Robot Software Education Institute
로봇 SW 교육원

로봇활용 소프트웨어 교육

파이썬 #1

광운대학교 로봇학부
박광현

일정

파이썬과 친해지기	13:00 ~ 14:00
진짜 햄스터와 만나자	14:10 ~ 15:10
똑똑한 햄스터	15:20 ~ 16:20
아이들을 위한 프로그램	16:30 ~ 17:30

일반적으로 제가 만드는 강의 자료는

강의자료만 봐서는 무슨 말인지 알기가 어렵습니다.

왜 이렇게 강의를 하는지도 이해하기 어렵구요.

그래서 강의 시간에 말로 설명했던 것 중

기억나는 대로 코멘트를 추가했습니다.

검은색 필기체로 표시된 것이 코멘트입니다.

파이썬과 친해지기

미션 #1

학생들에게 오늘의 재미난 미션을 설명합니다
소설로봇 지보를 예로 들었고
로봇과 대화하는 상상을 해보도록 합니다
물론 인공지능은 사용할 수 없으므로
미션에서는 정해진 시나리오대로 대화하는 것을 합니다

이름을 입력하세요: 광현

광현아 안녕? 내 이름을 불러 줘.

햄스터

그래, 내 이름은 햄스터야.

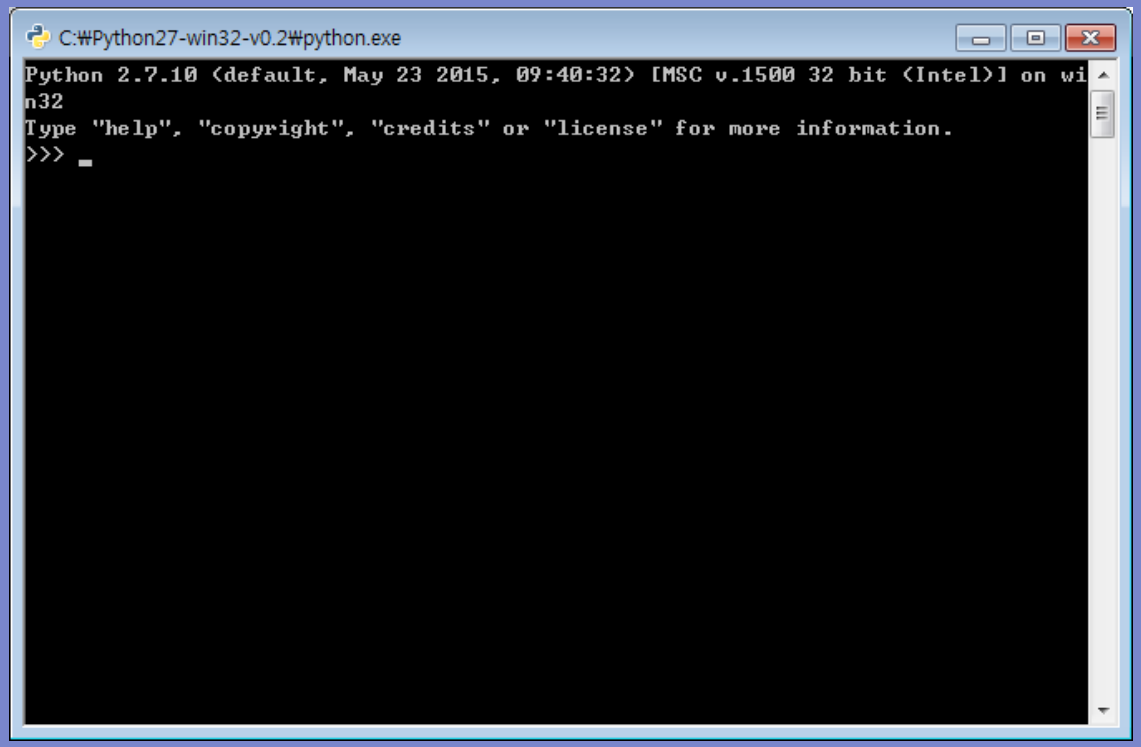
...

