PYTHON PROGRAMMING

2023/09/28 15:00 PM

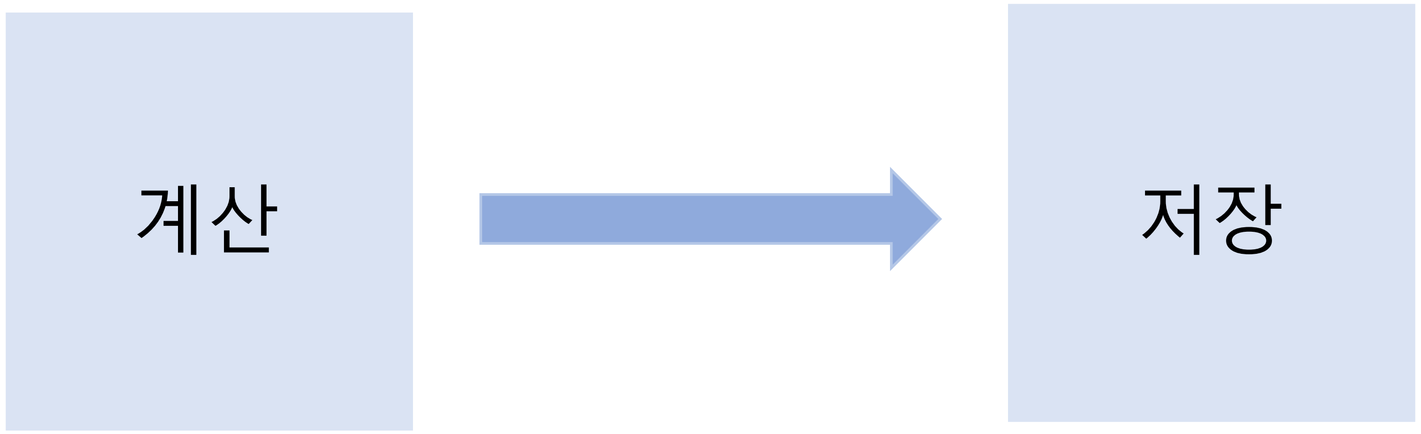
레슨 소개

- Programming 기초
- Python 3.6 이상 사용
- <The MIT Press Introduction to Computation and Programming Using Python, 2nd Edition>

CONTENTS

01. Getting Started

컴퓨터의 기능



Types of Knowledge

DECLARATIVE KNOWLEDGE	**IMPERATIVE KNOWLEDGE**
composed of statements of fact	how-to knowledge

`x = 0`
`y*y = x`
	Find the biggest number
	Find the square root

ALGORITHM 필요!

Algorithm

- == COMPUTATION

- Flow of control that specifies when each step is to be executed

- Example:
  - Q. 사용자에게 정수 3개를 입력받은 뒤 가장 큰 수 찾기

    1. 정수 입력받기
    2. 입력받은 숫자들을 '리스트'에 담기
    3. '리스트'의 원소들에 대해 for문을 사용하여 숫자들의 크기를 비교

       3-1. 가장 큰 수를 저장할 변수 초기화

       3-2. for문 설계 …

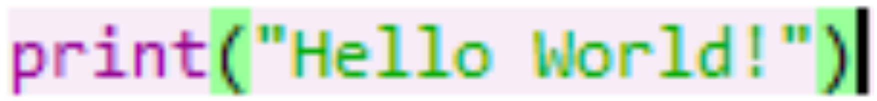
Types of errors

SYNTAX ERROR	SEMANTIC ERROR
문법적 오류	의미 오류
CAT DOG BOY	3.2/BOY

02. Introduction to Python

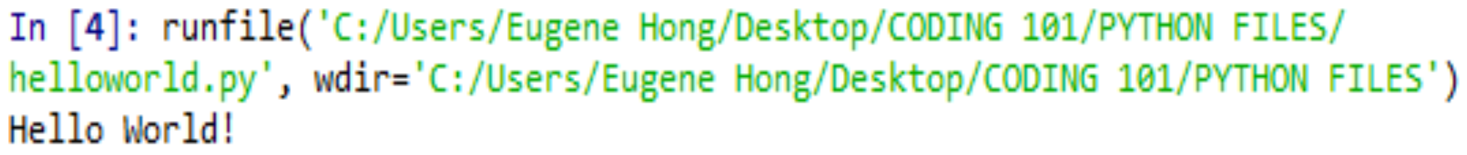
Running a Python Program

COMMAND



```
print("Hello World!")
```

OUTPUT



```
In [4]: runfile('C:/Users/Eugene Hong/Desktop/CODING 101/PYTHON FILES/
helloworld.py', wdir='C:/Users/Eugene Hong/Desktop/CODING 101/PYTHON FILES')
Hello World!
```

Object Type

- **SCALAR: atomic or invisible**
  - Integer (ex: 0, 1, 2)
  - Floating point number (ex: 0.0, -1.29, 0.009)
  - Boolean (ex: `True, False`)
  - None
- **NON-SCALAR**
  - String (ex: `“Programming”, “Algorithm”`)

Operator

- **int / float**
  - +, -, *, / (integer division: 몫과 나머지 보여 줌)
  - // (floating division: 몫만 보여 줌)
  - % (나머지만 보여 줌)
  - ** (제곱)
  - == (같음)
  - != (같지 않음)

- **Boolean**
  - `and`
  - `or`
  - `not`

Object (cont.)

