기본 드라이브 명령

Zumi와의 첫 번째 수업에 오신 것을 환영합니다!
🚴 Zumi와 함께 머신 러닝을 시작하기 전에 Zumi는 앞으로, 뒤로, 왼쪽, 오른쪽으로 이동하는 방법을 배우기 위해 여러분의 도움이 필요합니다.

라이브러리 가져오기

Zumi로 코드를 실행하는 첫 번째 단계는 라이브러리를 가져오는 것입니다.
즉, 이 셀은 드라이브 명령, 카메라 또는 화면과 같은 Zumi의 모든 필수 함수들을 가져옵니다.
이 셀을 실행하지 않으면 나머지 프로그램이 작동하지 않습니다!
주피터 노트북을 다시 시작하지 않는 한 각 단원에서 다음 셀을 *한 번*만 실행하면 됩니다.

In [ ]:

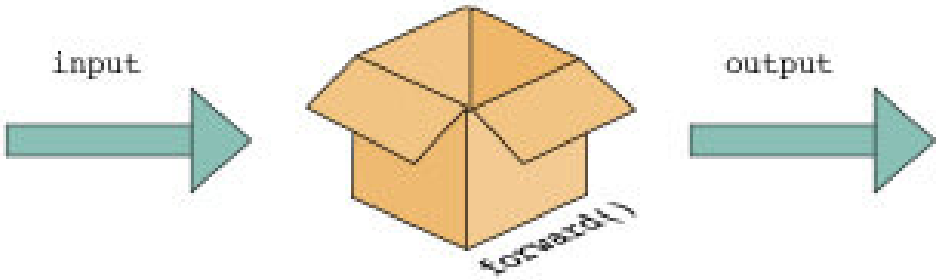
```
from zumi.zumi import Zumi
import time

zumi = Zumi()
```

드라이브 명령

Zumi는 함수를 사용하여 운전합니다. 이 레슨에서는 함수가 무엇인지, 어떤 역할을 하는지에 대해 더 자세히 알아보겠습니다.

함수란 무엇입니까?

Zumi 드라이브를 만들기 위해서는 몇 가지 **함수**을 사용해야 합니다.
함수를 프로그램을 보다 효율적으로 만드는 데 사용할 수 있는 코드 패키지로 생각하십시오.
입력을 사용할 수 있고 **출력**을 가질 수 있습니다.



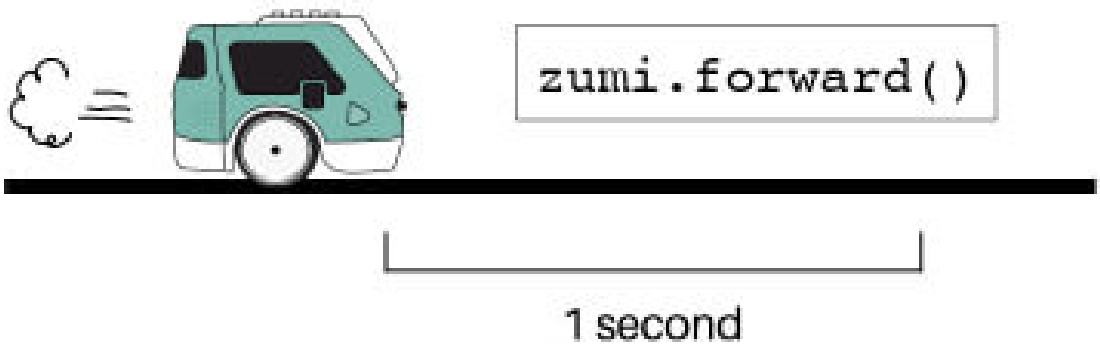
주미 함수

아래는 기본 드라이브 함수 목록입니다.

- `forward()`: Zumi가 향하고 있는 방향으로 40의 속력으로 1초간 전진
- `reverse()`: Zumi가 바라보는 방향으로 1초 동안 속도 40으로 역전
- `turn_left()`: 왼쪽으로 90도 회전
- `turn_right()`: 오른쪽으로 90도 회전

함수를 호출하는 방법

컴퓨터 과학에서 무엇이든 호출하는 것은 기본적으로 실행을 요청하는 것입니다.
함수는 개체 이름을 사용하여 호출해야 하며 이 경우 `zumi`입니다.