시작하기

C:\Python27-win32-v0.2

python.exe 실행

파이썬 쉘(Shell)



주의할 점 강의 이후에 추가된 페이지입니다

수업을 하다가 보니 hamster.school에서 배포한 파이썬에 dll이 빠져있다는 걸 알았습니다. 수정 후 다시 배포하겠지만 우선은 아래와 같이 하시면 됩니다.

1) www.python.org에서 정식 파이썬 2.7.10의 설치 파일을 내려 받아 설치합니다.
   C:\Python27에 설치됩니다.

2) hamster.school에서 파이썬 설치파일을 내려 받아 설치합니다.
   C:\Python27-win32-v0.2 또는 C:\Python27-win64-v0.2에 설치됩니다.

3) C:\Python27-win??-v0.2\Lib\site-packages 폴더에 있는
   roboid 폴더와 serial 폴더를
   C:\Python27\Lib\site-packages 폴더에 복사합니다.

4) C:\Python27에 있는 python.exe 실행

이후 버전에서 API 함수가 조금 수정될 것입니다

```
>>> print ‘hello’
```

```
>>> print 'hello'

>>> print "hamster"
```

간결함을 추구하는 파이썬에서 왜
작은 따옴표와 큰 따옴표를
모두 사용할 수 있게 하는지 생각해 봅시다
(추리)

```
>>> print ‘hello’

>>> print “hamster”

>>> print ‘hello’ + ‘hamster’
```

```
>>> print 'hello'

>>> print "hamster"

>>> print 'hello' + 'hamster'

>>> print 'hello', 'hamster'
```

둘 사이의 차이점을
관찰해 봅시다

규칙을 찾고 규칙을 확장해 봅니다
+ 'hamster' 뒤에 더 추가하면 어떻게 될까?
, 'hamster' 뒤에 더 추가하면 어떻게 될까?
추리하고 확인하고 추리하고 확인하는 과정을 해봅니다

```
>>> print 'hello'

>>> print "hamster"

>>> print 'hello' + 'hamster'

>>> print 'hello', 'hamster'

>>> print 'hello\nhamster'
```

복잡해 보이는데
내가 파이썬을 만든 사람이라면
이를 어떻게 간결하게 할 수 있도록
했을까 생각해 봅시다

```
>>> print 'hello'

>>> print "hamster"

>>> print 'hello' + 'hamster'

>>> print 'hello', 'hamster'

>>> print 'hello\nhamster'

>>> print '''hello
... hamster'''
```

복잡해 보이는데
내가 파이썬을 만든 사람이라면
이를 어떻게 간결하게 할 수 있도록
했을까 생각해 봅시다

생각해 본 후 보여줍니다

---

title

지금까지 배운 것을 바탕으로 화면에 위와 같이 표시되도록 해봅시다
아마 복잡할 겁니다
일단 각자의 방법으로 해 본 다음
간결함을 추구하는 파이썬에서 이를 어떻게 간결하게 처리하는지 보여 줍니다

```
------------------------------
title
------------------------------
```

```
>>> print '-'*30; print 'title'; print '-'*30
```

쉘에서 입력할 때는 한 줄 입력하고 엔터를 누르면 실행되어 버리기 때문에
여러 명령 줄을 입력하려면
하나의 명령과 그 다음 명령 사이에 세미콜론(;)을 넣어 주면 됩니다
지금 보니까 별로 좋은 예제는 아니네요 ^^;;

```
------------------------------
title
------------------------------

>>> print '-'*30; print 'title'; print '-'*30


------------------------------
I'm hamster
------------------------------
```

