페이스 바 키 다운  
--keyup(32) # 스페이스바 키 업
```

스페이스바를 눌렀을 때“<sub>stop</sub>”을 출력하는 블록입니다.



Figure 125: Image

## Javascript 코드

```
if (_keydown(32)) {  
    window.alert('stop!');  
}
```

## Python 코드

```
if __keydown(32):  
    print('stop!')
```

## 키 입력

키 입력 블록은 여러 개의 **키를 동시에 눌렀을 때** 동작을 수행하도록 설정할 수 있습니다.

이 블록을 활용하면 shift, ctrl, alt 와 같은 조합 키를 포함하여 최대 5 개까지 감지할 수 있습니다.



Figure 126: Image

## Javascript 코드

```
__keypressed([32, 38, 0, 0, 0]) // 32(스페이스 바), 38(위쪽 방향 키) 를 동시에 눌렀을 때
```

## Python 코드

```
__keypressed([32, 38, 0, 0, 0]) # 32(스페이스 바), 38(위쪽 방향 키) 를 동시에 눌렀을 때
```

Shift + Space 키가 눌리면 “Keypress event”를 출력하는 블록입니다.

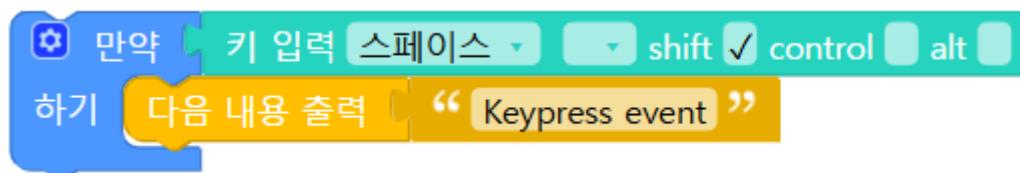


Figure 127: Image

## Javascript 코드

```
if (__keypressed([32, 0, 16, 0, 0])) { // Shift + Space 키 입력 시  
    window.alert('Keypress event');  
}
```

## Python 코드

```
if __keypressed([32, 0, 16, 0, 0]): # Shift + Space 키 입력 시  
    print('Keypress event')
```

## 로그 출력하기

로그 출력하기 블록은 특정 변수나 속성 값을 실시간으로 콘솔 창에 출력하여 프로그램 동작을 분석할 수 있도록 합니다. 디버깅 과정에서 값을 확인하거나 데이터 변화를 추적할 때 유용합니다.



Figure 128: Image

## Javascript 코드

```
__log(1, 'A', 'K') // 'A' 태그의 K 단위 1 값 로그 출력
```

## Python 코드

```
__log(1, 'A', 'K') # 'A' 태그의 K 단위 1 값 로그 출력
```

랜덤한 정수를 콘솔에서 확인하는 예시입니다.



Figure 129: Image

결과는 다음과 같습니다.

## Javascript 코드

```
var random;  
  
function mathRandomInt(a, b) { // 무작위 수 생성 함수  
    if (a > b) {
```

[A] 9K  
[A] 100K  
[A] 98K  
[A] 4K  
[A] 86K  
[A] 3K  
[A] 61K  
[A] 23K  
[A] 70K  
[A] 42K  
[A] 56K  
[A] 36K  
[A] 28K

Figure 130: Image

```

var c = a;
a = b;
b = c;
}
return Math.floor(Math.random() * (b - a + 1) + a);
}

random = mathRandomInt(1, 100);
__log(random, 'A', 'K'); // 1 ~ 100 사이 random 정수 'A' 태그 'K' 단위 로그 출력하기

```

## Python 코드

```

import random

random2 = None
random2 = random.randint(1, 100)
__log(random2, 'A', 'K') # 1 ~ 100 사이 random 정수 'A' 태그 'K' 단위 로그 출력하기

```

## 스코프 출력하기

**스코프 출력하기** 블록은 특정 값의 변화를 실시간 그래프 형태로 표현하여 **스코프** 창에서 데이터를 직관적으로 분석할 수 있도록 합니다.

그래프의 색상, 최소/최대 값, 범위를 설정할 수 있으며, 센서 **데이터 시각화**나 변수 변화 추적에 유용합니다.

스코프 출력하기 | 태그 [ A ] 최소 0 최대 1 색깔 #000000

Figure 131: Image

### Javascript 코드

```
--scope('_', 0, 1, '#000000', 0); // 0 의 값 0 ~ 1 범위에서 검은 색 스코프 출력
```

### Python 코드

```
--scope('_', 0, 1, '#000000', 0) # 0 의 값 0 ~ 1 범위에서 검은 색 스코프 출력
```

랜덤한 변수 값을 빨간색 그래프로 확인하는 예시입니다.

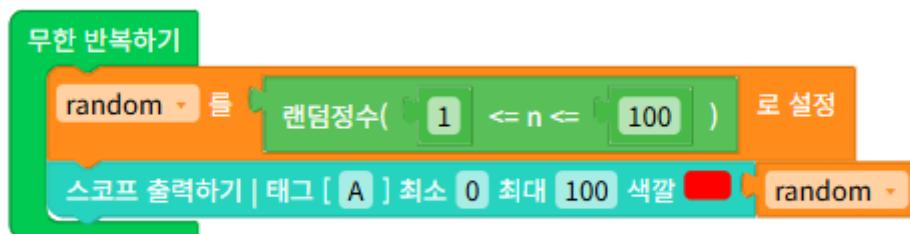


Figure 132: Image

결과는 다음과 같습니다.

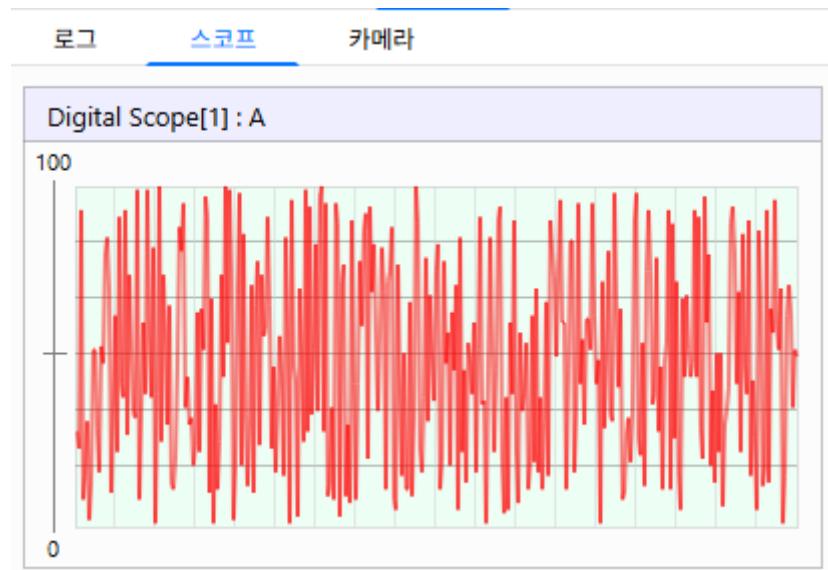


Figure 133: Image

## Javascript 코드

```
var random;

function mathRandomInt(a, b) { // 무작위 수 생성 함수
  if (a > b) {
    var c = a;
    a = b;
    b = c;
  }
  return Math.floor(Math.random() * (b - a + 1)) + a;
}

random = mathRandomInt(1, 100);
__scope('A', 0, 100, '#ff0000', random); // 0 ~ 100 범위에서 1 ~ 100 사이 random 정수 'A' 태그 빨간색 스크립트 출력하기
}
```

## Python 코드

```
import random

random2 = None
random2 = random.randint(1, 100)
__scope('A', 0, 100, '#ff0000', 0) # 0 ~ 100 범위에서 1 ~ 100 사이 random2 정수 'A' 태그 빨간색 스크립트 출력하기
```

## 조건문

---

조건문은 컴퓨터 프로그래밍에서 핵심적인 역할을 합니다. 조건문을 사용하면 다음과 같은 문장을 표현할 수 있습니다:

- 왼쪽에 길이 있다면, 왼쪽으로 돈다.
- 점수가 100 이면, “잘 했어요!”를 출력한다.

## 블록

### 만약

가장 간단한 조건문은 **만약** 블록입니다. 아래와 같이 나타낼 수 있습니다:



Figure 134: Image

이 블록이 실행되면, 변수 **x**의 값을 100과 비교합니다. 만약 값이 100보다 크면, “큰 수입니다.”가 출력됩니다. 그렇지 않으면 아무 일이 일어나지 않습니다.

## Javascript 코드

```
if(x > 100) {  
    window.alert("큰 수입니다.");  
}
```

## Python 코드

```
if x > 100:  
    print("큰 수입니다.")
```

## 만약-아니라면

조건이 참이 아닐 경우 무엇을 해야 할지 지정할 수도 있습니다. 아래 예시와 같습니다:

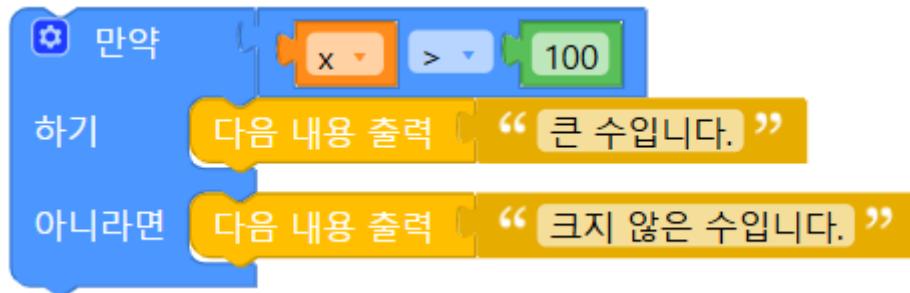


Figure 135: Image

이전 블록처럼, **x** 가 100 보다 크면“큰 수입니다.”가 출력됩니다. 그렇지 않으면, “크지 않은 수입니다.”가 출력됩니다.

**만약** 블록에는 하나의 **아니라면** 섹션이 있을 수 있으며, 그 이상은 있을 수 없습니다.

## Javascript 코드

```
if(x > 100) {  
    window.alert('큰 수입니다.');//  
} else {  
    window.alert('크지 않은 수입니다.');//  
}
```

## Python 코드

```
if x > 100:  
    print('큰 수입니다.')  
else:  
    print('크지 않은 수입니다.')
```

## 만약-다른 경우

하나의 **만약** 블록에서 여러 조건을 테스트할 수도 있습니다. **다른 경우** 절을 추가하여 가능해집니다:



Figure 136: Image

이 블록은 먼저 **x** 가 100 보다 큰지 확인하고, 그렇다면“큰 수입니다.”를 출력합니다. 그렇지 않으면, **x** 가 42 와 같은지 확인합니다. 만약 그렇다면“행운의 수입니다.”를 출력합니다. 그렇지 않으면 아무 일도 일어나지 않습니다.

**만약** 블록은 다양한 개수의 **다른 경우** 블록을 가질 수 있습니다. 조건은 위에서 아래로 차례대로 평가되며, 하나가 만족되거나 조건이 더 이상 남아있지 않으면 끝납니다.

## Javascript 코드

```
if(x > 100) {
    window.alert('큰 수입니다.');
} else if(x == 42) {
    window.alert('행운의 수입니다.');
}
```

## Python 코드

```
if x > 100:
    print('큰 수입니다.')
elif x == 42:
    print('행운의 수입니다.')
```

## 만약-다른 경우-아니라면

아래와 같이 **만약** 블록은 **다른 경우와 아니라면** 섹션을 모두 가질 수 있습니다:

**아니라면** 섹션은 앞의 조건들이 모두 참이 아니더라도 반드시 어떤 작업을 수행하게 만듭니다.

**아니라면** 섹션은 어떤 수의 **다른 경우** 섹션 뒤에 올 수 있으며, 그 수는 0 이상입니다.



Figure 137: Image

## Javascript 코드

```

if(x > 100) {
    window.alert('큰 수입니다.');
} else if(x === 42) {
    window.alert('행운의 수입니다.');
} else {
    window.alert('크지 않은 수입니다.');
}

```

## Python 코드

```

if x > 100:
    print('큰 수입니다.')
elif x == 42:
    print('행운의 수입니다.')
else:
    print('크지 않은 수입니다.')

```

## 블록 수정 방법

단순한 **만약** 블록만 도구 상자에 나타납니다:



Figure 138: Image

**다른 경우와 아니라면** 절을 추가하려면, 기어 아이콘을 클릭하여 새 창을 엽니다:

**다른 경우와 아니라면** 절을 **만약** 블록 아래로 드래그하고, 그것들을 재정렬하거나 제거할 수 있습니다. 완료되면 기어 아이콘을 클릭하여 창을 닫습니다:



Figure 139: Image

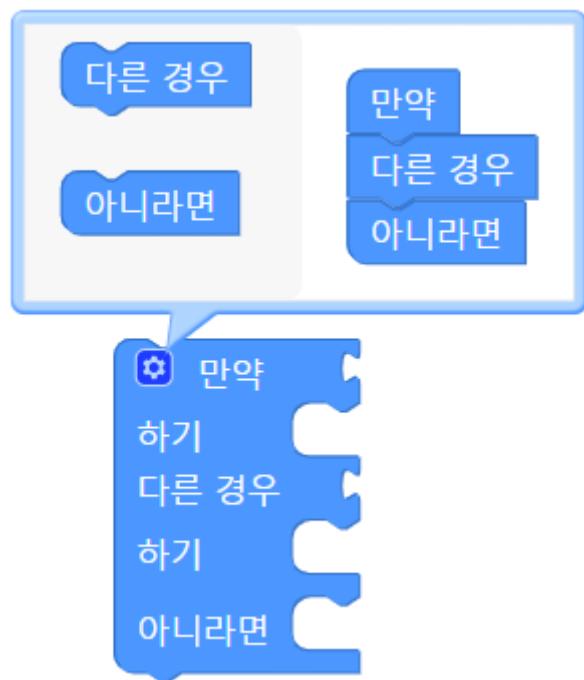


Figure 140: Image

다른 경우 서브 블록은 원하는 개수만큼 추가할 수 있지만, 아니라면 블록은 최대 한 개만 추가할 수 있다는 점에 유의하십시오. ## 터틀

## 블록

### 바퀴 속도 설정하기

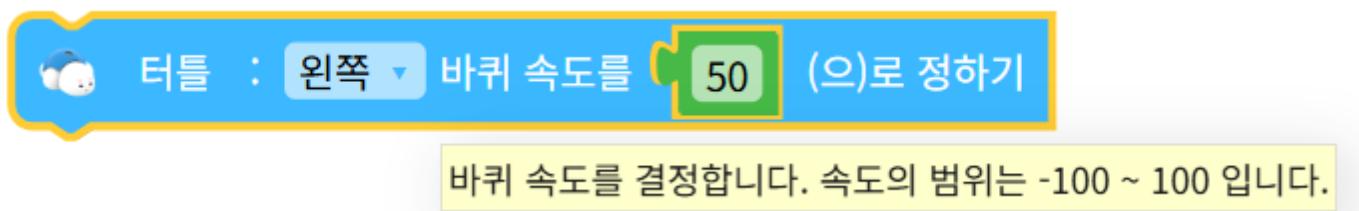
터틀의 바퀴 속도를 설정합니다.

바퀴 속도가 양수이면 앞쪽 방향으로 회전하고, 바퀴 속도가 음수이면 뒤쪽 방향으로 회전합니다.

예를 들어, 바퀴 속도가 100 이라면, 앞쪽 방향으로 100 의 속도로 회전하고,

바퀴 속도가 -100 이라면, 뒤쪽 방향으로 100 의 속도로 회전합니다.

한번 바퀴 속도를 설정하면, 다시 바퀴 속도를 설정하기 전까지 해당 속도로 터틀이 이동합니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	바퀴 속도	-100 ~ 100 정수, 0: 정지

### 자바스크립트 코드

```
// 원쪽 바퀴 속도를 50(으)로 정하기
if($('Turtle*0:wheel.move').d != 0) {
  $('Turtle*0:wheel.move').d = 0;
}
$('Turtle*0:wheel.speed.left').d = __getSpeed('Turtle*0', 50);

// 오른쪽 바퀴 속도를 40(으)로 정하기
if($('Turtle*0:wheel.move').d != 0) {
  $('Turtle*0:wheel.move').d = 0;
}
$('Turtle*0:wheel.speed.right').d = __getSpeed('Turtle*0', 40);

// 양쪽 바퀴 속도를 30(으)로 정하기
if($('Turtle*0:wheel.move').d != 0) {
  $('Turtle*0:wheel.move').d = 0;
}
```

```

$( 'Turtle*0:wheel.speed.left' ).d = __getSpeed( 'Turtle*0', 30 );
$( 'Turtle*0:wheel.speed.right' ).d = __getSpeed( 'Turtle*0', 30 );

```

## 파이썬 코드

```

# 左쪽 바퀴 속도를 50(으)로 정하기
if __('Turtle*0:wheel.move').d != 0:
    __('Turtle*0:wheel.move').d = 0
__('Turtle*0:wheel.speed.left').d = __getSpeed( 'Turtle*0', 50 )

# 오른쪽 바퀴 속도를 40(으)로 정하기
if __('Turtle*0:wheel.move').d != 0:
    __('Turtle*0:wheel.move').d = 0
__('Turtle*0:wheel.speed.right').d = __getSpeed( 'Turtle*0', 40 )

# 양쪽 바퀴 속도를 30(으)로 정하기
if __('Turtle*0:wheel.move').d != 0:
    __('Turtle*0:wheel.move').d = 0
__('Turtle*0:wheel.speed.left').d = __getSpeed( 'Turtle*0', 30 )
__('Turtle*0:wheel.speed.right').d = __getSpeed( 'Turtle*0', 30 )

```

## 거리 이동하기

터틀이 이동할 거리를 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

거리 값이 0 일 경우에는, 현재 설정되어 있는 바퀴 속도대로 멈추지 않고 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
distance	입력값	거리 값	0 이상 실수
unit	드롭다운 옵션	거리 단위	cm, mm, 인치 (inch)

## 자바스크립트 코드

```
// 50cm 이동하기 | 기다리기 0
$('Turtle*0:wheel.move').d = __getDistance('Turtle*0', 50, 'cm');
await $('Turtle*0:wheel.!move').w();

// 50mm 이동하기 | 기다리기 X
$('Turtle*0:wheel.move').d = __getDistance('Turtle*0', 50, 'mm');

// 50inch 이동하기 | 기다리기 X
$('Turtle*0:wheel.move').d = __getDistance('Turtle*0', 50, 'inch');
```

## 파이썬 코드

```
# 50cm 이동하기 | 기다리기 0
__('Turtle*0:wheel.move').d = __getDistance('Turtle*0', 50, 'cm')
await __('Turtle*0:wheel.!move').w()

# 50mm 이동하기 | 기다리기 X
__('Turtle*0:wheel.move').d = __getDistance('Turtle*0', 50, 'mm')

# 50inch 이동하기 | 기다리기 X
__('Turtle*0:wheel.move').d = __getDistance('Turtle*0', 50, 'inch')
```

## 시간 이동하기

터틀이 이동할 시간을 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
time	입력값	시간 값	0 이상 실수
unit	드롭다운 옵션	시간 단위	초 (seconds), 밀리초 (milliseconds)

옵션을 밀리초 (milliseconds)로 설정한 경우에는, time 값을 1000으로 나눈 값이 입력됩니다.

## 자바스크립트 코드

```
// 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Turtle*0', 5, true);

// 5 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Turtle*0', 0.005, false);
```

## 파이썬 코드

```
# 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Turtle*0', 5, True)

# 5 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Turtle*0', 0.005, False)
```

## 제자리 돌기

터틀이 제자리에서 회전할 방향과 각도를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	회전 방향	왼쪽 (left), 오른쪽 (right)
degree	입력값	회전 각도	0 이상 정수

## 자바스크립트 코드

```
// 원쪽으로 90 도 제자리 돌기 | 기다리기 0
await __turn_degree_left('Turtle*0', 90, true);

// 오른쪽으로 270 도 제자리 돌기 | 기다리기 X
__turn_degree_right('Turtle*0', 270, false);
```

## 파이썬 코드

```
# 왼쪽으로 90 도 제자리 돌기 | 기다리기 0
await __turn_degree_left('Turtle*0', 90, True)

# 오른쪽으로 270 도 제자리 돌기 | 기다리기 X
__turn_degree_right('Turtle*0', 270, False)
```

## 바퀴 속도 변경하기

터틀의 바퀴 속도를 변경합니다.

현재의 바퀴 속도에 입력한 속도를 더한 값이 새로운 바퀴 속도가 됩니다.

새롭게 설정된 바퀴 속도의 범위는 -100 ~ 100 으로 설정됩니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	현재 바퀴 속도에 더할 속도 값	-200 ~ 200 정수

## 자바스크립트 코드

```
// 왼쪽 바퀴 속도를 50 만큼 바꾸기
if($('Turtle*0:wheel.move').d != 0) {
    $('Turtle*0:wheel.move').d = 0;
}
$('Turtle*0:wheel.speed.left').d = $('Turtle*0:wheel.speed.left').d + __getSpeed('Turtle*0', 50);

// 오른쪽 바퀴 속도를 -40 만큼 바꾸기
if($('Turtle*0:wheel.move').d != 0) {
```

```

    $('Turtle*0:wheel.move').d = 0;
}
$('Turtle*0:wheel.speed.right').d = $('Turtle*0:wheel.speed.right').d + __getSpeed('Turtle*0', (-40));

// 양쪽 바퀴 속도를 200 만큼 바꾸기
if($('Turtle*0:wheel.move').d != 0) {
    $('Turtle*0:wheel.move').d = 0;
}
$('Turtle*0:wheel.speed.left').d = $('Turtle*0:wheel.speed.left').d + __getSpeed('Turtle*0', 200);
$('Turtle*0:wheel.speed.right').d = $('Turtle*0:wheel.speed.right').d + __getSpeed('Turtle*0', 200);

```

## 파이썬 코드

```

# 左쪽 바퀴 속도를 50 만큼 바꾸기
if ___('Turtle*0:wheel.move').d != 0:
    ___('Turtle*0:wheel.move').d = 0
___('Turtle*0:wheel.speed.left').d = ___('Turtle*0:wheel.speed.left').d + __getSpeed('Turtle*0', 50)

# 오른쪽 바퀴 속도를 -40 만큼 바꾸기
if ___('Turtle*0:wheel.move').d != 0:
    ___('Turtle*0:wheel.move').d = 0
___('Turtle*0:wheel.speed.right').d = ___('Turtle*0:wheel.speed.right').d + __getSpeed('Turtle*0', (-40))

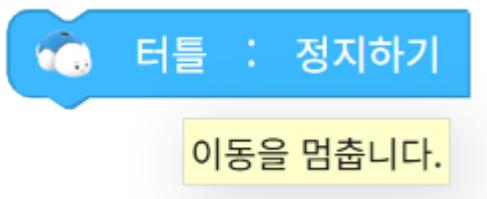
# 양쪽 바퀴 속도를 200 만큼 바꾸기
if ___('Turtle*0:wheel.move').d != 0:
    ___('Turtle*0:wheel.move').d = 0
___('Turtle*0:wheel.speed.left').d = ___('Turtle*0:wheel.speed.left').d + __getSpeed('Turtle*0', 200)
___('Turtle*0:wheel.speed.right').d = ___('Turtle*0:wheel.speed.right').d + __getSpeed('Turtle*0', 200)

```

## 정지하기

터틀의 이동을 멈춥니다.

터틀의 양쪽 바퀴 속도가 모두 0으로 초기화됩니다.



## 자바스크립트 코드

```

__stopMove('Turtle*0');

```

## 파이썬 코드

```

__stopMove('Turtle*0')

```

## 바퀴가 움직이는 중인가?

터틀의 바퀴가 움직이고 있는지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 터틀 : 바퀴가 움직이는 중인가?

바퀴가 움직이는 중이면 true, 멈춰있으면 false를 반환합니다.

### 자바스크립트 코드

