



**if statement**

**Condition & Loop**

**최성철 교수**  
**Director of TEAMLAB**

# 학점 프로그램을 개발해 보자!

점수	학점
38	
37	
7	
16	
95	
71	
63	
48	
49	
66	
37	

- ① 점수에 따른 학점의 기준을 만든다  
예) 95점 이상 A+, 60점 미만 F
- ② 기준을 바탕으로 첫번째 줄의 점수를 판단한다  
예) 38점은 60점 미만이므로 F
- ③ 다음 줄로 계속 이동하면서 ②를 반복한다  
예) 37점은 60점 미만이므로 F
- ④ 더 이상 점수가 없을 때 프로그램을 종료한다.

프로그램 작성 시,  
조건에 따른 판단과 반복은 필수

# 조건문이란?

조건에 따라 특정한 동작을 하게하는 명령어

프로그램 예시 in 생활

- 지하철 앞 차 간격이 10M 이하면 속도를 10km 이하로 늦춰라
- 사용자가 20세 이하면 VOD를 플레이 하지 마라
- 휴대폰 패턴이 5회 틀리면 20초 동안 대기 상태로 만들어라

조건문은 조건을 나타내는 기준과 실행해야 할 명령으로 구성됨  
조건에 참, 거짓에 따라 실행해야 할 명령이 수행되거나 되지 않음

파이썬은 조건문으로 `if` , `else` , `elif` 등의 명령 키워드를 사용함

# if-else문

가장 기본적인 조건문으로 조건에 따른 명령을 실행

```
print ("Tell me your age?")
myage = int(input()) # 나이를 입력 받아 myage 변수에 할당
if myage < 30:       # myage 가 30 미만일 때
    print ("Welcome to the Club")
else:                # myage 가 30 이상일 때
    print ("Oh! No. You are not accepted.")
```

Editor

입력 받은 값이 “30미만” 이라는 조건에 따라 명령 실행

# if-else문 문법

```
if <조건>:           # if를 쓰고 조건 삽입 후 ":" 입력
    <수행 명령1-1>    # 들여쓰기(indentation)후 수행명령 입력
    <수행 명령1-2>    # 같은 조건하에 실행일 경우 들여쓰기 유지
else:                # 조건이 불일치할 경우 수행할 명령 block
    <수행 명령2-1>    # 조건 불일치 시 수행할 명령 입력
    <수행 명령2-2>    # 조건 불일치 시 수행할 명령 들여쓰기 유지
```

- ① 조건 판단 방법
- ② 조건 일치 시 수행 명령 block ":"와 들여쓰기
- ③ 조건 불일치 시 수행 명령 block

# 조건 판단 방법

- if 다음에 조건을 표기하여 참 또는 거짓을 판단함
- 참/거짓의 구분을 위해서는 비교 연산자를 활용

비교연산자	비교상태	설명
$x < y$	~보다 작음	x과 y보다 작은지 검사
$x > y$	~ 보다 큼	x과 y보다 큰지 검사
$x == y$	같음	x와 y과 같은지 검사 (값과 메모리 주소)
$x \text{ is } y$		
$x != y$	같지 않음	x와 y과 다른지 검사 (값과 메모리 주소)
$x \text{ is not } y$		
$x \geq y$	크거나 같음	x과 y보다 이상인지 검사
$x \leq y$	작거나 같음	x과 y보다 이하인지 검사

# 조건 참/거짓의 구분

- 숫자형의 경우는 수학에서의 참/거짓 과 동일
- 컴퓨터는 존재하면 참 없으면 거짓이라고 판단함

```
>>> if 1:  
...     print "True"  
... else:  
...     print "False"  
...  
True
```

python shell

마찬가지로 if "abc": 는 참, if "": 은 거짓임

# 논리 키워드 사용: and, or, not

- 조건문을 표현할 때 집합의 논리 키워드를 함께 사용하여 참과 거짓을 판단하기도 함

a = 8, b = 5 일 때

if a == 8 and b == 4    # 거짓

if a > 7 or b > 7    # 참

if not (a > 7)    # 거짓, a>7인 것이 참 이므로 거짓으로 판단됨



# 조건 판단 연습 (1/3)

다음 프로그램 수행 결과는?

```
if score >= 90:  
    grade = 'A'  
if score >= 80:  
    grade = 'B'  
if score >= 70:  
    grade = 'C'  
if score >= 60:  
    grade = 'D'  
if score < 60:  
    grade = 'F'  
print grade
```