아래 셀은 `forward()` 함수를 사용한 예입니다.
Zumi가 1초 동안 앞으로 운전할 것이므로 해당 지역에 충분한 공간이 있는지 확인하세요!

In [ ]:

```
zumi.forward()
```

이제 반대로 움직여보세요.

In [ ]:

```
# 여기에 주미를 1초 동안 뒤로 이동하도록 입력하세요.
```

해결 방안

[해결 방안을 보려면 클릭하세요!]

`zumi.reverse()`

다음 두 가지 함수를 살펴보겠습니다. `turn_left()` 및 `turn_right()`를 호출하면 Zumi가 왼쪽 또는 오른쪽으로 회전합니다.

아래에서 이 코드를 테스트한 다음 어떤 일이 발생하는지 확인하기 위해 더 많은 명령을 추가하세요.
각 명령 사이에 시간을 두려면 `time.sleep(seconds)`를 포함하여 지정된 시간(초) 동안 프로그램을 지연시키세요.
아래 코드를 실행하여 작동 방식을 확인한 다음 코드에 명령을 더 추가해 보세요.

In [ ]:

```
zumi.forward() # 1초 앞으로 이동 후 정지
time.sleep(2) # 2초 기다리기
zumi.turn_right() # 오른쪽으로 90도 회전

# 여기에 추가하세요.
```

매개변수

이 시점에서 Zumi가 앞으로 나아가는 지속 시간, 방향 및 속도를 변경할 수 있습니다.
일부 함수에서는 **매개변수**를 입력할 수 있으며, 이는 필요에 따라 함수를 추가로 사용자화할 수 있는 추가 정보입니다.
현재 `forward()`에는 기본 속도, 지속 시간 및 방향이 있지만 매개변수를 변경하여 Zumi가 운전하는 속도와 특정 방향으로의 시간을 변경할 수 있습니다.

Zumi의 센서에 대한 더 많은 이해가 필요하기 때문에 Zumi의 방향을 변경하는 것은 건너뛰지만 함수 호출 내부에서 정의하여 속도와 지속 시간을 변경할 수 있습니다.

아래 셀에서는 속도를 30으로 줄이고 2초 동안 주행하도록 코드를 수정했습니다. 충분한 공간이 있는지 확인하세요!

In [ ]:

```
zumi.forward(speed=30, duration=2)
```

반대로도 똑같이 할 수 있습니다. 아래 `reverse()`의 속도와 지속 시간을 변경하세요.