- **Object + Operator = Expression**

- Operator type에 따라 다른 연산
  - ex: `3*4 = 12` (int * int)
    `3*'a' = 'aaa'` (int * string)

- 모든 숫자 (int, float) < 모든 문자열 (String)

Variable

- 변수: Symbolic Address Names
  - better to use easily read names
  - case-sensitive (`A != a, B != b`)
  - reserved-keyword (ex: `and, break, list, continue, class`)
  - assigned by =

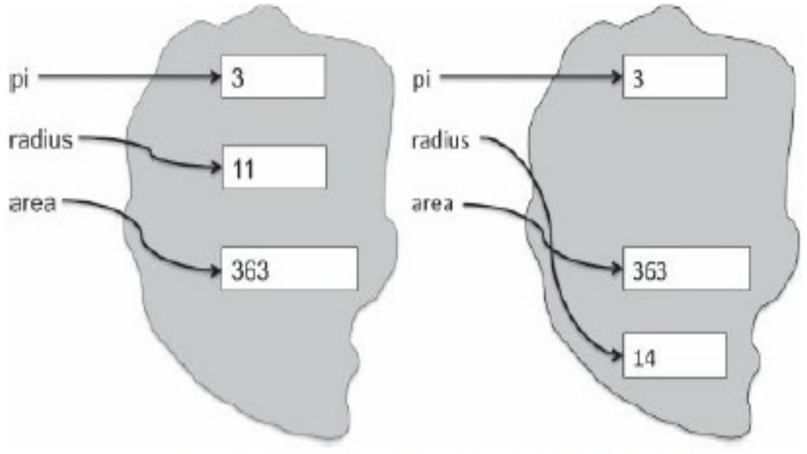


Figure 2.2 Binding of variables to objects

```
pi = 3
radius = 11
area = pi * (radius**2)
radius = 14
```

03. Basic Programming

Conditional Statement

```
if Boolean expression:
    block of code
else:
    block of code
```

* indentation 중요!

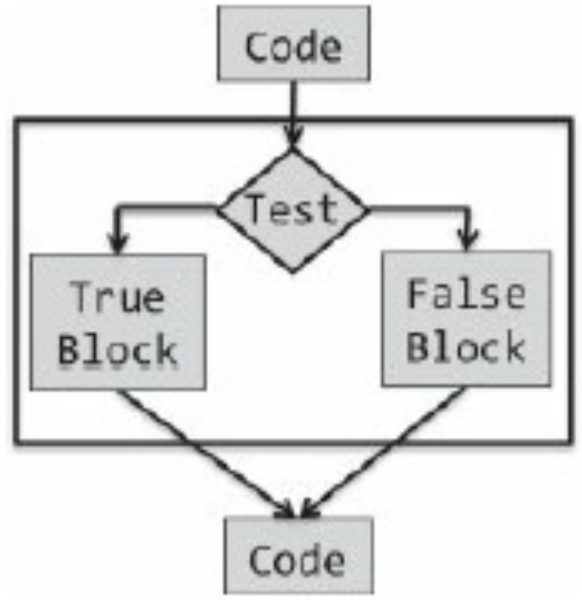
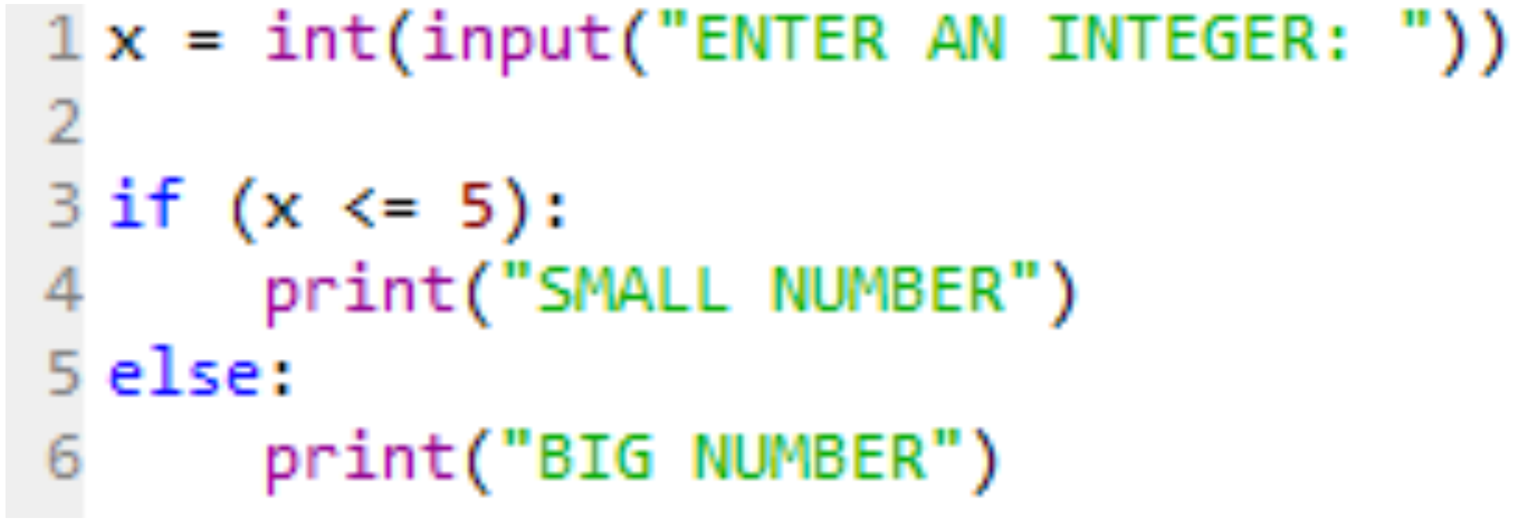


Figure 2.3 Flow chart for conditional statement

Conditional Statement (cont.)

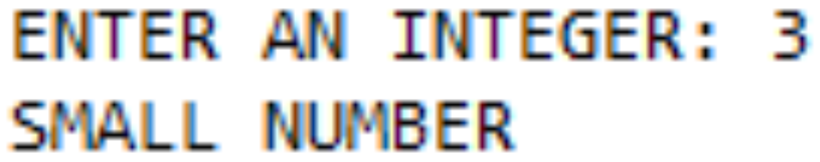
- Example:



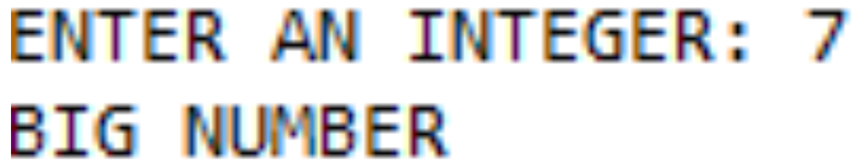
```
x = int(input("ENTER AN INTEGER: "))

if (x <= 5):
    print("SMALL NUMBER")
else:
    print("BIG NUMBER")
```

Conditional Statement (cont.)

