

**Zumi**  
*BLOCKLY* *for*  
**ROBOLINK**



00

소개

*Introduction*

# 블록클리 소개

Introduction to Blockly



# 1. 주미를 연결하고 코딩모드를 클릭합니다.

※ 주미연결에 대해서는 매뉴얼을 참조해주세요

v1.7 이상에서만 BLOCKLY가 활성화 됩니다.



## 2. 새로운 블럭클리를 만들어 줍니다.

※ 블럭클리 만들기를 눌러 새로운 블럭클리를 만들고 이름을 정해 프로젝트를 생성해 줍니다.

### 코드 모드

자신만의 주피터 노트북 프로젝트와 블럭클리 프로젝트를 만들어 보세요.

+ 새로운 주피터 만들기

+ 새로운 블럭클리 만들기

1



x



블럭클리 프로젝트 이름:

프로젝트 생성하기

2

# 3. 인터페이스를 살펴봅시다.

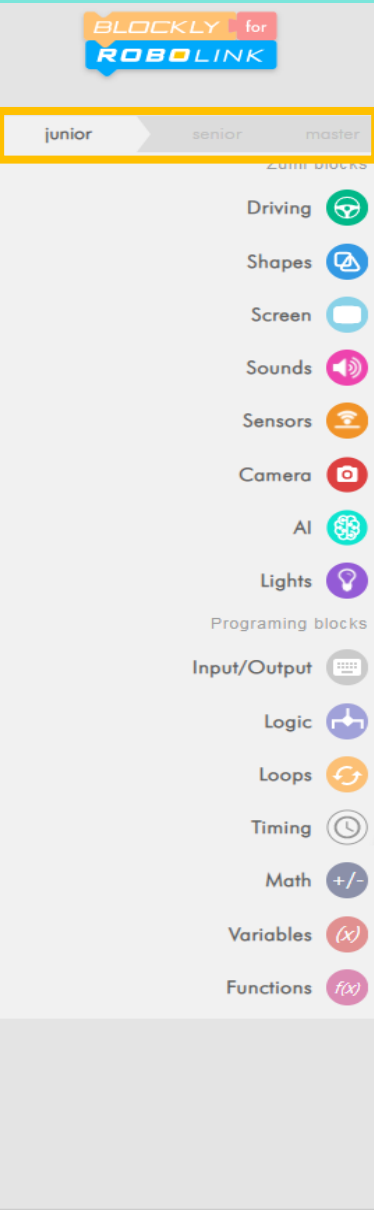
※ 업데이트 시 인터페이스가 변경 될 수 있습니다.

The image shows a screenshot of the Blockly for RoboLink interface. The interface is divided into several sections:

- Top Bar:** Contains the 'BLOCKLY for ROBO LINK' logo, a menu icon, a 'Run code' button (green play icon), and a 'Stop' button (red stop icon).
- Left Panel (Zumi blocks):** Lists various block categories: Driving, Shapes, Screen, Sounds, Sensors, Camera, AI, Lights, and Programming blocks. The '주미 블록' (Zumi blocks) category is highlighted.
- Right Panel (Callouts):** Explains the functions of the top bar buttons:
  - 메뉴 (Menu):** Opens a menu with options: Save (Save file), Undo (Restore file), Redo (Save file again), and System Requirements (Check system requirements).
  - 코드 실행 (Run code):** Executes the code on the 'blockboard' (※ '블록보드'의 코드를 실행 시킵니다.).
  - 정지 (Stop):** Stops the code on the 'blockboard' (※ '블록보드'의 코드를 정지 시킵니다.).
- Bottom Panel (Console):** Shows the 'console' area where the execution status of the Zumi robot can be confirmed (주미의 실행 상태를 확인 할 수 있습니다.). It includes a 'clear console' button.
- Right Panel (Callouts):** Explains the functions of the blockboard controls:
  - 블록이 있는 위치로 이동합니다.** (Move to the location of the block).
  - '블록보드'를 확대합니다.** (Zoom in the blockboard).
  - '블록보드'를 축소합니다.** (Zoom out the blockboard).
  - '블록보드'의 블록을 제거합니다.** (Remove blocks from the blockboard).
- Python Callout:** Explains that selected blocks are converted to Python code (짜여진 블록을 파이썬으로 변환합니다.).

### 3. 인터페이스를 살펴봅시다.

※ 업데이트 시 인터페이스가 변경 될 수 있습니다.



junior

주니어

Zumi 블록클리의 간단한 기능을 소개합니다.

senior

시니어

조금 더 상위 난위도의 블록클리를 지원합니다.

master

마스터

더 어려운 난이도의 블록클리 블록을 지원합니다.

# 01

## 주니어

*junior*

# 블록설명

Introduction to Blocks





Run code



Stop

junior senior master

Zumi blocks

Driving

Shapes

Screen

Sounds

Sensors

Camera

AI

Lights

Programing blocks

Input/Output

Logic

Loops

Timing

Math

Variables

Functions



# Driving

주미를 이동시키는 블록입니다.  
주미가 움직이는 시간을 설정하여  
주미를 이동시킬 수 있습니다.

↑ forward for 0 seconds

↓ reverse for 0 seconds

↶ turn left 30 °

↷ turn right 30 °

🖐 brake

↶ left U-turn with 0 speed

↷ right U-turn with 0 speed

🚗 parallel park

🔧 calibrate gyro



Python



clear console





	<p>주미를 0 초 만큼 전진 시킵니다.</p>
	<p>주미를 0 초 만큼 후진 시킵니다.</p>
	<p>주미를 0 도 만큼 왼쪽으로 회전 시킵니다.</p>
	<p>주미를 0 도 만큼 오른쪽으로 회전 시킵니다.</p>
	<p>주미를 이동을 멈춥니다.</p>
	<p>주미를 0 의 속도 만큼 왼쪽으로 U-턴 시킵니다.</p>
	<p>주미를 0 의 속도 만큼 오른쪽으로 U-턴 시킵니다.</p>
	<p>주미를 후진시켜 주차 합니다.</p>
	<p>자이로 센서를 캘리브레이션 합니다.</p>



Run code



Stop





Python



# Shapes

주미를 이동시키는 블록입니다.  
일정 모양과 방향을 설정해주면  
설정 방향으로 주미를 이동 시킵니다.

 left ▾ triangle

 left ▾ square

 rectangle


 left ▾ circle

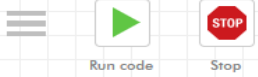
 figure 8

 J-turn

console

clear console

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



	주미를 왼쪽 방향으로 '삼각형 모양'으로 주행시킵니다.
	주미를 왼쪽 방향으로 '사각형 모양'으로 주행시킵니다.
	주미를 직사각형 모양으로 주행시킵니다.
	주미를 왼쪽 방향으로 '원 모양'으로 주행시킵니다.
	주미를 '8자 모양'으로 주행시킵니다.
	주미가 'J자 모양'으로 턴(회전) 합니다.

console

clear console

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions


















## Screen

주미의 화면을 변경시킵니다.  
메시지를 쓰거나 주미의 표정을  
변경시킬 수 있습니다.

- draw text message
- sad
- closed eyes
- sleepy eyes
- happy eyes
- glimmer eyes
- blinking eyes
- angry eyes
- open eyes

console

clear console

- junior
- senior
- master
- Zumi blocks
- Driving 
- Shapes 
- Screen 
- Sounds 
- Sensors 
- Camera 
- AI 
- Lights 
- Programing blocks
- Input/Output 
- Logic 
- Loops 
- Timing 
- Math 
- Variables 
- Functions 



draw text message	주미의 화면에 message 를 표시합니다.
sad	주미가 슬픈 표정을 짓습니다.
closed eyes	주미가 눈을 감습니다.
sleepy eyes	주미가 피곤한 표정을 짓습니다.
happy eyes	주미가 행복한 표정을 짓습니다.
glimmer eyes	주미가 눈을 희미하게 뜹니다.
blinking eyes	주미가 눈을 깜빡입니다.
angry eyes	주미가 화난 표정을 짓습니다.
open eyes	주미가 눈을 뜹니다.