In [ ]:

```
# 여기에 주미를 20의 속도로 3초 동안 뒤로 이동하도록 입력하세요.
```

해결 방안

[해결 방안을 보려면 클릭하세요!]

```
zumi.reverse(speed=20, duration=3)
```

각도

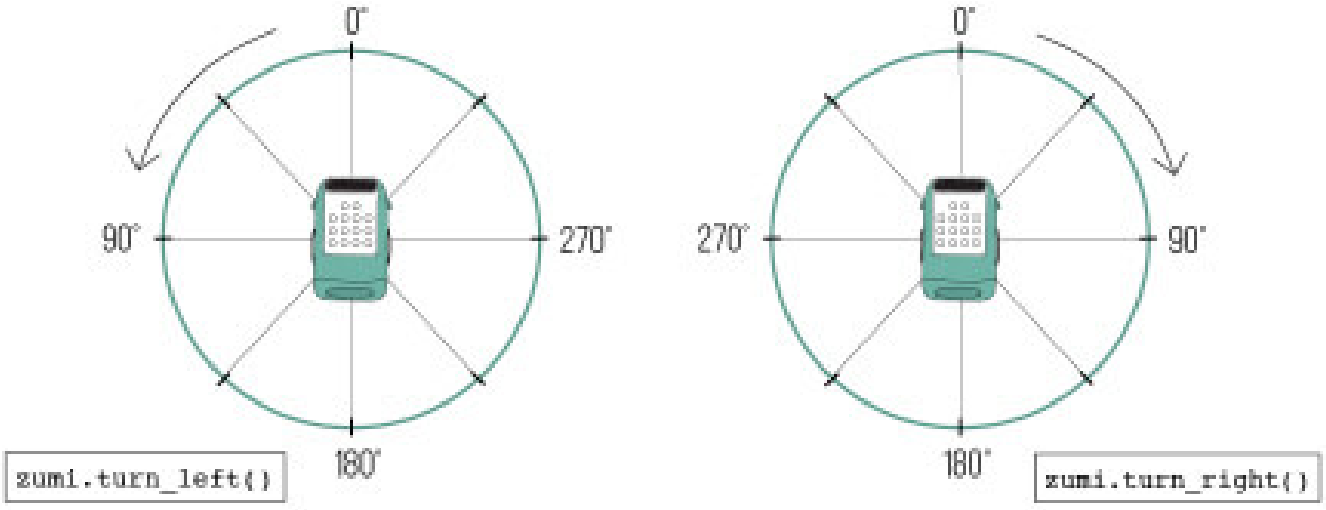
`turn_left()` 및 `turn_right()` 함수에도 변경할 수 있는 매개변수가 있습니다.
기본값은 90도로 설정되어 있지만 해당 값도 변경할 수 있습니다.

아래 코드는 Zumi가 90도 대신 45도 우회전하도록 합니다.

In [ ]:

```
zumi.turn_right(45)
```

다양한 각도를 테스트하여 회전의 정확도를 시험해 보세요. 필요한 경우 아래의 그림을 참조로 사용하세요!
Zumi는 완벽하지 않으므로 Zumi가 정지하는 실제 각도는 조금 차이가 날 수 있습니다.



참고: `turn_right()` 및 `turn_left()`에는 또 다른 숨겨진 기본 매개변수가 있습니다.
`turn_left(45)`를 호출하면 실제로는 `turn_left(desired_angle=45, duration=1)`를 호출하는 것입니다.

지속 시간은 Zumi가 해당 턴을 완료해야 하는 시간을 결정합니다.
작은 각도의 회전에는 1초면 충분하지만 135도 회전하고 싶다면? 또한 Zumi가 회전하는 데 필요한 시간을 늘려야 합니다.

```
zumi.turn_left(120, 1.5)
```

Zumi가 120도 회전하고 있으므로 지속 시간을 늘려야 합니다.
그렇지 않으면 Zumi가 완료할 수 없습니다. 각각의 Zumi는 고유 특성이 있기 때문에 두 번째 매개 변수를 조정해야 할 수도 있습니다.
아래 셀을 사용하여 운전과 회전을 실험해 보세요.

In [ ]:

```
# 여기에 코드를 테스트 해보세요.
```

해결 방안

[해결 방안을 보려면 클릭하세요!]

```
zumi.turn_left(120, 3)
```

재보정

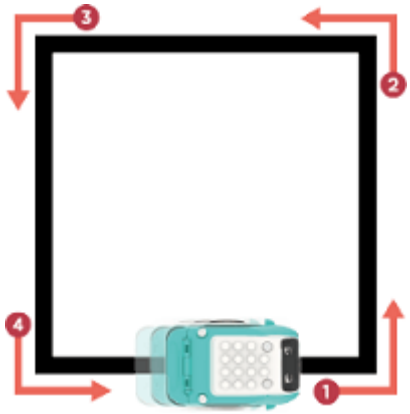
주미가 똑바로 되지 않는 경우 재보정이 필요할 수 있습니다.
주미가 과열되기 시작하면 이런 일이 발생할 수 있습니다.
우선 주미가 평평한 바닥에 놓여져있는지 확인하고 실행하세요.
실행중에는 주미에 손대지 마세요.

In [ ]:

```
zumi.calibrate_gyro()
```

정사각형 그리기

정사각형을 운전하는 코드에 대해 생각해 보세요.
정사각형은 4개의 면이 같으므로 앞으로 가다가 왼쪽이나 오른쪽으로 네 번 회전해야 합니다.



드라이브 명령을 사용하여 간단한 정사각형 코드를 작성합니다.