```
$(`'Turtle*0:wheel.speed.left').d != 0 || $(`'Turtle*0:wheel.speed.right').d != 0
```

### 파이썬 코드

```
__(`'Turtle*0:wheel.speed.left').d != 0 or __(`'Turtle*0:wheel.speed.right').d != 0
```

### 바퀴 기준 회전하기

터틀이 제자리에서 회전할 기준과 방향, 각도를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



✓ 左侧

右侧

회전할 기준과 방향, 각도를 설정합니다.  
기다리기를 체크하면, 회전이 완료될 때까지 기다립니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pivot	드롭다운 옵션	회전 기준	왼쪽 바퀴 (left wheel), 오른쪽 바퀴 (right wheel)

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	앞쪽 (forward), 뒤쪽 (backward)
degree	입력값	회전 각도	0 이상 정수

## 자바스크립트 코드

```
// 원쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('Turtle*0', 'left wheel', 'forward', 90, true);

// 원쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('Turtle*0', 'left wheel', 'backward', 90, false);

// 오른쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('Turtle*0', 'right wheel', 'forward', 90, true);

// 오른쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('Turtle*0', 'right wheel', 'backward', 90, false);
```

## 파이썬 코드

```
# 원쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('Turtle*0', 'left wheel', 'forward', 90, True)

# 원쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('Turtle*0', 'left wheel', 'backward', 90, False)

# 오른쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('Turtle*0', 'right wheel', 'forward', 90, True)

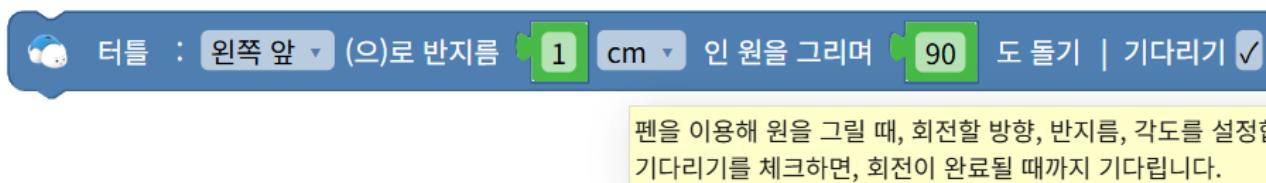
# 오른쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('Turtle*0', 'right wheel', 'backward', 90, False)
```

## 원 그리며 돌기

펜을 이용해 원을 그릴 때 회전할 방향, 반지름, 각도를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 앞 (left forward), 왼쪽 뒤 (left backward), 오른쪽 앞 (right forward), 오른쪽 뒤 (right backward)
radius	입력값	원의 반지름	0 이상 실수
unit	드롭다운 옵션	거리 단위	cm, mm, 인치 (inch)
degree	입력값	회전 각도	0 이상 실수

## 자바스크립트 코드

```
// 왼쪽 앞 (으)로 반지름 1cm 인 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('Turtle*0', '', 'left forward', __getDistance('Turtle*0', 1, 'cm'), 90, true); // (robot, pivot, direction, circle_info, degree, wait_w)

// 왼쪽 뒤 (으)로 반지름 1mm 인 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('Turtle*0', '', 'left backward', __getDistance('Turtle*0', 1, 'mm'), 90, true); // (robot, pivot, direction, circle_info, degree, wait_w)

// 오른쪽 앞 (으)로 반지름 1inch 인 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('Turtle*0', '', 'right forward', __getDistance('Turtle*0', 1, 'inch'), 90, false); // (robot, pivot, direction, circle_info, degree, wait_w)

// 오른쪽 뒤 (으)로 반지름 1cm 인 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('Turtle*0', '', 'right backward', __getDistance('Turtle*0', 1, 'cm'), 90, false); // (robot, pivot, direction, circle_info, degree, wait_w)
```

## 파이썬 코드

```
# 왼쪽 앞 (으)로 반지름 1cm 인 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('Turtle*0', '', 'left forward', __getDistance('Turtle*0', 1, 'cm'), 90, True) #(robot, pivot, direction, circle_info, degree, wait_w)

# 왼쪽 뒤 (으)로 반지름 1mm 인 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('Turtle*0', '', 'left backward', __getDistance('Turtle*0', 1, 'mm'), 90, True) #(robot, pivot, direction, circle_info, degree, wait_w)

# 오른쪽 앞 (으)로 반지름 1inch 인 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('Turtle*0', '', 'right forward', __getDistance('Turtle*0', 1, 'inch'), 90, False) #(robot, pivot, direction, circle_info, degree, wait_w)

# 오른쪽 뒤 (으)로 반지름 1cm 인 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('Turtle*0', '', 'right backward', __getDistance('Turtle*0', 1, 'cm'), 90, False) #(robot, pivot, direction, circle_info, degree, wait_w)
```

## 센서로 선 따라가기

터틀이 바닥의 컬러 센서를 이용하여 특정한 선을 따라갑니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



터틀 : 검정색 ▾ 선을 따라가기 | 기다리기 ✓

✓ 검정색

빨간색

초록색

파란색

아무 색

바닥의 컬러 센서를 이용해, 특정 색의 선을 따라 이동합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	선의 색	검정색 (8), 빨간색 (9), 초록색 (11), 파란색 (13), 아무색 (15)

## 자바스크립트 코드

```
// 검정색 선을 따라가기 (8) | 기다리기 0
$('Turtle*0:trace.mode').d = 8;
await $('Turtle*0:trace.!mode').w();

// 빨간색 선을 따라가기 (9) | 기다리기 0
$('Turtle*0:trace.mode').d = 9;
await $('Turtle*0:trace.!mode').w();

// 초록색 선을 따라가기 (11) | 기다리기 0
$('Turtle*0:trace.mode').d = 11;
await $('Turtle*0:trace.!mode').w();

// 파란색 선을 따라가기 (13) | 기다리기 X
$('Turtle*0:trace.mode').d = 13;

// 아무색 선을 따라가기 (15) | 기다리기 X
$('Turtle*0:trace.mode').d = 15;
```

## 파이썬 코드

```
# 검정색 선을 따라가기 (8) | 기다리기 0
__('Turtle*0:trace.mode').d = 8
await __('Turtle*0:trace.!mode').w()
```

```

# 빨간색 선을 따라가기 (9) | 기다리기 0
__('Turtle*0:trace.mode').d = 9
await __('Turtle*0:trace.!mode').w()

# 초록색 선을 따라가기 (11) | 기다리기 0
__('Turtle*0:trace.mode').d = 11
await __('Turtle*0:trace.!mode').w()

# 파란색 선을 따라가기 (13) | 기다리기 X
__('Turtle*0:trace.mode').d = 13

# 아무색 선을 따라가기 (15) | 기다리기 X
__('Turtle*0:trace.mode').d = 15

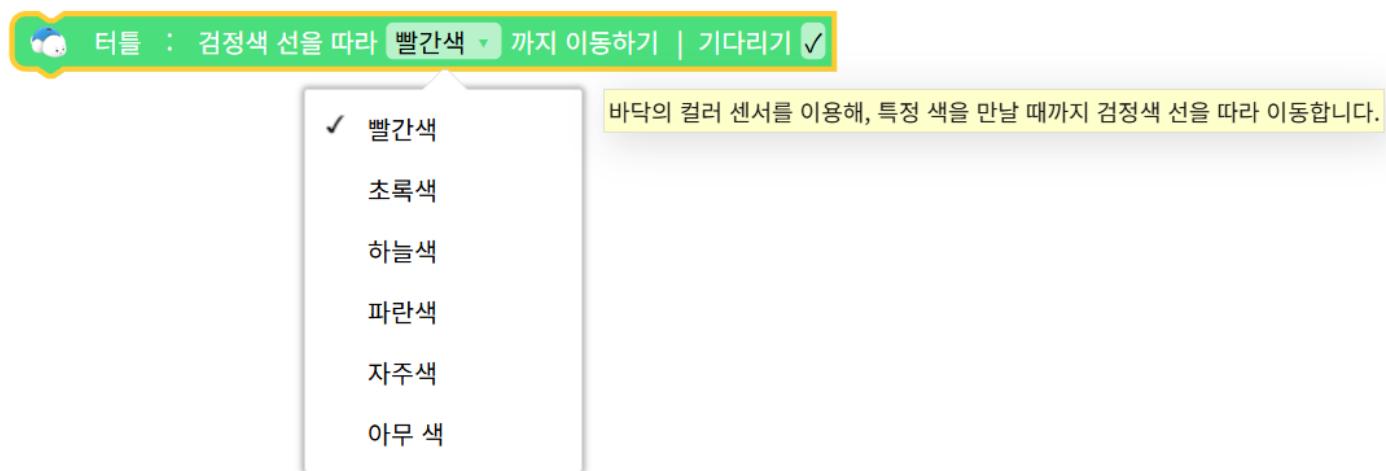
```

## 특정 색 까지 검정색 선을 따라 이동하기

터틀 바닥의 컬러 센서를 이용해, 특정 색을 만날 때 까지 검정색 선을 따라 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	선의 색	빨간색 (49), 초록색 (51), 하늘색 (52), 파란색 (53), 자주색 (54), 아무색 (55)

## 자바스크립트 코드

```

// 검정색 선을 따라 빨간색까지 이동하기 (49) | 기다리기 0
$('Turtle*0:trace.mode').d = 49;
await $('Turtle*0:trace.!mode').w();

// 검정색 선을 따라 초록색까지 이동하기 (51) | 기다리기 0
$('Turtle*0:trace.mode').d = 51;

```

```

await $('Turtle*0:trace.!mode').w();

// 검정색 선을 따라 하늘색까지 이동하기 (52) | 기다리기 0
$('Turtle*0:trace.mode').d = 52;
await $('Turtle*0:trace.!mode').w();

// 검정색 선을 따라 파란색까지 이동하기 (53) | 기다리기 X
$('Turtle*0:trace.mode').d = 53;

// 검정색 선을 따라 자주색까지 이동하기 (54) | 기다리기 X
$('Turtle*0:trace.mode').d = 54;

// 검정색 선을 따라 아무색까지 이동하기 (55) | 기다리기 X
$('Turtle*0:trace.mode').d = 55;

```

## 파이썬 코드

```

# 검정색 선을 따라 빨간색까지 이동하기 (49) | 기다리기 0
__('Turtle*0:trace.mode').d = 49
await __('Turtle*0:trace.!mode').w()

# 검정색 선을 따라 초록색까지 이동하기 (51) | 기다리기 0
__('Turtle*0:trace.mode').d = 51
await __('Turtle*0:trace.!mode').w()

# 검정색 선을 따라 하늘색까지 이동하기 (52) | 기다리기 0
__('Turtle*0:trace.mode').d = 52
await __('Turtle*0:trace.!mode').w()

# 검정색 선을 따라 파란색까지 이동하기 (53) | 기다리기 X
__('Turtle*0:trace.mode').d = 53

# 검정색 선을 따라 자주색까지 이동하기 (54) | 기다리기 X
__('Turtle*0:trace.mode').d = 54

# 검정색 선을 따라 아무색까지 이동하기 (55) | 기다리기 X
__('Turtle*0:trace.mode').d = 55

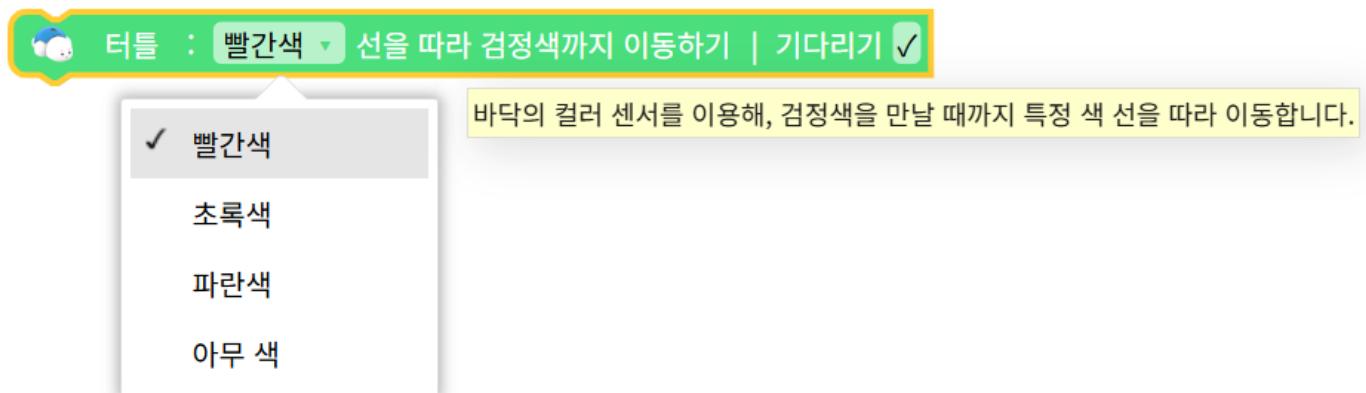
```

## 검정색 까지 특정 색 선을 따라 이동하기

터틀 바닥의 컬러 센서를 이용해, 검정 색을 만날 때 까지 특정 색 선을 따라 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	선의 색	빨간색 (57), 초록색 (59), 파란색 (61), 아무색 (63)

## 자바스크립트 코드

```
// 빨간색 선을 따라 검정색까지 이동하기 (57) | 기다리기 0
$('Turtle*0:trace.mode').d = 57;
await $('Turtle*0:trace.!mode').w();

// 초록색 선을 따라 검정색까지 이동하기 (59) | 기다리기 0
$('Turtle*0:trace.mode').d = 59;
await $('Turtle*0:trace.!mode').w();

// 파란색 선을 따라 검정색까지 이동하기 (61) | 기다리기 X
$('Turtle*0:trace.mode').d = 61;

// 아무 색 선을 따라 검정색까지 이동하기 (63) | 기다리기 X
$('Turtle*0:trace.mode').d = 63;
```

## 파이썬 코드

```
# 빨간색 선을 따라 검정색까지 이동하기 (57) | 기다리기 0
__('Turtle*0:trace.mode').d = 57
await __('Turtle*0:trace.!mode').w()

# 초록색 선을 따라 검정색까지 이동하기 (59) | 기다리기 0
__('Turtle*0:trace.mode').d = 59
await __('Turtle*0:trace.!mode').w()

# 파란색 선을 따라 검정색까지 이동하기 (61) | 기다리기 X
__('Turtle*0:trace.mode').d = 61

# 아무 색 선을 따라 검정색까지 이동하기 (63) | 기다리기 X
__('Turtle*0:trace.mode').d = 63
```

## 검정색 교차로에서 행동 설정하기

터틀 바닥의 컬러 센서를 이용해, 검정색 교차로를 인식하면 지정된 행동을 합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



터틀 : 검정색 교차로에서 앞으로 건너가기 ▾ | 기다리기 ✓

✓ 앞으로 건너가기

왼쪽으로 돌기

오른쪽으로 돌기

유턴하기

검정색 교차로에서 지정된 행동을 합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
behavior	드롭다운 옵션	검정색 교차로에서의 행동	앞으로 건너가기 (16), 왼쪽으로 돌기 (24), 오른쪽으로 돌기 (32), 유턴하기 (40)

## 자바스크립트 코드

```
// 검정색 교차로에서 앞으로 건너가기 (16) | 기다리기 0
$(`'Turtle*0:trace.mode').d = 16;
await $(`'Turtle*0:trace.!mode').w();

// 검정색 교차로에서 왼쪽으로 돌기 (24) | 기다리기 0
$(`'Turtle*0:trace.mode').d = 24;
await $(`'Turtle*0:trace.!mode').w();

// 검정색 교차로에서 오른쪽으로 돌기 (32) | 기다리기 X
$(`'Turtle*0:trace.mode').d = 32;

// 검정색 교차로에서 유턴하기 (40) | 기다리기 X
$(`'Turtle*0:trace.mode').d = 40;
```

## 파이썬 코드

```
# 검정색 교차로에서 앞으로 건너가기 (16) | 기다리기 0
__(`'Turtle*0:trace.mode').d = 16
await __(`'Turtle*0:trace.!mode').w()

# 검정색 교차로에서 왼쪽으로 돌기 (24) | 기다리기 0
__(`'Turtle*0:trace.mode').d = 24
await __(`'Turtle*0:trace.!mode').w()

# 검정색 교차로에서 오른쪽으로 돌기 (32) | 기다리기 X
__(`'Turtle*0:trace.mode').d = 32

# 검정색 교차로에서 유턴하기 (40) | 기다리기 X
__(`'Turtle*0:trace.mode').d = 40
```

## 선 따라가기 속도 설정

터틀의 선 따라가기 속도를 설정합니다.

속도의 범위는 1 ~ 10 입니다.

터틀 : 선 따라가기 속도를 5 (으)로 정하기

선 따라가기 속도를 설정합니다. 속도의 범위는 1 ~ 10 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
velocity	입력값	선 따라가기 속도 값	1 ~ 10 사이 정수

## 자바스크립트 코드

```
// 선 따라가기 속도 5 로 정하기  
$('Turtle*0:trace.speed').d = 5;
```

## 파이썬 코드

```
# 선 따라가기 속도 5 로 정하기  
__('Turtle*0:trace.speed').d = 5
```

## 선 따라가기 방향 변화량 설정

터틀의 선 따라가기 방향 변화량을 설정합니다.

변화량의 범위는 1 ~ 10 입니다.

터틀 : 선 따라가기 방향 변화량을 5 (으)로 정하기

선 따라가기 방향 변화량을 설정합니다. 변화량의 범위는 1 ~ 10 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
differential	입력값	선 따라가기 방향 변화량 값	1 ~ 10 사이 정수

## 자바스크립트 코드

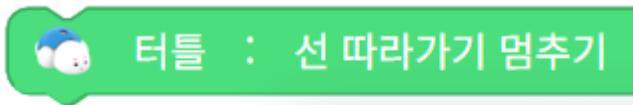
```
$(('Turtle*0:trace.gain')).d = 5; // 선 따라가기 방향 변화량 5로 설정하기
```

## 파이썬 코드

```
--('Turtle*0:trace.gain').d = 5 # 선 따라가기 방향 변화량 5로 설정하기
```

## 선 따라가기 멈추기

터틀의 선 따라가기 기능을 종료합니다.



선 따라가기 기능을 종료합니다.

## 자바스크립트 코드

```
$(('Turtle*0:trace.mode')).d = 0; // 선 따라가기 멈추기
```

## 파이썬 코드

```
--('Turtle*0:trace.mode').d = 0 # 선 따라가기 멈추기
```

## LED 색 설정하기

터틀의 머리 LED 색을 설정합니다.



터틀 : 머리 LED 색을 검정색 ▾ 으로 정하기

✓ 검정색

머리 LED 색을 설정합니다.

빨간색

노란색

초록색

청록색

파란색

자홍색

흰색

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	색상	검정색 ([0, 0, 0]), 빨간색 ([255, 0, 0]), 노란색 ([255, 255, 0]), 초록색 ([0, 255, 0]), 청록색 ([0, 255, 255]), 파란색 ([0, 0, 255]), 자홍색 ([255, 0, 255]), 흰색 ([255, 255, 255])

## 자바스크립트 코드

```
// 머리 LED 색을 검정색으로 정하기
$('Turtle*0:head_led').d = [0, 0, 0];

// 머리 LED 색을 빨간색으로 정하기
$('Turtle*0:head_led').d = [255, 0, 0];

// 머리 LED 색을 노란색으로 정하기
```

```

$(`'Turtle*0:head_led').d = [255, 255, 0];

// 머리 LED 색을 초록색으로 정하기
$(`'Turtle*0:head_led').d = [0, 255, 0];

// 머리 LED 색을 청록색으로 정하기
$(`'Turtle*0:head_led').d = [0, 255, 255];

// 머리 LED 색을 파란색으로 정하기
$(`'Turtle*0:head_led').d = [0, 0, 255];

// 머리 LED 색을 자홍색으로 정하기
$(`'Turtle*0:head_led').d = [255, 0, 255];

// 머리 LED 색을 흰색으로 정하기
$(`'Turtle*0:head_led').d = [255, 255, 255];

```

## 파이썬 코드

```

# 머리 LED 색을 검정색으로 정하기
__(`'Turtle*0:head_led').d = [0, 0, 0]

# 머리 LED 색을 빨간색으로 정하기
__(`'Turtle*0:head_led').d = [255, 0, 0]

# 머리 LED 색을 노란색으로 정하기
__(`'Turtle*0:head_led').d = [255, 255, 0]

# 머리 LED 색을 초록색으로 정하기
__(`'Turtle*0:head_led').d = [0, 255, 0]

# 머리 LED 색을 청록색으로 정하기
__(`'Turtle*0:head_led').d = [0, 255, 255]

# 머리 LED 색을 파란색으로 정하기
__(`'Turtle*0:head_led').d = [0, 0, 255]

# 머리 LED 색을 자홍색으로 정하기
__(`'Turtle*0:head_led').d = [255, 0, 255]

# 머리 LED 색을 흰색으로 정하기
__(`'Turtle*0:head_led').d = [255, 255, 255]

```

## LED 색상 카테고리 블록들로 정하기

색상 카테고리에 있는 블록들로 터틀의 머리 LED 색을 설정합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	입력값	LED 색상	RGB 배열 ([255,255,255])

## 자바스크립트 코드

```
// 머리 LED 색상을 R:255, G:0, B:0 색상으로 정하기
$('Turtle*0:head_led').d = [255, 0, 0];

// 머리 LED 색상을 R:255, G:255, B:0 색상으로 정하기
$('Turtle*0:head_led').d = [255, 255, 0];

// 머리 LED 색상을 무작위 색상으로 정하기
$('Turtle*0:head_led').d = __randomColor();
```

## 파이썬 코드

```
# 머리 LED 색상을 R:255, G:0, B:0 색상으로 정하기
__('Turtle*0:head_led').d = [255, 0, 0]

# 머리 LED 색상을 R:255, G:255, B:0 색상으로 정하기
__('Turtle*0:head_led').d = [255, 255, 0]

# 머리 LED 색상을 무작위 색상으로 정하기
__('Turtle*0:head_led').d = __randomColor()
```

## LED 색 색상 지정 RGB 만큼 변경하기

지정한 R,G,B 값만큼 터틀의 머리 LED 색을 변경합니다.



지정한 R, G, B 값만큼 머리 LED 색을 변경합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	입력값	변경 R,G,B 값	R,G,B 각각 -255~255 사이 정수

## 자바스크립트 코드

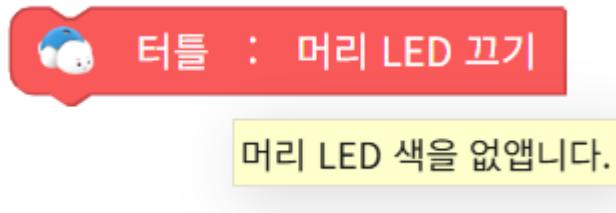
```
// 머리 LED 색을 R : 1, G : 2, B : 3 만큼 바꾸기  
$(('Turtle*0:head_led')).d = [$(('Turtle*0:head_led')).d[0] + 1, $(('Turtle*0:head_led')).d[1] + 2, $(('Turtle*0:head_led')).d[2] + 3];
```

## 파이썬 코드

```
# 머리 LED 색을 R : 1, G : 2, B : 3 만큼 바꾸기  
__($('Turtle*0:head_led')).d = [__($('Turtle*0:head_led')).d[0] + 1, __($('Turtle*0:head_led')).d[1] + 2, __($('Turtle*0:head_led')).d[2] + 3]
```

## LED 끄기

터틀의 머리 LED 색을 없앱니다.



## 자바스크립트 코드