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



# Sounds

주미의 소리를 설정 할 수 있습니다.  
정해져 있는 소리를 내게 하거나  
음과 시간 설정을 하여 소리를  
내는 것도 가능합니다.

play note for 10 millisec

angry

happy

blink
















celebrate

wake up

desoriented

oops front

oops back

- junior
- senior
- master
- Zumi blocks
- Driving 
- Shapes 
- Screen 
- Sounds 
- Sensors 
- Camera 
- AI 
- Lights 
- Programing blocks
- Input/Output 
- Logic 
- Loops 
- Timing 
- Math 
- Variables 
- Functions 



play note for 10 millisec	주미가 note 의 소리를 10초 동안 냅니다.
angry	주미가 화내는 소리를 냅니다.
happy	주미가 행복해 하는 소리를 냅니다.
blink	주미가 깜빡이는 소리를 냅니다.
celebrate	주미가 축하하는 소리를 냅니다.
wake up	주미가 깨어나는 소리를 냅니다.
desoriented	주미가 지속적인 소리를 냅니다.
oops front	주미가 앞쪽에 사고가 났을때의 소리를 냅니다.
oops back	주미가 뒤쪽에 사고가 났을때의 소리를 냅니다.

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors**
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



## Sensors

주미가 감지하는 센서 값을 받아옵니다.

- get IR reading front right ▾
- get z angle
- get x angle
- get y angle
- reset gyro
- get battery voltage
- get battery percentage

console

clear console

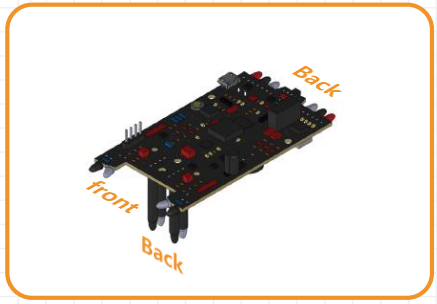


- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors**
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions

Run code Stop

get IR reading front right ▾	주미의 센서 값을 받아옵니다.
get z angle	주미 자이로 센서의 (Z값)을 받아옵니다.
get x angle	주미 자이로 센서의 (X값)을 받아옵니다.
get y angle	주미 자이로 센서의 (Y값)을 받아옵니다.
reset gyro	주미 자이로 센서를 리셋 합니다.
get battery voltage	주미의 배터리 전압 값을 받아옵니다.
get battery percentage	주미의 배터리 잔량(%) 값을 받아옵니다.

- ✓ front right
- bottom right
- back right
- bottom left
- back left
- front left



console

clear console



Run code



Stop

junior senior master

Zumi blocks

Driving

Shapes

Screen

Sounds

Sensors

Camera

AI

Lights

Programing blocks

Input/Output

Logic

Loops

Timing

Math

Variables

Functions



# Camera

주미의 카메라를 켜거나  
값을 불러오는 블록 입니다.

import camera

start camera

close camera

take picture

show image
















get QR code message

find stop sign

find face

console

clear console

- junior
- senior
- master
- Zumi blocks
- Driving 
- Shapes 
- Screen 
- Sounds 
- Sensors 
- Camera **
- AI 
- Lights 
- Programing blocks
- Input/Output 
- Logic 
- Loops 
- Timing 
- Math 
- Variables 
- Functions 



<b>import camera</b>	주미의 카메라 라이브러리를 포함시킵니다.
<b>start camera</b>	주미의 카메라를 ON 합니다
<b>close camera</b>	주미의 카메라를 OFF 합니다.
<b>take picture</b>	주미의 카메라로 사진을 찍습니다.
<b>show image</b>	괄호[] 안에 있는 이미지를 보여줍니다.
<b>get QR code message</b>	QR 코드의 메시지 값을 얻어옵니다.
<b>find stop sign</b>	Stop sign을 찾습니다.
<b>find face</b>	Face 이미지를 찾습니다.

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



**AI**

AI 훈련(러닝)을 필요로 하거나 값을 가져오는 블록 입니다.

Train a KNN model

Get model

predict from frame

load KNN model RGB

RED

BLUE

GREEN

X

Please select a model to load

RGB

Load model

console

clear console

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programming blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



<b>Train a KNN model</b>	KNN 모델을 트레이닝 하여 생성합니다
<b>Get model</b>	러닝된 모델을 불러 옵니다. (모델을 생성하고 가져오면 블록이 추가로 생성됩니다.)
<b>predict from frame</b>	괄호[]의 프레임 예측합니다.

console

clear console



Run code



Stop



Python

junior senior master

Zumi blocks

Driving

Shapes

Screen

Sounds

Sensors

Camera

AI

Lights

Programing blocks

Input/Output

Logic

Loops

Timing

Math

Variables

Functions



# Lights

주미의 LED를 조절합니다.  
각 LED를 켜거나 끌 수 있습니다.

lights on

hazard lights on

lights off

hazard lights off

headlights on

left signal on

headlights off

left signal off

brake lights on

right signal on

brake lights off

right signal off

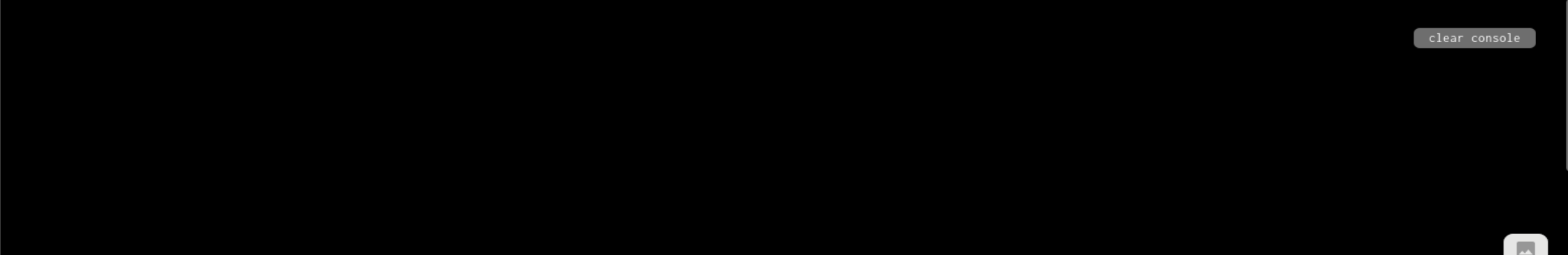
console

clear console



lights on	lights off	주미의 모든 LED를 켭니다/끽니다
headlights on	headlights off	주미의 헤드라이트 LED를 켭니다/끽니다
brake lights on	brake lights off	주미의 브레이크 라이트 LED를 켭니다/끽니다
hazard lights on	hazard lights off	주미의 비상등 LED를 켭니다/끽니다
left signal on	left signal off	왼쪽 신호 LED를 켭니다/끽니다
right signal on	right signal off	오른쪽 신호 LED를 켭니다/끽니다

console



clear console




Run code



Stop



Python

 **Input/Output**

입력·출력에 관련된 블록이 있습니다

print

“ ”

console

clear console





Run code



Stop



Python

	오른쪽에 끼워진 블록을 출력합니다.
	왼쪽에 끼워진 블록에 따옴표 안의 내용을 대입합니다.

console

clear console



# Logic

코드를 실행하기 위한 조건을 만들 수 있습니다.

A vertical stack of logic blocks from the Blockly interface. From top to bottom: an 'if' block with a gear icon and a 'do' block; an equals sign '=' block; an 'and' block; a 'not' block; and a 'true' block with a dropdown arrow.

console

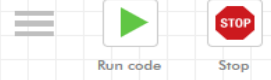
clear console



	만약 ( 오른쪽 블록 ) 이라면 (오른쪽 블록) 을 실행합니다.
	( 왼쪽 블록 ) 과 (오른쪽 블록) 이 같다
	( 왼쪽 블록 ) 그리고 (오른쪽 블록) 이
	(오른쪽 블록) 이 아니다
	(왼쪽 블록) 이 참

console

clear console



# Loops

코드에 대한 반복문을 설정할 수 있습니다.