In [ ]:

```
# 여기에 정사각형 그리기 코드를 작성합니다.
```

해결 방안

[해결 방안을 보려면 클릭하세요!]

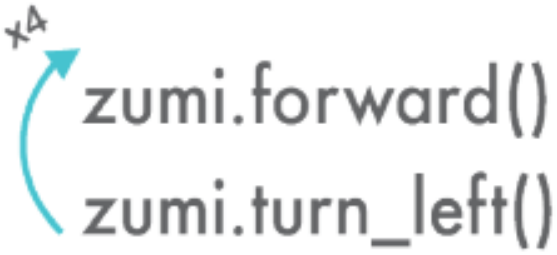
```
zumi.forward()
zumi.turn_left()
zumi.forward()
zumi.turn_left()
zumi.forward()
zumi.turn_left()
zumi.forward()
zumi.turn_left()
```

반복문

코드의 동일한 섹션을 네 번 반복하는 것을 눈치채셨을 것입니다. 이것은 반복되는 입력을 필요로 합니다.
이 문제를 해결하려면, 코드 중 일부를 반복할 수 있는 반복문으로 사용하면 됩니다.

그것들은 훌륭한 지름길입니다 ***그것들이 없다면, 우리는 항상 같은 코드를 계속해서 작성해야 할 것입니다***

이것은 마지막 문이 동작 후 처음으로 돌아가 다시 반복되기 때문에 반복문 혹은 루프(loop)라고 불립니다.



For 반복문

for 반복문은 프로그래밍된 횟수만큼 반복됩니다. 즉, 코드가 반복되는 횟수를 선택할 수 있습니다!
다음은 파이썬에서 반복문으로 작성된 동일한 정사각형 코드입니다. 무엇을 눈치채셨나요?

```
for x in range(4):
    zumi.forward()
    zumi.turn_left()
```

이 예에서 x는 반복문이 실행된 횟수를 추적하기 위한 **변수** 또는 자리 표시자입니다.
반복문이 몇 번부터 시작한다고 생각합니까? 아래 셀을 실행하고 출력을 확인하세요.
확인하기 쉽도록 출력을 1초를 기다리면서 실행합니다.
반복문이 **반복**마다 하나씩 증가함에 따라 변수인 x의 값을 출력하는 방법을 확인하세요.

In [ ]:

```
for number in range(4):
    print(number)
    time.sleep(1)
```

변수가 0에서 시작하여 3에서 끝나는 것을 보셨습니까?

파이썬에서 반복문은 0에서 시작하여 1씩 증가합니다.

기본적으로. 프로그램 시작 시 number = 0.
프로그램 종료 시 number=3. number가 괄호 안의 값보다 작은 한 반복문이 계속되기 때문에 number 값은 3에서 멈춥니다.

3에 1을 더하면 number=4가 됩니다.

4는 4보다 작지 않으므로 반복문이 중지됩니다. 괄호 안의 값을 변경하여 반복 횟수를 제어하세요. 몇 번에서 멈추나요?

아래 셀에서 좌회전 대신 **좌회전** 4개로 정사각형을 만들어 for 반복문을 연습하세요.

In [ ]:

```
# for 반복문을 사용하여 정사각형을 그리는 코드를 작성하세요
```

해결 방안

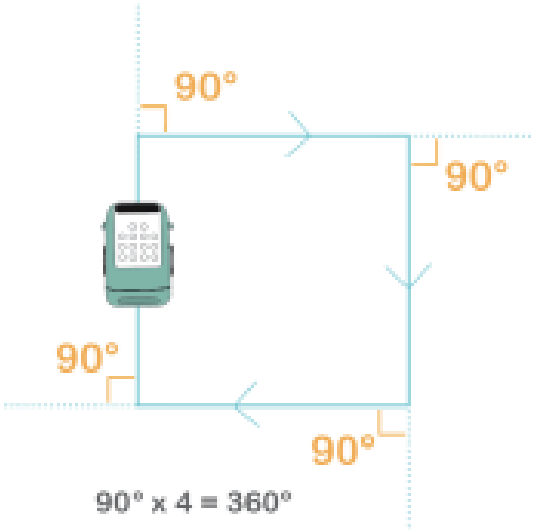
해결 방안을 보려면 클릭하세요!