지금까지 배운 것으로 위의 내용을 화면에 표시하는 것을 해봅시다
이 예제를 통해 왜 작은 따옴표와 큰 따옴표를 모두 사용할 수 있게 하는지
답을 얻을 수 있습니다

```
------------------------------
title
------------------------------

>>> print '-'*30; print 'title'; print '-'*30


------------------------------
I'm hamster
------------------------------


------------------------------
I'm albert
------------------------------
```

성공한 사람도 있고 해결 못한 사람도 있을 텐데
답을 설명해 주고 아래 예제를 한번 더 해봅니다
설명에 대한 확인을 위해서
또 다른 이유는 '패턴'이라는 걸 설명하기 위해서

```
>>> line = '-'*30
>>> name = 'hamster'
>>> print line; print "I'm " + name; print line
```

코딩에서 패턴을 찾는 것은 매우 중요합니다
패턴의 개념이 변수로, 반복으로, 함수로 연결되기 때문입니다
앞의 예제에서 패턴을 찾아 봅시다
바로 앞의 예제 두 가지(I'm hamster와 I'm albert)에서 변화가 일어나는 부분이 뭘까요?
변화가 일어나는 부분이 변수가 됩니다

```
>>> line = '-'*30
>>> name = 'hamster'
>>> print line; print "I'm " + name; print line


------------------------------
I'm hamster. Are you albert?
------------------------------
```

위의 예제를 또 해 봅시다

이 예제는 또 다른 것을 설명하기 위한 기반이 됩니다

지금까지 배운 것으로 해 보면 약간 복잡하게 코딩될 것입니다

```
>>> line = '-'*30
>>> name = 'hamster'
>>> print line; print "I'm " + name; print line


------------------------------
I'm hamster. Are you albert?
------------------------------
```

우선 간단한 것부터

```
------------------------------
I'm hamster
------------------------------

>>> line = '-'*30
>>> name = 'hamster'
>>> print line; print "I'm %s" % name;
print line
```

%s로 빈 칸을 만들고 그 빈 칸을 name으로 채운다는 걸 보여 줍니다

보통 이런 걸 가르칠 때 %d, %s 등을 테이블로 보여주고 한 번에 설명하는데 학생들은 기억 못합니다

예제를 통해 하나씩 자연스럽게 익히도록 하는 것이 좋습니다. 여기서는 %s만 사용하고 다른 것은 나중에

요거는 다음 줄이 아니라 PPT에 한 줄로 표시가 안되어서 이런 겁니다

I'm hamster. Are you albert?

%s를 배웠으니 %s를 이용해서 위 예제를 해봅시다
좀 복잡해질 것입니다
각자 고민을 해서 해보는 것이 중요합니다
그리고 간결함을 추구하는 파이썬에서 이를 어떻게 간결하게 만들까 생각해 봅시다

```
------------------------------
I'm hamster. Are you albert?
------------------------------

>>> line = '-'*30
>>> name1 = 'hamster'
>>> name2 = 'albert'
>>> print line; print "I'm %s. Are you %s" %
(name1, name2); print line
```

각자 해 본 다음에
일단 hamster와 albert 자리에 %s... 총 두 개의 %s가 사용될 것이라는 것까지는
모두 이해를 합니다
그 두 개의 %s를 어떻게 채우느냐가 문제인데
괄호로 묶어주면 된다는 것을 얘기해 주면 쉽게 기억할 수 있게 됩니다
여기서 tuple은 설명하지 않고 넘어 갑니다

```
>>> sentence = 'hello hamster.'

>>> print sentence[6:13]
```

위 문장에서 hamster만 뽑아 내는 문제입니다
일단 하나의 예제를 보여 주고 설명합니다
직관적으로 이해하기 쉬운 예제입니다
인덱스를 앞에서부터 세어 나가는 것의 불편함을 설명합니다
hamster가 아니라 100글자 정도 된다면?

```
>>> sentence = 'hello hamster.'

>>> print sentence[6:13]

>>> print sentence[6:-1]
```