The screenshot shows several Blockly loop blocks:

- A 'repeat 10 times' block with a 'do' sub-block.
- A 'repeat while' block with a 'do' sub-block.
- A 'count with i from 1 to 10 by 1' block with a 'do' sub-block.
- A 'for each item i in list' block with a 'do' sub-block.
- A 'break out of loop' block.

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



	10번 동안 (아래쪽 블록) 을 반복합니다.
	(오른쪽 블록) 만큼 (아래쪽 블록) 을 반복합니다.
	1 에서 10까지 1씩 카운터 하며 (아래쪽 블록) 을 반복합니다.
	(오른쪽 블록) 의 리스트 대로 (아래쪽 블록) 을 반복합니다.
	반복을 멈춥니다.

console

clear console



Run code



Stop



Python



# Timing

코드 사이의 간격 타이머를 설정  
할 수 있습니다.



A code block with a clock icon on the left, the text "wait 0 seconds" in the center, and a small tab on the left side.

console

clear console



Run code



Stop



# Math

수식 또는 함수의 값을 설정 할 수 있습니다.

0

1 + 1

square root 9

sin 45

π

0 is even

round 3.1

sum of list

remainder of 64 ÷ 10

constrain 50 low 1 high 100

random integer from 1 to 100

random fraction

console

clear console

Zumi blocks

- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programming blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions

Run code Stop

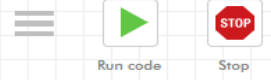
	0 을 대입합니다.
	1 + 1 을 대입합니다.
	$\sqrt{9}$ 를 대입합니다.
	sin 45 를 대입합니다.
	$\pi$ 를 대입합니다.
	0도 를 대입합니다.

console

clear console



	3.1 크기의 원 을 대입합니다.
	(왼쪽 블록) 리스트를 더하기 합니다
	64 나누기 10의 나머지 를 대입합니다.
	1 ~ 100 사이 중 50가지 숫자를 제안합니다.
	1 ~ 100 사이 랜덤한 정수를 대입합니다.
	랜덤한 분수를 대입합니다.



# Variables

함수명을 지정하거나 함수를 불러올수 있습니다.

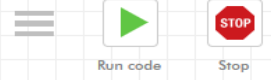
Create variable...

```
set i to
```

```
get i
```

console

clear console



$f(x)$  **Function**

특수 기능을 가진 블록을 만들거나 설정 할 수 있습니다.

do something

do something2

if return

do something

do something2

- junior
- senior
- master
- Zumi blocks
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights
- Programing blocks
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



여러 개의 블록을 한번에 사용하거나, 하나의 로직 블록을 여러 번 반복해야 할 때 이 블록을 사용합니다.

ear console

# 01

## 주니어



*junior*

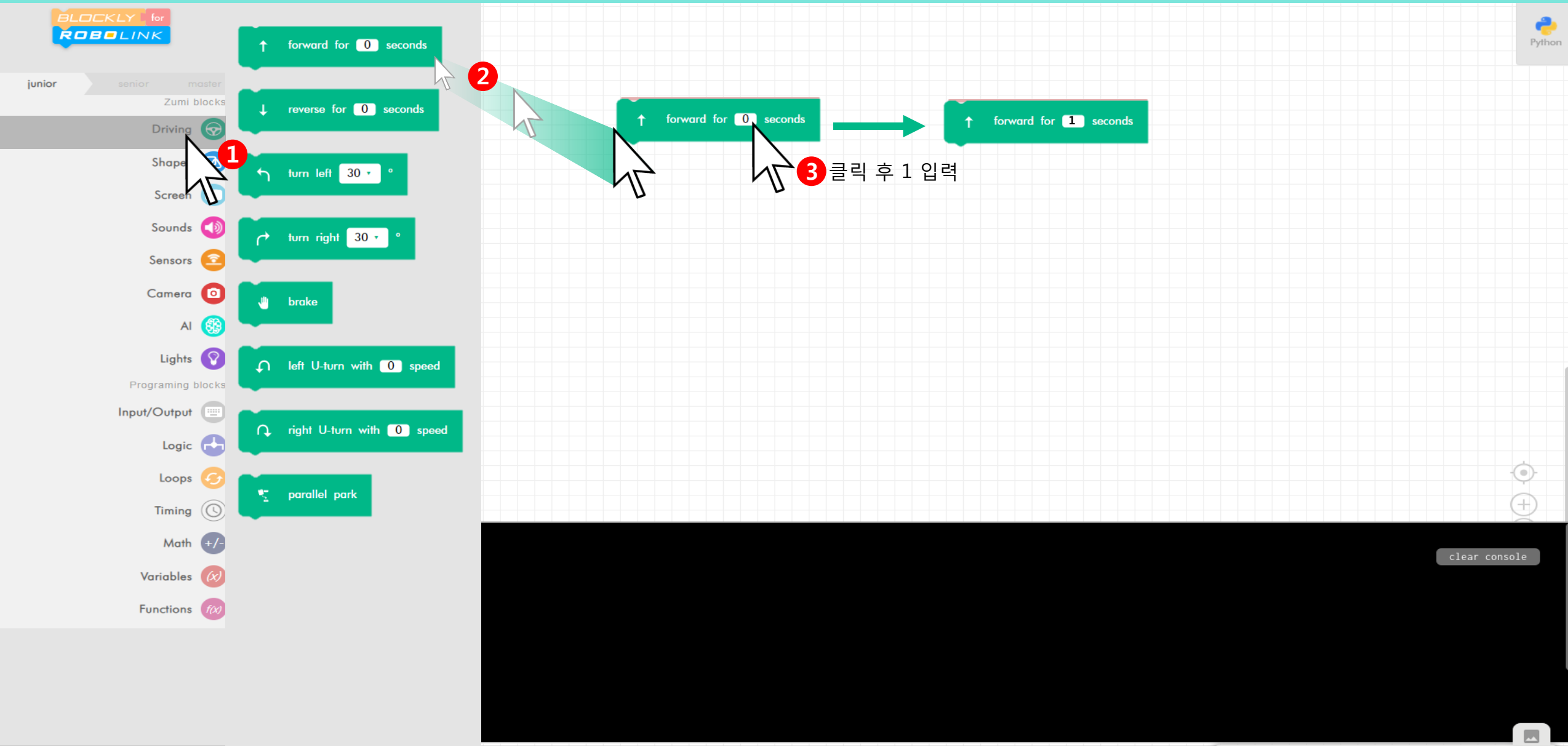
# 예제 따라하기

Follow the example



# 실행하기

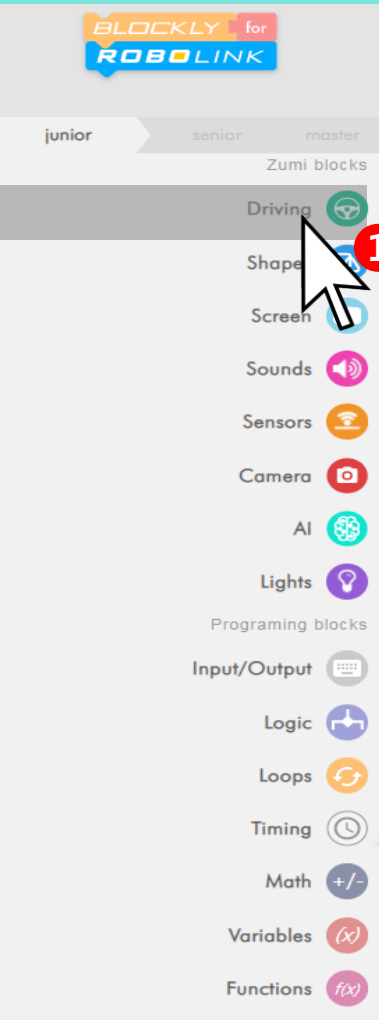
Driving  을 선택하고  를 드래그 하여 블록필드에 가져옵니다.