- Example answer:



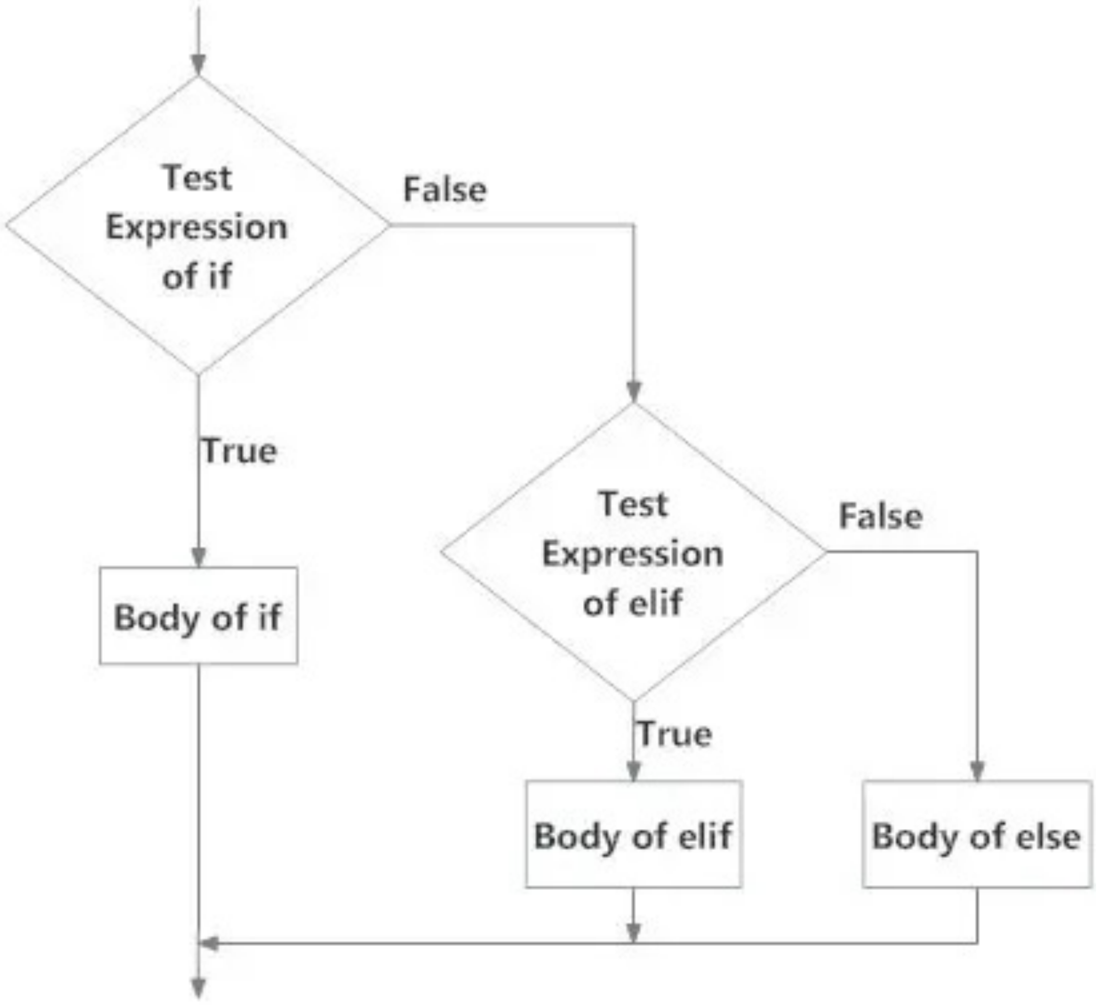
```
ENTER AN INTEGER: 3
SMALL NUMBER
```



```
ENTER AN INTEGER: 7
BIG NUMBER
```

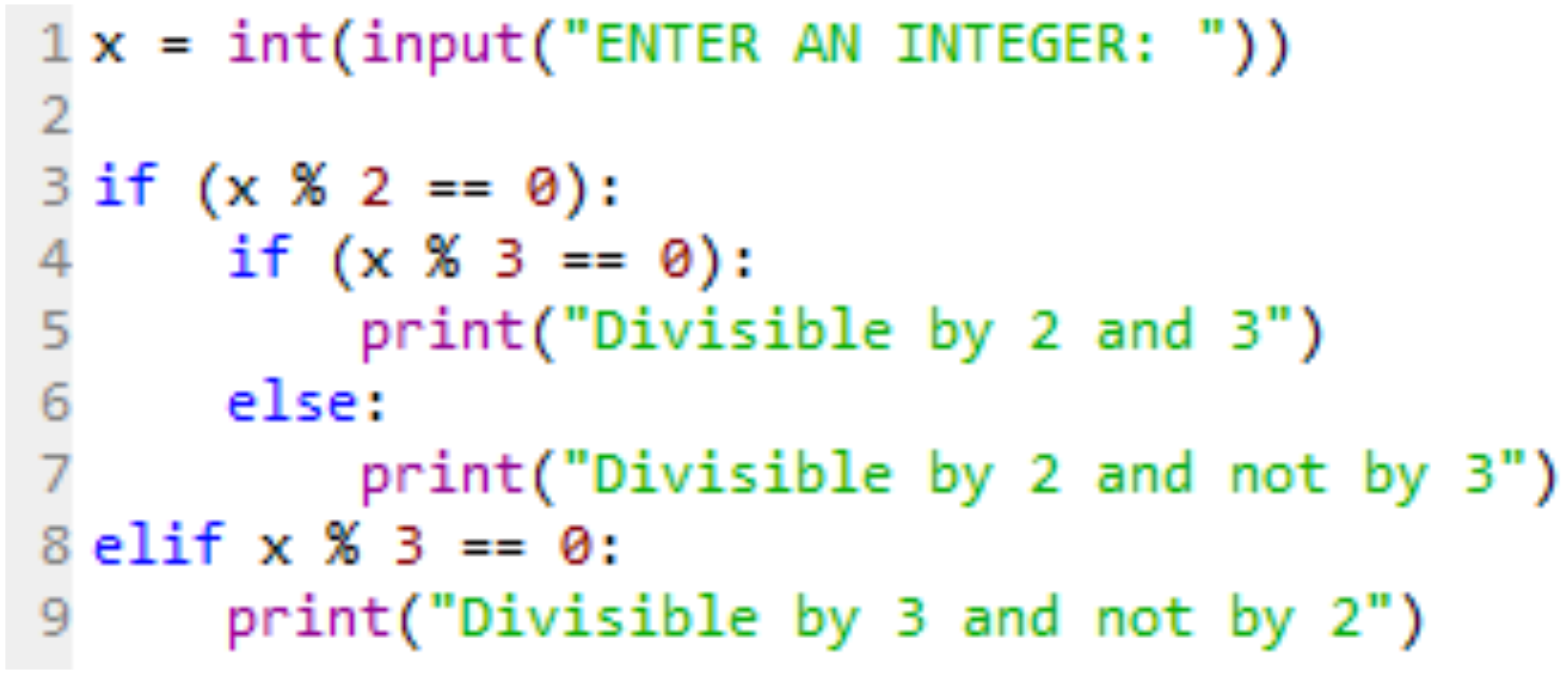
Nested Statement

```
if Boolean expression:
    block of code
    if Boolean expression:
        block of code
    else:
        block of code
elif Boolean expression:
    block of code
else:
    block of code
```