score	grade
38	
37	
7	
16	
95	
71	
63	
48	
49	
66	
37	

# 조건 판단 연습 (2/3)

다음 프로그램 수행 결과는?

```
if score >= 90:  
    grade = 'A'  
if score >= 80:  
    grade = 'B'  
if score >= 70:  
    grade = 'C'  
if score >= 60:  
    grade = 'D'  
if score < 60:  
    grade = 'F'  
print grade
```

score	grade
38	F
37	F
7	F
16	F
95	D
71	D
63	D
48	F
49	F
66	D
37	F

# 조건 판단 연습 (3/3)

```
if score >= 90:  
    grade = 'A'  
if score >= 80:  
    grade = 'B'  
if score >= 70:  
    grade = 'C'  
if score >= 60:  
    grade = 'D'  
if score < 60:  
    grade = 'F'  
print grade
```

- 모든 if문을 순차적으로 실행
- 95는 90초과지만 동시에 60초과이기도 하므로, 마지막 조건문에 따라 grade 값에 "D"가 할당됨
- 이 문제를 해결하기 위해 **elif**와 **else** 구문이 사용됨

# 조건 판단 연습 수정

```
if score >= 90: grade = 'A'      # 90 이상일 경우 A
elif score >= 80: grade = 'B'    # 80 이상일 경우 B
elif score >= 70: grade = 'C'    # 70 이상일 경우 C
elif score >= 60: grade = 'D'    # 60 이상일 경우 D
else: grade = 'F'                # 모든 조건에 만족하지 못할 경우 F
```

- 수행할 명령문이 한 줄이면 붙여쓰기 가능
- else 키워드 뒤엔 조건 삭제



**Human knowledge belongs to the world.**



**Lab: If Statement**

**Condition & Loop**

**최성철 교수**  
**Director of TEAMLAB**

# [연습] 무슨 학교 다니세요?

## 태어난 연도를 계산하여 학교 종류를 맞추는 프로그램

당신이 태어난 년도를 입력하세요

Terminal

1994 # 자신이 태어난 연도 입력

대학생

- 나이는 2017 - 태어난 년도 +1 로 계산
- 26세 이하 20세 이상 이면 "대학생", 20세 미만 17세 이상 이면 "고등학생"  
17세 미만 14세 이상 이면 "중학생", 14세 미만 8세 이상이면 "초등학생"  
그 외의 경우는 "학생이 아닙니다" 출력



**Human knowledge belongs to the world.**



**for statement**

**Condition & Loop**

**최성철 교수**  
**Director of TEAMLAB**

01100  
00110

---

# 반복문이란?

정해진 동작을 반복적으로 수행하게 하는 명령문

## 프로그램 예시 in 생활

- 100명의 학생에 성적을 산출할 때
- 왓챠에서 영화 추천하기
- 워드에서 단어 바꾸기 명령 실행

반복문은 반복 시작 조건, 종료 조건, 수행 명령으로 구성됨

반복문 역시 반복 구문은 들여쓰기와 block으로 구분됨

파이썬은 반복문으로 for, while 등의 명령 키워드를 사용함

# for문 (1/2)

기본적인 반복문, 반복 범위를 지정하여 반복문 수행

```
for loopier in [1,2,3,4,5]:  
    print ("hello")
```

- ① loopier 변수에 1 할당
- ② "Hello" 출력
- ③ 리스트(대괄호속 숫자들) 있는 값 차례로 loopier 할당
- ④ 5까지 할당한 후 반복 block 수행 후 종료

```
for loopier in [1,2,3,4,5]:  
    print (loopier)
```

만약 100번 반복해야 하는 프로그램을 짜야 한다면?

# for문 (2/2)

## range () 사용하기

```
for loop in [1,2,3,4,5]:  
    print ("hello")  
for loop in range(0,5):  
    print "hello"
```

Editor

### 왜 range(1,5) 과 아닌 range (0,5) 인가?

: range()는 마지막 숫자 바로 앞까지 리스트를 만들어줌

즉, range(1,5) = [1,2,3,4] 까지 같은 의미

※ range(0,5) = [0,1,2,3,4] = range(5)는 같은 의미

---

# [알아두면 상식] 반복문

## 반복문 변수명

- ✓ 임시적인 반복 변수는 대부분  $i, j, k$ 로 정함
- ✓ 이것은 수학에서 변수를  $x, y, z$ 로 정하는 것과 유사한 관례

## 0부터 시작하는 반복문

- ✓ 반복문은 대부분 0부터 반복을 시작
- ✓ 이것도 일종의 관례로 1부터 시작하는 언어도 존재
- ✓ 2진수가 0부터 시작하기 때문에 0부터 시작하는 걸 권장

