

CS101 – Python 프로그램 작성 예제

Lecture 2

School of Computing
KAIST

학습 목표:

- Python 프로그램의 형태 및 동작을 이해할 수 있다.
- 하향식 (Top-down) 프로그램 설계 방식을 이해 할 수 있다.

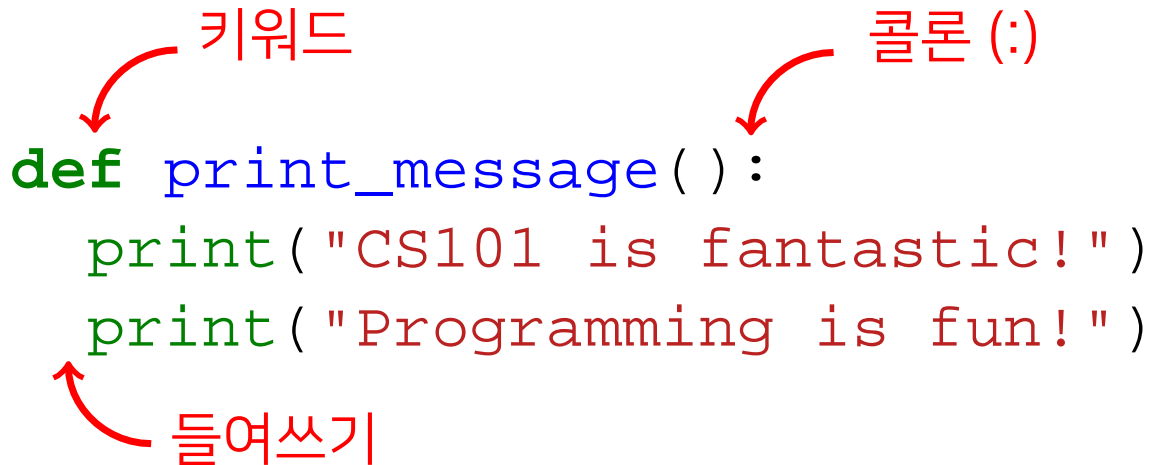
이제 몇 가지 Python 코드를 살펴봅시다.

- Python 상호작용
- Python 프로그램 (스크립트)
- 주석
- 함수
- 키워드
- for 반복문
- 들여쓰기

새로운 함수 만들기

함수는 여러 개의 프로그램 명령어들을 모아 놓은 것입니다.

즉, 함수는 새로운 함수의 이름과 함수가 호출될 때 실행될 명령들로 만듭니다
(함수의 정의란, 함수의 이름과 실행할 명령들을 모아놓은 코드입니다)



```
def print_message( ) :  
    print( "CS101 is fantastic!" )  
    print( "Programming is fun!" )
```

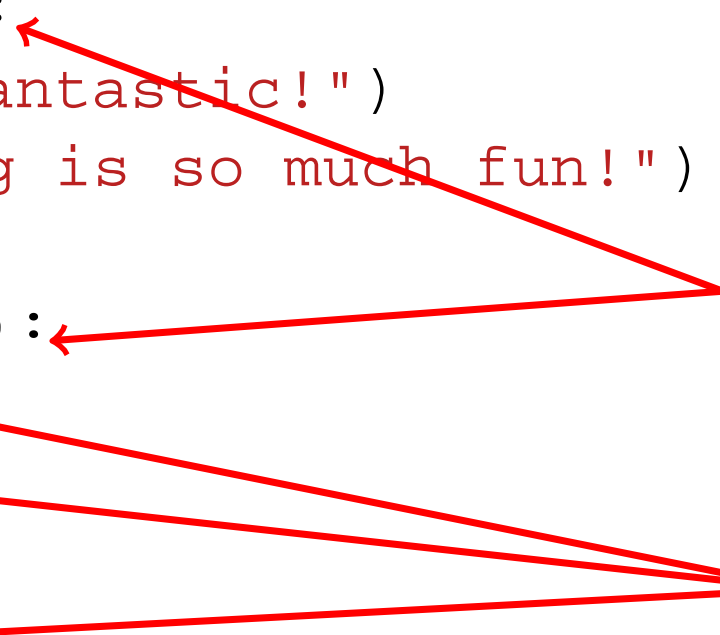
함수 안에서 다른 함수를 호출할 수 있습니다.

```
def repeat_message( ) :  
    print_message( )  
    print_message( )
```

```
def print_message():  
    print("CS101 is fantastic!")  
    print("Programming is so much fun!")  
  
def repeat_message():  
    print_message()  
    print_message()  
  
repeat_message()
```

함수의 정의

함수 호출



프로그램은 첫 번째 명령부터 실행합니다. 명령은 위에서 아래로 하나씩 실행합니다.
함수의 정의는 함수를 정의할 뿐, 정의한 함수가 자동으로 실행 되지는 않습니다.
함수를 호출하면, 그 함수의 정의대로 실행한 후 함수 호출이 종료됩니다.