```
// 머리 LED 끄기  
$('Turtle*0:head_led').d = [0, 0, 0];
```

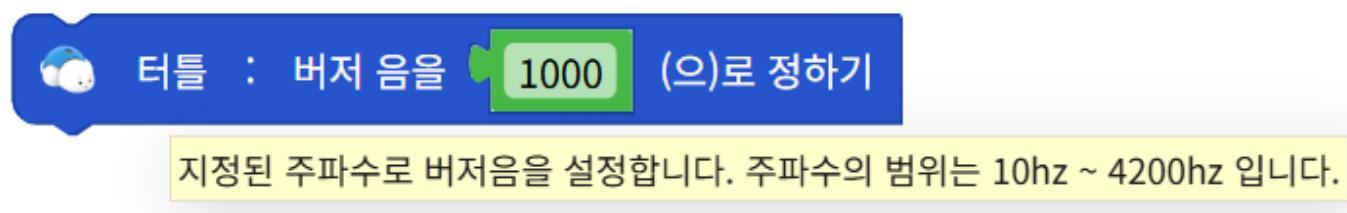
## 파이썬 코드

```
# 머리 LED 끄기  
__($('Turtle*0:head_led')).d = [0, 0, 0]
```

## 버저음 설정하기

지정된 주파수로 터틀의 버저음을 설정합니다.

주파수의 범위는 10hz ~ 4200hz 입니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound	입력값	버저음 주파수	10 ~ 4200(hz)

## 자바스크립트 코드

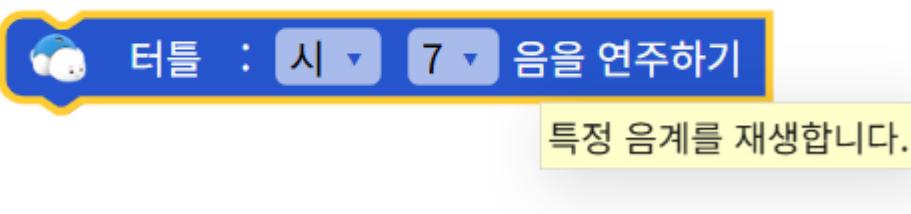
```
// 버저음 주파수 4200hz 로 설정하기
$(`'Turtle*0:sound.buzz').d = 4200;
```

## 파이썬 코드

```
# 버저음 주파수 4200hz 로 설정하기
__(`'Turtle*0:sound.buzz').d = 4200
```

## 음계 연주하기

터틀이 지정된 음계를 재생합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
note	드롭다운 옵션	음계	도 (Do), 도 #(Do#), 레 (Re), 레 #(Re#), 미 (Mi), 파 (Fa), 파 #(Fa#), 솔 (So), 솔 #(So#), 라 (La), 라 #(La#), 시 (Ti)
octave	드롭다운 옵션	옥타브	1 ~ 7

## 자바스크립트 코드

```
// 1 옥타브 도 (Do) 음을 연주하기  
$(`'Turtle*0:sound.note').d = 4;  
  
// 1 옥타브 레 (Re) 음을 연주하기  
$(`'Turtle*0:sound.note').d = 6;  
  
// 2 옥타브 도 (Do) 음을 연주하기  
$(`'Turtle*0:sound.note').d = 16;  
  
// 7 옥타브 시 (Ti) 음을 연주하기  
$(`'Turtle*0:sound.note').d = 87;
```

## 파이썬 코드

```
# 1 옥타브 도 (Do) 음을 연주하기  
__(`'Turtle*0:sound.note').d = 4  
  
# 1 옥타브 레 (Re) 음을 연주하기  
__(`'Turtle*0:sound.note').d = 6  
  
# 2 옥타브 도 (Do) 음을 연주하기  
__(`'Turtle*0:sound.note').d = 16  
  
# 7 옥타브 시 (Ti) 음을 연주하기  
__(`'Turtle*0:sound.note').d = 87
```

## 소리 재생하기

터틀이 특정 사운드 클립을 재생합니다.

기다리기를 체크하면, 재생이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



터틀 : beep ▾ 소리 재생하기 | 기다리기 ✓

- ✓ beep
- beep2
- beep3
- beep\_repeat
- beep\_random
- beep2\_random
- beep3\_random
- beep\_random\_repeat
- siren
- siren\_repeat
- engine

특정 사운드 클립을 재생합니다.

기다리기를 체크하면, 재생이 완료될 때까지 기다립니다.

드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound_clip	드롭다운 옵션	사운드 클립	beep(1), beep2(2), beep3(3), beep_repeat(4), beep_random(5), beep2_random(6), beep3_random(7), beep_random_repeat(8), siren(16), siren_repeat(17), engine(32), engine_repeat(33), noise_random(34), r2d2(48), toy_march(64), birthday(65), dibidibidip(66), mission_complete(67)

## 자바스크립트 코드

```
// dibidibidip(66) 소리 재생하기 | 기다리기 0
$('Turtle*0:sound.clip').d = 66;
await $('Turtle*0:sound.!clip').w();

// mission_complete(67) 소리 재생하기 | 기다리기 X
$('Turtle*0:sound.clip').d = 67;
```

## 파이썬 코드

```
# dibidibidip(66) 소리 재생하기 | 기다리기 0
__('Turtle*0:sound.clip').d = 66
await __('Turtle*0:sound.!clip').w()

# mission_complete(67) 소리 재생하기 | 기다리기 X
__('Turtle*0:sound.clip').d = 67
```

## 소리 끄기

터틀의 소리를 끕니다.



터틀 : 소리 끄기

소리를 끕니다.

## 자바스크립트 코드

```
// 터틀 소리 끄기  
__stopSound('Turtle*0');
```

## 파이썬 코드

```
# 터틀 소리 끄기  
__stopSound('Turtle*0')
```

## 소리가 재생 중인가?

터틀의 소리가 재생중인지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



터틀 : 소리가 재생 중인가?

소리가 재생 중이면 true, 재생 중이 아니면 false를 반환합니다.

## 자바스크립트 코드

```
// 터틀의 소리가 재생 중인가? - 재생 시 true, 아닐시 false  
$('Turtle*0:sound.playing').d;
```

## 파이썬 코드

```
# 터틀의 소리가 재생 중인가? - 재생 시 True, 아닐시 False  
__('Turtle*0:sound.playing').d
```

## 바퀴 속도 값

터틀의 지정한 바퀴 속도 값을 가져옵니다.



왼쪽

특정 바퀴의 속도

오른쪽

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
// 왼쪽 바퀴 속도
__getSpeedInput('Turtle*0', $('Turtle*0:wheel.speed.left').d);

// 오른쪽 바퀴 속도
__getSpeedInput('Turtle*0', $('Turtle*0:wheel.speed.right').d);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도
__getSpeedInput('Turtle*0', __('Turtle*0:wheel.speed.left').d)

# 오른쪽 바퀴 속도
__getSpeedInput('Turtle*0', __('Turtle*0:wheel.speed.right').d)
```

## 컬러 센서 값

터틀의 컬러 센서 값을 가져옵니다.



바닥의 컬러 센서 값

## 자바스크립트 코드

```
// 컬러 센서 값  
$('Turtle*0:line').d;
```

## 파이썬 코드

```
# 컬러 센서 값  
__('Turtle*0:line').d
```

### 카드 색깔 번호 값

터틀 바닥의 컬러 센서를 통해 읽은 카드의 색깔 번호를 반환합니다.



터틀 : 카드 색깔 번호

바닥의 컬러 센서를 통해 읽은 카드의 색깔의 번호

## 자바스크립트 코드

```
// 읽은 카드 색깔 번호 값  
$('Turtle*0:color.name').d;
```

## 파이썬 코드

```
# 읽은 카드 색깔 번호 값  
__('Turtle*0:color.name').d
```

### 카드 색깔 패턴 값

터틀 바닥의 컬러 센서를 통해 읽은 카드의 색깔의 패턴을 반환합니다.



터틀 : 카드 색깔 패턴

바닥의 컬러 센서를 통해 읽은 카드 색깔의 패턴

## 자바스크립트 코드

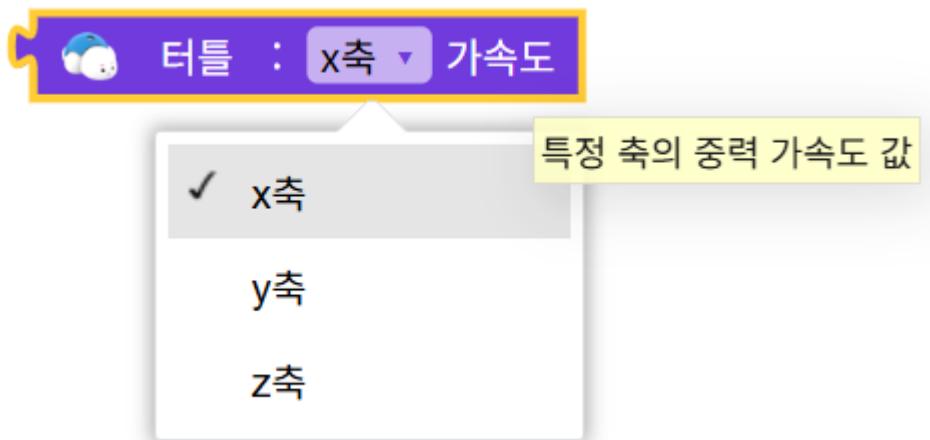
```
// 카드 색깔 패턴 값  
$('Turtle*0:card').d;
```

## 파이썬 코드

```
# 카드 색깔 패턴 값  
__('Turtle*0:card').d
```

## 중력 가속도 값

터틀의 특정 축의 중력 가속도 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	축 기준	x 축, y 축, z 축

## 자바스크립트 코드

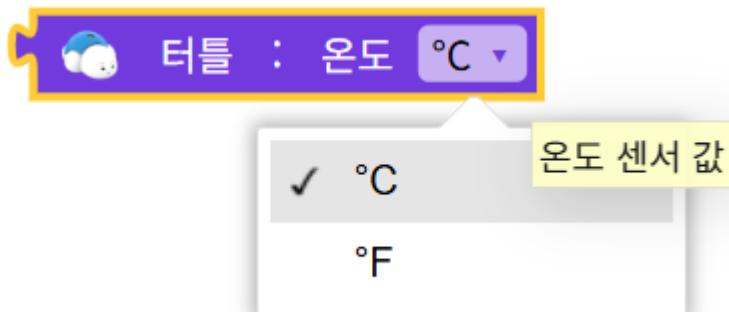
```
// x 축 기준 중력 가속도 값  
$('Turtle*0:acceleration.x').d;  
  
// y 축 기준 중력 가속도 값  
$('Turtle*0:acceleration.y').d;  
  
// z 축 기준 중력 가속도 값  
$('Turtle*0:acceleration.z').d;
```

## 파이썬 코드

```
# x 축 기준 중력 가속도 값  
__('Turtle*0:acceleration.x').d  
  
# y 축 기준 중력 가속도 값  
__('Turtle*0:acceleration.y').d  
  
# z 축 기준 중력 가속도 값  
__('Turtle*0:acceleration.z').d
```

## 온도 센서 값

터틀의 온도 센서 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
unit	드롭다운 옵션	온도 단위	섭씨 (°C), 화씨 (°F)

## 자바스크립트 코드

```
// 섭씨 기준 온도센서 값  
__getTemperature($('Turtle*0:temperature').d, '°C');  
  
// 화씨 기준 온도센서 값  
__getTemperature($('Turtle*0:temperature').d, '°F');
```

## 파이썬 코드

```
# 섭씨 기준 온도센서 값  
__getTemperature(__('Turtle*0:temperature').d, '°C')  
  
# 화씨 기준 온도센서 값  
__getTemperature(__('Turtle*0:temperature').d, '°F')
```

## 신호 세기 값

터틀의 신호 세기 값을 가져옵니다.



터틀 : 신호 세기

신호 세기

## 자바스크립트 코드

```
// 신호 세기 값  
$('Turtle*0:signal_strength').d;
```

## 파이썬 코드

```
# 신호 세기 값  
__('Turtle*0:signal_strength').d
```

## 배터리 충전 상태 값

터틀의 배터리 충전 상태 값을 가져옵니다.



터틀 : 배터리

배터리의 충전 상태 값

## 자바스크립트 코드

```
// 배터리 충전 상태 값  
$('Turtle*0:battery').d;
```

## 파이썬 코드

```
# 배터리 충전 상태 값  
__('Turtle*0:battery').d
```

## 특정 색에 닿았는가?

지정한 특정 색에 닿았는지 터틀 컬러센서를 통해 측정하여 참 (1) / 거짓 (0) (으)로 반환합니다.

터틀 : 빨간색 ▾에 닿았는가?

✓ 빨간색

특정 색에 닿았는지 여부

- ✓ 빨간색
- 노란색
- 초록색
- 청록색
- 파란색
- 자홍색
- 흰색

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	컬러센서 인식 색	빨간색 (1), 노란색 (2), 초록색 (3), 청록색 (4), 파란색 (5), 자홍색 (6), 흰색 (7)

## 자바스크립트 코드

```
// 빨간색에 닿았는가?
$('Turtle*0:color.name').d == 1;

// 노란색에 닿았는가?
$('Turtle*0:color.name').d == 2;

// 초록색에 닿았는가?
$('Turtle*0:color.name').d == 3;

// 청록색에 닿았는가?
$('Turtle*0:color.name').d == 4;

// 파란색에 닿았는가?
$('Turtle*0:color.name').d == 5;

// 자홍색에 닿았는가?
$('Turtle*0:color.name').d == 6;

// 흰색에 닿았는가?
$('Turtle*0:color.name').d == 7;
```

## 파이썬 코드

```
# 빨간색에 닿았는가?  
__('Turtle^0:color.name').d == 1  
  
# 노란색에 닿았는가?  
__('Turtle^0:color.name').d == 2  
  
# 초록색에 닿았는가?  
__('Turtle^0:color.name').d == 3  
  
# 청록색에 닿았는가?  
__('Turtle^0:color.name').d == 4  
  
# 파란색에 닿았는가?  
__('Turtle^0:color.name').d == 5  
  
# 자홍색에 닿았는가?  
__('Turtle^0:color.name').d == 6  
  
# 흰색에 닿았는가?  
__('Turtle^0:color.name').d == 7
```

## 카드 색깔 패턴이 ~ 인가?

컬러센서에 인식 된 카드 색깔의 패턴 일치 여부에 따라 참 (1) / 거짓 (0) (으)로 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pattern 1	드롭다운 옵션	카드 색깔 패턴	빨간색 (red), 노란색 (yellow), 초록색 (green), 청록색 (cyan), 파란색 (blue), 자홍색 (magenta)
pattern 2	드롭다운 옵션	카드 색깔 패턴	빨간색 (red), 노란색 (yellow), 초록색 (green), 청록색 (cyan), 파란색 (blue), 자홍색 (magenta)

## 자바스크립트 코드

```
// 카드 색깔 패턴이 빨간색 빨간색 인가?
$('Turtle*0:card').d == 9;

// 카드 색깔 패턴이 빨간색 노란색 인가?
$('Turtle*0:card').d == 10;

// 카드 색깔 패턴이 빨간색 초록색 인가?
$('Turtle*0:card').d == 11;

// 카드 색깔 패턴이 빨간색 청록색 인가?
$('Turtle*0:card').d == 12;

// 카드 색깔 패턴이 빨간색 파란색 인가?
$('Turtle*0:card').d == 13;

// 카드 색깔 패턴이 빨간색 자홍색 인가?
$('Turtle*0:card').d == 14;

// 카드 색깔 패턴이 노란색 빨간색 인가?
$('Turtle*0:card').d == 17;

// 카드 색깔 패턴이 초록색 빨간색 인가?
$('Turtle*0:card').d == 25;

// 카드 색깔 패턴이 청록색 빨간색 인가?
$('Turtle*0:card').d == 33;

// 카드 색깔 패턴이 파란색 빨간색 인가?
$('Turtle*0:card').d == 41;

// 카드 색깔 패턴이 자홍색 빨간색 인가?
$('Turtle*0:card').d == 49;

// 카드 색깔 패턴이 자홍색 자홍색 인가?
$('Turtle*0:card').d == 54;
```

## 파이썬 코드

```
# 카드 색깔 패턴이 빨간색 빨간색 인가?
__('Turtle*0:card').d == 9

# 카드 색깔 패턴이 빨간색 노란색 인가?
__('Turtle*0:card').d == 10
```

```

# 카드 색깔 패턴이 빨간색 초록색 인가?
--('Turtle*0:card').d == 11

# 카드 색깔 패턴이 빨간색 청록색 인가?
--('Turtle*0:card').d == 12

# 카드 색깔 패턴이 빨간색 파란색 인가?
--('Turtle*0:card').d == 13

# 카드 색깔 패턴이 빨간색 자홍색 인가?
--('Turtle*0:card').d == 14

# 카드 색깔 패턴이 노란색 빨간색 인가?
--('Turtle*0:card').d == 17

# 카드 색깔 패턴이 초록색 빨간색 인가?
--('Turtle*0:card').d == 25

# 카드 색깔 패턴이 청록색 빨간색 인가?
--('Turtle*0:card').d == 33

# 카드 색깔 패턴이 파란색 빨간색 인가?
--('Turtle*0:card').d == 41

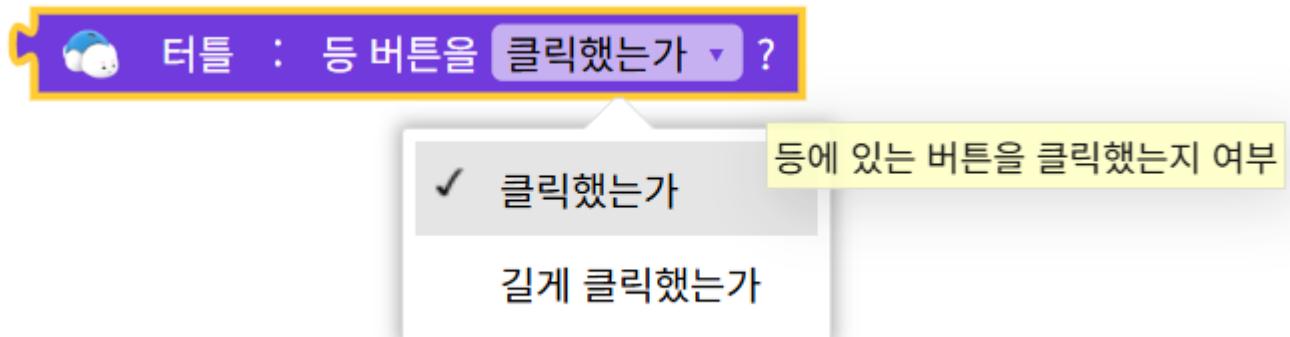
# 카드 색깔 패턴이 자홍색 빨간색 인가?
--('Turtle*0:card').d == 49

# 카드 색깔 패턴이 자홍색 자홍색 인가?
--('Turtle*0:card').d == 54

```

## 등 버튼을 클릭했는가?

등 버튼 클릭 여부에 따라 참 (1) / 거짓 (0) (으)로 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
action	드롭다운 옵션	버튼 클릭 방식	클릭했는가 (click), 길게 클릭했는가 (long_click)

## 자바스크립트 코드

```
// 등 버튼을 클릭했는가?  
$('Turtle*0:button.click').e;  
  
// 등 버튼을 길게 클릭했는가?  
$('Turtle*0:button.long_click').e;
```

## 파이썬 코드

```
# 등 버튼을 클릭했는가?  
__('Turtle*0:button.click').e  
  
# 등 버튼을 길게 클릭했는가?  
__('Turtle*0:button.long_click').e
```

## 상태 변경 여부

터틀의 상태 변경 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
condition	드롭다운 옵션	위치 조건	앞으로 기울였는가, 뒤로 기울였는가, 왼쪽으로 기울였는가, 오른쪽으로 기울였는가, 거꾸로 뒤집어졌는가, 뒤집어지지 않았는가

## 자바스크립트 코드

```
// 터틀이 앞으로 기울였는가?
$('Turtle*0:acceleration.x').d > 5000;

// 터틀이 뒤로 기울였는가?
$('Turtle*0:acceleration.x').d < -5000;

// 터틀이 왼쪽으로 기울였는가?
$('Turtle*0:acceleration.y').d > 5000;

// 터틀이 오른쪽으로 기울였는가?
$('Turtle*0:acceleration.y').d < -5000;

// 터틀이 거꾸로 뒤집어졌는가?
$('Turtle*0:acceleration.z').d > 0;

// 터틀이 뒤집어지지 않았는가?
$('Turtle*0:acceleration.z').d < -3000;
```

## 파이썬 코드

```
# 터틀이 앞으로 기울였는가?
__('Turtle*0:acceleration.x').d > 5000

# 터틀이 뒤로 기울였는가?
__('Turtle*0:acceleration.x').d < -5000

# 터틀이 왼쪽으로 기울였는가?
__('Turtle*0:acceleration.y').d > 5000

# 터틀이 오른쪽으로 기울였는가?
__('Turtle*0:acceleration.y').d < -5000

# 터틀이 거꾸로 뒤집어졌는가?
__('Turtle*0:acceleration.z').d > 0

# 터틀이 뒤집어지지 않았는가?
__('Turtle*0:acceleration.z').d < -3000
```

## 함수

함수 (Function) 는 특정 작업을 수행하는 블록 (명령어) 들의 모음입니다.

반복적으로 사용되는 동작을 하나의 함수로 정의하면, 코드를 간결하고 효율적으로 관리할 수 있습니다.

함수는 다음과 같은 특징을 가집니다.

- **재사용 가능:** 한 번 정의하면 여러 번 호출하여 사용할 수 있습니다.
- **입력과 출력:** 매개변수 (입력 값)를 받아 처리한 후 결과 (출력 값)를 반환할 수 있습니다.
- **코드의 가독성 향상:** 프로그램의 흐름을 논리적으로 구성할 수 있습니다.

## 함수 블록

블록 코딩에서는 **함수 블록**을 활용하여 사용자가 직접 원하는 기능을 정의할 수 있습니다.

### 함수 정의 블록

함수를 정의하는 블록을 사용하면 **새로운 함수**를 만들 수 있습니다.

아래 예제는 `func` 이라는 이름의 함수를 정의하는 방법을 보여줍니다.

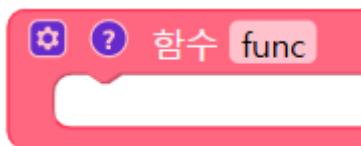


Figure 141: Image

## Javascript 코드

```
function func() { // func 함수 정의  
}
```

## Python 코드

```
def func(): # func 함수 정의  
    pass
```

### 매개변수를 가지는 함수

함수 블록에는 기어 버튼이 있으며, 이를 클릭하여 **매개변수 (입력 값)**를 추가할 수 있습니다.

- **매개변수 (Parameter)**는 함수에 전달하는 입력 값입니다.

### 매개변수 설정 방법

1. 매개 변수 블록에서 필요한 변수 (`x`, `y` 등)를 추가합니다.
2. 오른쪽 매개변수 목록에 연결하면, 매개변수를 가지는 함수가 생성됩니다.



Figure 142: Image



Figure 143: Image

매개변수 x, y 를 가지는 함수 `xyFunction` 을 정의하고, 내부에서 매개변수를 활용할 수 있습니다.

## Javascript 코드

```
var x, y;

function xyFunction(x, y) { // 매개변수 x, y 를 가지는 xyFunction 정의
}
```

## Python 코드

```
x = None
y = None

def xyFunction(x, y): # 매개변수 x, y 를 가지는 xyFunction 정의
    pass
```

## 함수 호출 (사용자 정의 함수 블록 생성)

함수를 정의하면, 사용자 정의 함수 블록이 자동으로 생성됩니다.

이를 통해 미리 정의한 함수를 호출하여 실행할 수 있습니다.

다음은 이전에 생성한 `xyFunction` 함수를 호출한 블록입니다.