## 무한 loop

- ✓ 반복 명령이 끝나지 않는 프로그램 오류
- ✓ CPU와 메모리 등 컴퓨터의 리소스를 과다하게 점유

# for문의 다양한 반복문 조건 표현 (1/2)

## 문자열을 한자씩 리스트로 처리

```
for i in 'abcdefg':  
    print (i)
```

Editor

## 각각의 문자열 리스트로 처리

```
for i in ['Americano', 'latte', 'fratuchino']:  
    print (i)
```

Editor

# for문의 다양한 반복문 조건 표현 (2/2)

## 간격을 두고 세기

```
for i in range(1, 10, 2):  
    # 1부터 10까지 2씩 증가시키면서 반복문 수행  
    print (i)
```

Editor

## 역순으로 반복문 수행

```
for i in range(10, 1, -1):  
    # 10부터 1까지 -1씩 감소시키면서 반복문 수행  
    print (i)
```

Editor

# while문

조건이 만족하는 동안 반복 명령문을 수행

```
i = 1
while i < 10:
    print (i)
    i += 1
```

- ① i 변수에 1 할당
- ② i가 10 미만인지 판단
- ③ 조건에 만족할 때 i 출력, i에 1을 더함
- ④ i가 10이 되면 반복 종료

Editor

## for문은 while문으로 변환 가능

반복 실행횟수를 명확히 알 때

```
for i in range(0,5):
    print (i)
```

Editor

반복 실행횟수가 명확하지 않을 때

```
i = 0
while i < 5:
    print (i)
    i = i + 1
```

Editor



# 반복의 제어 – break, continue

## break 특정 조건에서 반복 종료

```
for i in range(10):  
    if i == 5: break      # i가 5가 되면 반복 종료  
    print (i)  
print ("EOP")           # 반복 종료 후 "EOP" 출력
```

Editor

## continue 특정 조건에서 남은 반복 명령 skip

```
for i in range(10):  
    if i == 5: continue  # i가 5가 되면 i를 출력하지 않음  
    print (i)  
print ("EOP")           # 반복 종료 후 "EOP" 출력
```

Editor

# 반복의 제어 – else

반복 조건이 만족하지 않을 경우 반복 종료 시 1회 수행

```
for i in range(10):  
    print (i),  
else:  
    print ("EOP")
```

Editor

```
i = 0  
while i < 10:  
    print (i),  
    i += 1  
else:  
    print ("EOP")
```

Editor

※ break로 종료된 반복문은 else block이 수행되지 않음



**Human knowledge belongs to the world.**



**Lab: Multiple Table Game**

**Condition & Loop**

**최성철 교수**  
**Director of TEAMLAB**

# [연습] 구구단 계산기

아래와 같이 출력되는 프로그램을 만드시오

구구단 몇단을 계산할까요?

5

구구단 5단을 계산합니다.

5 X 1 = 5

5 X 2 = 10

5 X 3 = 15

.....

5 X 8 = 40

5 X 9 = 45

CMD

# Loop Review

```
sentence = "I love you"
reverse_sentence = ""
for char in sentence:
    reverse_sentence = char + reverse_sentence
print(reverse_sentence)
```

<u>Loop</u>	<u>reverse_sentence<sup>1</sup></u>	<u>reverse_sentence<sup>2</sup></u>	<u>char</u>
0		I	I
1	I	I	
2	I	II	I
3	II	oI	o
4	oI	voI	v
5	voI	evol	e
6	evol	evol	
7	evol	yevol	y
8	yevol	oyevol	o
9	oyevol	uoyevol	u

```

decimal = 10
result = ""
while (decimal > 0):
    remainder = decimal % 2
    decimal = decimal / 2
    result = str(remainder) + result
print (result)

```

수식

$$\begin{array}{r}
 2 \overline{) 10} \dots 0 \\
 2 \overline{) 5} \dots 1 \\
 2 \overline{) 2} \dots 0 \\
 2 \overline{) 1}
 \end{array}$$

<u>Loop</u>	<u>decimal<sup>1</sup></u>	<u>remainder</u>	<u>decimal<sup>2</sup></u>	<u>result<sup>1</sup></u>	<u>result<sup>2</sup></u>
0	10	0	5		0
1	5	1	2	0	10
2	2	0	1	10	010
3	1	1	0	010	1010