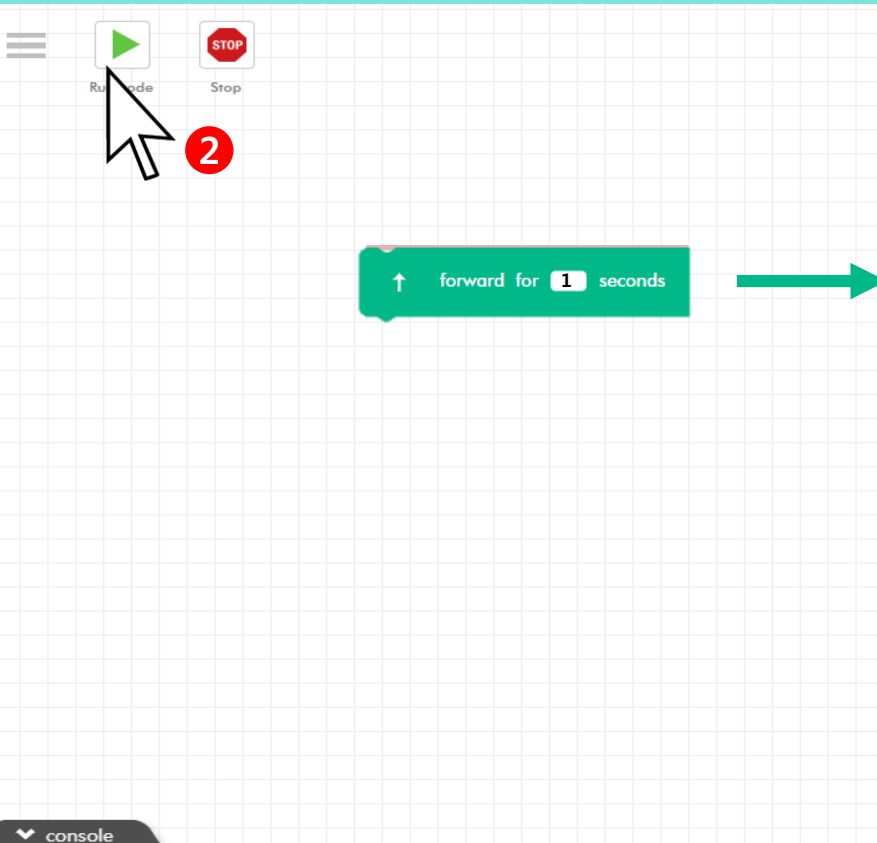
The screenshot shows the Blockly for RoboLink interface. On the left, the 'Driving' category is selected, and the 'forward for 0 seconds' block is highlighted with a red circle '1'. A mouse cursor is shown dragging this block from the palette to the workspace, indicated by a red circle '2'. In the workspace, the block is placed on a grid. A second mouse cursor is shown clicking on the '0' input field, with a red circle '3' and the text '클릭 후 1 입력' (Click and input 1). An arrow points from the '0' field to the '1' field, showing the change. The workspace also contains another 'forward for 1 seconds' block. The interface includes a sidebar with categories like 'Zumi blocks', 'Driving', 'Shape', 'Screen', 'Sounds', 'Sensors', 'Camera', 'AI', 'Lights', 'Input/Output', 'Logic', 'Loops', 'Timing', 'Math', 'Variables', and 'Functions'. A 'clear console' button is visible in the bottom right corner.

# 실행하기

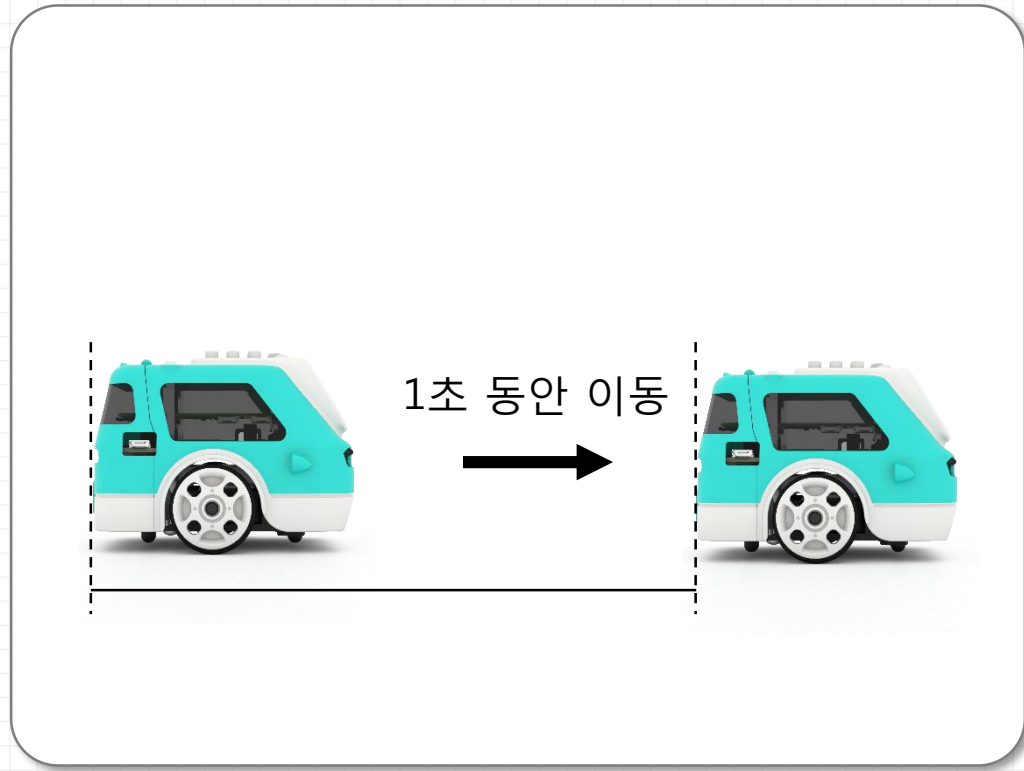
Driving  을 선택하여 창을 닫고  을 눌러  를 실행합니다.



Blockly interface sidebar showing categories: junior, senior, master. Under 'Zumi blocks', 'Driving' is selected and highlighted with a red circle and a mouse cursor icon labeled '1'. Other categories include Shape, Screen, Sounds, Sensors, Camera, AI, Lights, and Programming blocks. Under 'Programming blocks', 'Input/Output', Logic, Loops, Timing, Math, Variables, and Functions are listed.



Blockly workspace showing a 'Run code' button with a green play icon and a 'Stop' button with a red stop sign, both with red circles and mouse cursor icons labeled '2'. Below them is a code block: 'forward for 1 seconds' with a green arrow pointing to the right.