뒤에서부터 인덱스를 세는 방법을 설명해 줍니다

이제 마지막 마침표까지 포함해서 hamster.을 뽑아 내려면
어떻게 하면 되는지 질문하고 각자 답을 말해 봅니다

```
>>> sentence = 'hello hamster.'

>>> print sentence[6:13]

>>> print sentence[6:-1]

>>> print sentence[6:]
```

각자 생각해 본 다음 간결한 표현법을 알려 줍니다

이제 hello만 뽑아 내려면 어떻게 하면 좋을지
지금까지 배운 것으로 추리해 봅니다

```
>>> sentence = 'hello hamster.'

>>> print sentence[6:13]

>>> print sentence[6:-1]

>>> print sentence[6:]

>>> print sentence[:5]
```

추리한 것이 대부분 맞을 것입니다
이처럼 코딩 교육을 할 때 처음부터 문법을 설명하지 말고
예제를 통해 하나씩 추리해 나가는 것이 중요합니다

```
>>> sentence = 'hello hamster.'

>>> print sentence[6:13]

>>> print sentence[6:-1]

>>> print sentence[6:]

>>> print sentence[:5]

hello albert.
```

지금까지 배운 것을 이용해서 hello hamster를 hello albert로
수정하여 화면에 출력해 봅시다
정답은 없고 사람마다 다를 수 있습니다

```
>>> sentence = 'hello hamster.'

>>> print sentence[6:13]

>>> print sentence[6:-1]

>>> print sentence[6:]

>>> print sentence[:5]

hello albert.

>>> print sentence[:6] + 'albert' +
sentence[-1]
```

파이썬 파일 만들기

쉘에서 명령어 입력하는 것은 그만하고
파일에 코딩하여 실행하는 방법을 설명합니다

???.py

C:₩Python27 폴더에
확장자가 py인 텍스트 파일을 하나 만듭니다
예를 들어 test.py

```
name = raw_input('Enter name: ')
print "I'm %s." % name
```

만들어진 test.py 파일을 메모장 같은 것으로 열어서
위 코드를 입력합니다
워드패드 등은 유니코드로 저장되므로 ASCII로 저장할 수 있는
메모장을 사용하세요... 간결하니까요 ^^;;

python test.py

C:₩Python27에서 도스창 하나 열고
도스창에서 입력하여 test.py를 실행합니다

```
>>> name = raw_input('Enter name: ')
```

```
I'm ???
```

미션을 수행하기 위해 글자를 입력 받는 방법을 설명해 줍니다

미션 #1

대화하는 프로그램

파이썬에서는 한글 쓰기가 좀 까다로우니
일단은 영어로~ ^^

진짜 햄스터와 만나자

미션 #2

햄스터는
댄싱 머신

앞으로 1초 이동하기

```
from roboid.hamster import Hamster

# create a hamster
robot = Hamster()

robot.motors(30, 30, 1)
```

왼쪽 바퀴 속도 / 오른쪽 바퀴 속도 / 시간, 초 단위

정품 햄스터(판매하는 제품)와 비품 햄스터(샘플로 나눠 준 것)는
동작 방식이 좀 다른데
정품 햄스터는 코드 실행이 끝나면 스스로 멈추지만
비품 햄스터는 코드 실행이 끝나도 마지막 명령을 계속 수행합니다

앞으로 1초 이동하기

```
from roboid.hamster import Hamster

# create a hamster
robot = Hamster()

robot.motors(30, 30, 1)

# for a sample hamster
robot.reset()
robot.wait(1000)
```

비품 햄스터의 경우
동작을 멈추는 코드를 추가합니다
정품 햄스터는 이 코드가 필요 없구요

1초 기다리기
msec 단위

모든 디바이스의 값을 초기화

뒤로 1초 이동하기

```
from roboid.hamster import Hamster

# create a hamster
robot = Hamster()

robot.motors(-30, -30, 1)

# for a sample hamster
robot.reset()
robot.wait(1000)
```