# Debugging Loop

```
print ("input decimal number: ",)
decimal = int(input())
result = ""
loop_counter = 0
while (decimal > 0):
    temp_decimal_input = decimal
    temp_result_input=result

    remainder = decimal % 2
    decimal = decimal // 2
    result = str(remainder) + result

    print ("-----", loop_counter, "loop value check ----- ")
    print ("Initial decimal:", temp_decimal_input,
          ", Remainder:", remainder,
          ", Initial result", temp_result_input)
    print ("Output decimal:", decimal,
          "Output result:", result)
    print ("-----")
    print ("")

    loop_counter += 1
print ("Binary number is", result)
```

binary\_converter.py

Editor

**Loop 내에 변수들의 값을  
Print문으로 확인**



**Human knowledge belongs to the world.**

**Lab: Condition and Loop**

**Condition & Loop**

**최성철 교수**  
**Director of TEAMLAB**

01100  
00110

---

# 가변적인 중첩 반복문 (variable nested loops)

실제 프로그램에서는 반복문은

사용자의 입력에 따라 가변적으로 반복되고

하나의 반복이 아닌 중첩되어 반복이 일어남

# [연습] 숫자 찾기 게임

## 1~100 임의의 숫자를 맞추시오

guess\_number.py

Editor

```
# -*- coding: utf-8 -*-
import random                # 난수 발생 함수 호출
guess_number = random.randint(1, 100)  # 1~100 사이 정수 난수 발생
print ("숫자를 맞춰보세요 (1 ~ 100)")
users_input = int(input())    # 사용자 입력을 받음
while (users_input is not guess_number):    # 사용자 입력과 난수가 같은지 판단
    if users_input > guess_number:          # 사용자 입력이 클 경우
        print ("숫자가 너무 큼니다")
    else:                                   # 사용자 입력이 작은 경우
        print ("숫자가 너무 작습니다")
    users_input = int(input())              # 다시 사용자 입력을 받음
else: print ("정답입니다. ", "입력한 숫자는 ", users_input , "입니다")  # 종료 조건
```

# [연습] 연속적인 구구단 입력

1~9 입력 받아 구구단을 출력, 0을 입력 시 종료

구구단 몇 단을 계산할까요(1~9)?

5

구구단 5단을 계산합니다.

$$5 \times 1 = 5$$

$$5 \times 2 = 10$$

$$5 \times 3 = 15$$

.....

$$5 \times 8 = 40$$

$$5 \times 9 = 45$$

구구단 몇 단을 계산할까요(1~9)?

10

0

구구단 게임을 종료합니다

# [정답] 연속적인 구구단 입력

nested\_gugudan.py

Editor

```
print ("구구단 몇 단을 계산할까요(1~9)?")
```

```
x = 1
```

```
while (x is not 0):
```

```
    x = int(input())
```

```
    if x == 0: break
```

```
    if not(1 <= x <= 9):
```

```
        print ("잘못 입력하셨습니다", "1부터 9사이 숫자를 입력해주세요")
```

```
        continue
```

```
    else:
```

```
        print ("구구단 " + str(x) + "단을 계산합니다.")
```

```
        for i in range(1,10):
```

```
            print (str(x) + " X " + str(i) + " = " + str(x*i))
```

```
        print ("구구단 몇 단을 계산할까요(1~9)?")
```

```
print ("구구단 게임을 종료합니다")
```

전체 loop

0이 입력될 때까지  
게임이 실행됨

구구단 loop

구구단 계산용 loop

# [연습] 이차원 리스트 처리하기

사람 별 평균을 구하라

```
kor_score = [49, 79, 20, 100, 80]
math_score = [43, 59, 85, 30, 90]
eng_score = [49, 79, 48, 60, 100]
midterm_score = [kor_score, math_score, eng_score]
print (midterm_score[0][2])
```

	A	B	C	D	E
국어점수	49	79	20	100	80
수학점수	43	59	85	30	90
영어점수	49	79	48	60	100



# [연습] 이차원 리스트처리하기

two\_dim\_list\_loop.py

Editor

```
student_score = [0,0,0,0,0]
i = 0
for subject in midterm_score:
    for score in subject:
        student_score[i] += score    # 각 학생마다 개별로 교과 점수를 저장
        i += 1                      # 학생 index 구분
    i = 0                          # 과목이 바뀔 때 학생 인덱스 초기화
else:
    a, b, c, d, e = student_score    # 학생 별 점수를 unpacking
    student_average = [a/3,b/3,c/3,d/3,e/3] #
    print (student_average)
```



**Human knowledge belongs to the world.**



How to debug code

Condition & Loop

최성철 교수  
Director of TEAMLAB



[www.phdcomics.com](http://www.phdcomics.com)