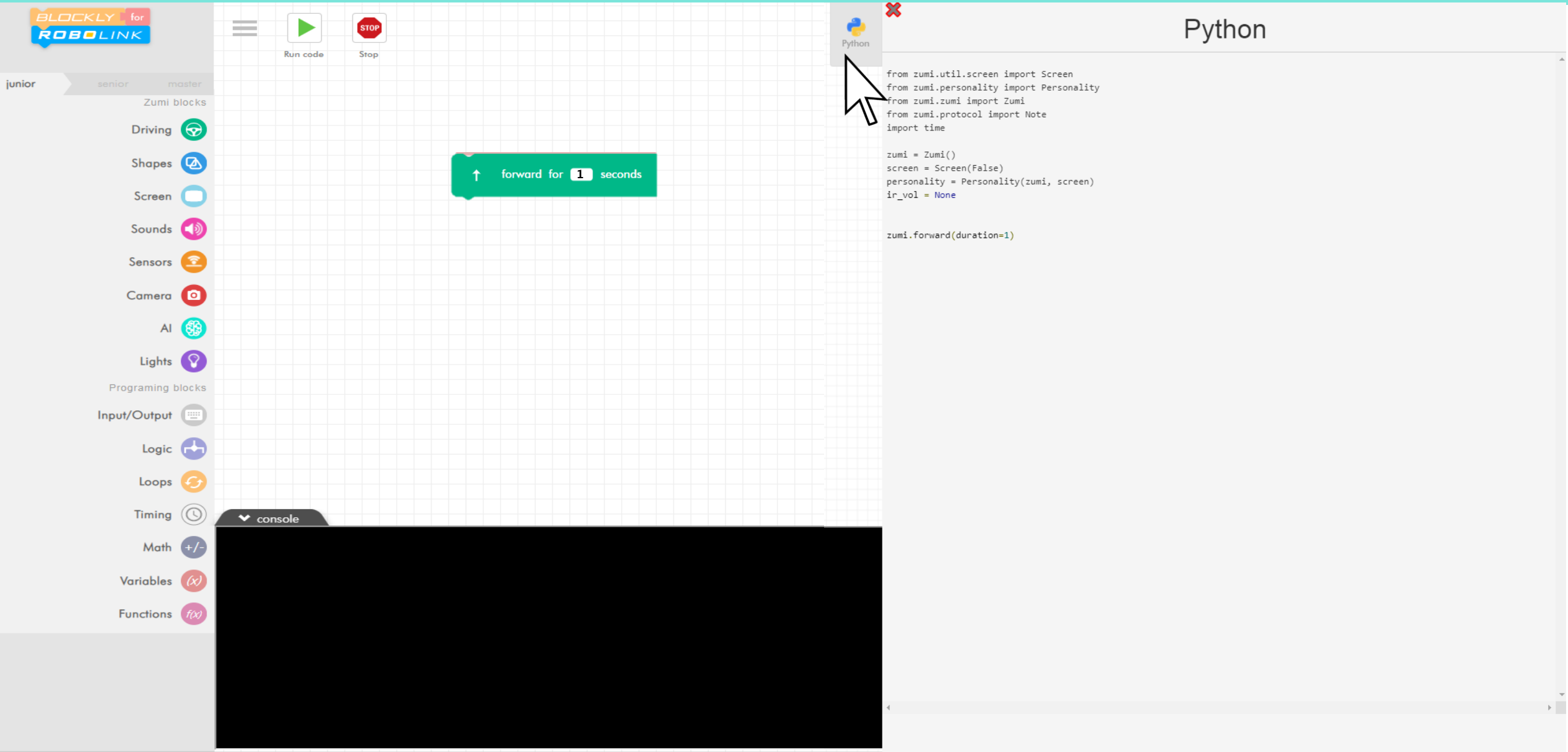


Console area with a 'console' dropdown menu and a 'clear console' button. The area is currently empty.



# 실행하기

우측의  버튼을 누르면 코딩한 블록을 Python 코드로 변환 할 수도 있습니다.



The screenshot displays the Blockly for RoboLink environment. On the left, a sidebar contains various block categories: Zumi blocks (Driving, Shapes, Screen, Sounds, Sensors, Camera, AI, Lights), Programming blocks (Input/Output, Logic, Loops, Timing, Math, Variables, Functions), and a console window at the bottom. The main workspace features a green block with the text "forward for 1 seconds". Above the workspace are "Run code" and "Stop" buttons. On the right, a "Python" panel is open, showing the equivalent Python code:

```
from zumi.util.screen import Screen
from zumi.personality import Personality
from zumi.zumi import Zumi
from zumi.protocol import Note
import time

zumi = Zumi()
screen = Screen(False)
personality = Personality(zumi, screen)
ir_vol = None

zumi.forward(duration=1)
```



# 센서 값 가져오기

주미가 읽는 적외선 센서 값을 가져와 봅시다.

BLOCKLY for ROBOTLINK

junior senior master

Zumi blocks

- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights

Programing blocks

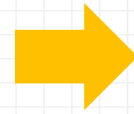
- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



## get IR reading

get IR reading front right ▾

- ✓ front right
- bottom right
- back right
- bottom left
- back left
- front left



print get IR reading front right ▾

145

## 파이썬 라이브러리

[https://learn.robolink.com/docs/zumi-library/get\\_all\\_ir\\_data](https://learn.robolink.com/docs/zumi-library/get_all_ir_data)



# 센서 값 가져오기

x, y, z 축의 각도값을 가져와 봅시다.

BLOCKLY for ROBOTLINK

junior senior master

Zumi blocks

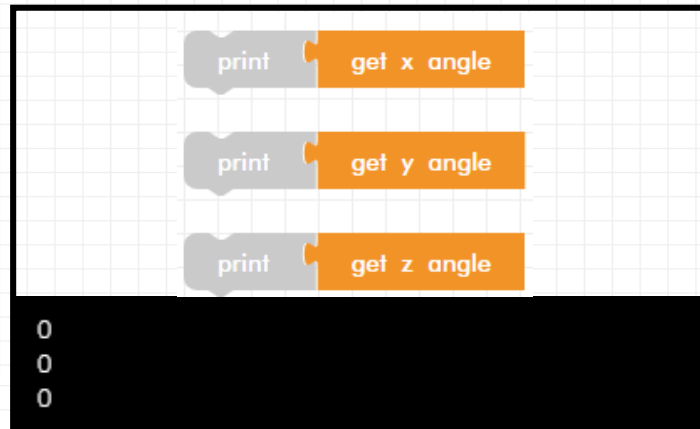
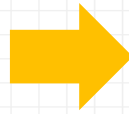
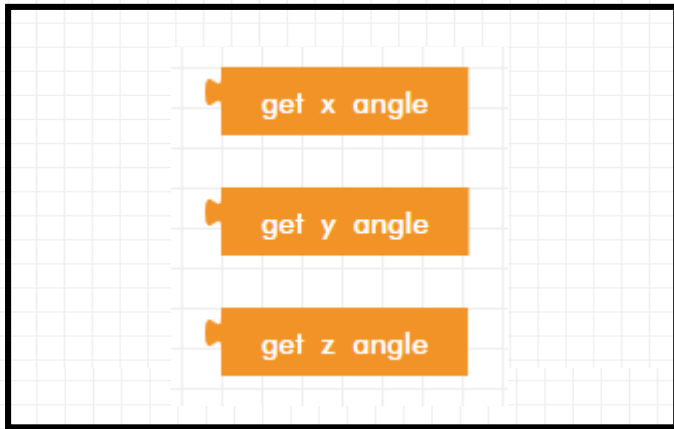
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights

Programing blocks

- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



## get x,y,z angle



## 파이썬 라이브러리

[https://learn.robolink.com/docs/zumi-library/read\\_x\\_angle](https://learn.robolink.com/docs/zumi-library/read_x_angle)

[https://learn.robolink.com/docs/zumi-library/read\\_y\\_angle](https://learn.robolink.com/docs/zumi-library/read_y_angle)

[https://learn.robolink.com/docs/zumi-library/read\\_z\\_angle](https://learn.robolink.com/docs/zumi-library/read_z_angle)

# 센서 값 가져오기

x, y, z 축의 각도값을 가져와 봅니다.

BLOCKLY for ROBO LINK

junior senior master

Zumi blocks

- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights

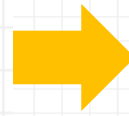
Programing blocks

- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



## While을 사용한 실시간 센서값 출력

```
repeat while true
do print get IR reading front right
```



```
0
0
0
0
0
0
```

# 센서 값 가져오기

자이로 센서를 초기화 해봅시다.