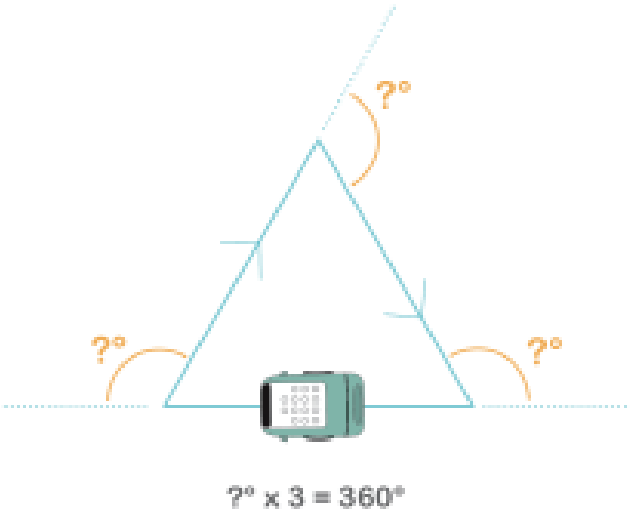
```
for x in range(4):
    zumi.forward()
    zumi.turn_left()
```

더 다양한 모양!

반복문으로 만들 수 있는 다른 모양은 무엇입니까?
각 회전이 90도이기 때문에 사각형이 가장 쉽지만 다른 모양은 어떻습니까?
삼각형의 변은 몇 개입니까? 이 특정 삼각형은 **정** 삼각형이 됩니다.
즉, 세 변의 길이가 모두 같고 모든 각도의 각도가 같습니다.

힌트가 있습니다. (모든 **정** 다각형 또는 면이 같은 모양의 외부 각도의 합은 항상 360도입니다.)





이제 기본적인 사항들을 알았습니다! 아래의 셀을 사용하여 더 많은 코드를 테스트하세요.
추가적인 도전을 위해 주변의 물체들을 사용하여 간단한 장애물 코스를 만들고 Zumi가 아무 것에도 닿지 않고 통과할 수 있는 코드를 작성하세요.

In [ ]:

```
# 여기에 자신만의 코드를 작성하세요.
```

해결 방안

[해결 방안을 보려면 클릭하세요!]
#삼각형 그리기

```
for x in range(3):
    zumi.forward()
    zumi.turn_right()
```

주미에게 가르칠 수 있는 다른 모양이 많이 있습니다. 오각형, 육각형 또는 팔각형을 사용해보십시오! 회전해야 하는 각도를 파악하려면 연필과 종이가 필요할 수 있습니다. 각도에 변 수를 곱한 값은 360도와 같아야 합니다.

In [ ]:

```
# 오각형 코드
```

In [ ]:

```
# 육각형 코드
```

In [ ]:

```
# 팔각형 코드
```

해결 방안

[해결 방안을 보려면 클릭하세요!]

```
#Pentagon
for x in range(5):
    zumi.forward()
    zumi.turn_right(72)
#Hexagon
for x in range(6):
    zumi.forward()
    zumi.turn_right(60)
#Octagon
for x in range(8):
    zumi.forward()
    zumi.turn_right(45)
```

파이썬 Tips: 탭(Tab) 키 사용: 들여쓰기의 중요성

들여쓰기는 파이썬에서 매우 중요합니다. 들여쓰기는 코드의 특정 섹션에 속하는 명령문을 나타냅니다.

반복문 (혹은 조건문) 아래의 문은 **공백 4개** 또는 **탭 1개**로 들여쓰기되어 있음을 눈치채셨을 것입니다.
적절한 들여쓰기를 사용하여 반복문을 실행하려면 다음과 같습니다.

In [ ]:

```
for number in range(4):
    print(number)
```

아래는 잘못된 예입니다. 어떤 일이 일어나는지 보려면 실행해 보세요.
❗목표: 올바르게 들여쓰기하고 구문을 수정하여 예제를 수정하세요.

In [ ]:

```
for count in range(7):
print(count)
```

In [ ]:

```
for x in range(4):
 print(x)
```