실습

```
# create a robot with one beeper
```

```
hubo = Robot(b beepers = 1)
```

인자의 기본값을 가진 객체

```
# move one step forward
```

```
hubo.move()
```

```
# turn left 90 degrees
```

```
hubo.turn_left()
```

멤버 함수: 점 (.)을 사용하여 표기

hubo는 어떻게 우회전을 할 수 있을까요?

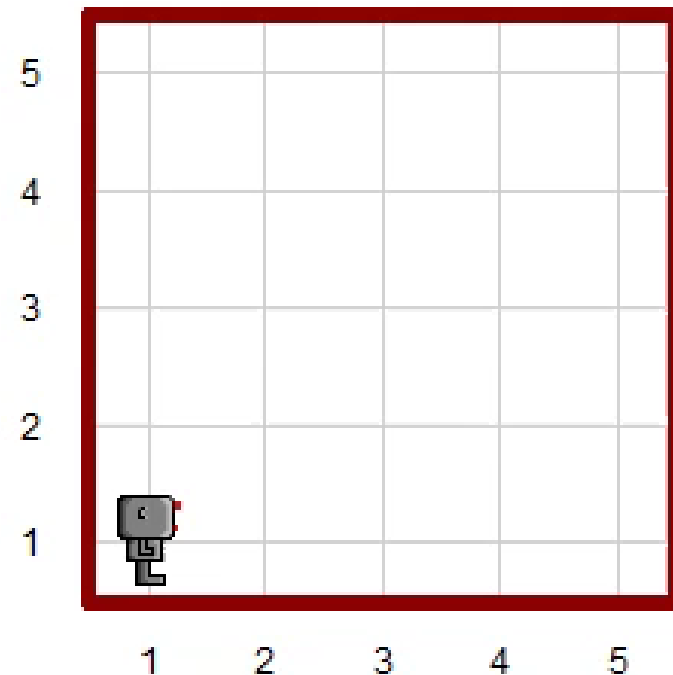
새로운 함수를 만들어서 사용합시다.

```
def turn_right():
```

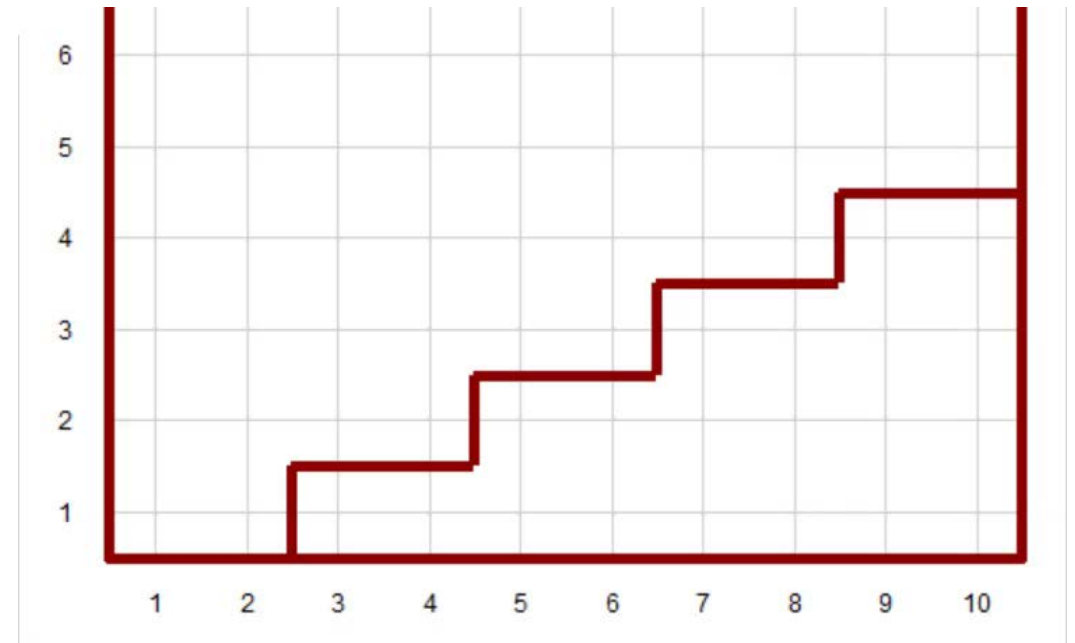
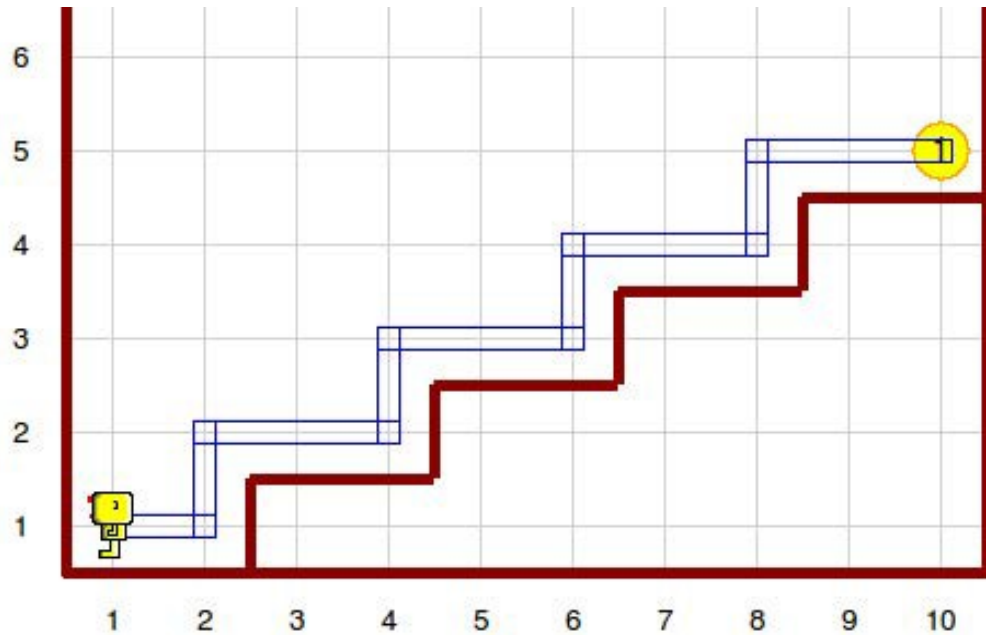
```
    hubo.turn_left()
```

```
    hubo.turn_left()
```

```
    hubo.turn_left()
```



