

코딩 규칙

1. 블록 코드 기본 구조

모든 블록 코드 제시 시, 프로그램의 진입점 역할을 하는 최상위 함수 블록인 시작하기와 무한 반복하기를 기본 구조로 항상 포함하여 제시합니다.

이 규칙을 추가하면, 앞으로 모든 블록 코드는 아래와 같은 기본 구조를 갖게 됩니다.

블록 구조 (Block Composer)	제시 방법 (텍스트 형식)
시작하기	시작하기
(내부 블록)	(내부 블록)
무한 반복하기	무한 반복하기
(내부 블록)	(내부 블록)

2. 블록 코드 형식 (줄 바꿈 및 들여쓰기 규칙)

- 최상위 블록 (시작하기, 무한 반복하기 등)은 왼쪽 정렬합니다.
- 각 명령어 블록은 반드시 개행 문자를 사용하여 분리하여, 한 줄에 하나의 블록만 출력해야 합니다.
- 내부 실행 영역을 가지는 블록 (만약, 반복하기, 함수 정의 등)의 내부에 포함되는 하위 블록은 들여쓰기를 적용하여 계층 구조를 명확하게 표현합니다.

3. 내부 블록 및 조건 표현 규칙

드롭다운 메뉴의 선택 값 또는 입력 값은 블록의 기능적 인자 (Argument)에 해당하며, 블록의 문구 내에서 해당 값이 위치하는 곳에 대괄호 [...]를 사용하여 직접 삽입하여 표현합니다.

이는 블록의 고유 문구와 사용자가 선택/입력한 값을 통합하여 시각적으로 재현하는 역할을 합니다.

모든 블록 코드는 블록의 고유 명칭, 드롭다운 메뉴 선택 값, 그리고 사용자가 입력한 값을 대괄호 [...]를 사용하여 모두 포함하는 형태로,

실제 Block Composer의 블록 모양을 텍스트로 최대한 가깝게 재현하여 제시해야 합니다.

블록 구조 (Block Composer)	제시 방법 (텍스트 형식)
만약 [조건] 하기 [명령] 아니라면 [명령]	만약 [조건] 하기 [명령] 아니라면 [명령]
라쿤봇: [속도] 제어 모드로 설정하기	라쿤봇: [속도] 제어 모드로 설정하기
라쿤봇: 관절 [1] 속도를 [100] (으)로 정하기	라쿤봇: 관절 [1] 속도를 [100] 으로 정하기

4. 스크립트 코드 기본 구조

모든 스크립트 코드 (파이썬 / 자바스크립트) 제시 시, 프로그램의 진입점 역할을 하는 함수인 `setup()` 과 `loop()` 를 기본 구조로 항상 포함하여 제시합니다.

이 규칙을 추가하면, 앞으로 모든 스크립트 코드 (파이썬 / 자바스크립트) 는 아래와 같은 기본 구조를 갖게 됩니다.

```
// 자바스크립트 코드 기본 구조
```

```
async function setup() {  
    // 내부 코드  
}
```

```
function loop() {
```

```
    // 내부 코드  
}
```

```
# 파이썬 코드 기본 구조
```

```
import asyncio
```

```
async def setup():
```

```
    # 내부 코드  
    return
```

```
def loop():
```

```
    # 내부 코드  
    return
```

5. 스크립트 코드 형식 (줄 바꿈 및 들여쓰기 규칙)

- 최상위 함수 (`setup`, `loop` 등) 은 왼쪽 정렬합니다.
- 줄 바꿈 시에 발생하는 들여쓰기 (' ') 는 반드시 빈칸 3 칸을 기준으로 합니다.

6. 블록 코드 및 스크립트 코드 제시 규칙

제공된 문서 (“코딩-가이드”, “기본-블록”, “햄스터-S”등) 에 명시된 함수 및 객체만 생성할 수 있습니다.

로봇 제어 시 문서에 정의되지 않은 새로운 로봇 제어 함수나 커스텀 함수를 생성하여 사용하는 것은 엄격히 금지됩니다.

단, 사용자의 함수 생성 요청 시 함수를 생성할 수 있으나, 다음과 같은 규칙을 따라야 합니다.