Nested Statement (cont.)

- Example:



```python
x = int(input("ENTER AN INTEGER: "))

if (x % 2 == 0):
    if (x % 3 == 0):
        print("Divisible by 2 and 3")
    else:
        print("Divisible by 2 and not by 3")
elif x % 3 == 0:
    print("Divisible by 3 and not by 2")
```

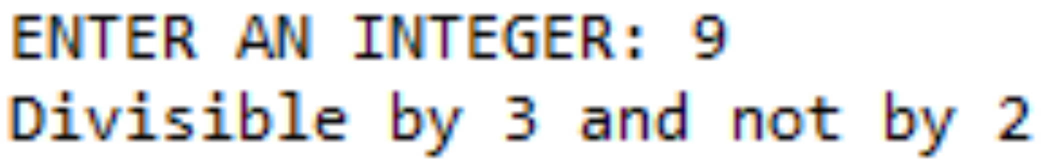
1) x = 9 ?

2) x = 4 ?

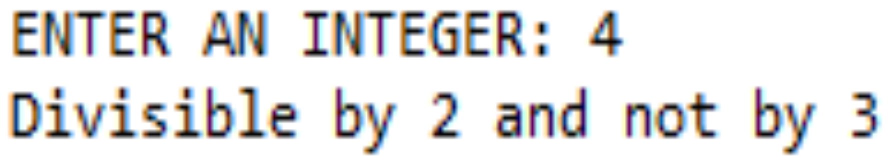
3) x =12 ?

Nested Statements (cont.)

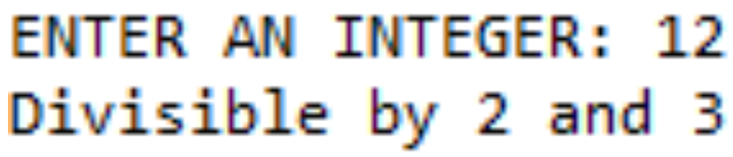
- Example answer:



```
ENTER AN INTEGER: 9
Divisible by 3 and not by 2
```



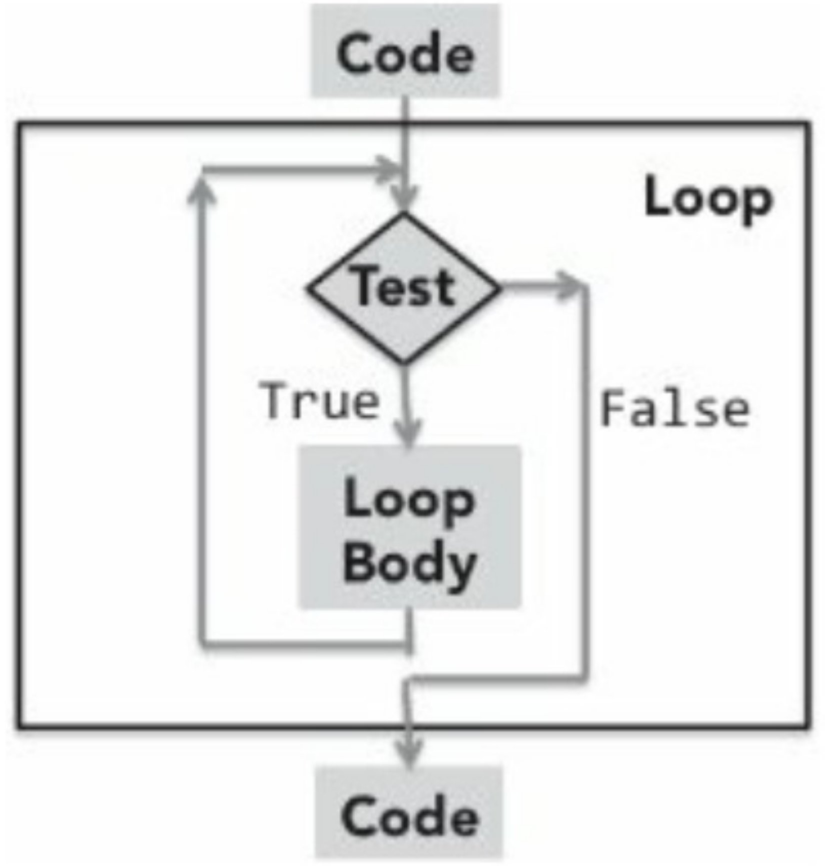
```
ENTER AN INTEGER: 4
Divisible by 2 and not by 3
```



```
ENTER AN INTEGER: 12
Divisible by 2 and 3
```

Iteration

- When we want a program to do the same thing many times, we can use **iteration**
- consists of *test condition* and *loop body*
- can be written by using `while` statement