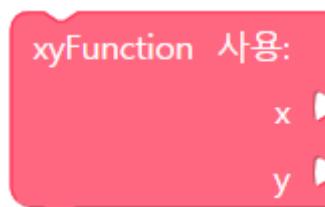


Figure 144: Image

## Javascript 코드

```
var x, y;

function xyFunction(x, y) {
}

xyFunction(null, null); // 함수 호출
```

## Python 코드

```
x = None
y = None

def xyFunction(x, y):
    pass
```

```
xyFunction(None, None) # 함수 호출
```

또한, 아이콘을 클릭하면 함수에 대한 **설명 (주석)**을 추가할 수 있어, 함수를 쉽게 설명할 수 있습니다.

아래 그림은 함수의 **주석**을 추가하는 예시입니다.



Figure 145: Image

## Javascript 코드

```
// 이 함수를 설명하세요...
function func() {
}
```

## Python 코드

```
# 이 함수를 설명하세요...
def func():
    pass
```

## 반환값 (Return) 이 없는 함수

반환값이 없는 함수는 특정 동작을 수행하지만 **값을 반환하지 않는 함수**입니다.

다음은 “hello”를 출력하는 함수입니다. 이 함수는 “hello”를 출력하기만 하고, 특정한 값을 반환하지 않습니다.



Figure 146: Image

## Javascript 코드

```
function func() { // 반환값이 없는 함수 정의
    window.alert('hello');
}
```

## Python 코드

```
def func(): # 반환값이 없는 함수 정의
    print('hello')
```

아래 블록을 실행하면, "hello" 가 화면에 출력됩니다.



Figure 147: Image

## Javascript 코드

```
function func() {
    window.alert('hello');
}
func(); // 반환값이 없는 함수 호출
```

## Python 코드

```
def func():
    print('hello') # 반환값이 없는 함수 호출
func()
```

## 반환값 (Return) 이 있는 함수

반환값이 있는 함수는 특정 작업을 수행한 후, 결과 값을 반환하여 다른 블록에서 활용할 수 있습니다.

아래 함수는 두 숫자의 합을 반환하는 함수입니다. 매개변수  $x$ ,  $y$ 를 입력받아  $x + y$ 의 결과를 반환합니다.



Figure 148: Image

## Javascript 코드

```
var x, y, sum;

function xyFunction(x, y) { // 반환값이 있는 함수 정의
    sum = x + y;
    return sum;
}
```

## Python 코드

```
x = None
y = None
sum2 = None # sum 은 예약어 이기 때문에 sum2 사용

def xyFunction(x, y): # 반환값이 있는 함수 정의
    global sum2
    sum2 = x + y
    return sum2
```

아래 함수를 호출하여 결과를 result 변수에 저장하고 사용할 수 있습니다.

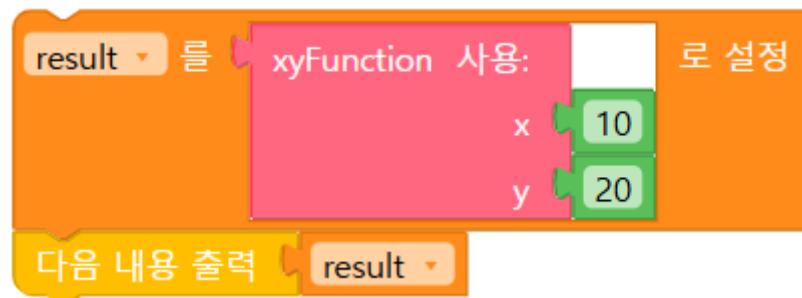


Figure 149: Image

함수 실행 예시 ( $10 + 20 = 30$  반환)

$_{30}$  이 출력됩니다.

## Javascript 코드

```
var x, y, result, sum;

function xyFunction(x, y) {
    sum = x + y;
    return sum;
}

result = xyFunction(10, 20); // 반환값이 있는 함수 호출
window.alert(result);
```

## Python 코드

```

x = None
y = None
result = None
sum2 = None # sum 은 예약어 이기 때문에 sum2 사용

def xyFunction(x, y): # 반환값이 있는 함수 호출
    global result, sum2
    sum2 = x + y
    return sum2

result = xyFunction(10, 20)
print(result)

```

## 특정 조건에서 값을 반환하는 블록 (만약 다음을 돌려줌 블록)

함수 내에서 특정 조건을 만족하면 즉시 값을 반환하고 함수를 종료하는 기능을 수행하는 블록입니다.

이 블록은 함수 내부에서만 사용할 수 있으며, 다른 곳에서는 비활성화됩니다.



Figure 150: Image

x 와 y 의 합을 반환하지만, 합이 0 보다 작으면 -1 을 반환하는 함수입니다.

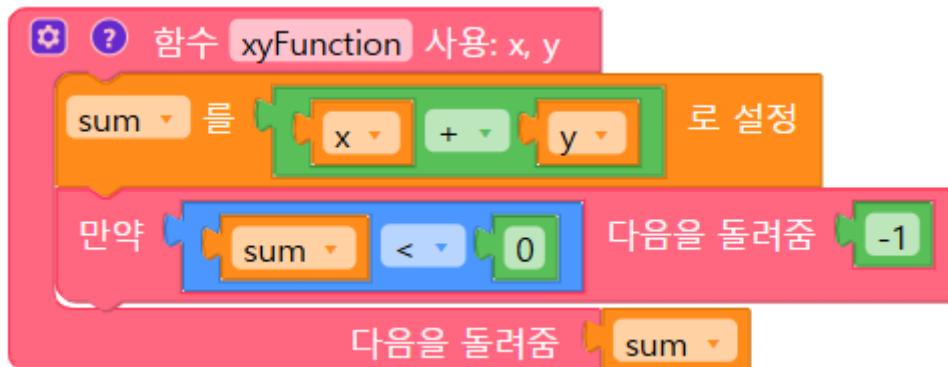


Figure 151: Image

## Javascript 코드

```

var x, y, sum;

function xyFunction(x, y) {
    sum = x + y;
    if (sum < 0) { // sum < 0 일 시, -1 반환
        return -1;
    }
    return sum;
}

```

## Python 코드

```
x = None
y = None
sum2 = None # sum 은 예약어 이기 때문에 sum2 사용

def xyFunction(x, y):
    global sum2
    sum2 = x + y
    if sum2 < 0: # sum < 0 일 시, -1 반환
        return -1
    return sum2
```

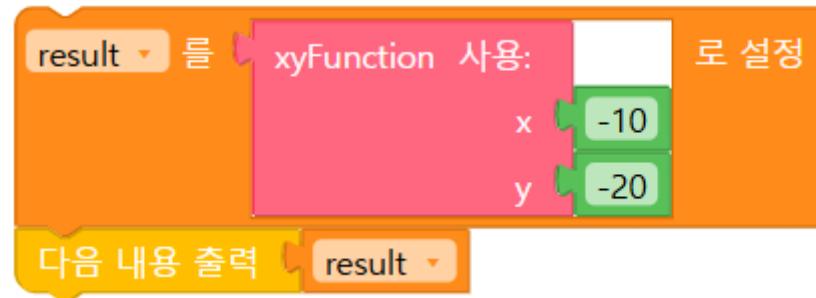


Figure 152: Image

-1이 출력됩니다.

## Javascript 코드

```
var x, y, result, sum;

function xyFunction(x, y) {
    sum = x + y;
    if (sum < 0) {
        return -1;
    }
    return sum;
}

result = xyFunction(-10, -20); // (-10) + (-20) < 0 이므로 -1 반환
window.alert(result);
```

## Python 코드

```
x = None
y = None
result = None
sum2 = None # sum 은 예약어 이기 때문에 sum2 사용

def xyFunction(x, y):
    global result, sum2
    sum2 = x + y
    if sum2 < 0:
        return -1
    return sum2

result = xyFunction(-10, -20) # (-10) + (-20) < 0 이므로 -1 반환
print(result)
```

## 햄스터

### 블록

#### 바퀴 속도 설정하기

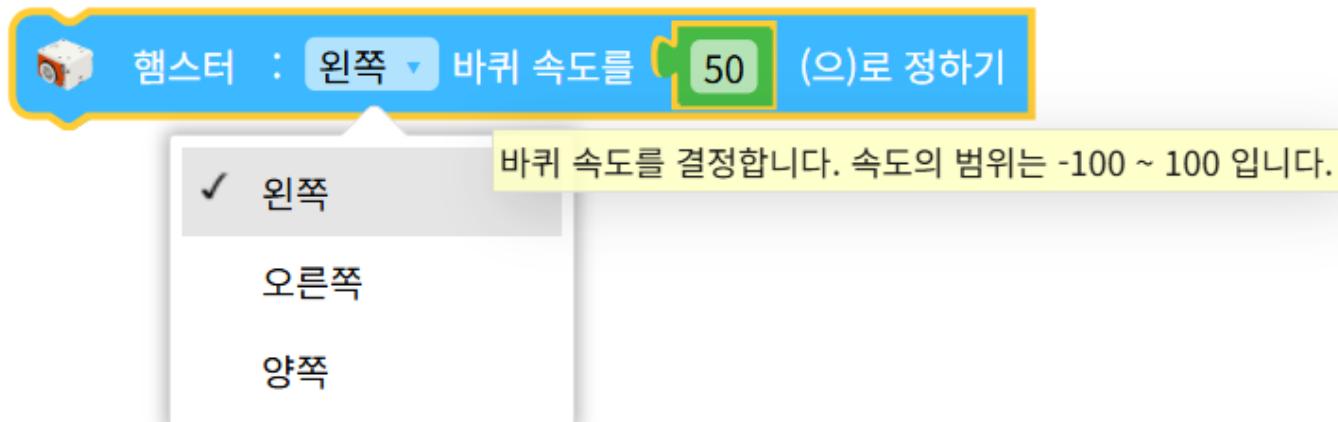
햄스터의 바퀴 속도를 설정합니다.

바퀴 속도가 양수이면 앞쪽 방향으로 회전하고, 바퀴 속도가 음수이면 뒤쪽 방향으로 회전합니다.

예를 들어, 바퀴 속도가 100 이라면, 앞쪽 방향으로 100 의 속도로 회전하고,

바퀴 속도가 -100 이라면, 뒤쪽 방향으로 100 의 속도로 회전합니다.

한번 바퀴 속도를 설정하면, 다시 바퀴 속도를 설정하기 전까지 해당 속도로 햄스터가 이동합니다.



#### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	바퀴 속도	-100 ~ 100 정수, 0: 정지

#### 자바스크립트 코드

```
// 햄스터 왼쪽 바퀴 속도를 50 으로 정하기
$(Hamster*0:wheel.speed.left').d = __getSpeed('Hamster*0', 50); // (robot, speed)

// 햄스터 오른쪽 바퀴 속도를 30 으로 정하기
$(Hamster*0:wheel.speed.right').d = __getSpeed('Hamster*0', 30); // (robot, speed)

// 햄스터 양쪽 바퀴 속도를 -10 으로 정하기
```

```

$(Hamster*0:wheel.speed.left').d = __getSpeed('Hamster*0', -10); // (robot, speed)
$(Hamster*0:wheel.speed.right').d = __getSpeed('Hamster*0', -10); // (robot, speed)

```

## 파이썬 코드

```

# 햄스터 左 바퀴 속도를 50 으로 정하기
__('Hamster*0:wheel.speed.left').d = __getSpeed('Hamster*0', 50) # (robot, speed)

# 햄스터 오른쪽 바퀴 속도를 30 으로 정하기
__('Hamster*0:wheel.speed.right').d = __getSpeed('Hamster*0', 30) # (robot, speed)

# 햄스터 양쪽 바퀴 속도를 -10 으로 정하기
__('Hamster*0:wheel.speed.left').d = __getSpeed('Hamster*0', -10) # (robot, speed)
__('Hamster*0:wheel.speed.right').d = __getSpeed('Hamster*0', -10) # (robot, speed)

```

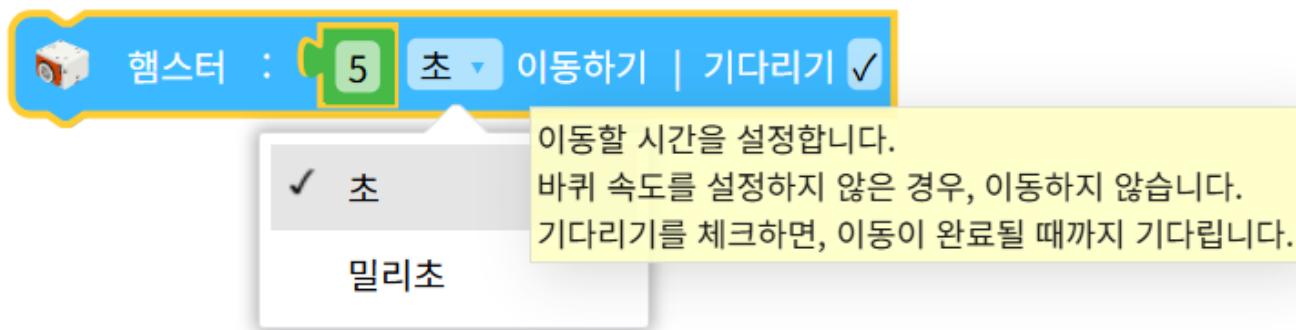
## 시간 이동하기

햄스터가 이동할 시간을 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
time	입력값	시간 값	0 이상 실수
unit	드롭다운 옵션	시간 단위	초 (seconds), 밀리초 (milliseconds)

옵션을 밀리초 (milliseconds)로 설정한 경우에는, `time` 값을 1000으로 나눈 값이 입력됩니다.

## 자바스크립트 코드

```
// 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Hamster*0', 5, true); // (robot, time, wait_w)

// 1 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Hamster*0', 0.001, false); // (robot, time, wait_w)
```

## 파이썬 코드

```
# 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('Hamster*0', 5, True) # (robot, time, wait_w)

# 1 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('Hamster*0', 0.001, False) # (robot, time, wait_w)
```

## 바퀴 속도 변경하기

햄스터의 바퀴 속도를 변경합니다.

현재의 바퀴 속도에 입력한 속도를 더한 값이 새로운 바퀴 속도가 됩니다.

새롭게 설정된 바퀴 속도의 범위는 -100 ~ 100으로 설정됩니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	현재 바퀴 속도에 더할 속도 값	-200 ~ 200 정수, 0: 정지

## 자바스크립트 코드

```
// 햄스터의 왼쪽 바퀴 속도를 50 만큼 바꾸기
$(`Hamster*0:wheel.speed.left`).d = $($(`Hamster*0:wheel.speed.left`).d + __getSpeed('Hamster*0', 50));

// 햄스터의 오른쪽 바퀴 속도를 10 만큼 바꾸기
$(`Hamster*0:wheel.speed.right`).d = $($(`Hamster*0:wheel.speed.right`).d + __getSpeed('Hamster*0', 10));

// 햄스터의 양쪽 바퀴 속도를 -20 만큼 바꾸기
$(`Hamster*0:wheel.speed.left`).d = $($(`Hamster*0:wheel.speed.left`).d + __getSpeed('Hamster*0', (-20)));
$(`Hamster*0:wheel.speed.right`).d = $($(`Hamster*0:wheel.speed.right`).d + __getSpeed('Hamster*0', (-20)));



```

## 파이썬 코드

```
# 햄스터의 왼쪽 바퀴 속도를 50 만큼 바꾸기
__(`Hamster*0:wheel.speed.left`).d = __(`Hamster*0:wheel.speed.left`).d + __getSpeed('Hamster*0', 50)

# 햄스터의 오른쪽 바퀴 속도를 10 만큼 바꾸기
__(`Hamster*0:wheel.speed.right`).d = __(`Hamster*0:wheel.speed.right`).d + __getSpeed('Hamster*0', 10)

# 햄스터의 양쪽 바퀴 속도를 -20 만큼 바꾸기
__(`Hamster*0:wheel.speed.left`).d = __(`Hamster*0:wheel.speed.left`).d + __getSpeed('Hamster*0', (-20))
__(`Hamster*0:wheel.speed.right`).d = __(`Hamster*0:wheel.speed.right`).d + __getSpeed('Hamster*0', (-20))
```

## 정지하기

햄스터의 이동을 멈춥니다.

햄스터의 양쪽 바퀴 속도가 모두 0으로 초기화됩니다.



## 자바스크립트 코드

```
__stopMove('Hamster*0');
```

## 파이썬 코드

```
__stopMove('Hamster*0')
```

## 말판 앞으로 한 칸 이동하기

햄스터가 말판 위에서 한 칸 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 햄스터 : 말판 앞으로 한 칸 이동하기 | 기다리기

말판 위에서 정해진 대로 한 칸씩 움직입니다.

### 자바스크립트 코드

```
// 말판 앞으로 한 칸 이동하기 | 기다리기 0
await __grid_move_forward('Hamster*0', true); // (robot, wait_w)

// 말판 앞으로 한 칸 이동하기 | 기다리기 X
__grid_move_forward('Hamster*0', false); // (robot, wait_w)
```

### 파이썬 코드

```
# 말판 앞으로 한 칸 이동하기 | 기다리기 0
await __grid_move_forward('Hamster*0', True) # (robot, wait_w)

# 말판 앞으로 한 칸 이동하기 | 기다리기 X
__grid_move_forward('Hamster*0', False) # (robot, wait_w)
```

### 말판에서 한번 돌기

말판 위 햄스터가 입력받은 방향으로 90 도 회전합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 햄스터 : 말판 **왼쪽** 으로 한 번 돌기 | 기다리기 ✓

말판 위에서 정해진 방향으로 90도 회전합니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	회전 방향	왼쪽 (left), 오른쪽 (right)

### 자바스크립트 코드

```

// 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_left('Hamster*0', true); // (robot, wait_w)

// 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 X
__grid_turn_left('Hamster*0', false); // (robot, wait_w)

// 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_right('Hamster*0', true); // (robot, wait_w)

// 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 X
__grid_turn_right('Hamster*0', false); // (robot, wait_w)

```

## 파이썬 코드

```

# 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_left('Hamster*0', True) # (robot, wait_w)

# 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 X
__grid_turn_left('Hamster*0', False) # (robot, wait_w)

# 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_right('Hamster*0', True) # (robot, wait_w)

# 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 X
__grid_turn_right('Hamster*0', False) # (robot, wait_w)

```

## 센서로 선 따라가기

햄스터가 바닥 센서를 이용하여 특정한 선을 따라갑니다.



바닥 센서를 이용해 특정한 색의 선을 따라 이동합니다.

왼쪽

오른쪽

가운데

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	바닥 센서 방향	왼쪽 (left), 오른쪽 (right), 가운데 (middle)
color	드롭다운 옵션	선의 색	검정색 (black), 흰색 (white)

## 자바스크립트 코드

```
$('Hamster*0:wheel.trace.mode').d = 1; // 왼쪽 바닥 센서로 검정색 선을 따라가기
$('Hamster*0:wheel.trace.mode').d = 2; // 오른쪽 바닥 센서로 검정색 선을 따라가기
$('Hamster*0:wheel.trace.mode').d = 3; // 가운데 바닥 센서로 검정색 선을 따라가기
$('Hamster*0:wheel.trace.mode').d = 8; // 왼쪽 바닥 센서로 흰색 선을 따라가기
$('Hamster*0:wheel.trace.mode').d = 9; // 오른쪽 바닥 센서로 흰색 선을 따라가기
$('Hamster*0:wheel.trace.mode').d = 10; // 가운데 바닥 센서로 흰색 선을 따라가기
```

## 파이썬 코드

```
--('Hamster*0:wheel.trace.mode').d = 1 # 왼쪽 바닥 센서로 검정색 선을 따라가기
--('Hamster*0:wheel.trace.mode').d = 2 # 오른쪽 바닥 센서로 검정색 선을 따라가기
--('Hamster*0:wheel.trace.mode').d = 3 # 가운데 바닥 센서로 검정색 선을 따라가기
--('Hamster*0:wheel.trace.mode').d = 8 # 왼쪽 바닥 센서로 흰색 선을 따라가기
--('Hamster*0:wheel.trace.mode').d = 9 # 오른쪽 바닥 센서로 흰색 선을 따라가기
--('Hamster*0:wheel.trace.mode').d = 10 # 가운데 바닥 센서로 흰색 선을 따라가기
```

## 회전 후 선 따라가기 & 교차로에서 멈추기

햄스터가 지정한 방향으로 회전한 뒤, 다음 교차로를 만날 때까지 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



✓ 좌회전

우회전

전진

유턴

지정한 방향으로 회전한 뒤, 다음 교차로를 만날 때까지 이동합니다.  
기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	이동 방향	좌회전 (left), 우회전 (right), 전진 (forward), 유턴 (back)
color	드롭다운 옵션	선의 색	검정색 (black), 흰색 (white)

## 자바스크립트 코드

```
// 좌회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
$('Hamster*0:wheel.trace.mode').d = 4;
await $('Hamster*0:wheel.trace.!mode').w();

// 우회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
$('Hamster*0:wheel.trace.mode').d = 5;

// 전진 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
$('Hamster*0:wheel.trace.mode').d = 6;
await $('Hamster*0:wheel.trace.!mode').w();

// 유턴 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
$('Hamster*0:wheel.trace.mode').d = 7;

// 좌회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
$('Hamster*0:wheel.trace.mode').d = 11;
await $('Hamster*0:wheel.trace.!mode').w();

// 우회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
$('Hamster*0:wheel.trace.mode').d = 12;

// 전진 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
$('Hamster*0:wheel.trace.mode').d = 13;
await $('Hamster*0:wheel.trace.!mode').w();

// 유턴 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
$('Hamster*0:wheel.trace.mode').d = 14;
```

## 파이썬 코드

```
# 좌회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
__('Hamster*0:wheel.trace.mode').d = 4
await __('Hamster*0:wheel.trace.!mode').w()

# 우회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
__('Hamster*0:wheel.trace.mode').d = 5

# 전진 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
__('Hamster*0:wheel.trace.mode').d = 6
await __('Hamster*0:wheel.trace.!mode').w()

# 유턴 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
__('Hamster*0:wheel.trace.mode').d = 7

# 좌회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
__('Hamster*0:wheel.trace.mode').d = 11
await __('Hamster*0:wheel.trace.!mode').w()
```

```

# 우회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
__('Hamster*0:wheel.trace.mode').d = 12

# 전진 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
__('Hamster*0:wheel.trace.mode').d = 13
await __('Hamster*0:wheel.trace.!mode').w()

# 유턴 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
__('Hamster*0:wheel.trace.mode').d = 14

```

## 선 따라가기 속도 설정

햄스터의 선 따라가기 속도를 설정합니다.

속도의 범위는 1 ~ 10 입니다.



**햄스터 : 선 따라가기 속도를 5 (으)로 정하기**

선 따라가기 속도를 설정합니다. 속도의 범위는 1 ~ 10 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
velocity	입력값	선 따라가기 속도 값	1 ~ 10 사이 정수

## 자바스크립트 코드

```
$('Hamster*0:wheel.trace.speed').d = 5; // 선 따라가기 속도 5 로 설정하기
```

## 파이썬 코드

```
__('Hamster*0:wheel.trace.speed').d = 5 # 선 따라가기 속도 5 로 설정하기
```

## 선 따라가기 멈추기

햄스터의 선 따라가기 기능을 종료합니다.



## 햄스터 : 선 따라가기 멈추기

선 따라가기 기능을 종료합니다.

### 자바스크립트 코드

```
$(('Hamster*0:wheel.trace.mode').d = 0; // 선 따라가기 멈추기
```

### 파이썬 코드

```
--('Hamster*0:wheel.trace.mode').d = 0 # 선 따라가기 멈추기
```

### LED 색 설정하기

햄스터의 LED 색을 설정합니다.

왼쪽, 오른쪽 또는 양쪽의 색을 설정할 수 있습니다.