<http://phdcomics.com/>

---

# Debugging (디버깅)

- 코드의 오류를 발견하여 수정하는 과정
- 오류의 '원인'을 알고 '해결책'을 찾아야 함
- 문법적 에러를 찾기 위한 에러 메시지 분석
- 논리적 에러를 찾기 위한 테스트도 중요

**문법적 에러**

# 초보자에게 흔히 생기는 문법 오류

## 들여쓰기 (Indentation Error)

```
x = 2
  y = 5 # IndentationError
print (x+y)
```

# 초보자에게 흔히 생기는 문법 오류

## 오타자

```
pront (x+y) # Not Print, But Pront
```

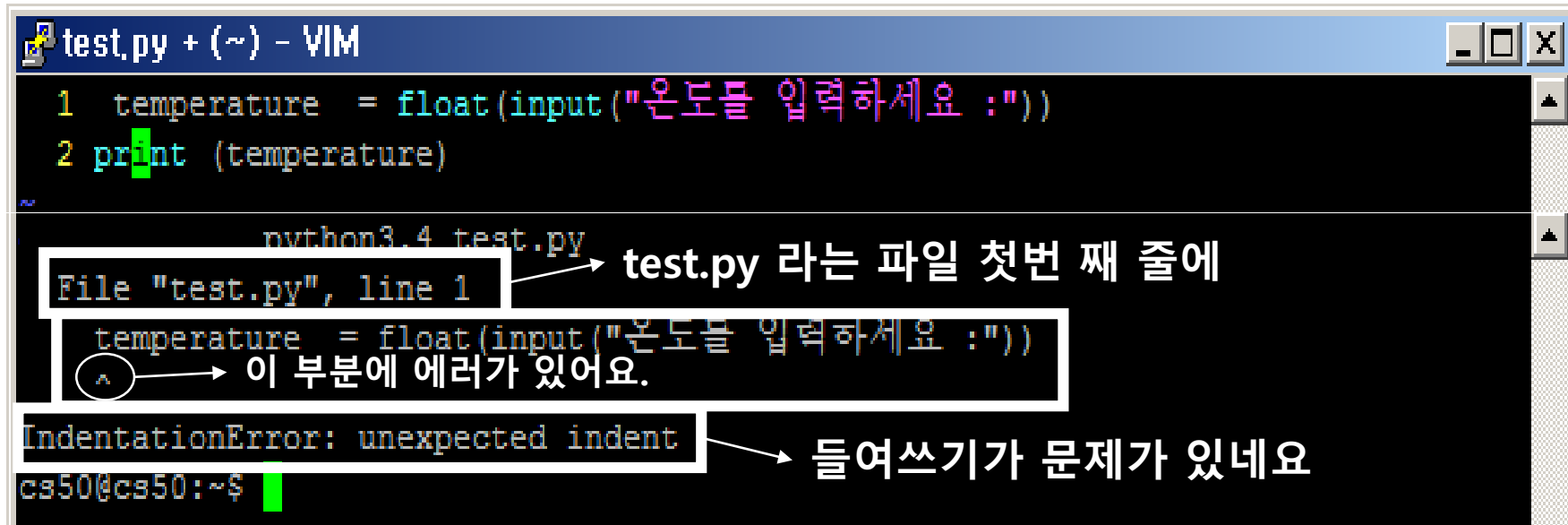
## 대소문자 구분 안 함

```
gachon = "ACE"  
print (Gachon) # g는 소문자
```



# 문법 에러 – Error 메시지 분석

## - 에러가 발생하면 파이썬이 알려줌



```
test.py + (~) - VIM
1 temperature = float(input("온도를 입력하세요 :"))
2 print (temperature)

python3.4 test.py
File "test.py", line 1
temperature = float(input("온도를 입력하세요 :"))
^
IndentationError: unexpected indent
cs50@cs50:~$
```

test.py 라는 파일 첫번 째 줄에

이 부분에 에러가 있어요.

들여쓰기가 문제가 있네요

Indentation Error – 흔히 발생하는 에러  
들여쓰기를 **Space**로 했나 **Tab**으로 했냐?

# 문법 에러 – Error 메시지 분석

```
test.py (~) - VIM
1 temperature = float(input("온도를 입력하세요 :"))
2 print (temperatre) temperature != temperatre
~

cs50@cs50: ~
cs50@cs50:~$ python3.4 test.py
온도를 입력하세요 :10.3
Traceback (most recent call last):
  File "test.py", line 2, in <module>
    print (temperatre)
NameError: name 'temperatre' is not defined
cs50@cs50:~$
```

test.py 라는 파일 두 번째 줄에  
이 부분에 에러가 있어요.

tempearatre라는 이름이 뭔지 모르겠어요.