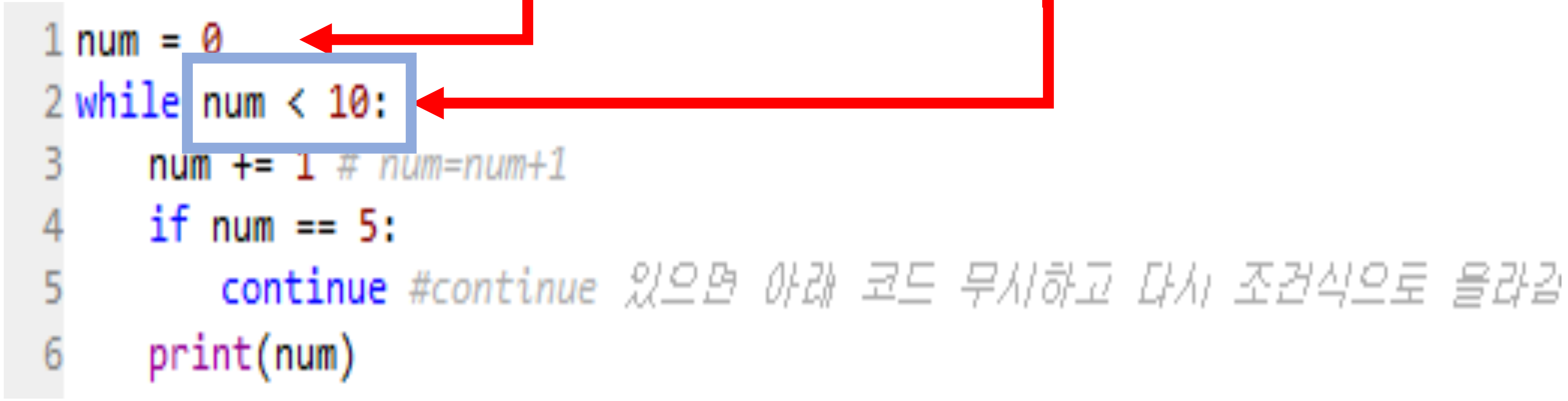


Iteration (cont.)

- Example #1:

변수 num 초기화

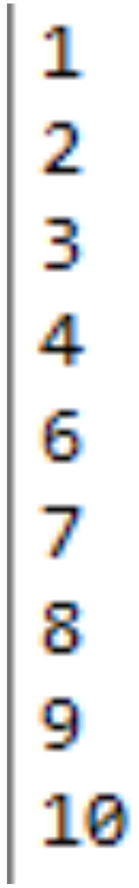
while loop 안의 code block의 조건 불만족할 때까지 실행



```
num = 0
while num < 10:
    num += 1 # num=num+1
    if num == 5:
        continue #continue 있으면 아래 코드 무시하고 다시 조건식으로 올라감
    print(num)
```

Iteration (cont.)

- Example #1 (answer)



```
1
2
3
4
6
7
8
9
10
```

Result:

`num == 5` 였을 때 (`while` loop 조건 불만족)

⇒ `print(num)` 문이 수행되지 않음

Iteration (cont.)

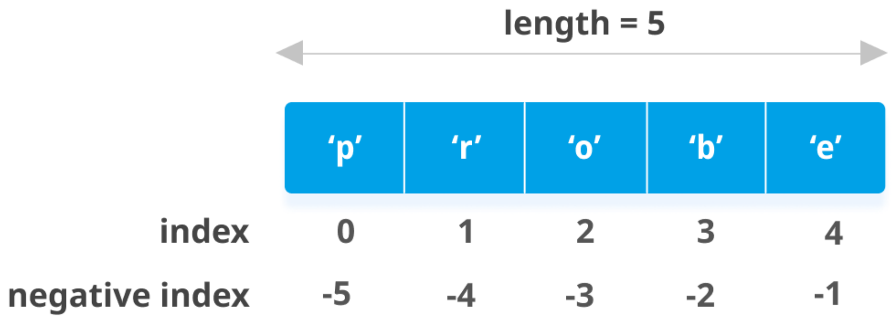
- Example #2:

```
# Square an integer, the hard way
x = 3
ans = 0
itersLeft = x
while (itersLeft != 0):
   ans = ans + x
   itersLeft = itersLeft - 1
print(str(x) + '*' + str(x) + ' = ' + str(ans))
```

Test #	x	ans	itersLeft
1	3	0	3
2	3	3	2
3	3	6	1
4	3	9	0

String Manipulation

- length : *len*(“*string* ”)
- indexing : *string*[0]
    - 특정 위치의 문자를 추출
    - index는 0부터 시작



- slicing : *string*[start : end-1]
- input : name = input(“*string* ”)
- type conversions (type casting): type(another type)

String Manipulation (cont.)

- Example
  - len(“abc”) $\longrightarrow$ 3
  - coding[0] $\longrightarrow$ ‘c’
  - python[0 : 2] $\longrightarrow$ ‘py’
  - name = input(“Enter a name: ”)

04. Numerical Programs

For Loop

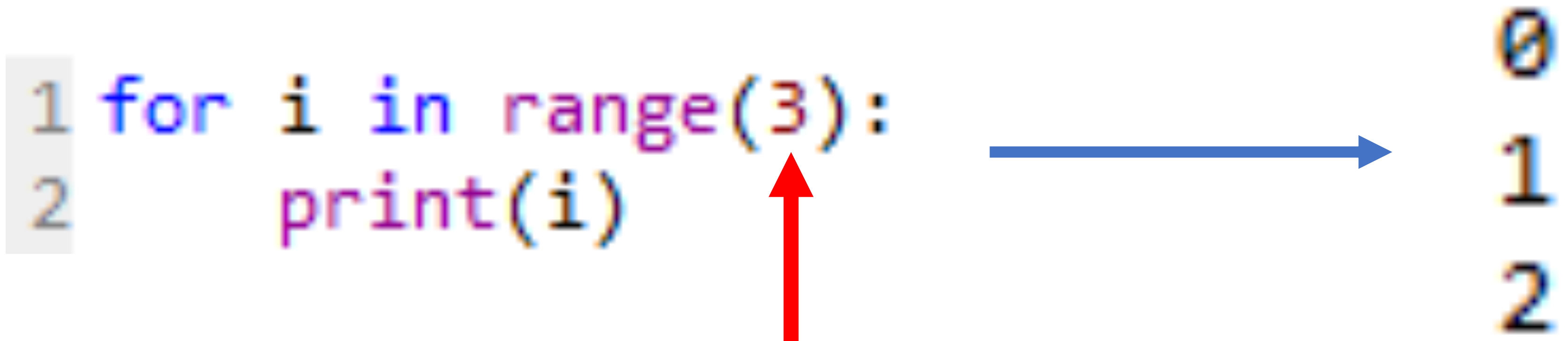
```
for i in range(a, b, c):
    code block
```

range(a, b, c) 안에서 a부터 b-1까지, c step으로 실행

✓ a와 c는 비울 수 있음 ⟹ a=0, c=1로 default 설정

For Loop (cont.)