공통 코드

공통 코드, 정품 햄스터는 필요 없음

이제부터는 하얀 색 코드 부분만 수정합니다

제자리 돌기 1초

```
robot.motors(-30, 30, 1)
```

피봇 턴 1초

```
robot.motors(0, 30, 1)
```

라운드 턴 1초

```
robot.motors(20, 50, 1)
```

코딩을 가르칠 때 다음과 같은 순서로 하는 것이 좋은데
이는 블록 코딩을 기준으로 한 것이고 텍스트 코딩이나 언령에 따라
순서는 좀 바뀔 수 있습니다

1 **순차**

명령어 하나, 같은 명령 순서대로,
다른 명령 섞어서 순서대로

2 **횟수 반복**

단순 반복 (한 가지 명령 반복),
규칙(패턴) 있는 반복 (패턴을 반복)
순차 반복 (단순 반복 + 순차, 단순 반복 + 단순 반복)
규칙(패턴) 있는 순차 반복, 내포 반복*

3 **디버깅**

명령어 추가, 삭제, 수정
순차를 반복으로 수정 (패턴 찾기)
반복 내에서 추가, 삭제, 수정

*고급 과정

4 **~까지 반복**

단순 반복
규칙(패턴) 있는 반복
순차 반복
규칙(패턴) 있는 순차 반복, 내포 반복*

5 **조건**

만약 ~이라면
만약 ~이라면, 아니면~
~까지 반복과 조합

6 **~동안 반복**

7 **논리**

AND, OR, NOT

8 **이벤트**

이벤트 처리

9 **함수 호출** 이미 존재하는 함수를 호출만

10 **변수**

11 **for 반복**

12 **함수 만들기** 함수 구현

13 **병렬 처리***

순차

1) 같은 명령 순서대로

2) 다른 명령 섞어서 순서대로

횟수 반복

```
for i in range(10):
    robot.motors(30, 30, 1)
```

파이썬에서는 들여쓰기가 중요합니다
for 문의 경우 반복할 부분은 모두 들여 씁니다
for 문의 제일 뒤에는 콜론(:)이 있는데
규칙은... 들여 써야 할 것 같다 생각되면
콜론 붙이면 됩니다 (for, while, if 등등)

1) 단순 반복

2) 패턴 반복

3) 순차 반복

4) 패턴 있는 순차 반복

파이썬에는 for(i=0;i<10;++i) 같은 건 없습니다
간결함을 추구하니까요 ^^ 파이썬은 같은 걸 여러 방법으로 표현할 수
있다면 그 중에서 가장 좋은 방법 하나만 제공합니다
for... in 문장과 range에 대해 글로 다 설명하려니 너무 기네요
각자 찾아 보시기를...

파이썬에서 들여쓰기는 보통 탭 1번으로 하거나
스페이스(빈 칸) 4개를 합니다
같은 파일 내에서 이 두 가지 방식을 섞어서 사용하면 안됩니다
그리고 들여쓰기를 정확하게 하지 않으면 파이썬에서는 에러가 납니다
개발자들 사이에서는 스페이스 4개를 하는 방식을 표준으로 하는데
탭으로 하는 경우 에디터마다 탭 간격이 달라서 다르게 보일 수 있기
때문입니다
하지만 탭을 스페이스 4개로 자동 변환해주는 에디터나 툴들이 있기
때문에 일단은 편한 방식으로 하셔도 됩니다
아직은 전문 개발자가 아니니까요 ^^;;

미션 #2

햄스터는 댄싱 머신

똑똑한 햄스터

조건, ~동안 반복

```
while True:
      proximity = robot.proximity()
      print proximity
```

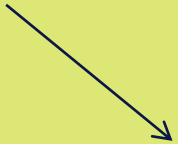
직진하다가 앞에 장애물이 나타나면 일정 거리를 후진한 후 다시 직진하는 프로그램을 작성하시오

```
while True:
    proximity = robot.proximity()
    if proximity[0] > 40:
        robot.motors(-30, -30)
    else:
        robot.motors(30, 30)
```