BLOCKLY for ROBOTLINK

junior senior master

Zumi blocks

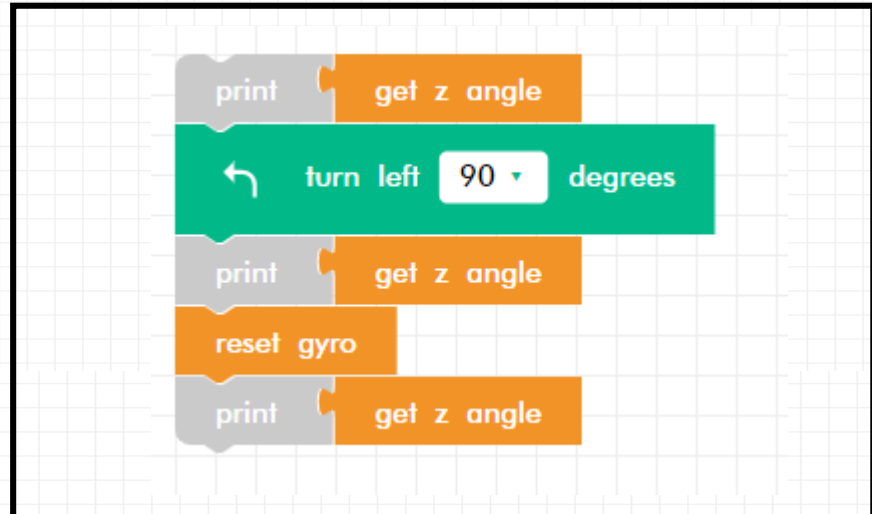
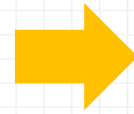
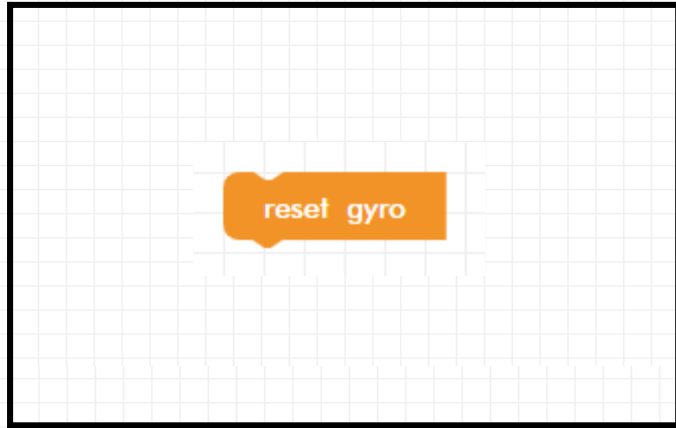
- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights

Programing blocks

- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



## Reset gyro



```
0  
87  
1
```

## 파이썬 라이브러리

[https://learn.robolink.com/docs/zumi-library/get\\_all\\_ir\\_data](https://learn.robolink.com/docs/zumi-library/get_all_ir_data)



# 배터리 값 가져오기

배터리 값을 가져와 봅시다.

BLOCKLY for ROBOTLINK

junior senior master

Zumi blocks

- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights

Programing blocks

- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions

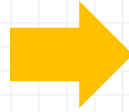


## get battery voltage

get battery voltage    배터리 전압

## get battery percentage

get battery percentage    배터리 잔량



print    get battery voltage

4.103448275862069

print    get battery percentage

100

## 파이썬 라이브러리

[https://learn.robolink.com/docs/\\_old\\_1.2\\_01.zumi-library/get battery voltage](https://learn.robolink.com/docs/_old_1.2_01.zumi-library/get%20battery%20voltage)



# 카메라 사용하기

카메라를 이용하여 사진을 찍어 봅니다.



junior senior master

Zumi blocks

Driving

Shapes

Screen

Sounds

Sensors

Camera

AI

Lights

Programing blocks

Input/Output

Logic

Loops

Timing

Math

Variables

Functions



Run code



Stop

```
import camera
start camera
print take picture
close camera
```

```
Starting PiCamera
[[[103 109 112]
 [103 109 112]
 [104 110 113]
 ...
 [167 202 211]
 [166 201 210]
 [167 202 211]]
 [[103 109 112]
 [103 109 112]
 [103 109 112]
 ...
```

```
import camera
start camera
set frame to take picture
close camera
```

```
...
[175 137 114]
[174 136 115]
[174 136 115]]
[[156 107 83]
 [156 107 83]
 [155 106 82]
 ...
 [174 136 113]
 [174 136 115]
 [174 136 115]]]
Closing PiCamera
```

파이썬 라이브러리

[https://learn.robolink.com/docs/\\_old\\_1.2\\_01.zumi-library/capture](https://learn.robolink.com/docs/_old_1.2_01.zumi-library/capture)



Python

# 카메라 사용하기

카메라를 사용하여 이미지 보는 예제를 해봅시다.



junior senior master

Zumi blocks

Driving

Shapes

Screen

Sounds

Sensors

Camera

AI

Lights

Programing blocks

Input/Output

Logic

Loops

Timing

Math

Variables

Functions



```
import camera
start camera
show image [take picture]
close camera
```

이미지 서랍 상단의 이미지를 클릭하면 해당 이미지를 크게 볼 수 있습니다.

```
import camera
start camera
set frame to [take picture]
show image [get frame]
close camera

import camera
start camera
repeat 5 times
do [show image [take picture]]
close camera
```



# 카메라 사용하기

카메라를 사용하여 QR 코드를 가져오는 예제를 해봅시다.

The image shows the Blockly for RoboLink interface. On the left is a sidebar with various block categories: Zumi blocks, Driving, Shapes, Screen, Sounds, Sensors, Camera, AI, Lights, Programming blocks, Input/Output, Logic, Loops, Timing, Math, Variables, and Functions. The main workspace contains two Python scripts. The first script is a linear sequence of blocks: 'import camera', 'start camera', 'set frame to take picture', 'print get QR code message get frame', and 'close camera'. The second script is a loop-based script: 'import camera', 'start camera', 'repeat 10 times', 'do' block containing 'set frame to take picture', 'print get QR code message get frame', and 'show image get frame', followed by 'close camera'. Below the scripts is a terminal window with the following output: 'Starting PiCamera', 'LEFT', 'None', 'None', 'None', 'LEFT', 'LEFT', 'LEFT', 'LEFT', 'LEFT', 'LEFT', 'LEFT', 'Closing PiCamera'. To the right of the workspace is a live camera feed window showing a QR code with a pink bounding box and the word 'LEFT' in pink text above it.

파이썬 라이브러리

[https://learn.robolink.com/docs/zumi-library/get\\_qr\\_message](https://learn.robolink.com/docs/zumi-library/get_qr_message)



# 카메라 사용하기

카메라를 사용하여 정지 표시를 찾는 예제를 해봅시다.

The image shows the Blockly for RoboLink interface. On the left is a sidebar with various block categories like Driving, Shapes, Screen, Sounds, Sensors, Camera, AI, Lights, and Programming blocks. The main workspace contains a Python script for finding a stop sign. The script starts with 'import camera' and 'start camera'. It then enters a 'repeat while true' loop. Inside the loop, it 'set frame to take picture', 'set result to find stop sign' (using a 'get frame' block), 'print get result', and 'show image get frame'. An 'if' block checks 'get result == true', and if true, it 'do break out of loop'. Finally, it 'close camera'. A terminal window on the right shows the output: 'Starting PiCamera', 'False', 'False', 'True', 'Closing PiCamera'. A video player window shows a sequence of frames from the camera, with the final frame showing a stop sign.

파이썬 라이브러리

[https://learn.robolink.com/docs/zumi-library/find\\_stop\\_sign](https://learn.robolink.com/docs/zumi-library/find_stop_sign)

# 카메라 사용하기

카메라를 사용하여 얼굴인식을 해봅니다.

BLOCKLY for ROBO LINK

junior senior master

Zumi blocks

- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights

Programing blocks

- Input/Output
- Logic
- Loops
- Timing
- Math
- Variables
- Functions



```
import camera
start camera
repeat 10 times
do
  print
  find face
  take picture
close camera
```

```
Starting PiCamera
False
True
False
False
False
False
False
False
False
True
Closing PiCamera
```

파이썬 라이브러리

[https://learn.robolink.com/docs/zumi-library/find\\_face](https://learn.robolink.com/docs/zumi-library/find_face)



# AI 예제

인공지능 블록을 사용해봅니다.

BLOCKLY for ROBOTLINK

junior senior master

Zumi blocks

- Driving
- Shapes
- Screen
- Sounds
- Sensors
- Camera
- AI
- Lights

Programming blocks

Input/Output

- Logic
- Loops
- Timing
- Math
- Variables
- Functions

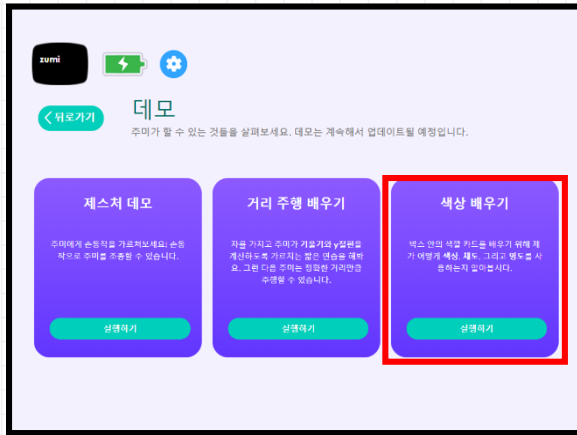


## load KNN model

KNN 모델 가져오기

load KNN model blockly\_test\_1

※ 다음의 블록을 사용하려면 먼저 데모 → 색상 배우기를 통해서 모델을 생성 해야합니다.  
생성된 모델을 블록을 통해 불러와서 사용하게 됩니다.