- Example #1



```
for i in range(3):
    print(i)
```

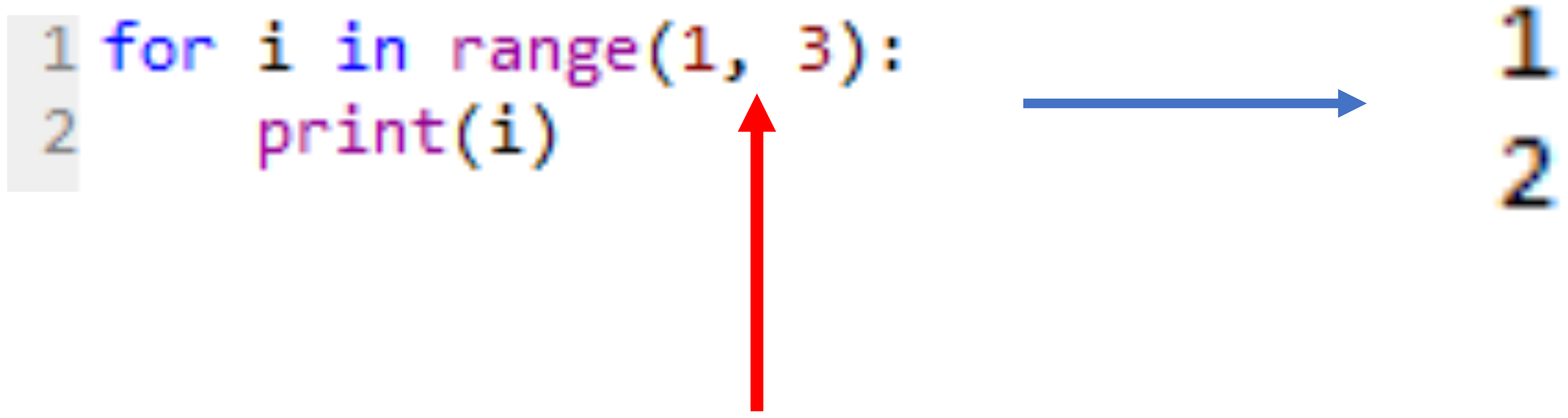
```
0
1
2
```

`range(a)` 안에서 변수 `i`에 0부터 `a-1`까지 bind

⟹ 0, 1, 2

For Loop (cont.)

- Example #2



```python
for i in range(1, 3):
    print(i)
```

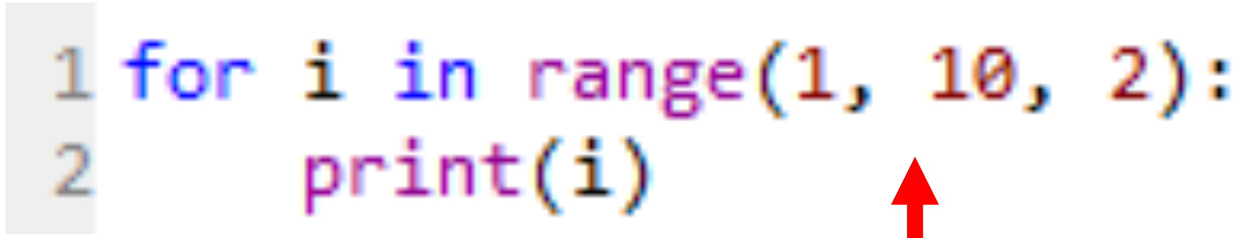
```
1
2
```

- `range(a, b)` 안에서 변수 `i`에 `a` (1)부터 `b-1` (2)까지 bind
- a부터 ( ) 안에 든 숫자 횟수만큼 `for loop` 실행

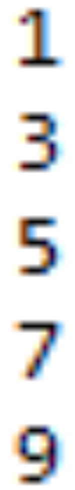
  ⇒ 1, 2

For Loop (cont.)

- Example #3



```python
for i in range(1, 10, 2):
    print(i)
```



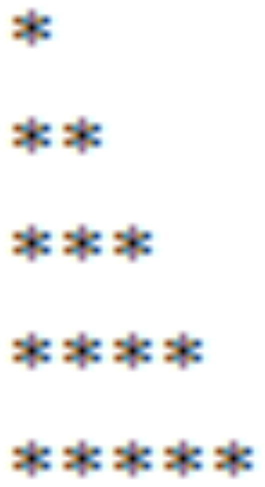
```
1
3
5
7
9
```

`range(a, b, c)` 안에서 변수 `i`에 `a`부터 `b-1`까지 bind, `c` step으로 실행!

⟹ 1, 3, 5, 7, 9 (`c`값인 2씩 띄어서 실행)

For Loop (cont.)

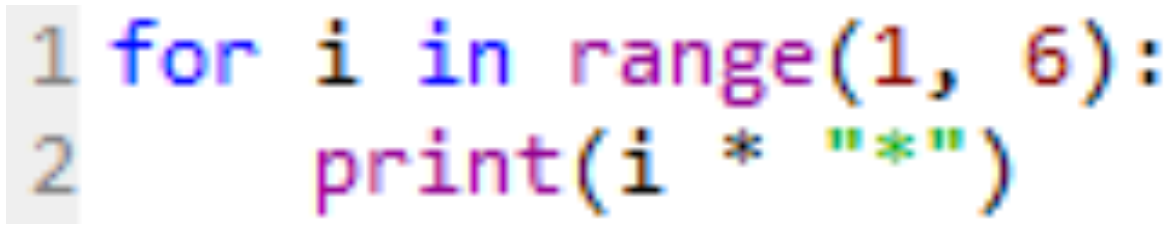
- Quiz #1
  - 아래와 같은 출력 결과가 나오도록 코드를 작성하세요.



```
*
**
***
****
*****
```

For Loop (cont.)