휴보가 계단을 올라가서 문 앞에 신문을 놓고, 처음 위치로 돌아가게 하고 싶습니다.



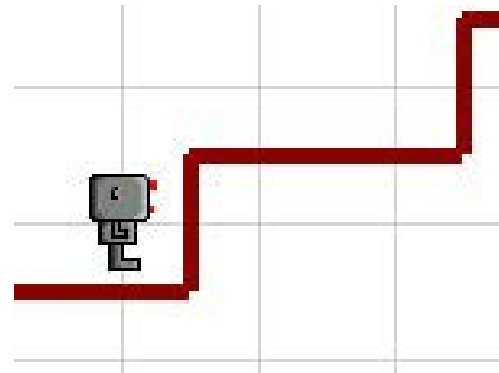
문제의 개요:

- 계단을 4칸 올라가서
- 신문을 놓고
- 돌아서서
- 계단을 4칸 내려간다

Python으로 만들면

```
climb_up_four_stairs()  
hubo.drop_beeper()  
turn_around()  
climb_down_four_stairs()
```

```
def turn_around():  
    hubo.turn_left()  
    hubo.turn_left()  
  
def climb_up_four_stairs():  
    climb_up_one_stair()  
    climb_up_one_stair()  
    climb_up_one_stair()  
    climb_up_one_stair()  
  
def climb_up_one_stair():  
    hubo.turn_left()  
    hubo.move()  
    turn_right()  
    hubo.move()  
    hubo.move()
```



하향식 설계 (Top-down design)

하나의 큰 문제를 중간 크기의 여러 문제로 나누고, 중간 크기의 문제 하나 하나를 어떻게 해결할 지 파악합니다.

그리고, 중간 크기의 문제에 대한 해결책을 작성하기 위해 더 작은 크기의 문제들로 나눕니다.

문제가 쉽게 풀수 있을 만큼 작아지면, 작은 문제에 대한 해결책을 작성한 뒤, 그 해결책들을 모아서 보다 큰 문제에 대한 해결책으로 활용합니다.

간단한 반복

동일한 명령을 4번 반복해서 실행해봅시다.

```
for i in range(4):  
    print("CS101 is fantastic!")
```

← for-반복문

↑ 들여쓰기를 잊지 마세요!

다음 코드와

```
for i in range(4):  
    print("CS101 is great!")  
    print("I love programming!")
```

다음 코드의 차이점은 무엇일까요?

```
for i in range(4):  
    print("CS101 is great!")  
print("I love programming!")
```

실습

반복된 코드 제거하기

```
def climb_up_four_stairs():  
    climb_up_one_stair()  
    climb_up_one_stair()  
    climb_up_one_stair()  
    climb_up_one_stair()
```

프로그램을 작성 할 때, 동일한 내용의 코드를 여러 번 쓰는 것은 피해야 합니다.
for 반복문은 위 코드를 좀 더 깔끔하게 쓸 수 있게 도와줍니다.

```
def climb_up_four_stairs():  
    for i in range(4):  
        climb_up_one_stair()
```

정리 및 예습

본 강의 학습 목표:

- Python 프로그램의 형태 및 동작을 이해할 수 있다.
- 하향식 (Top-down) 프로그램 설계 방식을 이해 할 수 있다.

다음 강의 학습 목표:

- `if` 조건문의 형태 및 동작을 이해 할 수 있다.
- `while` 반복문의 형태 및 동작을 이해할 수 있다.