# AI 예제

인공지능 블록을 사용해봅니다.



junior senior master

Zumi blocks

Driving

Shapes

Screen

Sounds

Sensors

Camera

AI

Lights

Programing blocks

Input/Output

Logic

Loops

Timing

Math

Variables

Functions



## predict from frame

프레임에서 예측하기

```
import camera
start camera
load KNN model blockly_test_1
if (
  predict from frame take picture = red
)
do
  stop
close camera
```

```
import camera
start camera
load KNN model blockly_test_1
print
predict from frame take picture
close camera
```

```
Starting PiCamera
Loaded blockly_test_1 successfully
red
Closing PiCamera
```



# 02

## 시니어

*Senior*

# 블록설명

Introduction to Blocks





# Driving

주니어의 블록 보다  
보다 정밀한 설정값을 통하여  
주미를 이동 시킬 수 있습니다.

```
↑ forward_step( angle = 0 )
```

```
↓ reverse_step( angle = 0 )
```

```
move_to_coordinate( x = 0 , y = 0 , inches )
```

```
line_follow_gyro( speed = 20 , duration = 2 , angle = 2 , left_IR = 100 , right_IR = 100 )
```

```
funnel_align( speed = 20 , duration = 1 , angle_adj = 1 )
```

```
forward_avoid_collision( speed = 20 , duration = 1 , angle = 0 )
```

```
reverse_avoid_collision( speed = 20 , duration = 1 , angle = 0 )
```

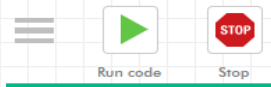
```
set_PID( P = 2.9 , I = 0.01 , D = 0.05 )
```

```
set_speed_prediction( speed = 40 , slope(inches) = 6 , intercept = 0 )
```

```
reset_PID()
```

```
speed_calibration()
```

- junior
- senior
- master
- Zumi blocks
- Driving
- Camera
- Eyes
- Programming blocks
- Lights
- Logic
- Loops
- Timing
- Math
- Variables



↑ forward_step( angle = 0 )	각도 (0도)만큼 전진 시킵니다.
↓ reverse_step( angle = 0 )	각도 (0도)만큼 후진 시킵니다. ※반드시  블록을 사용하여야 합니다. 사용하지 않을 시 계속 회전
move_to_coordinate( x = 0 , y = 0 , inches )	X=0 Y=0 좌표로 으로 주미를 이동시킵니다 단위 : 인치 (inches)
line_follow_gyro( speed = 20 , duration = 2 , angle = 2 , left_IR = 100 , right_IR = 100 )	설정된 IR 센서값에 따라 주미를 라인을 따라 주행하게 합니다.
funnel_align( speed = 20 , duration = 1 , angle_adj = 1 )	주미가 꼬깔을 인식하고 그 앞에 정렬합니다.
forward_avoid_collision( speed = 20 , duration = 1 , angle = 0 )	설정 값에 따라 전방 센서가 감지되면 회피하는 블록 입니다.
reverse_avoid_collision( speed = 20 , duration = 1 , angle = 0 )	설정 값에 따라 후방 센서가 감지되면 회피하는 블록 입니다.
set_PID( P = 2.9 , I = 0.01 , D = 0.05 )	설정 값에 따라 후방 센서가 감지되면 회피하는 블록 입니다.
set_speed_prediction( speed = 40 , slope(inches) = 6 , intercept = 0 )	속도의 최적 맞춤 라인 데이터를 설정합니다.
reset_PID()	주미의 PID 값을 리셋 합니다.
speed_calibration()	주미의 속도를 보정(캘리브레이션)합니다.

# 02

## 시니어

*Senior*

# 예제 따라하기

Follow the example







Run code



Stop



Python

junior

senior

master

Zumi blocks

Driving

Camera

Eyes

Programming blocks

Lights

Logic

Loops

Timing

Math

Variables

repeat 100 times

do  
↑ forward\_step( angle = 30 )

brake

repeat 100 times

do  
↓ reverse\_step( angle = 0 )

brake

**Angle(각도)** : 앞으로 주행할 각도입니다.

0°에서 100단계( 100 ) 동안 주미를 각도(30°)로 전진시킵니다.

반드시 을 사용하여 멈춰주세요  
멈추지 않으면 Zumi는 영원히 회전 합니다.

블록도 같습니다.

[https://learn.robolink.com/docs/zumi-library/forward\\_step](https://learn.robolink.com/docs/zumi-library/forward_step)

[https://learn.robolink.com/docs/zumi-library/reverse\\_step](https://learn.robolink.com/docs/zumi-library/reverse_step)



Run code

Stop



Python

junior

senior

master

Zumi blocks

Driving

Camera

Eyes

Programming blocks

Lights

Logic

Loops

Timing

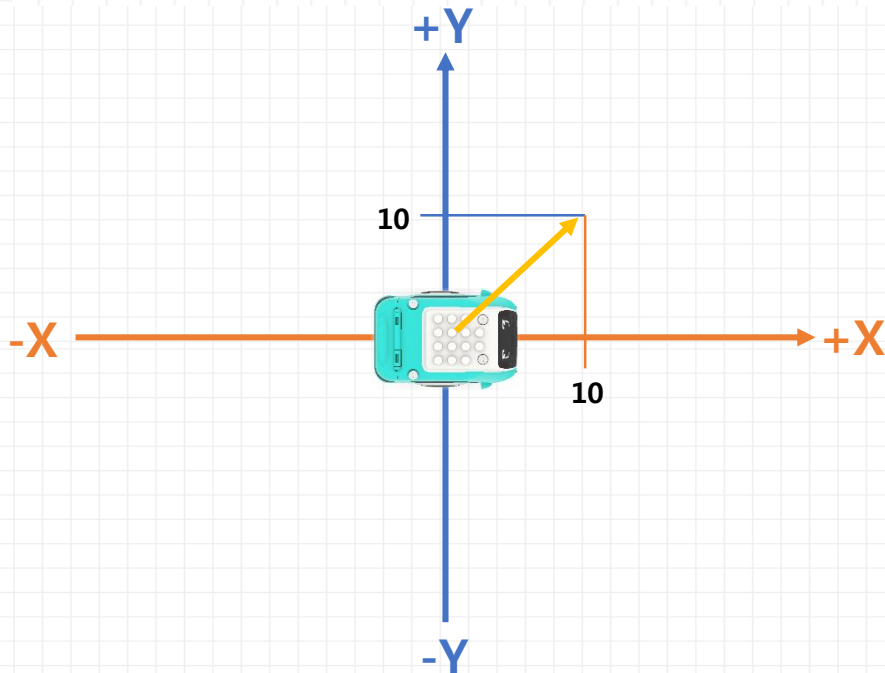
Math

Variables

```
move_to_coordinate( x = 10 , y = 10 , inches )
```

inches

centimeter



X 및 Y 좌표면에서 Zumi를 원하는 좌표로 이동시킵니다.  
기본 단위는 **인치**이지만 **센티미터**로 변경할 수 있습니다.

왼쪽의 코드의 경우 Zumi는

**X축(10) y축(10)**을 따라 **각각 10인치** 좌표 방향으로  
주행합니다.

설정된 **속도 보정값**에 따라 주행합니다.