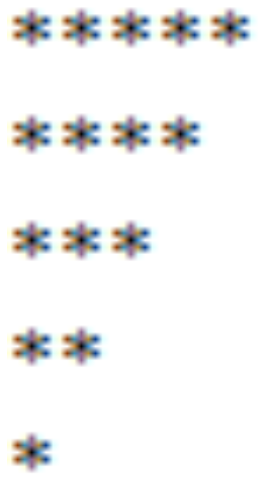
- Quiz #1 (answer)



```python
for i in range(1, 6):
    print(i * "*")
```

For Loop (cont.)

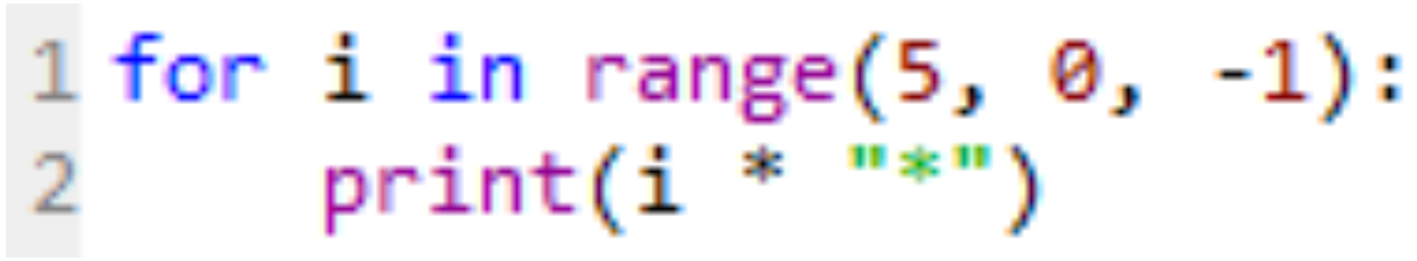
- Quiz #2
  - 아래와 같은 출력 결과가 나오도록 코드를 작성하세요.



```
*****
****
***
**
*
```

For Loop (cont.)

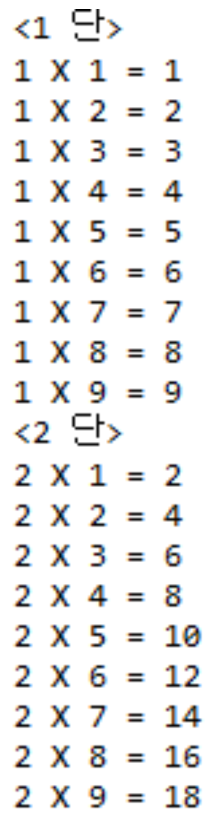
- Quiz #2 (answer)



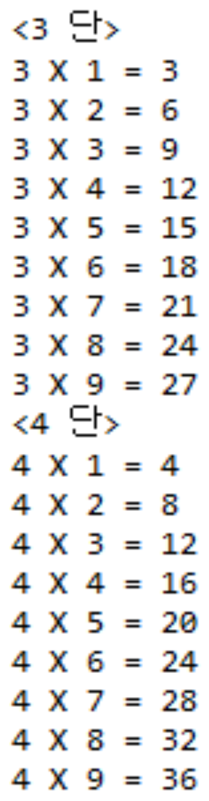
```python
for i in range(5, 0, -1):
    print(i * "*")
```

For Loop (cont.)

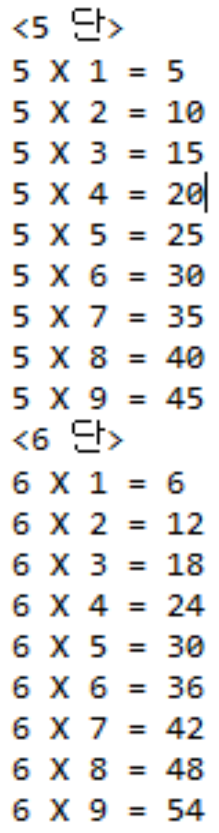
- Quiz #3
  - 1단부터 9단까지 구구단을 출력하도록 코드를 작성하세요.



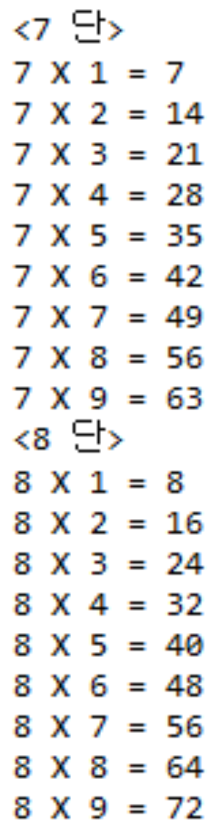
```
<1 단>
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
<2 단>
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
```



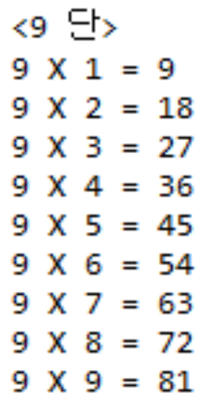
```
<3 단>
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
3 X 8 = 24
3 X 9 = 27
<4 단>
4 X 1 = 4
4 X 2 = 8
4 X 3 = 12
4 X 4 = 16
4 X 5 = 20
4 X 6 = 24
4 X 7 = 28
4 X 8 = 32
4 X 9 = 36
```



```
<5 단>
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
<6 단>
6 X 1 = 6
6 X 2 = 12
6 X 3 = 18
6 X 4 = 24
6 X 5 = 30
6 X 6 = 36
6 X 7 = 42
6 X 8 = 48
6 X 9 = 54
```