## 햄스터 : 왼쪽 ▾ LED 색을 검정색 ▾ 으로 정하기

✓ 왼쪽

오른쪽

양쪽

LED 색을 설정합니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (both)

이름	구분	설명	범위 / 종류
color	드롭다운 옵션	색상	검정색 (black, 0), 파란색 (blue, 1), 초록색 (green, 2), 청록색 (cyan, 3), 빨간색 (red, 4), 자홍색 (magenta, 5), 노란색 (yellow, 6), 흰색 (white, 7)

## 자바스크립트 코드

```
// 左側 LED 색을 빨간색 (4) 으로 정하기
$(Hamster*0:led.left').d = 4;

// 오른쪽 LED 색을 초록색 (2) 으로 정하기
$(Hamster*0:led.right').d = 2;

// 양쪽 LED 색을 파란색 (1) 으로 정하기
$(Hamster*0:led.left').d = 1;
$(Hamster*0:led.right').d = 1;
```

## 파이썬 코드

```
# 左쪽 LED 색을 빨간색 (4) 으로 정하기
__('Hamster*0:led.left').d = 4

# 오른쪽 LED 색을 초록색 (2) 으로 정하기
__('Hamster*0:led.right').d = 2

# 양쪽 LED 색을 파란색 (1) 으로 정하기
__('Hamster*0:led.left').d = 1
__('Hamster*0:led.right').d = 1
```

## LED 끄기

햄스터의 LED 색을 없앱니다.

왼쪽, 오른쪽 또는 양쪽의 LED 를 끌 수 있습니다.



LED 색을 없앱니다.

✓ 왼쪽

오른쪽

양쪽

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)

## 자바스크립트 코드

```
// 왼쪽 LED 끄기
$('Hamster*0:led.left').d = 0;

// 오른쪽 LED 끄기
$('Hamster*0:led.right').d = 0;

// 양쪽 LED 끄기
$('Hamster*0:led.left').d = 0;
$('Hamster*0:led.right').d = 0;
```

## 파이썬 코드

```
# 왼쪽 LED 끄기
__('Hamster*0:led.left').d = 0

# 오른쪽 LED 끄기
__('Hamster*0:led.right').d = 0

# 양쪽 LED 끄기
__('Hamster*0:led.left').d = 0
__('Hamster*0:led.right').d = 0
```

## 버저음 설정하기

지정된 주파수로 햄스터의 버저음을 설정합니다.

주파수의 범위는 10hz ~ 4200hz 입니다.



햄스터 : 버저 음을 1000 (으)로 정하기

지정된 주파수로 버저음을 설정합니다. 주파수의 범위는 10hz ~ 4200hz 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound	입력값	버저음 주파수	10 ~ 4200(hz)

## 자바스크립트 코드

```
// 버저음 주파수 4200hz 로 설정하기
$('Hamster*0: sound.buzz').d = 4200;
```

## 파이썬 코드

```
# 버저음 주파수 4200hz 로 설정하기
__('Hamster*0: sound.buzz').d = 4200
```

## 음계 연주하기

햄스터가 지정된 음계를 재생합니다.



특정 음계를 재생합니다.

✓ 도

도#(레 ↞ )

레

레#(미 ↞ )

미

파

파#(솔 ↞ )

솔

솔#(라 ↞ )

라

#### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
note	드롭다운 옵션	음계	도 (Do), 도 #(Do#), 레 (Re), 레 #(Re#), 미 (Mi), 파 (Fa), 파 #(Fa#), 솔 (So), 솔 #(So#), 라 (La), 라 #(La#), 시 (Ti)
octave	드롭다운 옵션	옥타브	1 ~ 7

## 자바스크립트 코드

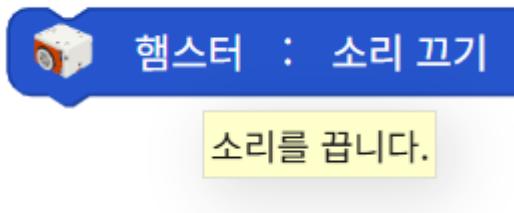
```
// 1 옥타브 도 (Do) 음을 연주하기  
$(‘Hamster*0:sound.note’).d = 4;  
  
// 1 옥타브 레 (Re) 음을 연주하기  
$(‘Hamster*0:sound.note’).d = 6;  
  
// 2 옥타브 도 (Do) 음을 연주하기  
$(‘Hamster*0:sound.note’).d = 16;  
  
// 7 옥타브 시 (Ti) 음을 연주하기  
$(‘Hamster*0:sound.note’).d = 87;
```

## 파이썬 코드

```
# 1 옥타브 도 (Do) 음을 연주하기  
__('Hamster*0:sound.note').d = 4  
  
# 1 옥타브 레 (Re) 음을 연주하기  
__('Hamster*0:sound.note').d = 6  
  
# 2 옥타브 도 (Do) 음을 연주하기  
__('Hamster*0:sound.note').d = 16  
  
# 7 옥타브 시 (Ti) 음을 연주하기  
__('Hamster*0:sound.note').d = 87
```

## 소리 끄기

햄스터의 소리를 끕니다.



## 자바스크립트 코드

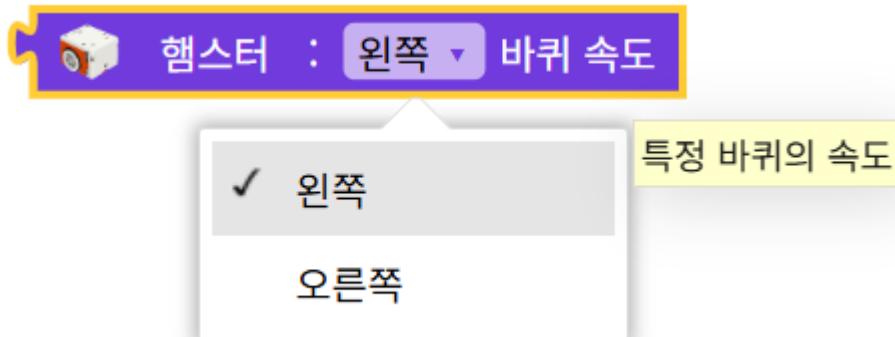
```
// 햄스터 소리 끄기  
__stopSound(‘Hamster*0’);
```

## 파이썬 코드

```
# 햄스터 소리 끄기  
__stopSound(‘Hamster*0’)
```

## 바퀴 속도 값

햄스터의 지정한 바퀴 속도 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
//왼쪽 바퀴 속도  
__getSpeedInput('Hamster*0', $('Hamster*0:wheel.speed.left').d);  
  
//오른쪽 바퀴 속도  
__getSpeedInput('Hamster*0', $('Hamster*0:wheel.speed.right').d);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도  
__getSpeedInput('Hamster*0', __('Hamster*0:wheel.speed.left').d)  
  
# 오른쪽 바퀴 속도  
__getSpeedInput('Hamster*0', __('Hamster*0:wheel.speed.right').d)
```

## 근접 센서 값

햄스터의 지정한 근접 센서 값을 가져옵니다.



✓ 왼쪽

특정 근접 센서의 값

오른쪽

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
//왼쪽 근접 센서 값
$(Hamster*0:proximity.left').d;

//오른쪽 근접 센서 값
$(Hamster*0:proximity.right').d;
```

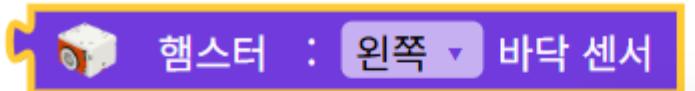
## 파이썬 코드

```
# 왼쪽 근접 센서 값
__('Hamster*0:proximity.left').d

# 오른쪽 근접 센서 값
__('Hamster*0:proximity.right').d
```

## 바닥 센서 값

햄스터의 지정한 바닥 센서 값을 가져옵니다.



✓ 왼쪽

특정 바닥 센서의 값

오른쪽

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
//왼쪽 바닥 센서 값
$(Hamster*0:floor.left').d;

//오른쪽 바닥 센서 값
$(Hamster*0:floor.right').d;
```

## 파이썬 코드

```
# 왼쪽 바닥 센서 값
__($('Hamster*0:floor.left').d

# 오른쪽 바닥 센서 값
__($('Hamster*0:floor.right').d
```

## 중력 가속도 값

햄스터의 특정 축의 중력 가속도 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	축 기준	x 축, y 축, z 축

## 자바스크립트 코드

```
// x 축 기준 중력 가속도 값
$(('Hamster*0:acceleration.x')).d;

// y 축 기준 중력 가속도 값
$(('Hamster*0:acceleration.y')).d;

// z 축 기준 중력 가속도 값
$(('Hamster*0:acceleration.z')).d;
```

## 파이썬 코드

```
# x 축 기준 중력 가속도 값
___('Hamster*0:acceleration.x').d

# y 축 기준 중력 가속도 값
___('Hamster*0:acceleration.y').d

# z 축 기준 중력 가속도 값
___('Hamster*0:acceleration.z').d
```

## 밝기 센서 값

햄스터의 밝기 센서 값을 가져옵니다.



밝기 센서 값

## 자바스크립트 코드

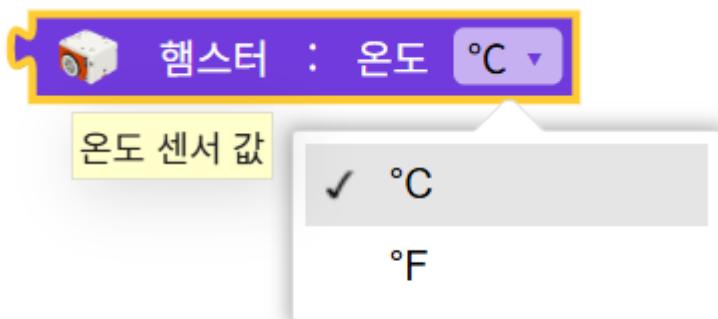
```
// 밝기 센서 값
$(Hamster*0:Light).d;
```

## 파이썬 코드

```
# 밝기 센서 값
__('Hamster*0:Light').d
```

## 온도 센서 값

햄스터의 온도 센서 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
unit	드롭다운 옵션	온도 단위	섭씨 (°C), 화씨 (°F)

## 자바스크립트 코드

```
// 섭씨 기준 온도센서 값
__getTemperature($('Hamster*0:temperature').d, '°C');
```

```
// 화씨 기준 온도센서 값  
__getTemperature($('Hamster*0:temperature').d, '"F")
```

## 파이썬 코드

```
# 섭씨 기준 온도센서 값  
__getTemperature(__('Hamster*0:temperature').d, '"C')  
  
# 화씨 기준 온도센서 값  
__getTemperature(__('Hamster*0:temperature').d, '"F')
```

## 신호 세기 값

햄스터의 신호 세기 값을 가져옵니다.



신호 세기

## 자바스크립트 코드

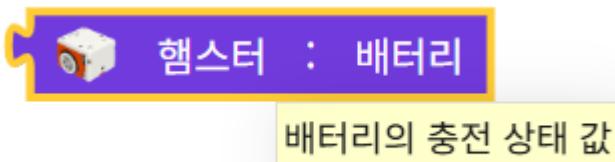
```
// 신호 세기 값  
$('Hamster*0:signal_strength').d;
```

## 파이썬 코드

```
# 신호 세기 값  
__('Hamster*0:signal_strength').d
```

## 배터리 충전 상태 값

햄스터의 배터리 충전 상태 값을 가져옵니다.



배터리의 충전 상태 값

## 자바스크립트 코드

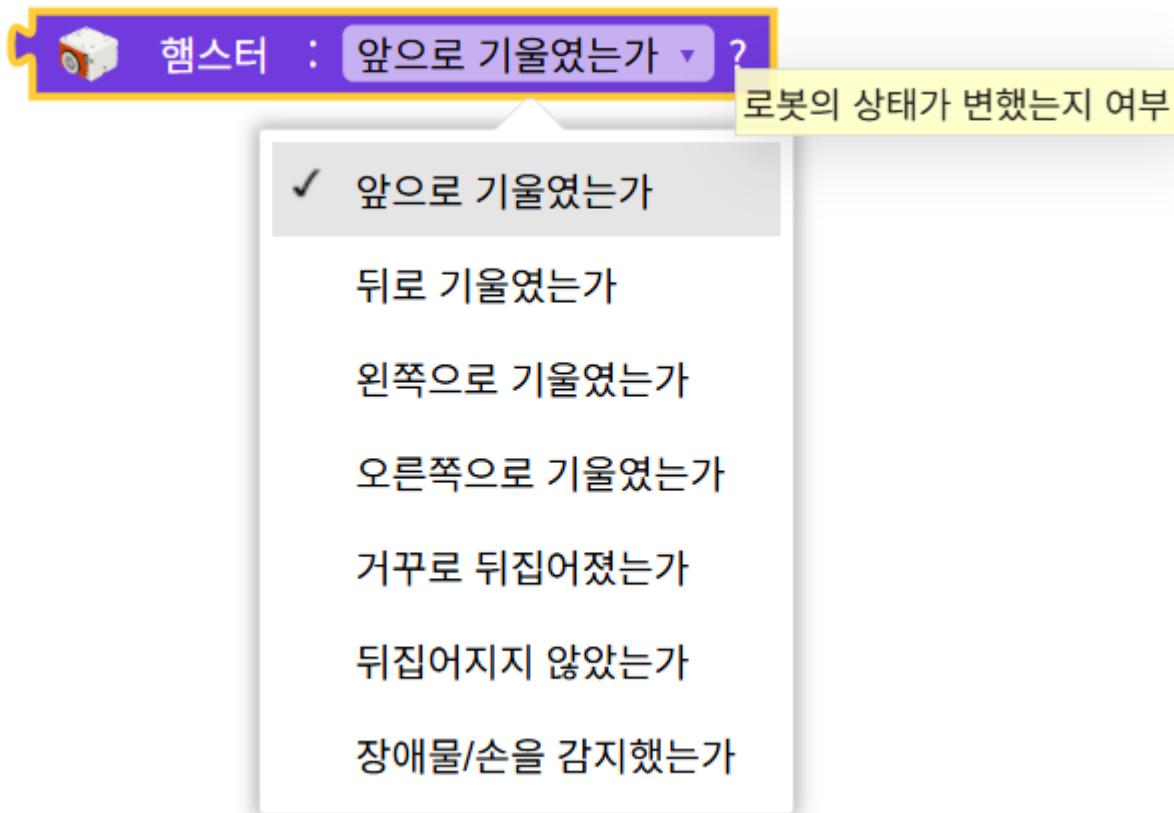
```
// 배터리 충전 상태 값  
$(Hamster*0:battery.level').d;
```

## 파이썬 코드

```
# 배터리 충전 상태 값  
__('Hamster*0:battery.level').d
```

## 상태 변경 여부

햄스터의 상태 변경 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
condition	드롭다운 옵션	위치 조건	앞으로 기울였는가, 뒤로 기울였는가, 왼쪽으로 기울였는가, 오른쪽으로 기울였는가, 거꾸로 뒤집어졌는가, 뒤집어지지 않았는가, 장애물/손을 감지했는가

## 자바스크립트 코드

```
// 햄스터가 앞으로 기울였는가?
$('Hamster*0:acceleration.x').d > 5000;

// 햄스터가 뒤로 기울였는가?
$('Hamster*0:acceleration.x').d < -5000;

// 햄스터가 왼쪽으로 기울였는가?
$('Hamster*0:acceleration.y').d > 5000;

// 햄스터가 오른쪽으로 기울였는가?
$('Hamster*0:acceleration.y').d < -5000;

// 햄스터가 거꾸로 뒤집어졌는가?
$('Hamster*0:acceleration.z').d > 0;

// 햄스터가 뒤집어지지 않았는가?
$('Hamster*0:acceleration.z').d < -3000;

// 햄스터가 장애물/손을 감지했는가?
$('Hamster*0:proximity.left').d > 50 || $('Hamster*0:proximity.right').d > 50;
```

## 파이썬 코드

```
# 햄스터가 앞으로 기울였는가?
__('Hamster*0:acceleration.x').d > 5000

# 햄스터가 뒤로 기울였는가?
__('Hamster*0:acceleration.x').d < -5000

# 햄스터가 왼쪽으로 기울였는가?
__('Hamster*0:acceleration.y').d > 5000

# 햄스터가 오른쪽으로 기울였는가?
__('Hamster*0:acceleration.y').d < -5000

# 햄스터가 거꾸로 뒤집어졌는가?
__('Hamster*0:acceleration.z').d > 0

# 햄스터가 뒤집어지지 않았는가?
__('Hamster*0:acceleration.z').d < -3000

# 햄스터가 장애물/손을 감지했는가?
__('Hamster*0:proximity.left').d > 50 or __('Hamster*0:proximity.right').d > 50
```

## 입출력 포트 입력 모드 설정하기

햄스터 입출력 포트의 입력 모드를 설정합니다.

A 포트, B 포트 또는 A 와 B 포트의 입력 모드를 설정할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	설정 입출력 포트	A, B, A 와 B
mode	드롭다운 옵션	포트 동작 모드	아날로그 입력, 디지털 입력, 서보 출력, pwm 출력, 디지 털 출력

## 자바스크립트 코드

```
// 햄스터 포트 A 를 아날로그 입력으로 정하기
$('Hamster*0:io.a.mode').d = 0;

// 햄스터 포트 A 를 디지털 입력으로 정하기
$('Hamster*0:io.a.mode').d = 1;

// 햄스터 포트 B 를 서보 출력으로 정하기
$('Hamster*0:io.b.mode').d = 8;

// 햄스터 포트 B 를 pwm 출력으로 정하기
$('Hamster*0:io.b.mode').d = 9;

// 햄스터 포트 A,B 를 디지털 출력으로 정하기
$('Hamster*0:io.a.mode').d = 10;
$('Hamster*0:io.b.mode').d = 10;
```

## 파이썬 코드

```
# 햄스터 포트 A 를 아날로그 입력으로 정하기  
__('Hamster*0:io.a.mode').d = 0  
  
# 햄스터 포트 A 를 디지털 입력으로 정하기  
__('Hamster*0:io.a.mode').d = 1  
  
# 햄스터 포트 B 를 서보 출력으로 정하기  
__('Hamster*0:io.b.mode').d = 8  
  
# 햄스터 포트 B 를 pwm 출력으로 정하기  
__('Hamster*0:io.b.mode').d = 9  
  
# 햄스터 포트 A,B 를 디지털 출력으로 정하기  
__('Hamster*0:io.a.mode').d = 10  
__('Hamster*0:io.b.mode').d = 10
```

## 입출력 포트 출력값 설정하기

햄스터 입출력 포트의 출력값을 설정합니다.

A 포트, B 포트 또는 A 와 B 포트의 입력 모드를 설정할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	설정 입출력 포트	A(a), B(b), A 와 B(a,b)
output	입력값	입출력 포트 출력값	0 ~ 180

## 자바스크립트 코드

```
// 햄스터 포트 A 출력값을 50 으로 정하기  
$(Hamster*0:io.a.out').d = 50;  
  
// 햄스터 포트 B 출력값을 40 으로 정하기  
$(Hamster*0:io.b.out').d = 40;
```

```
// 햄스터 포트 A, B 출력값을 30 으로 정하기
$('Hamster*0:io.a.out').d = 30;
$('Hamster*0:io.b.out').d = 30;
```

## 파이썬 코드

```
# 햄스터 포트 A 출력값을 50 으로 정하기
__('Hamster*0:io.a.out').d = 50

# 햄스터 포트 B 출력값을 40 으로 정하기
__('Hamster*0:io.b.out').d = 40

# 햄스터 포트 A, B 출력값을 30 으로 정하기
__('Hamster*0:io.a.out').d = 30
__('Hamster*0:io.b.out').d = 30
```

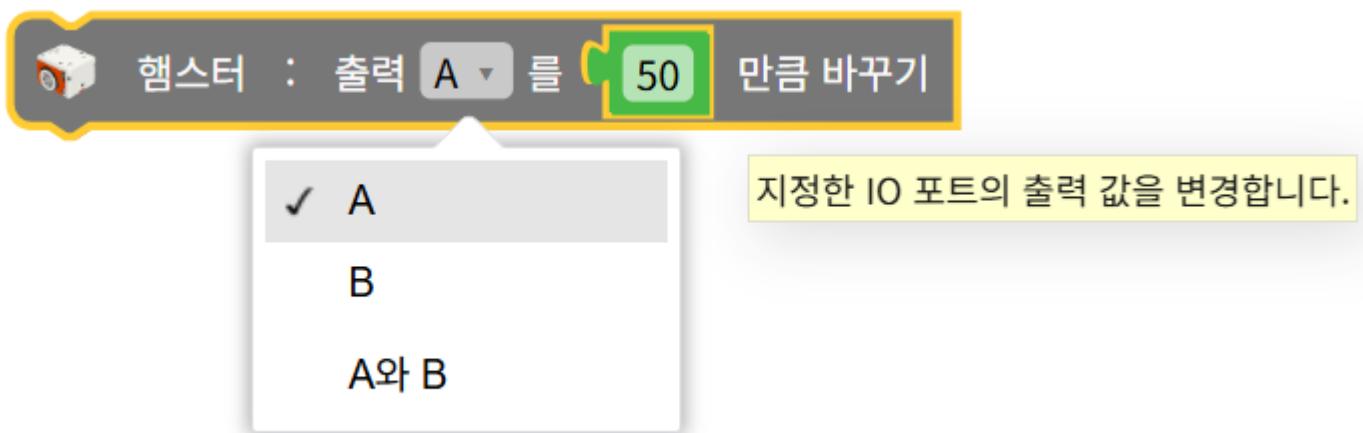
## 입출력 포트 출력값 변경하기

햄스터 입출력 포트의 출력값을 변경합니다.

현재의 포트 출력값에 입력한 출력값을 더한 값이 새로운 포트 출력값이 됩니다.

새롭게 설정된 포트 출력값 범위는 0 ~ 180 으로 설정됩니다.

A 포트, B 포트 또는 A 와 B 포트의 입력 모드를 설정할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	설정 입출력 포트	A(a), B(b), A 와 B(a,b)
value	입력값	입출력 포트 변경값	정수

## 자바스크립트 코드

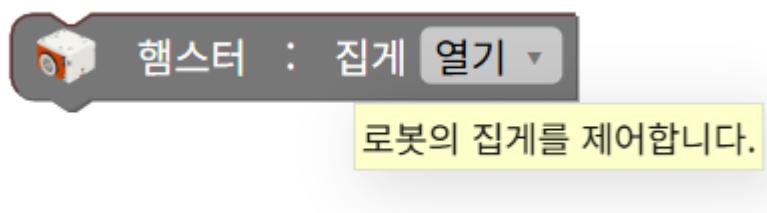
```
// 햄스터 포트 A 출력값을 10 만큼 변경하기  
$(Hamster*0:io.a.out').d = $(Hamster*0:io.a.out').d + 10;  
  
// 햄스터 포트 B 출력값을 20 만큼 변경하기  
$(Hamster*0:io.b.out').d = $(Hamster*0:io.b.out').d + 20;  
  
// 햄스터 포트 A,B 출력값을 30 만큼 변경하기  
$(Hamster*0:io.a.out').d = $(Hamster*0:io.a.out').d + 30;  
$(Hamster*0:io.b.out').d = $(Hamster*0:io.b.out').d + 30;
```

## 파이썬 코드

```
# 햄스터 포트 A 출력값을 10 만큼 변경하기  
__('Hamster*0:io.a.out').d = __('Hamster*0:io.a.out').d + 10  
  
# 햄스터 포트 B 출력값을 20 만큼 변경하기  
__('Hamster*0:io.b.out').d = __('Hamster*0:io.b.out').d + 20  
  
# 햄스터 포트 A,B 출력값을 30 만큼 변경하기  
__('Hamster*0:io.a.out').d = __('Hamster*0:io.a.out').d + 30  
__('Hamster*0:io.b.out').d = __('Hamster*0:io.b.out').d + 30
```

## 집게 열기 / 닫기

햄스터의 집게를 열거나 닫습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	집게 토글	열기 (1), 닫기 (2)

## 자바스크립트 코드

```
// 햄스터 집게 열기  
$(Hamster*0:io.gripper').d = 1;  
  