오타자 / 대문자 등에 의한 에러 발생

논리적 에러

---

# 논리적 에러

- 논리적 에러 - 뜻대로 실행이 안되는 코드
- 중간 중간 프린터 문을 찍어서 확인
- Loop Review 처럼 해보기!

# 논리적 에러

<http://dodnet.tistory.com/186>

Trapezoid

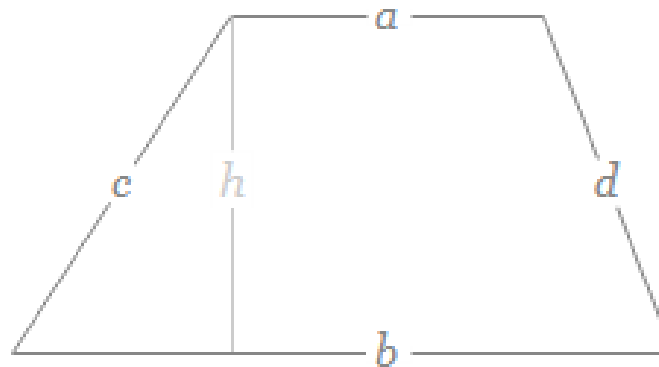
Solve for area ▾

$$A = \frac{a+b}{2}h$$

*a* Base

*b* Base

*h* Height



# 논리적 에러 – 함수 Check

```
def addition(x, y):  
    return x+y
```

```
def multiplication(x, y):  
    return x*y
```

```
def divided_by_2(x):  
    return x/2
```

Editor

trapezium\_area.py

```
>>> import trapezium_area as ta  
      #trapezium_area 파일을 ta라는 이름으로 부름  
>>> ta.addition(10,5)  
15  
>>> ta.multiplication(10,5)  
50  
>>> ta.divided_by_2(50)  
25  
>>>
```

Python Shell

# 논리적 에러 – 함수 Check Print문

Editor

```
def addition(x, y):  
    return x+y
```

```
def multiplication(x, y):  
    return x*y
```

```
def divided_by_2(x):  
    return x/2
```

# Python Shell에서 호출 할 경우 실행되지 않음

```
if __name__ == '__main__':  
    print(addition(10,5))  
    print(multiplication(10,5))  
    print(multiplication(50))
```

trapezium\_area\_test.py

# 논리적 에러 – 함수 Check Print문

```
python trapezium_area_test.py
```

```
addition : 15
```

```
multiplication : 50
```

```
divided_by_2 : 25.0    #결과가 올바르게 출력 됐는지 확인 가능
```

Editor

※ if `__name__ == "__main__"` 차이

```
>>> import trapezium_area_test
```

```
>>>
```

있을 경우

```
>>> import trapezium_area_test
```

```
addtion : 15
```

```
multiplication : 50
```

```
divided_by_2 : 25.0
```

없을 경우



# 함수 합치기 – 사다리꼴 넓이

Editor

```
def addition(x, y):  
    return x+y  
def divided_by_2(x):  
    return x/2  
  
def main():  
    base_line = float(input("밑변의 길이는?"))  
    upper_edge = float(input("윗변의 길이는?"))  
    height = float(input("높이는?"))  
  
    print("넓이는 :", divided_by_2(addition(base_line, upper_edge) *  
        height))  
  
if __name__ == "__main__":  
    main()
```

스스로 해결 해 보기

---

# 인터넷에 물어보기

- 모든 문제는 Google + stack overflow 로 해결 가능
- stack overflow: 전 세계 개발자들의 코딩 Q&A 사이트  
코딩계의 네이버 지식인

# 인터넷에 물어보기



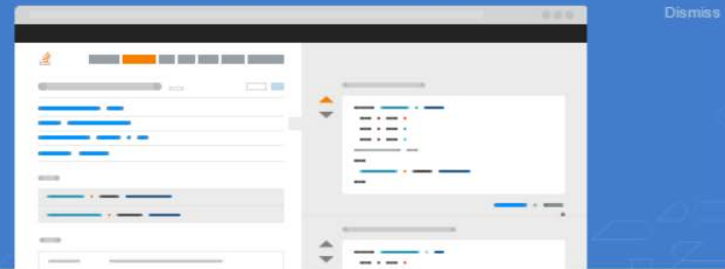
Questions Jobs Documentation Beta Tags Users Badges Ask Question

## Announcing Stack Overflow Documentation

We started with Q&A. Technical documentation is next, and we need your help.  
Whether you're a beginner or an experienced developer, you *can* contribute.