[https://learn.robolink.com/docs/zumi-library/move to coordinate%28%29](https://learn.robolink.com/docs/zumi-library/move%20to%20coordinate%28%29)



Run code



Stop

junior

senior

master

Zumi blocks

Driving

Camera

Eyes

Programing blocks

Lights

Logic

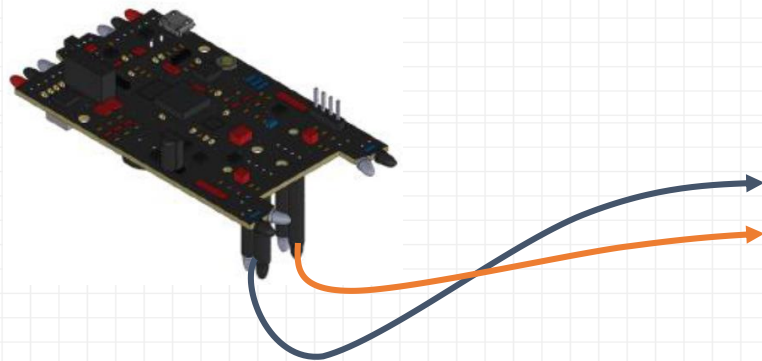
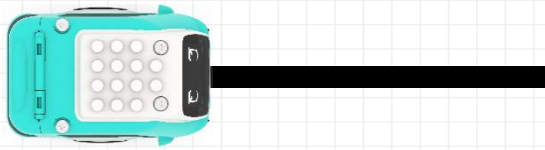
Loops

Timing

Math

Variables

```
line_follow_gyro( speed = 20 , duration = 2 , angle = 2 , left_IR = 100 , right_IR = 100 )
```



Zumi가 검은색 선이 감지되면, 검은색 선을 따라 주행하며 두개의 하단의 IR 센서 중 하나가 흰색이 감지될 경우 주행 각도를 조정합니다.

**Speed (속도)** : 모터가 0-80으로 이동하는 정수 값입니다.

**duration (지속시간)** : 명령이 지속되는 시간입니다.

**angle (각도)** : 이 값은 센서가 흰색을 감지할 경우 Zumi가 각도를 변경하는 양입니다.

**Left IR** : 하단 좌측의 IR 센서 임계 값

**right IR**: 하단 우측 IR 센서의 임계 값

[https://learn.robolink.com/docs/zumi-library/line\\_follow\\_gyro\\_assist](https://learn.robolink.com/docs/zumi-library/line_follow_gyro_assist)



Run code



Stop



Python

junior

senior

master

Zumi blocks

Driving

Camera

Eyes

Programing blocks

Lights

Logic

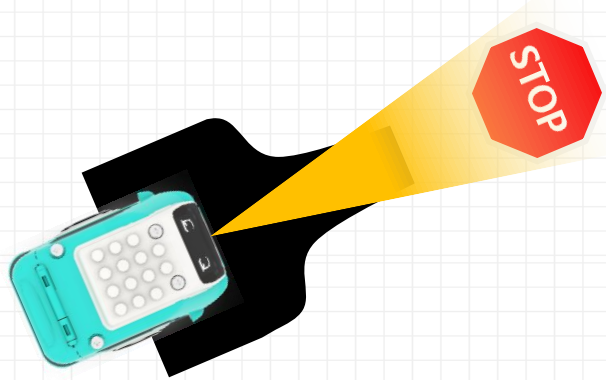
Loops

Timing

Math

Variables

```
funnel_align( speed = 20 , duration = 1 , angle_adj = 1 )
```



주미 대회에서 사용되는 깔때기를 인식하고,  
그 앞에 정렬합니다.

**Speed(스피드)** : 순방향 속도는 0 - 80 사이의 정수 값입니다.

**duration(지속 시간)** : 명령을 실행할 시간(초)입니다.

**Angle\_adjust(각도\_조정)** : 검은색 선이 Zumi 아래에 중앙에 있지  
않을 때 Zumi가 수행하는 자이로의 값입니다.

[https://learn.robolink.com/docs/zumi-library/funnel\\_align](https://learn.robolink.com/docs/zumi-library/funnel_align)



Run code



Stop



Python

junior

senior

master

Zumi blocks

Driving

Camera

Eyes

Programming blocks

Lights

Logic

Loops

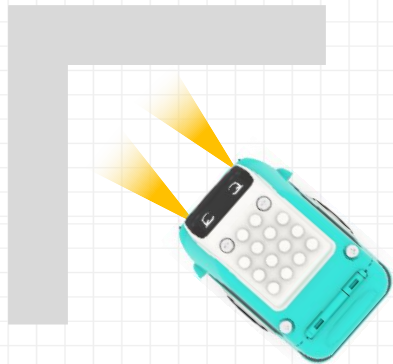
Timing

Math

Variables

```
forward_avoid_collision( speed = 20 , duration = 1 , angle = 0 )
```

```
reverse_avoid_collision( speed = 20 , duration = 1 , angle = 0 )
```



전방 또는 후방의 센서를 통하여 무언가를 감지할 때까지  
설정된 속도,시간,각도로 주행합니다.

**Speed(스피드)** : 모터 속도입니다.

**duration(지속 시간)** : 계속 앞으로 나아가고 싶은 시간입니다.

**Angle(각도)** : 는 주행하고자 하는 방향입니다.

[https://learn.robolink.com/docs/zumi-library/forward\\_avoid\\_collision](https://learn.robolink.com/docs/zumi-library/forward_avoid_collision)



Run code

Stop

junior

senior

master

Zumi blocks

Driving

Camera

Eyes

Programing blocks

Lights

Logic

Loops

Timing

Math

Variables

`speed_calibration()``set_speed_prediction( speed = 40 , slope(inches) = 6 , intercept = 0 )`

console

```
Zumi board detected
OLED Screen detected
Gyroscope & Accelerometer detected
Save these values
zumi.PRED_SPEED_INCHES_SEC = 6.919576445977846   주미.PRED_SPEED_INCH_SEC = 6.919576445977846
zumi.PRED_SLOPE_INT_INCH = -0.5973887956673474  주미.PRED_SLOPE_INT_INCH = -0.5973887956673474
zumi.PRED_SET_SPEED = 40                       zumi.PRED_SET_SPEED = 40
```

speed\_calibration을 통하여 zumi가 주행하기 가장 적합한 3가지의 주행 설정 값을 찾을 수 있습니다.

이 값을 찾아서 적정 값을 move\_to\_coordinate 및 move\_inches 명령에 대입하여 작동하는 것이 좋습니다.

속도 보정의 값을 얻은 후에는 set speed prediction 명령을 사용하여 최적 맞춤 라인 데이터를 설정할 수 있습니다.

[https://learn.robolink.com/docs/zumi-library/speed calibration%28%29](https://learn.robolink.com/docs/zumi-library/speed%20calibration%28%29)

- junior
- senior
- master
- Zumi blocks
- Driving
- Camera
- Eyes
- Programing blocks
- Lights
- Logic
- Loops
- Timing
- Math
- Variables

# reset\_PID()

주미가 주행할 때 사용하는 PID 값을 리셋 합니다.

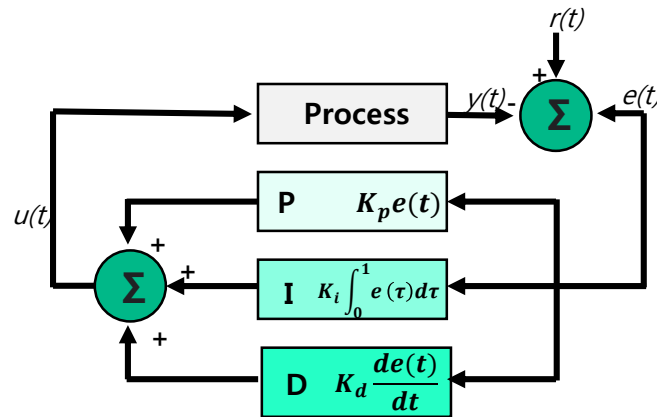
**P (Proportion) :** 비례

**I (Integral) :** 적분

**D (Differential) :** 미분

※ PID : 로봇의 자세 제어를 위한 연산 입니다.

연산 식과 프로세서가 다소 어렵기 때문에, 이 블록은 보정을 위해 값을 초기화 하는 작업이라고 생각하시면 됩니다.



< PID 계산 >

**Zumi**  
*BLOCKLY* *for*  
**ROBOLINK**