// 햄스터 집게 닫기  
$(Hamster*0:io.gripper').d = 2;
```

## 파이썬 코드

```
# 햄스터 집게 열기  
__('Hamster*0:io.gripper').d = 1  
  
# 햄스터 집게 닫기  
__('Hamster*0:io.gripper').d = 2
```

## 슈터 각도 설정하기

햄스터의 슈터 각도를 설정하여 제어합니다.

각도의 범위는 0 ~ 255 입니다.



슈터 각도를 설정하여 제어합니다. 각도의 범위는 0 ~ 255 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
angle	입력값	슈터 각도	0 ~ 255 사이 정수

## 자바스크립트 코드

```
// 햄스터 슈터 각도 255 도로 정하기  
$('Hamster*0:io.shooter').d = 255;  
  
// 햄스터 슈터 각도 0 도로 정하기  
$('Hamster*0:io.shooter').d = 0;
```

## 파이썬 코드

```
# 햄스터 슈터 각도 255 도로 정하기  
__('Hamster*0:io.shooter').d = 255  
  
# 햄스터 슈터 각도 0 도로 정하기  
__('Hamster*0:io.shooter').d = 0
```

## 입출력 포트 입력 값

햄스터의 입출력 포트 입력 값을 가져옵니다.



입출력 포트의 입력 값

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	입출력 포트	A(a), B(b)

## 자바스크립트 코드

```
// 햄스터 A 포트 입력값  
$(Hamster*0:io.a.in').d;  
  
// 햄스터 B 포트 입력값  
$(Hamster*0:io.b.in').d;
```

## 파이썬 코드

```
# 햄스터 A 포트 입력값  
__('Hamster*0:io.a.in').d  
  
# 햄스터 B 포트 입력값  
__('Hamster*0:io.b.in').d
```

## 햄스터 S

## 블록

### 바퀴 속도 설정하기

햄스터 S의 바퀴 속도를 설정합니다.

바퀴 속도가 양수이면 앞쪽 방향으로 회전하고, 바퀴 속도가 음수이면 뒤쪽 방향으로 회전합니다.

예를 들어, 바퀴 속도가 100이라면, 앞쪽 방향으로 100의 속도로 회전하고,  
 바퀴 속도가 -100이라면, 뒤쪽 방향으로 100의 속도로 회전합니다.  
 한번 바퀴 속도를 설정하면, 다시 바퀴 속도를 설정하기 전까지 해당 속도로 햄스터 S가 이동합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	바퀴 속도	-100 ~ 100 정수, 0: 정지

## 자바스크립트 코드

```
// 왼쪽 바퀴 속도를 50 (으)로 정하기
if($('HamsterS*0:wheel.move').d != 0) {
  $('HamsterS*0:wheel.move').d = 0;
}
$('HamsterS*0:wheel.speed.left').d = __getSpeed('HamsterS*0', 50);

// 오른쪽 바퀴 속도를 50 (으)로 정하기
if($('HamsterS*0:wheel.move').d != 0) {
  $('HamsterS*0:wheel.move').d = 0;
}
$('HamsterS*0:wheel.speed.right').d = __getSpeed('HamsterS*0', 50);
```

## 파이썬 코드

```
# 양쪽 바퀴 속도를 50 (으)로 정하기
if ___('HamsterS*0:wheel.move').d != 0:
  ___('HamsterS*0:wheel.move').d = 0
___('HamsterS*0:wheel.speed.left').d = __getSpeed('HamsterS*0', 50)
___('HamsterS*0:wheel.speed.right').d = __getSpeed('HamsterS*0', 50)
```

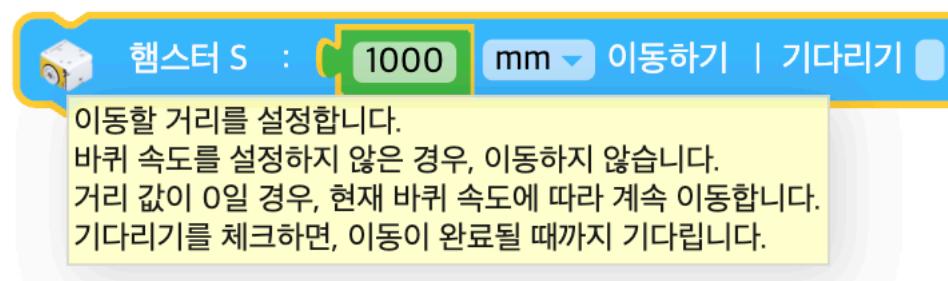
## 거리 이동하기

햄스터 S가 이동할 거리를 설정합니다.  
 바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

거리 값이 0 일 경우에는, 현재 설정되어 있는 바퀴 속도대로 멈추지 않고 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
distance	입력값	거리 값	0 이상 실수
unit	드롭다운 옵션	거리 단위	cm, mm, 인치 (inch)

## 자바스크립트 코드

```
// 5 cm 이동하기 | 기다리기 0
$(`'HamsterS*0:wheel.move'`).d = __getDistance('HamsterS*0', 5, 'cm'); // (robot, distance, unit)
await $(`'HamsterS*0:wheel.!move'`).w();

// 1000 mm 이동하기 | 기다리기 X
$(`'HamsterS*0:wheel.move'`).d = __getDistance('HamsterS*0', 1000, 'mm'); // (robot, distance, unit)
```

## 파이썬 코드

```
# 5 cm 이동하기 | 기다리기 0
__(`'HamsterS*0:wheel.move'`).d = __getDistance('HamsterS*0', 5, 'cm') # (robot, distance, unit)
await __(`'HamsterS*0:wheel.!move'`).w()

# 1000 mm 이동하기 | 기다리기 X
__(`'HamsterS*0:wheel.move'`).d = __getDistance('HamsterS*0', 1000, 'mm') # (robot, distance, unit)
```

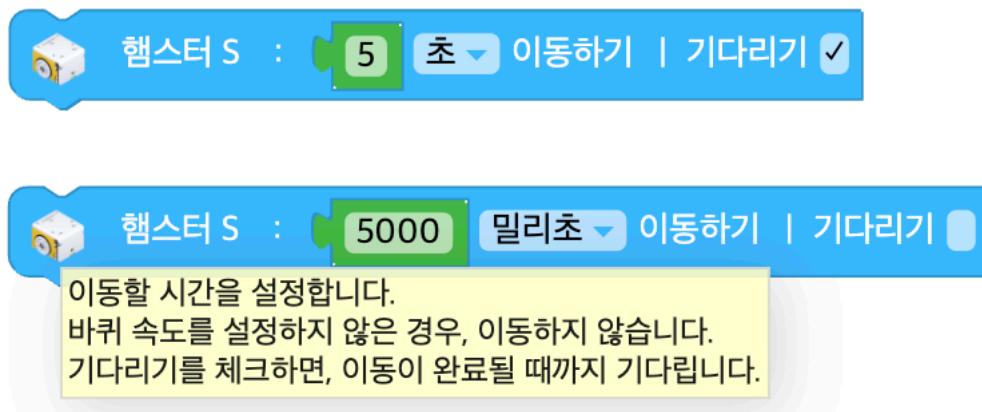
## 시간 이동하기

햄스터 S 가 이동할 시간을 설정합니다.

바퀴 속도가 설정되어 있지 않은 경우, 이동하지 않습니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
time	입력값	시간 값	0 이상 실수
unit	드롭다운 옵션	시간 단위	초 (seconds), 밀리초 (milliseconds)

옵션을 밀리초 (milliseconds)로 설정한 경우에는 `time` 값을 1000으로 나눈 값이 입력됩니다.

## 자바스크립트 코드

```
// 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('HamsterS*0', 5, true); // (robot, time, wait_w)

// 5000 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('HamsterS*0', 5, false); // (robot, time, wait_w)
```

## 파이썬 코드

```
# 5 초 이동하기 | 기다리기 0
await __stopAfterDelay('HamsterS*0', 5, True) # (robot, time, wait_w)

# 5000 밀리초 이동하기 | 기다리기 X
__stopAfterDelay('HamsterS*0', 5, False) # (robot, time, wait_w)
```

## 제자리 돌기

햄스터 S가 제자리에서 회전할 방향과 각도를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



햄스터 S : 왼쪽 ▾ 으로 90 도 제자리 돌기 | 기다리기 ✓



햄스터 S : 오른쪽 ▾ 으로 270 도 제자리 돌기 | 기다리기

제자리에서 회전할 방향과 각도를 설정합니다.  
기다리기를 체크하면, 회전이 완료될 때까지 기다립니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	회전 방향	왼쪽 (left), 오른쪽 (right)
degree	입력값	회전 각도	0 이상 정수

## 자바스크립트 코드

```
// 왼쪽으로 90 도 제자리 돌기 | 기다리기 0
await __turn_degree_left('HamsterS*0', 90, true); // (robot, degree, wait_w)

// 오른쪽으로 270 도 제자리 돌기 | 기다리기 X
__turn_degree_right('HamsterS*0', 270, false); // (robot, degree, wait_w)
```

## 파이썬 코드

```
# 왼쪽으로 90 도 제자리 돌기 | 기다리기 0
await __turn_degree_left('HamsterS*0', 90, True) # (robot, degree, wait_w)

# 오른쪽으로 270 도 제자리 돌기 | 기다리기 X
__turn_degree_right('HamsterS*0', 270, False) # (robot, degree, wait_w)
```

## 바퀴 속도 변경하기

햄스터 S의 바퀴 속도를 변경합니다.

현재의 바퀴 속도에 입력한 속도를 더한 값이 새로운 바퀴 속도가 됩니다.

새롭게 설정된 바퀴 속도의 범위는 -100 ~ 100으로 설정됩니다.



햄스터 S : 왼쪽 ▾ 바퀴 속도를 50 만큼 바꾸기

50

만큼 바꾸기

✓ 왼쪽

오른쪽

양쪽

바퀴 속도를 변경합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
wheel	드롭다운 옵션	바퀴 종류	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
velocity	입력값	현재 바퀴 속도에 더할 속도 값	-200 ~ 200 정수, 0: 정지

## 자바스크립트 코드

```
// 왼쪽 바퀴 속도를 50 만큼 바꾸기
if($('HamsterS*0:wheel.move').d != 0) {
    $('HamsterS*0:wheel.move').d = 0;
}
$('HamsterS*0:wheel.speed.left').d = $('HamsterS*0:wheel.speed.left').d + __getSpeed('HamsterS*0', 50);

// 오른쪽 바퀴를 50 만큼 바꾸기
if($('HamsterS*0:wheel.move').d != 0) {
    $('HamsterS*0:wheel.move').d = 0;
}
$('HamsterS*0:wheel.speed.right').d = $('HamsterS*0:wheel.speed.right').d + __getSpeed('HamsterS*0', 50);

// 양쪽 바퀴를 50 만큼 바꾸기
if($('HamsterS*0:wheel.move').d != 0) {
    $('HamsterS*0:wheel.move').d = 0;
}
$('HamsterS*0:wheel.speed.left').d = $('HamsterS*0:wheel.speed.left').d + __getSpeed('HamsterS*0', 50);
$('HamsterS*0:wheel.speed.right').d = $('HamsterS*0:wheel.speed.right').d + __getSpeed('HamsterS*0', 50);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도를 50 만큼 바꾸기
if __('HamsterS*0:wheel.move').d != 0:
    __('HamsterS*0:wheel.move').d = 0
__('HamsterS*0:wheel.speed.left').d = __('HamsterS*0:wheel.speed.left').d + __getSpeed('HamsterS*0', 50)

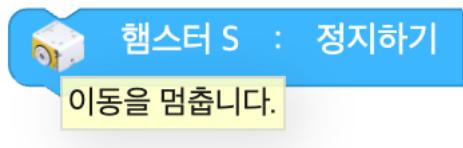
# 오른쪽 바퀴 속도를 50 만큼 바꾸기
if __('HamsterS*0:wheel.move').d != 0:
    __('HamsterS*0:wheel.move').d = 0
__('HamsterS*0:wheel.speed.right').d = __('HamsterS*0:wheel.speed.right').d + __getSpeed('HamsterS*0', 50)
```

```
# 양쪽 바퀴 속도를 50 만큼 바꾸기
if __('HamsterS*0:wheel.move').d != 0:
    __('HamsterS*0:wheel.move').d = 0
__('HamsterS*0:wheel.speed.left').d = __('HamsterS*0:wheel.speed.left').d + __getSpeed('HamsterS*0', 50)
__('HamsterS*0:wheel.speed.right').d = __('HamsterS*0:wheel.speed.right').d + __getSpeed('HamsterS*0', 50)
```

## 정지하기

햄스터 S 의 이동을 멈춥니다.

햄스터 S 의 양쪽 바퀴 속도가 모두 0 으로 초기화됩니다.



## 자바스크립트 코드

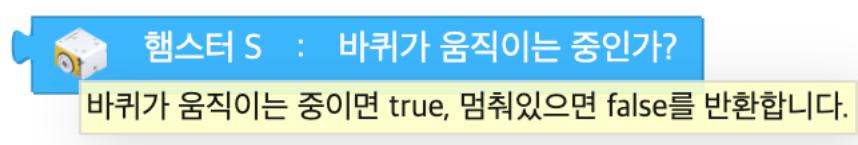
```
__stopMove('HamsterS*0');
```

## 파이썬 코드

```
__stopMove('HamsterS*0')
```

## 바퀴가 움직이는 중인가?

햄스터 S 의 바퀴가 움직이고 있는지 아닌지 여부를 참 (1) / 거짓 (0) (으) 로 반환합니다.



## 자바스크립트 코드

```
$(('HamsterS*0:wheel.moving')).d
```

## 파이썬 코드

```
__('HamsterS*0:wheel.moving').d
```

## 말판 앞으로 한 칸 이동하기

햄스터 S 가 말판 위에서 한 칸 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



말판 위에서 정해진 대로 한 칸씩 움직입니다.

## 자바스크립트 코드

```
// 말판 앞으로 한 칸 이동하기 | 기다리기 0
await __grid_move_forward('HamsterS*0', true); // (robot, wait_w)

// 말판 앞으로 한 칸 이동하기 | 기다리기 X
__grid_move_forward('HamsterS*0', false); // (robot, wait_w)
```

## 파이썬 코드

```
# 말판 앞으로 한 칸 이동하기 | 기다리기 0
await __grid_move_forward('HamsterS*0', True) # (robot, wait_w)

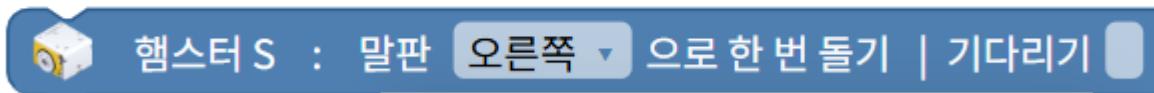
# 말판 앞으로 한 칸 이동하기 | 기다리기 X
__grid_move_forward('HamsterS*0', False) # (robot, wait_w)
```

## 말판에서 한번 돌기

말판 위 햄스터 S 가 입력받은 방향으로 90 도 회전합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



말판 위에서 정해진 방향으로 90도 회전합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	회전 방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
// 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_left('HamsterS*0', true); // (robot, wait_w)

// 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 X
__grid_turn_left('HamsterS*0', false); // (robot, wait_w)

// 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_right('HamsterS*0', true); // (robot, wait_w)

// 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 X
__grid_turn_right('HamsterS*0', false); // (robot, wait_w)
```

## 파이썬 코드

```
# 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_left('HamsterS*0', True) # (robot, wait_w)

# 말판 위에서 왼쪽으로 한번 돌기 | 기다리기 X
__grid_turn_left('HamsterS*0', False) # (robot, wait_w)

# 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 0
await __grid_turn_right('HamsterS*0', True) # (robot, wait_w)

# 말판 위에서 오른쪽으로 한번 돌기 | 기다리기 X
__grid_turn_right('HamsterS*0', False) # (robot, wait_w)
```

## 펜 홀더 기준 회전하기

펜 홀더 사용 중에, 햄스터 S 가 제자리에서 회전할 방향과 각도를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 async 함수 내에서만 사용할 수 있습니다.



펜 홀더를 사용할 때, 회전할 기준과 방향, 각도를 설정합니다.  
기다리기를 체크하면, 회전이 완료될 때까지 기다립니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pivot	드롭다운 옵션	회전 기준	왼쪽 펜 (left pen), 오른쪽 펜 (right pen), 왼쪽 바퀴 (left wheel), 오른쪽 바퀴 (right wheel)
direction	드롭다운 옵션	방향	앞쪽 (forward), 뒤쪽 (backward)
degree	입력값	회전 각도	0 이상 정수

## 자바스크립트 코드

```
//왼쪽 펜 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'left pen', 'forward', 90, true); // (robot, pivot, direction, degree, wait_w)

//왼쪽 펜 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'left pen', 'backward', 90, false); // (robot, pivot, direction, degree, wait_w)

//오른쪽 펜 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'right pen', 'forward', 90, true); // (robot, pivot, direction, degree, wait_w)

//오른쪽 펜 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'right pen', 'backward', 90, false); // (robot, pivot, direction, degree, wait_w)

//왼쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'left wheel', 'forward', 90, true); // (robot, pivot, direction, degree, wait_w)

//왼쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'left wheel', 'backward', 90, false); // (robot, pivot, direction, degree, wait_w)

//오른쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'right wheel', 'forward', 90, true); // (robot, pivot, direction, degree, wait_w)

//오른쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'right wheel', 'backward', 90, false); // (robot, pivot, direction, degree, wait_w)
```

## 파이썬 코드

```
# 왼쪽 펜 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'left pen', 'forward', 90, True) # (robot, pivot, direction, degree, wait_w)

# 왼쪽 펜 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'left pen', 'backward', 90, False) # (robot, pivot, direction, degree, wait_w)

# 오른쪽 펜 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'right pen', 'forward', 90, True) # (robot, pivot, direction, degree, wait_w)

# 오른쪽 펜 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'right pen', 'backward', 90, False) # (robot, pivot, direction, degree, wait_w)

# 왼쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'left wheel', 'forward', 90, True) # (robot, pivot, direction, degree, wait_w)

# 왼쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'left wheel', 'backward', 90, False) # (robot, pivot, direction, degree, wait_w)
```

```
# 오른쪽 바퀴 기준 앞쪽 방향으로 90 도 돌기 | 기다리기 0
await __pivot('HamsterS*0', 'right wheel', 'forward', 90, True) # (robot, pivot, direction, degree, wait_w)

# 오른쪽 바퀴 기준 뒤쪽 방향으로 90 도 돌기 | 기다리기 X
__pivot('HamsterS*0', 'right wheel', 'backward', 90, False) # (robot, pivot, direction, degree, wait_w)
```

## 펜 홀더 기준 원 그리며 돌기

펜 홀더 사용 중에, 햄스터 S 가 원을 그릴 때 회전할 기준, 각도, 방향 그리고 원의 반지름 크기를 설정합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
pivot	드롭다운 옵션	회전 기준	왼쪽 펜 (left pen), 오른쪽 펜 (right pen)
direction	드롭다운 옵션	방향	왼쪽 앞 (left forward), 왼쪽 뒤 (left backward), 오른쪽 앞 (right forward), 오른쪽 뒤 (right backward)
radius	입력값	원의 반지름	0 이상 실수
unit	드롭다운 옵션	거리 단위	cm, mm, 인치 (inch)
degree	입력값	회전 각도	0 이상 실수

## 자바스크립트 코드

```
// 왼쪽 펜 기준 왼쪽 앞으로 1cm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'left pen', 'left forward', __getDistance('HamsterS*0', 1, 'cm'), 90, false); // (robot, pivot, direction, circle_info, degree, wait_w)

// 왼쪽 펜 기준 왼쪽 뒤로 1cm 원을 그리며 90 도 돌기 | 기다리기 0
```

```

await __pivot_circle('HamsterS*0', 'left pen', 'left backward', __getDistance('HamsterS*0', 1, 'cm'), 90, true); // (robot, pivot, direction, circle_info, degree, wait_w)

// 왼쪽 펜 기준 오른쪽 앞으로 1cm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'left pen', 'right forward', __getDistance('HamsterS*0', 1, 'cm'), 90, false); // (robot, pivot, direction, circle_info, degree, wait_w)

// 왼쪽 펜 기준 오른쪽 뒤로 1cm 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('HamsterS*0', 'left pen', 'right backward', __getDistance('HamsterS*0', 1, 'cm'), 90, true); // (robot, pivot, direction, circle_info, degree, wait_w)

// 오른쪽 펜 기준 왼쪽 앞으로 1mm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'right pen', 'left forward', __getDistance('HamsterS*0', 1, 'mm'), 90, false); // (robot, pivot, direction, circle_info, degree, wait_w)

// 오른쪽 펜 기준 왼쪽 뒤로 1mm 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('HamsterS*0', 'right pen', 'left backward', __getDistance('HamsterS*0', 1, 'mm'), 90, true); // (robot, pivot, direction, circle_info, degree, wait_w)

// 오른쪽 펜 기준 오른쪽 앞으로 1mm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'right pen', 'right forward', __getDistance('HamsterS*0', 1, 'mm'), 90, false); // (robot, pivot, direction, circle_info, degree, wait_w)

// 오른쪽 펜 기준 오른쪽 뒤로 1mm 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('HamsterS*0', 'right pen', 'right backward', __getDistance('HamsterS*0', 1, 'mm'), 90, true); // (robot, pivot, direction, circle_info, degree, wait_w)

```

## 파이썬 코드

```

# 왼쪽 펜 기준 왼쪽 앞으로 1cm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'left pen', 'left forward', __getDistance('HamsterS*0', 1, 'cm'), 90, False) # (robot, pivot, direction, circle_info, degree, wait_w)

# 왼쪽 펜 기준 왼쪽 뒤로 1cm 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('HamsterS*0', 'left pen', 'left backward', __getDistance('HamsterS*0', 1, 'cm'), 90, True) # (robot, pivot, direction, circle_info, degree, wait_w)

# 왼쪽 펜 기준 오른쪽 앞으로 1cm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'left pen', 'right forward', __getDistance('HamsterS*0', 1, 'cm'), 90, False) # (robot, pivot, direction, circle_info, degree, wait_w)

# 왼쪽 펜 기준 오른쪽 뒤로 1cm 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('HamsterS*0', 'left pen', 'right backward', __getDistance('HamsterS*0', 1, 'cm'), 90, True) # (robot, pivot, direction, circle_info, degree, wait_w)

# 오른쪽 펜 기준 왼쪽 앞으로 1mm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'right pen', 'left forward', __getDistance('HamsterS*0', 1, 'mm'), 90, False) # (robot, pivot, direction, circle_info, degree, wait_w)

# 오른쪽 펜 기준 왼쪽 뒤로 1mm 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('HamsterS*0', 'right pen', 'left backward', __getDistance('HamsterS*0', 1, 'mm'), 90, True) # (robot, pivot, direction, circle_info, degree, wait_w)

# 오른쪽 펜 기준 오른쪽 앞으로 1mm 원을 그리며 90 도 돌기 | 기다리기 X
__pivot_circle('HamsterS*0', 'right pen', 'right forward', __getDistance('HamsterS*0', 1, 'mm'), 90, False) # (robot, pivot, direction, circle_info, degree, wait_w)

# 오른쪽 펜 기준 오른쪽 뒤로 1mm 원을 그리며 90 도 돌기 | 기다리기 0
await __pivot_circle('HamsterS*0', 'right pen', 'right backward', __getDistance('HamsterS*0', 1, 'mm'), 90, True) # (robot, pivot, direction, circle_info, degree, wait_w)

```

## 센서로 선 따라가기

햄스터 S 가 바닥 센서를 이용하여 특정한 선을 따라갑니다.



햄스터 S : 왼쪽 ▾ 바닥 센서로 검정색 ▾ 선을 따라가기



햄스터 S : 오른쪽 ▾ 바닥 센서로 검정색 ▾ 선을 따라가기



햄스터 S : 가운데 ▾ 바닥 센서로 흰색 ▾ 선을 따라가기

바닥 센서를 이용해 특정한 색의 선을 따라 이동합니다.

### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	바닥 센서 방향	왼쪽 (left), 오른쪽 (right), 가운데 (middle)
color	드롭다운 옵션	선의 색	검정색 (black), 흰색 (white)

### 자바스크립트 코드

```
$('HamsterS*0:wheel.trace.mode').d = 1; // 왼쪽 바닥 센서로 검정색 선을 따라가기
$('HamsterS*0:wheel.trace.mode').d = 2; // 오른쪽 바닥 센서로 검정색 선을 따라가기
$('HamsterS*0:wheel.trace.mode').d = 3; // 가운데 바닥 센서로 검정색 선을 따라가기
$('HamsterS*0:wheel.trace.mode').d = 8; // 왼쪽 바닥 센서로 흰색 선을 따라가기
$('HamsterS*0:wheel.trace.mode').d = 9; // 오른쪽 바닥 센서로 흰색 선을 따라가기
$('HamsterS*0:wheel.trace.mode').d = 10; // 가운데 바닥 센서로 흰색 선을 따라가기
```

### 파이썬 코드

```
__('HamsterS*0:wheel.trace.mode').d = 1 # 왼쪽 바닥 센서로 검정색 선을 따라가기
__('HamsterS*0:wheel.trace.mode').d = 2 # 오른쪽 바닥 센서로 검정색 선을 따라가기
__('HamsterS*0:wheel.trace.mode').d = 3 # 가운데 바닥 센서로 검정색 선을 따라가기
__('HamsterS*0:wheel.trace.mode').d = 8 # 왼쪽 바닥 센서로 흰색 선을 따라가기
__('HamsterS*0:wheel.trace.mode').d = 9 # 오른쪽 바닥 센서로 흰색 선을 따라가기
__('HamsterS*0:wheel.trace.mode').d = 10 # 가운데 바닥 센서로 흰색 선을 따라가기
```

### 회전 후 선 따라가기 & 교차로에서 멈추기

햄스터 S 가 지정한 방향으로 회전한 뒤, 다음 교차로를 만날 때까지 이동합니다.

기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



햄스터 S : 좌회전 ▾ 한 뒤에 검정색 ▾ 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 ✓



햄스터 S : 우회전 ▾ 한 뒤에 흰색 ▾ 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기

지정한 방향으로 회전한 뒤, 다음 교차로를 만날 때까지 이동합니다.  
기다리기를 체크하면, 이동이 완료될 때까지 기다립니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	이동 방향	좌회전 (left), 우회전 (right), 전진 (forward), 유턴 (back)
color	드롭다운 옵션	선의 색	검정색 (black), 흰색 (white)

## 자바스크립트 코드

```
$(HamsterS*0:wheel.trace.mode').d = 4; // 좌회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await $(HamsterS*0:wheel.trace.!mode').w();
$(HamsterS*0:wheel.trace.mode').d = 5; // 우회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
$(HamsterS*0:wheel.trace.mode').d = 6; // 전진 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await $(HamsterS*0:wheel.trace.!mode').w();
$(HamsterS*0:wheel.trace.mode').d = 7; // 유턴 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
$(HamsterS*0:wheel.trace.mode').d = 11; // 좌회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await $(HamsterS*0:wheel.trace.!mode').w();
$(HamsterS*0:wheel.trace.mode').d = 12; // 우회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
$(HamsterS*0:wheel.trace.mode').d = 13; // 전진 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await $(HamsterS*0:wheel.trace.!mode').w();
$(HamsterS*0:wheel.trace.mode').d = 14; // 유턴 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
```

## 파이썬 코드

```
--('HamsterS*0:wheel.trace.mode').d = 4 # 좌회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await --('HamsterS*0:wheel.trace.!mode').w()
--('HamsterS*0:wheel.trace.mode').d = 5 # 우회전 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
--('HamsterS*0:wheel.trace.mode').d = 6 # 전진 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await --('HamsterS*0:wheel.trace.!mode').w()
--('HamsterS*0:wheel.trace.mode').d = 7 # 유턴 한 뒤에 검정색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
--('HamsterS*0:wheel.trace.mode').d = 11 # 좌회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await --('HamsterS*0:wheel.trace.!mode').w()
--('HamsterS*0:wheel.trace.mode').d = 12 # 우회전 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
--('HamsterS*0:wheel.trace.mode').d = 13 # 전진 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 0
await --('HamsterS*0:wheel.trace.!mode').w()
--('HamsterS*0:wheel.trace.mode').d = 14 # 유턴 한 뒤에 흰색 선을 따라가다가 다음 교차로에서 멈추기 | 기다리기 X
```

## 선 따라가기 속도 설정

햄스터 S 의 선 따라가기 속도를 설정합니다.

속도의 범위는 1 ~ 10 입니다.



햄스터 S : 선 따라가기 속도를 5 (으)로 정하기

5

(으)로 정하기

선 따라가기 속도를 설정합니다. 속도의 범위는 1 ~ 10 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
velocity	입력값	선 따라가기 속도 값	1 ~ 10 사이 정수

## 자바스크립트 코드

```
$('HamsterS*0:wheel.trace.speed').d = 1; // 선 따라가기 속도 1 로 설정하기
```

## 파이썬 코드

```
--('HamsterS*0:wheel.trace.speed').d = 1 # 선 따라가기 속도 1 로 설정하기
```

## 선 따라가기 방향 변화량 설정

햄스터 S 의 선 따라가기 방향 변화량을 설정합니다.

변화량의 범위는 1 ~ 10 입니다.



햄스터 S : 선 따라가기 방향 변화량을 5 (으)로 정하기

5

(으)로 정하기

선 따라가기 방향 변화량을 설정합니다. 변화량의 범위는 1 ~ 10 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
differential	입력값	선 따라가기 방향 변화량 값	1 ~ 10 사이 정수

## 자바스크립트 코드

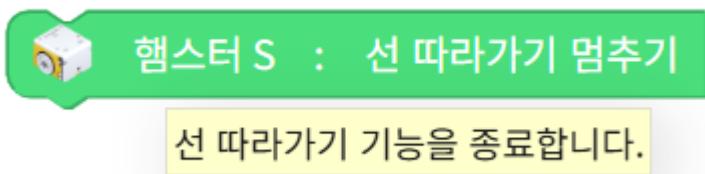
```
$(('HamsterS*0:wheel.trace.gain')).d = 5; // 선 따라가기 방향 변화량 5로 설정하기
```

## 파이썬 코드

```
--('HamsterS*0:wheel.trace.gain').d = 5 # 선 따라가기 방향 변화량 5로 설정하기
```

## 선 따라가기 멈추기

햄스터 S 의 선 따라가기 기능을 종료합니다.



## 자바스크립트 코드

```
$(('HamsterS*0:wheel.trace.mode')).d = 0; // 선 따라가기 멈추기
```

## 파이썬 코드

```
--('HamsterS*0:wheel.trace.mode').d = 0 # 선 따라가기 멈추기
```

## LED 색 설정하기

햄스터 S 의 LED 색을 설정합니다.

왼쪽, 오른쪽 또는 양쪽의 색을 설정할 수 있습니다.



햄스터 S : 왼쪽 ▾ LED 색을 검정색 ▾ 으로 정하기



햄스터 S : 오른쪽 ▾ LED 색을 초록색 ▾ 으로 정하기



햄스터 S : 양쪽 ▾ LED 색을 자홍색 ▾ 으로 정하기

LED 색을 설정합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
color	드롭다운 옵션	색상	검정색 ([0, 0, 0]), 빨간색 ([255, 0, 0]), 노란색 ([255, 255, 0]), 초록색 ([0, 255, 0]), 청록색 ([0, 255, 255]), 파란색 ([0, 0, 255]), 자홍색 ([255, 0, 255]), 흰색 ([255, 255, 255])

## 자바스크립트 코드

```
// 왼쪽 LED 색을 빨간색으로 정하기
$('#HamsterS*0:led.left').d = [255, 0, 0];

// 오른쪽 LED 색을 초록색으로 정하기
$('#HamsterS*0:led.right').d = [0, 255, 0];

// 양쪽 LED 색을 파란색으로 정하기
$('#HamsterS*0:led.left').d = [0, 0, 255];
$('#HamsterS*0:led.right').d = [0, 0, 255];
```

## 파이썬 코드

```
# 左쪽 LED 색을 빨간색으로 정하기  
__('HamsterS*0:led.left').d = [255, 0, 0]  
  
# 오른쪽 LED 색을 초록색으로 정하기  
__('HamsterS*0:led.right').d = [0, 255, 0]  
  
# 양쪽 LED 색을 파란색으로 정하기  
__('HamsterS*0:led.left').d = [0, 0, 255]  
__('HamsterS*0:led.right').d = [0, 0, 255]
```

## LED 색 색상 카테고리 블록들로 설정하기

색상 카테고리에 있는 블록들로 햄스터 S의 LED 색을 설정합니다.

왼쪽, 오른쪽 또는 양쪽의 색을 설정할 수 있습니다.



햄스터 S : 왼쪽 ▾ LED 색을 기본 색상 으로 정하기



햄스터 S : 왼쪽 ▾ LED 색을 기본 색상 으로 정하기



햄스터 S : 왼쪽 ▾ LED 색을 기본 색상 으로 정하기

색상 카테고리에 있는 블록들로 LED 색을 설정합니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
color	입력값	LED 색상	RGB 배열 ([255,255,255])

## 자바스크립트 코드

```
// 左쪽 LED 색상을 R:255, G:255, B:255 색상으로 정하기  
$('HamsterS*0:led.left').d = [255, 255, 255];  
  
// 오른쪽 LED 색상을 R:255, G:255, B:255 색상으로 정하기
```

```

$(HamsterS*0:led.right').d = [255, 255, 255];

// 양쪽 LED 색상을 무작위 색상으로 정하기
$(HamsterS*0:led.left').d = __randomColor();
$(HamsterS*0:led.right').d = __randomColor();

```

## 파이썬 코드

```

# 왼쪽 LED 색상을 R:255, G:255, B:255 색상으로 정하기
__('HamsterS*0:led.left').d = [255, 255, 255]

# 오른쪽 LED 색상을 R:255, G:255, B:255 색상으로 정하기
__('HamsterS*0:led.right').d = [255, 255, 255]

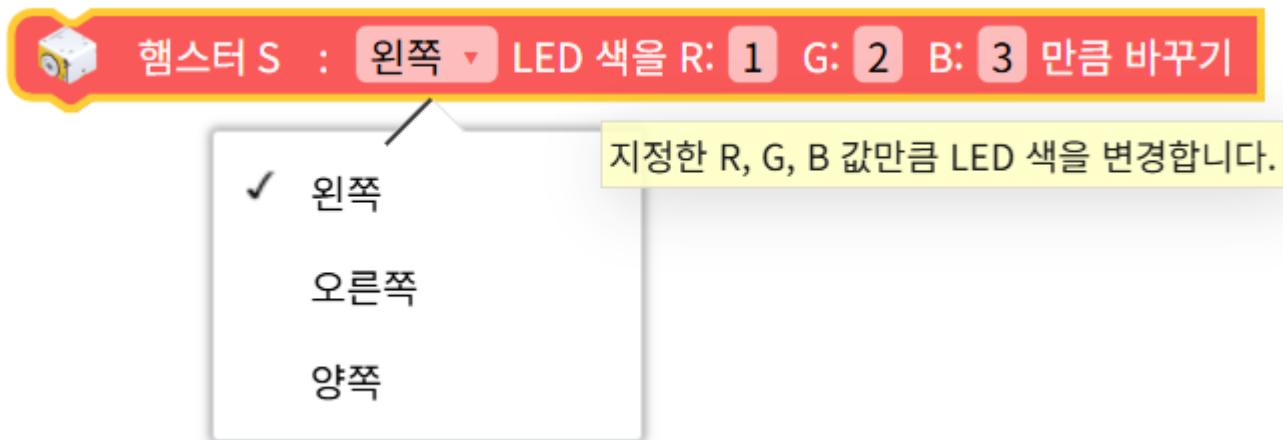
# 양쪽 LED 색상을 무작위 색상으로 정하기
__('HamsterS*0:led.left').d = __randomColor()
__('HamsterS*0:led.right').d = __randomColor()

```

## LED 색 색상 지정 RGB 만큼 변경하기

지정한 R,G,B 값만큼 햄스터 S 의 LED 색을 변경합니다.

왼쪽, 오른쪽 또는 양쪽의 색을 설정할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)
color	입력값	변경 R,G,B 값	R,G,B 각각 -255~255 사이 정수

## 자바스크립트 코드

```
// 왼쪽 LED 색을 R : 10, G : 10, B : 10 만큼 바꾸기  
$('HamsterS*0:led.left').d = [$(('HamsterS*0:led.left').d[0] + 10, $('HamsterS*0:led.left').d[1] + 10, $('HamsterS*0:led.left').d[2] + 10);  
  
// 오른쪽 LED 색을 R : 100, G : 100, B : 100 만큼 바꾸기  
$('HamsterS*0:led.right').d = [$(('HamsterS*0:led.right').d[0] + 100, $('HamsterS*0:led.right').d[1] + 100, $('HamsterS*0:led.right').d[2] + 100);  
  
// 양쪽 LED 색을 R : -50, G : -50, B : -50 만큼 바꾸기  
$('HamsterS*0:led.left').d = [$(('HamsterS*0:led.left').d[0] + -50, $('HamsterS*0:led.left').d[1] + -50, $('HamsterS*0:led.left').d[2] + -50);  
$('HamsterS*0:led.right').d = [$(('HamsterS*0:led.right').d[0] + -50, $('HamsterS*0:led.right').d[1] + -50, $('HamsterS*0:led.right').d[2] + -50);
```

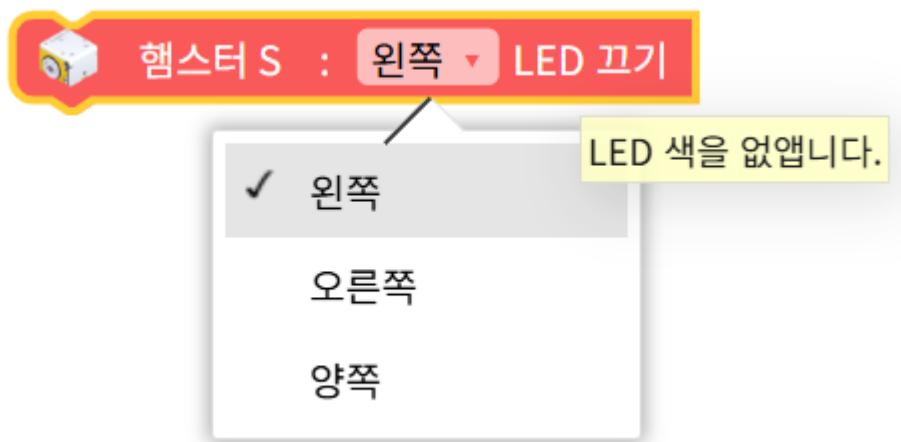
## 파이썬 코드

```
# 왼쪽 LED 색을 R : 10, G : 10, B : 10 만큼 바꾸기  
___('HamsterS*0:led.left').d = [__(('HamsterS*0:led.left').d[0] + 10, ___('HamsterS*0:led.left').d[1] + 10, ___('HamsterS*0:led.left').d[2] + 10)]  
  
# 오른쪽 LED 색을 R : 100, G : 100, B : 100 만큼 바꾸기  
___('HamsterS*0:led.right').d = [__(('HamsterS*0:led.right').d[0] + 100, ___('HamsterS*0:led.right').d[1] + 100, ___('HamsterS*0:led.right').d[2] + 100)]  
  
# 양쪽 LED 색을 R : -50, G : -50, B : -50 만큼 바꾸기  
___('HamsterS*0:led.left').d = [__(('HamsterS*0:led.left').d[0] + -50, ___('HamsterS*0:led.left').d[1] + -50, ___('HamsterS*0:led.left').d[2] + -50)]  
___('HamsterS*0:led.right').d = [__(('HamsterS*0:led.right').d[0] + -50, ___('HamsterS*0:led.right').d[1] + -50, ___('HamsterS*0:led.right').d[2] + -50)]
```

## LED 끄기

햄스터 S 의 LED 색을 없앱니다.

왼쪽, 오른쪽 또는 양쪽의 LED 를 끌 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	적용 LED 방향	왼쪽 (left), 오른쪽 (right), 양쪽 (left, right)

## 자바스크립트 코드

```
// 왼쪽 LED 끄기  
$(HamsterS*0:led.left').d = [0, 0, 0];  
  
// 오른쪽 LED 끄기  
$(HamsterS*0:led.right').d = [0, 0, 0];  
  
// 양쪽 LED 끄기  
$(HamsterS*0:led.left').d = [0, 0, 0];  
$(HamsterS*0:led.right').d = [0, 0, 0];
```

## 파이썬 코드

```
# 왼쪽 LED 끄기  
__('HamsterS*0:led.left').d = [0, 0, 0]  
  
# 오른쪽 LED 끄기  
__('HamsterS*0:led.right').d = [0, 0, 0]  
  
# 양쪽 LED 끄기  
__('HamsterS*0:led.left').d = [0, 0, 0]  
__('HamsterS*0:led.right').d = [0, 0, 0]
```

## 버저음 설정하기

지정된 주파수로 햄스터 S 의 버저음을 설정합니다.

주파수의 범위는 10hz ~ 4200hz 입니다.



지정된 주파수로 버저음을 설정합니다. 주파수의 범위는 10hz ~ 4200hz 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound	입력값	버저음 주파수	10 ~ 4200(hz)

## 자바스크립트 코드

```
// 버저음 주파수 4200hz 로 설정하기  
$(HamsterS*0:sound.buzz').d = 4200;
```

## 파이썬 코드

```
# 버저음 주파수 4200hz 로 설정하기  
__('HamsterS*0:sound.buzz').d = 4200
```

## 음계 연주하기

햄스터 S 가 지정된 음계를 재생합니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
note	드롭다운 옵션	음계	도 (Do), 도 #(Do#), 레 (Re), 레 #(Re#), 미 (Mi), 파 (Fa), 파 #(Fa#), 솔 (So), 솔 #(So#), 라 (La), 라 #(La#), 시 (Ti)
octave	드롭다운 옵션	옥타브	1 ~ 7

## 자바스크립트 코드

```
// 1 옥타브 도 (Do) 음을 연주하기  
$(('HamsterS*0:sound.note').d = 4;  
  
// 1 옥타브 레 (Re) 음을 연주하기  
$(('HamsterS*0:sound.note').d = 6;  
  
// 2 옥타브 도 (Do) 음을 연주하기  
$(('HamsterS*0:sound.note').d = 16;  
  
// 7 옥타브 시 (Ti) 음을 연주하기  
$(('HamsterS*0:sound.note').d = 87;
```

## 파이썬 코드

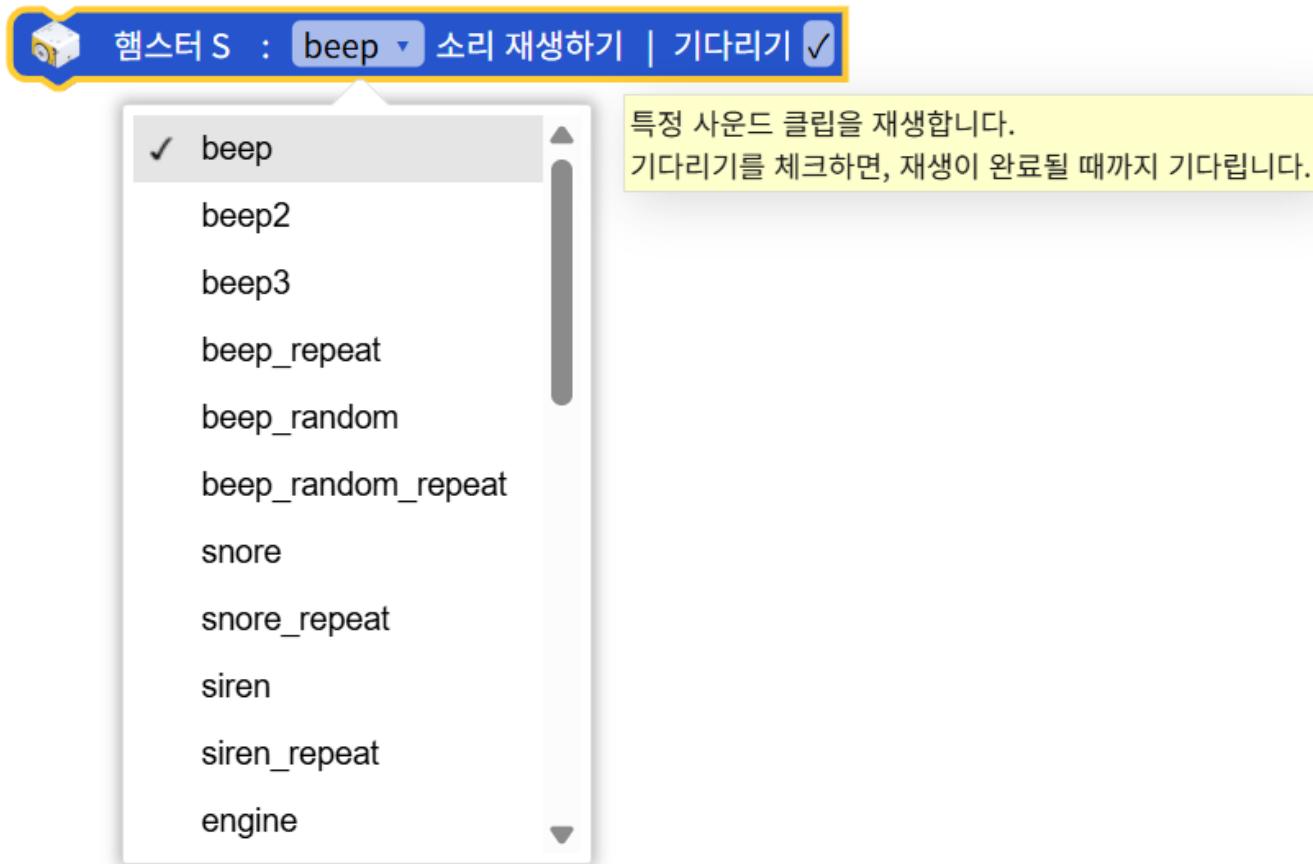
```
# 1 옥타브 도 (Do) 음을 연주하기  
__('HamsterS*0:sound.note').d = 4  
  
# 1 옥타브 레 (Re) 음을 연주하기  
__('HamsterS*0:sound.note').d = 6  
  
# 2 옥타브 도 (Do) 음을 연주하기  
__('HamsterS*0:sound.note').d = 16  
  
# 7 옥타브 시 (Ti) 음을 연주하기  
__('HamsterS*0:sound.note').d = 87
```

## 소리 재생하기

햄스터 S 가 특정 사운드 클립을 재생합니다.

기다리기를 체크하면, 재생이 완료될 때까지 기다립니다.

단, 기다리기를 체크한 경우에는 `async` 함수 내에서만 사용할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
sound_clip	드롭다운 옵션	사운드 클립	beep(1), beep2(2), beep3(3), beep_repeat(4), beep_random(5), beep_random_repeat(6), snore(7), snore_repeat(8), siren(9), siren_repeat(10), engine(11), engine_repeat(12), fart_a(13), fart_b(14), noise(15), noise_repeat(16), whistle(17), chop_chop(18), chop_chop_repeat(19), robot(32), dibidibidip(33), melody(34), mission_complete(35), happy(48), angry(49), sad(50), sleep(51), toy_march(52), birthday(53)

## 자바스크립트 코드

```
// beep 소리 재생하기 | 기다리기 0
$('HamsterS*0:sound.clip').d = 1;
await $('HamsterS*0:sound.!clip').w();

// birthday 소리 재생하기 | 기다리기 X
$('HamsterS*0:sound.clip').d = 53;
```

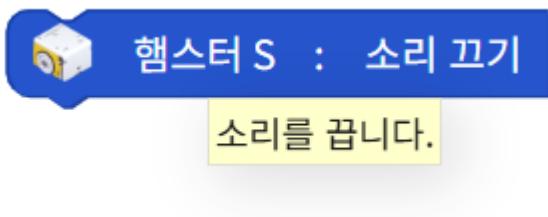
## 파이썬 코드

```
# beep 소리 재생하기 | 기다리기 0
__('Hamster$0:sound.clip').d = 1
await __('Hamster$0:sound.!clip').w()

# birthday 소리 재생하기 | 기다리기 X
__('Hamster$0:sound.clip').d = 53
```

## 소리 끄기

햄스터 S 의 소리를 끕니다.