Sign up and start helping →

Learn more about Documentation →



## Top Questions

interesting

454 featured

hot

week

month

0 votes 0 answers 1 view

Wireless Communication and Network Research

networking wireless telecommunication communication-protocol

asked 6 secs ago Abdulhameed 49

0 votes 1 answer 8 views

Bootstrap, Modal with link inside data-content once clicked close the current modal and shows another modal?

javascript html twitter-bootstrap modal-dialog

answered 35 secs ago Bùi văn Nguyễn 129

0 votes 0 answers 4 views

Hi i'm using swrevealviewController and pushing to other view controller by using the bellow code

ios objective-c

modified 1 min ago NDoc 7,339

0 votes 0 answers 47 views

Register with Node/Express and Passport

## Looking for a job?

### Senior Software Engineer

Buzzvil  Seoul, South Korea

\$50,000 - \$60,000

python

### Full Stack Developer - \$60k

Crossover  No office location

REMOTE

python react-native

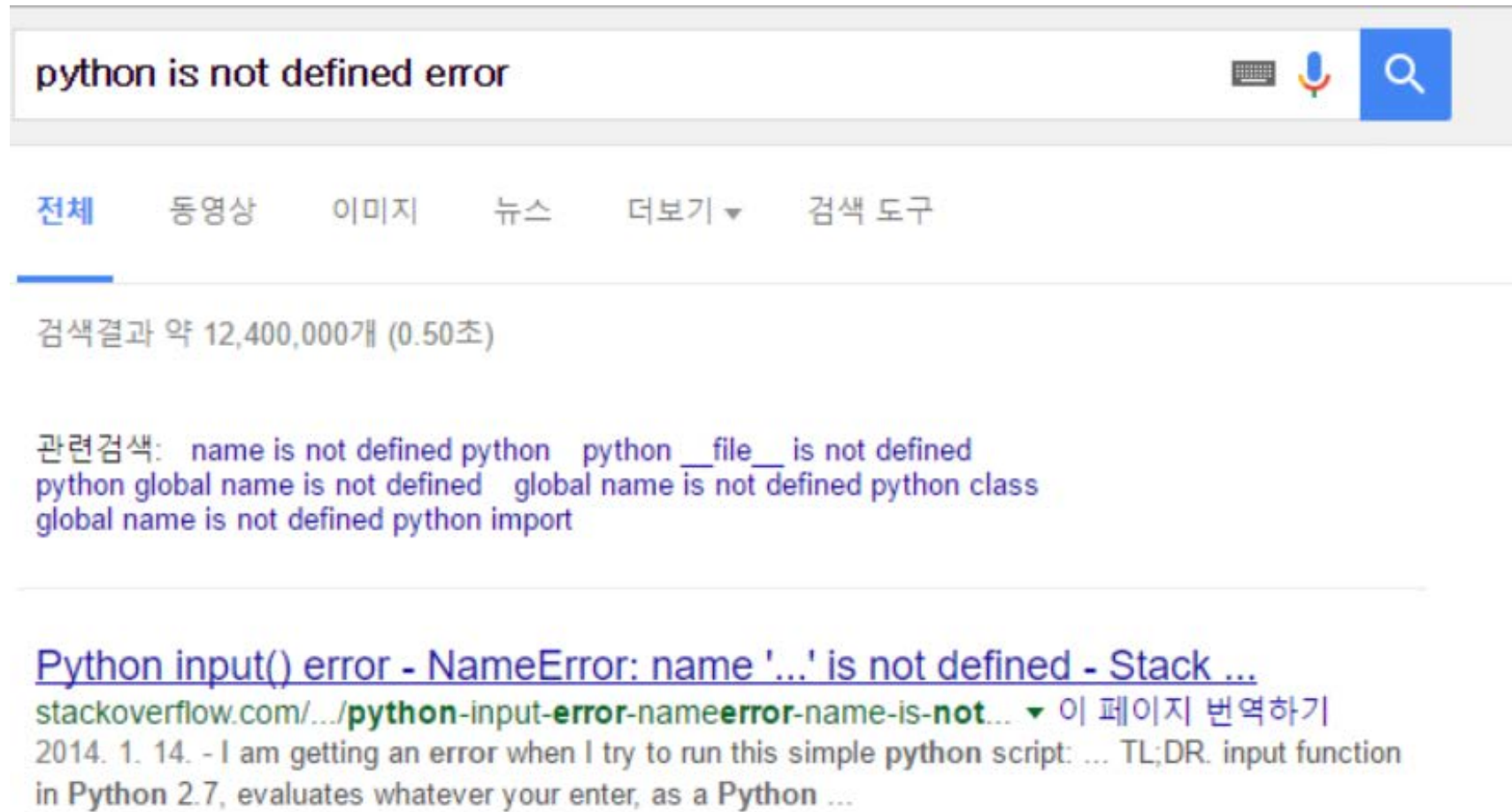
### 웹서비스 백엔드 개발자 (Web Service Back-End Developer)