1) 함수 이름은 영어로 작성되어야 합니다. 2) 비동기 작업을 수행하는 모든 함수는 정의 시 함수 이름 앞에 반드시 `async` 키워드를 붙여야 합니다.

문서에 명시된 변환 예시에 나타나지 않는 `const`, `let`, `var` 등의 임시 변수 선언은 절대 금지하며, 사용자의 변수 생성 요청 시에만 생성할 수 있습니다.

또한, 로봇 ID 및 매개값은 반드시 리터럴로 작성해야 합니다.

```
// 올바른 예시
__getSpeed('HamsterS*0', 100)

// 잘못된 예시
const ROBOT_ID = 'HamsterS*0';
__getSpeed(ROBOT_ID, 100)
```

7. 로봇 Device 객체 호출 규칙

스크립트 코드 (파이썬 / 자바스크립트)에서 로봇 Device 객체를 호출할 때에는 다음과 같은 규칙을 따라야 합니다.

파이썬 Device 객체: 반드시 `__` (언더바 2 개)를 사용하여 `__('{Device 객체 urn}')` 형식으로 작성합니다.

```
__('HamsterS*0:wheel.speed.left').d = 50
```

자바스크립트 Device 객체: 반드시 `$` (달러 기호)를 사용하여 `$('{Device 객체 urn}')` 형식으로 작성합니다.

```
$( 'HamsterS*0:wheel.speed.left' ).d = 50;
```

금지 사항: `_('...')` (언더바 1 개) 표기를 사용하거나, 자바스크립트에서 `$$('...')` (달러 기호 2 개) 표기를 사용해서는 안 됩니다.

8. 커스텀/유ти리티 함수 호출 규칙

스크립트 코드에서 커스텀/유ти리티 함수를 호출할 때에는 파이썬, 자바스크립트 구분 없이 다음과 같은 공통된 규칙을 따라야 합니다.

함수 이름은 반드시 `__` (언더바 2 개)로 시작해야 합니다. 함수 이름이 `_` (언더바 1 개)로 시작하는 함수를 생성하거나 호출하는 행위는 엄격히 금지됩니다.

```
// 올바른 예시
__getSpeed('HamsterS', 50)
```

```
// 잘못된 예시  
_getSpeed('HamsterS', 50)
```

9. 로봇 전용 블록 및 코드 우선 사용 규칙

로봇 하드웨어 (바퀴 속도, LED, 소리 등) 제어 시,

‘기본-블록’문서에 정의된 유ти리티 함수보다 로봇 (예: ‘햄스터-S’) 문서에 정의된 Device 객체 및 함수를 우선해서 사용해야 합니다.

예를 들어, 로봇을 이용해 소리를 내야 하는 경우,

유ти리티 함수인 `_playSound()` 함수보다 ‘햄스터-S’문서에 명시된 `sound.clip` 객체를 우선해서 사용해야 합니다.

10. 로봇 이동 제어 시 필수 실행 조건

로봇 (햄스터 S, 햄스터, 빠오, 터틀, 비글) 의 바퀴 속도를 설정하거나 변경하는 모든 코드를 작성할 때,
속도 설정 명령을 실행하기 직전에 `wheel.move` 객체의 상태를 초기화하는 과정이 필수적입니다.

`wheel.speed`로 로봇의 지속적인 이동 속도를 제어하기 전에, 혹시 남아 있을지 모르는 다른 이동 명령 (`wheel.move`) 의 값을 0 으로 초기화하여, 새로운 속도 설정이 문제 없이 정확하게 적용되도록 보장하기 위함입니다.

프로그래밍 언어

바퀴 속도 설정 시, 필수로 삽입해야 할 초기화 로직

파이썬

자바스크립트

11. Import 최소화 규칙

파이썬 코드 작성 시에는, 코드 실행에 필수적인 모듈만 `import` 해야 합니다. - `await` 키워드를 사용하는 모든 코드에는 `import asyncio` 만 기본적으로 포함됩니다.

자바스크립트 코드 작성 시에는, 그 어떠한 `import` 도 사용하지 않습니다.