## 자바스크립트 코드

```
// 햄스터 S 소리 끄기
stopSound('Hamster$0');
```

## 파이썬 코드

```
# 햄스터 S 소리 끄기
__stopSound('Hamster$0')
```

## 소리가 재생 중인가?

햄스터 S 의 소리가 재생중인지 아닌지 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



## 자바스크립트 코드

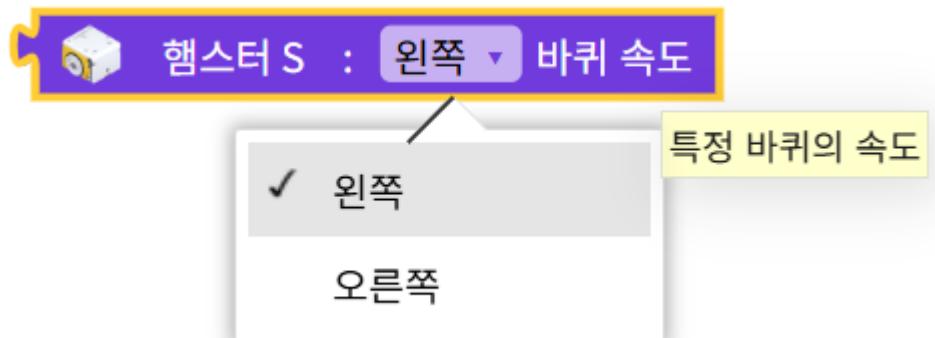
```
// 햄스터 S 의 소리가 재생 중인가? - 재생시 true, 아닐시 false
$('Hamster$0:sound.playing').d;
```

## 파이썬 코드

```
# 햄스터 S 의 소리가 재생 중인가? - 재생시 True, 아닐시 False  
__('HamsterS*0:sound.playing').d
```

## 바퀴 속도 값

햄스터 S 의 지정한 바퀴 속도 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

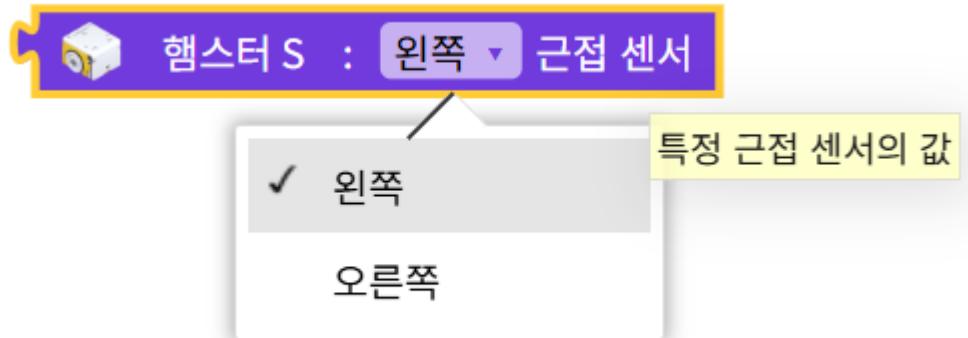
```
//왼쪽 바퀴 속도  
__getSpeedInput('HamsterS*0', $('HamsterS*0:wheel.speed.left').d);  
  
//오른쪽 바퀴 속도  
__getSpeedInput('HamsterS*0', $('HamsterS*0:wheel.speed.right').d);
```

## 파이썬 코드

```
# 왼쪽 바퀴 속도  
__getSpeedInput('HamsterS*0', __('HamsterS*0:wheel.speed.left').d)  
  
# 오른쪽 바퀴 속도  
__getSpeedInput('HamsterS*0', __('HamsterS*0:wheel.speed.right').d)
```

## 근접 센서 값

햄스터 S 의 지정한 근접 센서 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
//왼쪽 근접 센서 값
$('HamsterS*0:proximity.left').d;

//오른쪽 근접 센서 값
$('HamsterS*0:proximity.right').d;
```

## 파이썬 코드

```
# 왼쪽 근접 센서 값
__('HamsterS*0:proximity.left').d

# 오른쪽 근접 센서 값
__('HamsterS*0:proximity.right').d
```

## 바닥 센서 값

햄스터 S 의 지정한 바닥 센서 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
direction	드롭다운 옵션	방향	왼쪽 (left), 오른쪽 (right)

## 자바스크립트 코드

```
//왼쪽 바닥 센서 값
$(('HamsterS*0:floor.left')).d;

//오른쪽 바닥 센서 값
$(('HamsterS*0:floor.right')).d;
```

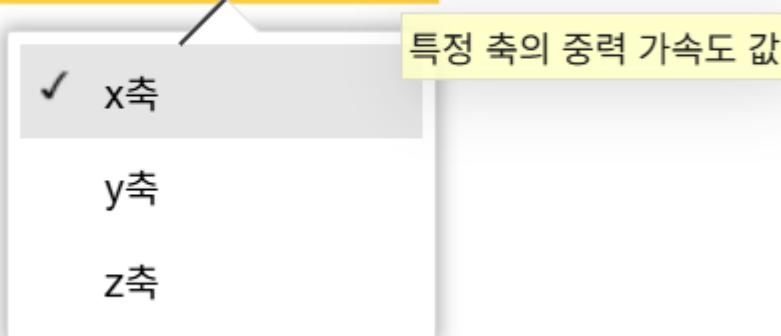
## 파이썬 코드

```
# 왼쪽 바닥 센서 값
__($('HamsterS*0:floor.left')).d

# 오른쪽 바닥 센서 값
__($('HamsterS*0:floor.right')).d
```

## 중력 가속도 값

햄스터 S 의 특정 축의 중력 가속도 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
axis	드롭다운 옵션	축 기준	x 축, y 축, z 축

## 자바스크립트 코드

```
// x 축 기준 중력 가속도 값
$(HamsterS$0:acceleration.x').d;

// y 축 기준 중력 가속도 값
$(HamsterS$0:acceleration.y').d;

// z 축 기준 중력 가속도 값
$(HamsterS$0:acceleration.z').d;
```

## 파이썬 코드

```
# x 축 기준 중력 가속도 값
__($('HamsterS$0:acceleration.x').d

# y 축 기준 중력 가속도 값
__($('HamsterS$0:acceleration.y').d

# z 축 기준 중력 가속도 값
__($('HamsterS$0:acceleration.z').d
```

## 밝기 센서 값

햄스터 S 의 밝기 센서 값을 가져옵니다.



밝기 센서 값

## 자바스크립트 코드

```
// 밝기 센서 값
$(HamsterS*0:Light).d;
```

## 파이썬 코드

```
# 밝기 센서 값
__('HamsterS*0:Light').d
```

## 온도 센서 값

햄스터 S 의 온도 센서 값을 가져옵니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
unit	드롭다운 옵션	온도 단위	섭씨 (°C), 화씨 (°F)

## 자바스크립트 코드

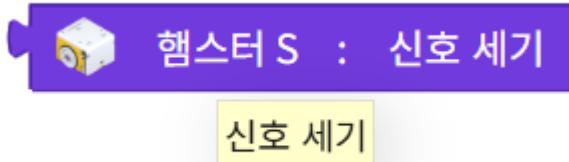
```
// 섭씨 기준 온도센서 값  
__getTemperature($('HamsterS*0:temperature').d, '°C');  
  
// 화씨 기준 온도센서 값  
__getTemperature($('HamsterS*0:temperature').d, '°F');
```

## 파이썬 코드

```
# 섭씨 기준 온도센서 값  
__getTemperature(__('HamsterS*0:temperature').d, '°C')  
  
# 화씨 기준 온도센서 값  
__getTemperature(__('HamsterS*0:temperature').d, '°F')
```

## 신호 세기 값

햄스터 S 의 신호 세기 값을 가져옵니다.



## 자바스크립트 코드

```
// 신호 세기 값  
$('HamsterS*0:signal_strength').d;
```

## 파이썬 코드

```
# 신호 세기 값  
__('HamsterS*0:signal_strength').d
```

## 배터리 충전 상태 값

햄스터 S 의 배터리 충전 상태 값을 가져옵니다.



## 햄스터 S : 배터리

배터리의 충전 상태 값

### 자바스크립트 코드

```
// 배터리 충전 상태 값  
$(HamsterS*0:battery.level).d;
```

### 파이썬 코드

```
# 배터리 충전 상태 값  
__('HamsterS*0:battery.level').d
```

### 상태 변경 여부

햄스터 S 의 상태 변경 여부를 참 (1) / 거짓 (0) (으)로 반환합니다.



로봇의 상태가 변했는지 여부

✓ 앞으로 기울였는가

뒤로 기울였는가

왼쪽으로 기울였는가

오른쪽으로 기울였는가

거꾸로 뒤집어졌는가

뒤집어지지 않았는가

장애물/손을 감지했는가

두드렸는가

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
condition	드롭다운 옵션	위치 조건	앞으로 기울였는가, 뒤로 기울였는가, 왼쪽으로 기울였는가, 오른쪽으로 기울였는가, 거꾸로 뒤집어졌는가, 뒤집어지지 않았는가, 장애물/손을 감지했는가, 두드렸는가

## 자바스크립트 코드

```
// 햄스터 S 가 앞으로 기울였는가?  
$(('HamsterS*0:acceleration.x')).d > 5000;  
  
// 햄스터 S 가 뒤로 기울였는가?  
$(('HamsterS*0:acceleration.x')).d < -5000;
```

```

// 햄스터 S 가 원쪽으로 기울였는가?
$('HamsterS*0:acceleration.y').d > 5000;

// 햄스터 S 가 오른쪽으로 기울였는가?
$('HamsterS*0:acceleration.y').d < -5000;

// 햄스터 S 가 거꾸로 뒤집어졌는가?
$('HamsterS*0:acceleration.z').d > 0;

// 햄스터 S 가 뒤집어지지 않았는가?
$('HamsterS*0:acceleration.z').d < -3000;

// 햄스터 S 가 장애물/손을 감지했는가?
$('HamsterS*0:proximity.left').d > 50 || $('HamsterS*0:proximity.right').d > 50;

// 햄스터 S 를 두드렸는가?
$('HamsterS*0:acceleration.tap').e;

```

## 파이썬 코드

```

# 햄스터 S 가 앞으로 기울였는가?
__('HamsterS*0:acceleration.x').d > 5000

# 햄스터 S 가 뒤로 기울였는가?
__('HamsterS*0:acceleration.x').d < -5000

# 햄스터 S 가 왼쪽으로 기울였는가?
__('HamsterS*0:acceleration.y').d > 5000

# 햄스터 S 가 오른쪽으로 기울였는가?
__('HamsterS*0:acceleration.y').d < -5000

# 햄스터 S 가 거꾸로 뒤집어졌는가?
__('HamsterS*0:acceleration.z').d > 0

# 햄스터 S 가 뒤집어지지 않았는가?
__('HamsterS*0:acceleration.z').d < -3000

# 햄스터 S 가 장애물/손을 감지했는가?
__('HamsterS*0:proximity.left').d > 50 or __('HamsterS*0:proximity.right').d > 50

# 햄스터 S 를 두드렸는가?
__('HamsterS*0:acceleration.tap').e

```

## 입출력 포트 입력 모드 설정하기

햄스터 S 입출력 포트의 입력 모드를 설정합니다.

A 포트, B 포트 또는 A 와 B 포트의 입력 모드를 설정할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	설정 입출력 포트	A(a), B(b), A 와 B(a,b)
mode	드롭다운 옵션	포트 동작 모드	아날로그 입력, 디지털 입력, 디지털 입력(풀업), 디지털 입력(풀다운), 전압 입력, 서보 출력, pwm 출력, 디지털 출력

## 자바스크립트 코드

```
// 햄스터 S 포트 A 를 아날로그 입력으로 정하기  
$(HamsterS*0:io.a.mode').d = 0;
```

```
// 햄스터 S 포트 A 를 디지털 입력으로 정하기  
$(HamsterS*0:io.a.mode').d = 1;
```

```

// 햄스터 S 포트 B 를 디지털 입력 (풀업) 으로 정하기
$('HamsterS*0:io.b.mode').d = 2;

// 햄스터 S 포트 B 를 디지털 입력 (풀다운) 으로 정하기
$('HamsterS*0:io.b.mode').d = 3;

// 햄스터 S 포트 A 와 B 를 전압 입력으로 정하기
$('HamsterS*0:io.a.mode').d = 4;
$('HamsterS*0:io.b.mode').d = 4;

// 햄스터 S 포트 A 와 B 를 서보 출력으로 정하기
$('HamsterS*0:io.a.mode').d = 8;
$('HamsterS*0:io.b.mode').d = 8;

// 햄스터 S 포트 A 와 B 를 pwm 출력으로 정하기
$('HamsterS*0:io.a.mode').d = 9;
$('HamsterS*0:io.b.mode').d = 9;

// 햄스터 S 포트 A 와 B 를 디지털 출력으로 정하기
$('HamsterS*0:io.a.mode').d = 10;
$('HamsterS*0:io.b.mode').d = 10;

```

## 파이썬 코드

```

# 햄스터 S 포트 A 를 아날로그 입력으로 정하기
__('HamsterS*0:io.a.mode').d = 0

# 햄스터 S 포트 A 를 디지털 입력으로 정하기
__('HamsterS*0:io.a.mode').d = 1

# 햄스터 S 포트 B 를 디지털 입력 (풀업) 으로 정하기
__('HamsterS*0:io.b.mode').d = 2

# 햄스터 S 포트 B 를 디지털 입력 (풀다운) 으로 정하기
__('HamsterS*0:io.b.mode').d = 3

# 햄스터 S 포트 A 와 B 를 전압 입력으로 정하기
__('HamsterS*0:io.a.mode').d = 4
__('HamsterS*0:io.b.mode').d = 4

# 햄스터 S 포트 A 와 B 를 서보 출력으로 정하기
__('HamsterS*0:io.a.mode').d = 8
__('HamsterS*0:io.b.mode').d = 8

# 햄스터 S 포트 A 와 B 를 pwm 출력으로 정하기
__('HamsterS*0:io.a.mode').d = 9
__('HamsterS*0:io.b.mode').d = 9

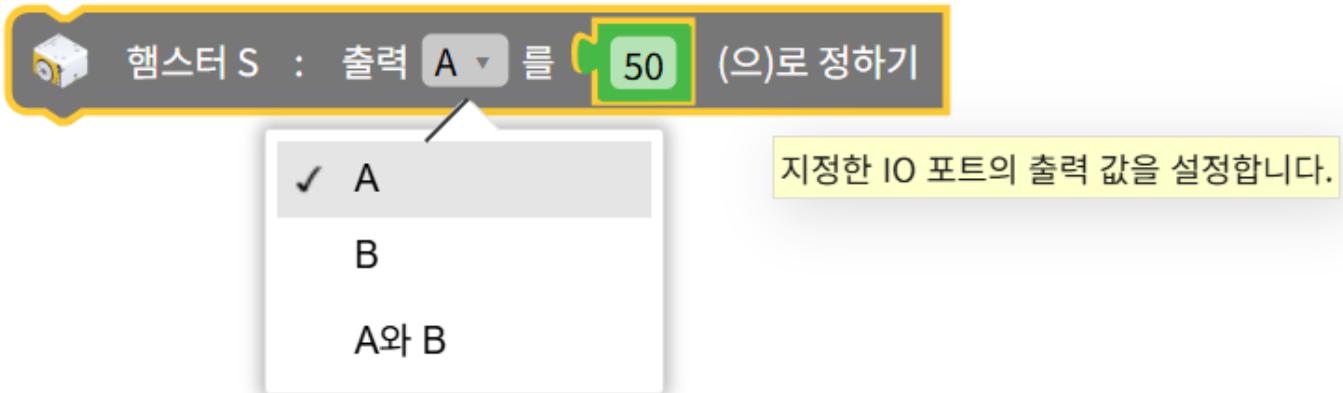
# 햄스터 S 포트 A 와 B 를 디지털 출력으로 정하기
__('HamsterS*0:io.a.mode').d = 10
__('HamsterS*0:io.b.mode').d = 10

```

## 입출력 포트 출력값 설정하기

햄스터 S 입출력 포트의 출력값을 설정합니다.

A 포트, B 포트 또는 A 와 B 포트의 입력 모드를 설정할 수 있습니다.



## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	설정 입출력 포트	A(a), B(b), A 와 B(a,b)
output	입력값	입출력 포트 출력값	0 ~ 180

## 자바스크립트 코드

```
// 햄스터 S 포트 A 출력값을 180 으로 정하기
$(HamsterS*0:io.a.out').d = 180;

// 햄스터 S 포트 B 출력값을 0 으로 정하기
$(HamsterS*0:io.b.out').d = 0;

// 햄스터 S 포트 A,B 출력값을 100 으로 정하기
$(HamsterS*0:io.a.out').d = 100;
$(HamsterS*0:io.b.out').d = 100;
```

## 파이썬 코드

```
# 햄스터 S 포트 A 출력값을 180 으로 정하기
__('HamsterS*0:io.a.out').d = 180

# 햄스터 S 포트 B 출력값을 0 으로 정하기
__('HamsterS*0:io.b.out').d = 0

# 햄스터 S 포트 A,B 출력값을 100 으로 정하기
__('HamsterS*0:io.a.out').d = 100
__('HamsterS*0:io.b.out').d = 100
```

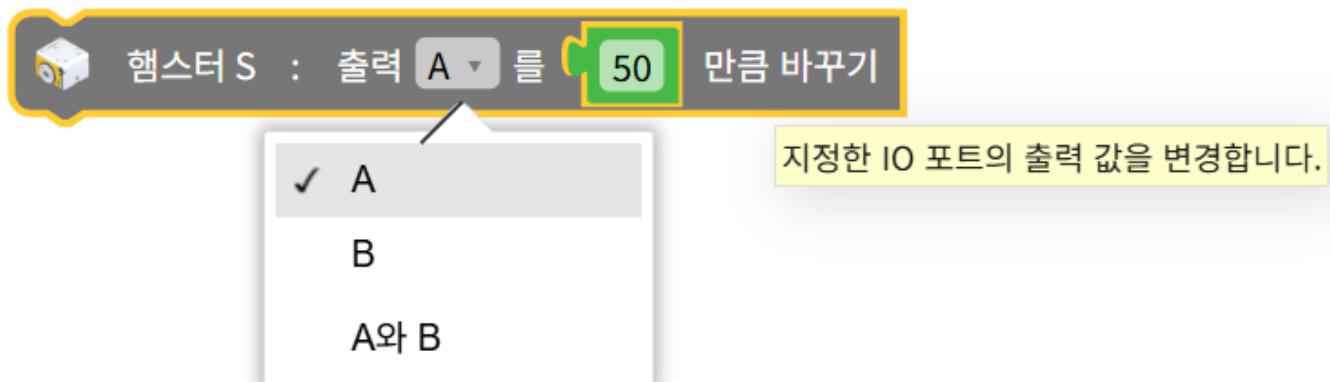
## 입출력 포트 출력값 변경하기

햄스터 S 입출력 포트의 출력값을 변경합니다.

현재의 포트 출력값에 입력한 출력값을 더한 값이 새로운 포트 출력값이 됩니다.

새롭게 설정된 포트 출력값 범위는 0 ~ 180으로 설정됩니다.

A 포트, B 포트 또는 A 와 B 포트의 입력 모드를 설정할 수 있습니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	설정 입출력 포트	A(a), B(b), A 와 B(a,b)
value	입력값	입출력 포트 변경값	정수

### 자바스크립트 코드

```
// 햄스터 S 포트 A 출력값을 10 만큼 변경하기
$(HamsterS*0:io.a.out').d = $(HamsterS*0:io.a.out').d + 10;

// 햄스터 S 포트 B 출력값을 20 만큼 변경하기
$(HamsterS*0:io.b.out').d = $(HamsterS*0:io.b.out').d + 20;

// 햄스터 S 포트 A,B 출력값을 30 만큼 변경하기
$(HamsterS*0:io.a.out').d = $(HamsterS*0:io.a.out').d + 30;
$(HamsterS*0:io.b.out').d = $(HamsterS*0:io.b.out').d + 30;
```

### 파이썬 코드

```
# 햄스터 S 포트 A 출력값을 10 만큼 변경하기
__('HamsterS*0:io.a.out').d = __('HamsterS*0:io.a.out').d + 10

# 햄스터 S 포트 B 출력값을 20 만큼 변경하기
__('HamsterS*0:io.b.out').d = __('HamsterS*0:io.b.out').d + 20

# 햄스터 S 포트 A,B 출력값을 30 만큼 변경하기
__('HamsterS*0:io.a.out').d = __('HamsterS*0:io.a.out').d + 30
__('HamsterS*0:io.b.out').d = __('HamsterS*0:io.b.out').d + 30
```

### 집게 열기 / 닫기

햄스터 S 의 집게를 열거나 닫습니다.



### 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
toggle	드롭다운 옵션	집게 토글	열기 (1), 닫기 (2)

### 자바스크립트 코드

```
// 햄스터 S 집게 열기
$('HamsterS*0:io.gripper').d = 1;

// 햄스터 S 집게 닫기
$('HamsterS*0:io.gripper').d = 2;
```

### 파이썬 코드

```
# 햄스터 S 집게 열기
__('HamsterS*0:io.gripper').d = 1

# 햄스터 S 집게 닫기
__('HamsterS*0:io.gripper').d = 2
```

### 슈터 각도 설정하기

햄스터 S 의 슈터 각도를 설정하여 제어합니다.

각도의 범위는 0 ~ 255 입니다.



햄스터 S : 슈터 각도를 45 (으)로 정하기

45

슈터 각도를 설정하여 제어합니다. 각도의 범위는 0 ~ 255 입니다.

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
angle	입력값	슈터 각도	0 ~ 255 사이 정수

## 자바스크립트 코드

```
// 햄스터 S 슈터 각도 255 도로 정하기
$(`'HamsterS*0:io.shooter').d = 255;

// 햄스터 S 슈터 각도 0 도로 정하기
$(`'HamsterS*0:io.shooter').d = 0;
```

## 파이썬 코드

```
# 햄스터 S 슈터 각도 255 도로 정하기
__(`'HamsterS*0:io.shooter').d = 255

# 햄스터 S 슈터 각도 0 도로 정하기
__(`'HamsterS*0:io.shooter').d = 0
```

## 입출력 포트 입력 값

햄스터 S 의 입출력 포트 입력 값을 가져옵니다.

The screenshot shows a Scratch-like programming environment. A yellow-bordered block labeled "햄스터 S : 입력" is selected. To its right, a callout box contains the text "입출력 포트의 입력 값". Below the main block, there is a dropdown menu with two options: "A" (selected) and "B".

## 드롭다운 옵션 및 입력값

이름	구분	설명	범위 / 종류
port	드롭다운 옵션	입출력 포트	A(a), B(b)

## 자바스크립트 코드

```
// 햄스터 S A 포트 입출력 포트 값  
$('HamsterS*0:io.a.in').d;  
  
// 햄스터 S B 포트 입출력 포트 값  
$('HamsterS*0:io.b.in').d;
```

## 파이썬 코드

```
# 햄스터 S A 포트 입출력 포트 값  
__('HamsterS*0:io.a.in').d  
  
# 햄스터 S B 포트 입출력 포트 값  
__('HamsterS*0:io.b.in').d
```