초를 생략하면 입력한 값을 계속 수행
즉, 원래는 왼쪽 바퀴 속도, 오른쪽 바퀴 속도를 '설정'하게 되는데
초를 입력하면 입력한 시간 후에 설정된 값을 리셋하는 것임

제자리에서 회전하다가 장애물을 만나면 제자리 회전을 중지하고 청색 LED를 깜박이는 프로그램을 작성하시오

```
robot.leds(LEFT_LED, RIGHT_LED, SECONDS)
```



요걸 그대로 타이핑 하는 건 아니고
왼쪽 LED 색상, 오른쪽 LED 색상,
시간(초)를 입력하라는 의미

제자리에서 회전하다가 장애물을 만나면 제자리 회전을 중지하고 청색 LED를 깜박이는 프로그램을 작성하시오

```
robot.leds(LEFT_LED, RIGHT_LED, SECONDS)
```

```
Hamster.LED_OFF
Hamster.LED_BLUE
Hamster.LED_GREEN
Hamster.LED_CYAN
Hamster.LED_RED
Hamster.LED_MAGENTA
Hamster.LED_YELLOW
Hamster.LED_WHITE
```

LED 색상은 1~7까지 숫자를 입력하면 되는데 각 숫자의 색상이 뭔지 기억하기 어려우니까 이를 미리 상수 값으로 정의해 놓았습니다
색상 넣는 자리에 왼쪽에 정의된 것을 입력하면 됩니다

예) robot.leds(Hamster.LED_BLUE, Hamster.LED_RED, 3)

논리 and, or, not

```
while True:
    proximity = robot.proximity()
    if proximity[0] > 40 or proximity[1] > 40:
        robot.motors(-30, -30)
    else:
        robot.motors(30, 30)
```

직진하다가 정지선을 만나면 정지

```
while True:
    leftFloor = robot.read(Hamster.LEFT_FLOOR)
    rightFloor = robot.read(Hamster.RIGHT_FLOOR)
    if leftFloor < 50 or rightFloor < 50:
        robot.motors(0, 0)
    else:
        robot.motors(30, 30)
```

정지선은 A4 용지에 검은색 선을 그려서 프린트 하거나
A4 용지에 검은색 매직, 펜 등으로 선을 그리면 됩니다
햄스터 이동 속도가 빠르면 센싱 못하고 지나갈 수 있으니 선은 굵게

직진하다가 네 번째 검은색 라인을 만나면 정지하는 프로그램을 작성하시오

직진하면서 왼쪽 바닥 센서가 검은색 라인을 감지하면 왼쪽 LED를 깜박이고 오른쪽 바닥 센서가 검은색 라인을 감지하면 오른쪽 LED를 깜박이는 프로그램을 작성하시오

```
if      :
    ????
elif    :
    ????
else:
    ????
```

함수

```
def function(arg1, arg2):
    ...
    return ...
```

```
def motors(left, right, seconds=0):
    robot.write(Hamster.LEFT_WHEEL, left)
    robot.write(Hamster.RIGHT_WHEEL, right)
    if seconds > 0:
        robot.wait(seconds * 1000)
        robot.write(Hamster.LEFT_WHEEL, 0)
        robot.write(Hamster.RIGHT_WHEEL, 0)
```

아이들을 위한 프로그램

문제

아이들이 사용하기 쉽도록
함수를 구현하여 제공하기

라이브러리에서 제공하는 API가
로보이드 프레임워크 철학에 따라 만들어지진 것이어서
아이들이 사용하기에는 좀 복잡해 보일 수 있습니다
따라서 제공되는 API를 랩핑해서
아이들이 이해하기 쉬운 함수로 만들어 봅니다
마치 이 강의를 위해 API를 일부러 그렇게 만든 것인 양... ^^;;