IDK2 (아이디케이즈소프트)

 Seoul, South Korea

aws node.js

# 인터넷에 물어보기



# 인터넷에 물어보기

Python input() error - NameError: name '...' is not defined



▲ I am getting an error when I try to run this simple python script:

43

```
input_variable = input ("Enter your name: ")  
print ("your name is" + input_variable)
```



Let's say I type in "dude", the error I am getting is:



20

```
line 1, in <module>  
input_variable = input ("Enter your name: ")  
File "<string>", line 1, in <module>  
NameError: name 'dude' is not defined
```

I am running Mac OS X 10.9.1 and I am using the Python Launcher app that came with the install of python 3.3 to run the script.

Edit: I realized I am somehow running these scripts with 2.7. I guess the real question is how do I run my scripts with version 3.3? I thought if I dragged and dropped my scripts on top of the Python Launcher app that is inside the Python 3.3 folder in my applications folder that it would launch my scripts using 3.3. I guess this method still launches scripts with 2.7. So How do I use 3.3?

# 인터넷에 물어보기



You are running Python 2, not Python 3. For this to work in Python 2, use `raw_input` .

11



```
input_variable = raw_input ("Enter your name: ")
print ("your name is" + input_variable)
```

[share](#) [improve this answer](#)

edited Mar 6 at 21:03



davidism

32.2k ● 10 ● 45 ● 71

answered Jan 14 '14 at 19:53



llamadillo

171 ● 1 ● 10

[add a comment](#)



Since you are writing for Python 3.x, you'll want to begin your script with:

0



```
#!/usr/bin/env python3
```

If you use:

```
#!/usr/bin/env python
```

It will default to Python 2.x. These go on the first line of your script, if there is nothing that starts with `#!` (aka the shebang).

If your scripts just start with:

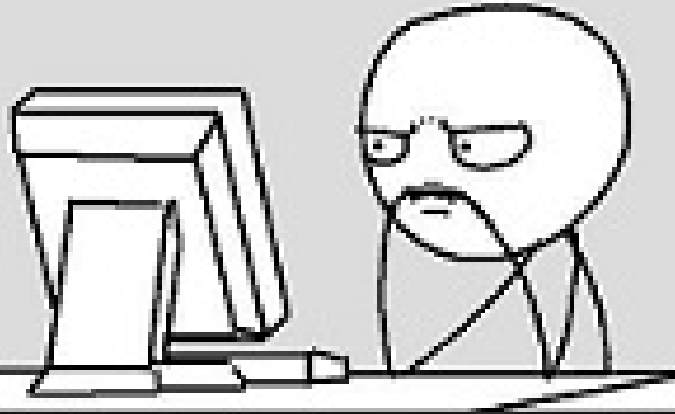
---

# 추천 사이트

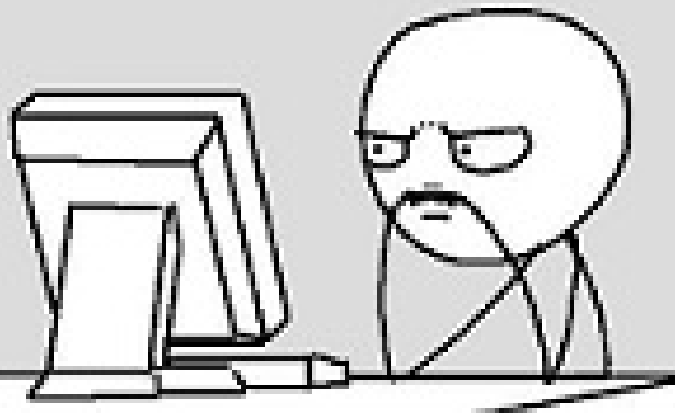
- 점프 투 파이썬 - <https://wikidocs.net/book/1>
- 코드카데미 - <https://www.codecademy.com/>
- 코드파이트 - <https://codefights.com/>
- 생활 코딩 - <https://opentutorials.org/course/1750>
- Couseira – python 검색



It doesn't work..... why?



It works..... why?



http://www.4mat.org

<http://goo.gl/PSvBDp>



**Human knowledge belongs to the world.**