디바이스 아이디를 나타내는 상수 값들입니다

Hamster.LEFT_WHEEL	-100 ~ 100	%
Hamster.RIGHT_WHEEL	-100 ~ 100	%
Hamster.BUZZER	0 ~ 167772.15	Hz
Hamster.LEFT_LED	0 ~ 7	
Hamster.RIGHT_LED	0 ~ 7	
Hamster.NOTE	0 ~ 88	
Hamster.SIGNAL_STRENGTH	-128 ~ 0	dB
Hamster.LEFT_PROXIMITY	0 ~ 255	
Hamster.RIGHT_PROXIMITY	0 ~ 255	
Hamster.LEFT_FLOOR	0 ~ 255	
Hamster.RIGHT_FLOOR	0 ~ 255	
Hamster.ACCELERATION	-32768 ~ 32767	
Hamster.LIGHT	0 ~ 65535	
Hamster.TEMPERATURE	-40 ~ 88	℃
Hamster.BATTERY	0 ~ 100	%

```
robot = Hamster()

robot.read(디바이스 아이디, index)
robot.write(디바이스 아이디, index, value)
```

로봇에 대한 API는 함수가 2개 밖에 없습니다
로보이드 프레임워크도 간결함을 추구하니까요 ^^;;
그리고 파이썬 뿐만 아니라 자바스크립트, 프로세싱, 자바, C, C++ 등
모든 곳에서 같은 API 형식을 사용하므로 한 번 익혀 두면
모든 언어에서 같은 방법으로 사용할 수 있습니다

```
robot = Hamster()
```

```
robot.read(디바이스 아이디, index)
robot.write(디바이스 아이디, index, value)
```

디바이스 아이디는 햄스터의 디바이스를 구분하기 위한 것으로
앞앞 페이지에서 나열한 상수 값을 넣으면 됩니다
각 디바이스는 데이터를 배열(파이썬에서는 리스트)에 저장하는데
index는 이 데이터 배열의 인덱스를 의미합니다
value는 디바이스의 데이터 배열에 쓸 값입니다

햄스터의 경우 Acceleration을 제외하고는 모두 값이 하나라서
데이터 배열 크기가 1입니다
즉 index는 무조건 0이라는 뜻이 되구요
Acceleration은 x축, y축, z축 3개가 있으므로 배열 크기는 3이며
x축 값은 인덱스 0, y축 값은 인덱스 1, z축 값은 인덱스 2입니다

대부분의 디바이스는 인덱스가 0이기 때문에
간결함을 위해 index는 생략해도 됩니다
index를 생략하면 index를 0으로 간주합니다

```
robot = Hamster()

robot.read(디바이스 아이디)
robot.write(디바이스 아이디, value)
```

이제 아이들을 위한 함수들을 만들어 봅시다
팀을 짜서 어떤 함수를 만들면 좋을지 토론해서
함수를 결정하고 구현하세요

예)

```
def move(left, right, seconds=0):

def light(left, right, seconds=0):

def beep():
```

코딩 교육 시 주의할 점 몇 가지

1) 문법 위주로 가르치지 않는다
2) 예제를 통해 추리하고 확인하고 추리하고 확인하는 과정을 수행한다
3) 예제를 통해 규칙을 찾고 생각한 규칙이 맞는지 확인해 보면서
   자연스럽게 문법을 익히게 한다
4) 한꺼번에 너무 많은 것을 알려 주려 하지 않는다
5) 아이들에게 생각할 시간과 기회를 많이 주고
   답을 바로 알려 주지 않는다
6) 무언가를 해결하는 것은 그렇게 중요하지 않다
   해결하는 것보다는 해결해 나가는 '과정'이 중요하다

추후 일정

파이썬 #2	8월 15일 강의실 A 좀더 자세히

다음 문제: 물류 로봇 만들기
아마존 키바 시스템

수고하셨습니다 !

akaii@kw.ac.kr