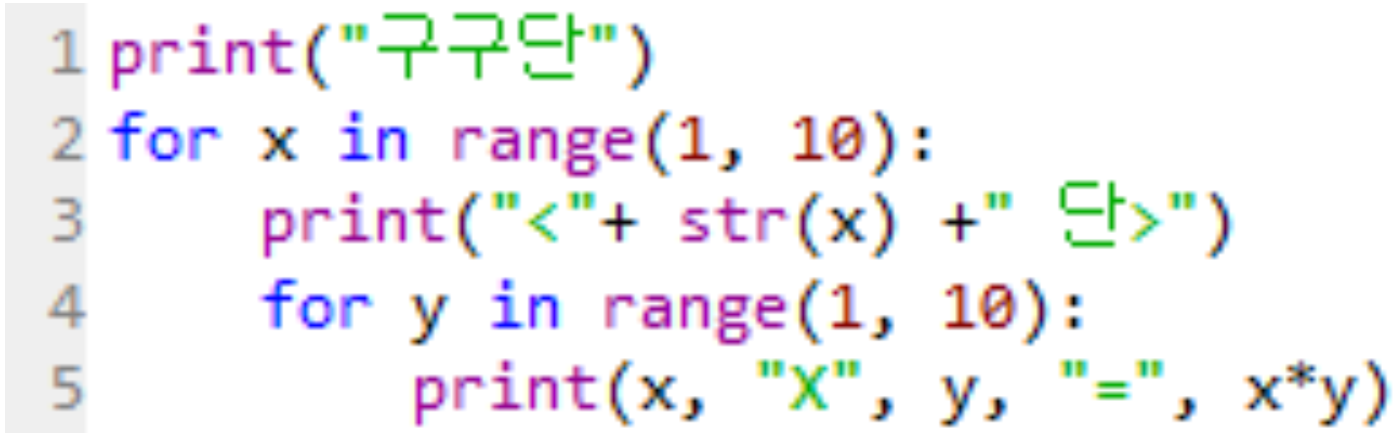
```
<7 단>
7 X 1 = 7
7 X 2 = 14
7 X 3 = 21
7 X 4 = 28
7 X 5 = 35
7 X 6 = 42
7 X 7 = 49
7 X 8 = 56
7 X 9 = 63
<8 단>
8 X 1 = 8
8 X 2 = 16
8 X 3 = 24
8 X 4 = 32
8 X 5 = 40
8 X 6 = 48
8 X 7 = 56
8 X 8 = 64
8 X 9 = 72
```



```
<9 단>
9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
```

For Loop (cont.)

- Quiz #3 (answer)



```python
print("구구단")
for x in range(1, 10):
    print("<"+ str(x) +" 단>")
    for y in range(1, 10):
        print(x, "X", y, "=", x*y)
```

Nested For Loops

```
for (i in range):
      code block (outer block)
      for j in range:
            code block (inner block)
      code block (outer block)
```

- The range function in *the outer loop* is evaluated only once.
- But the range function in *the inner loop* is evaluated each time *the inner for statement* is reached.

Nested For Loops (cont.)

- The arguments to the `range` function in the line with `for` are evaluated just before the first iteration of the loop, and not reevaluated for `for` subsequent iterations.

Nested For Loops (cont.)

- Example #1:

```
x = 4
for j in range(x):
    for i in range(x):
        print(i)
        x = 2
```

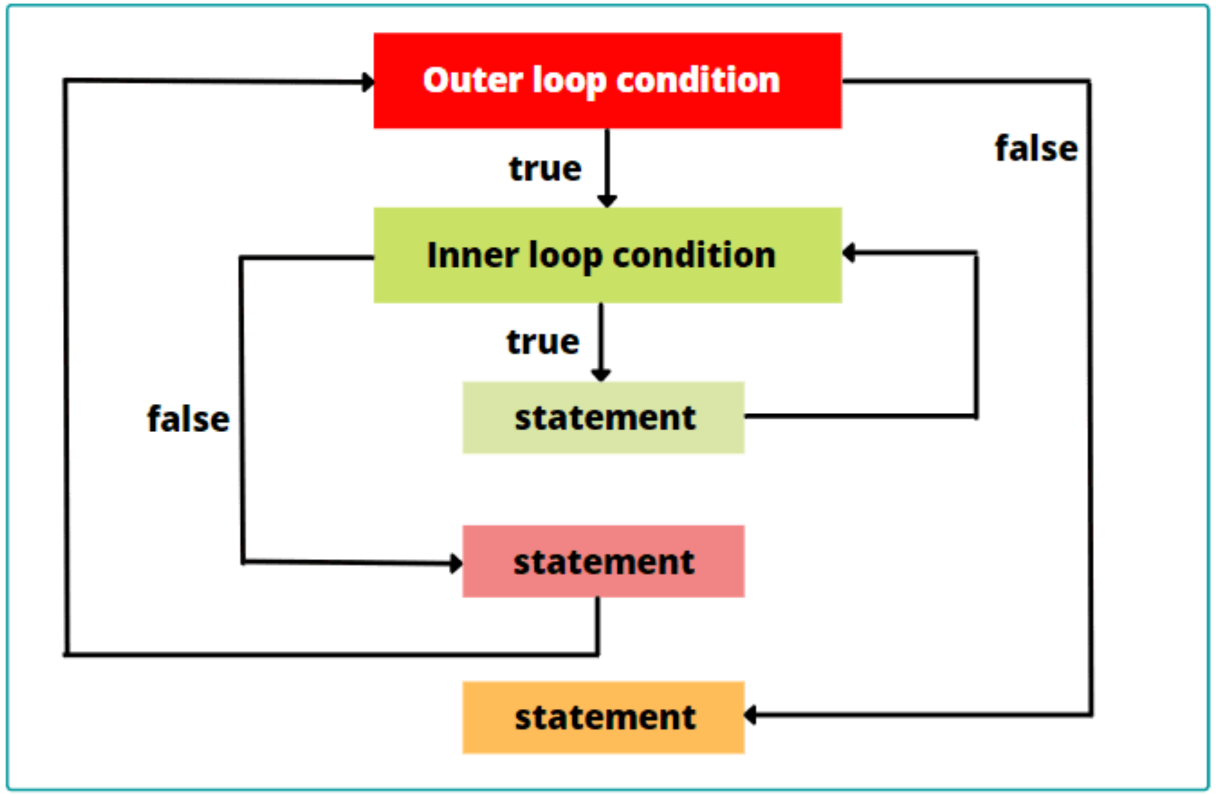
```
0
1
2
3
0
1
0
1
0
1
```

- The range function in *the outer loop* is evaluated only once.

- But the range function in *the inner loop* is evaluated each time *the inner for statement* is reached.

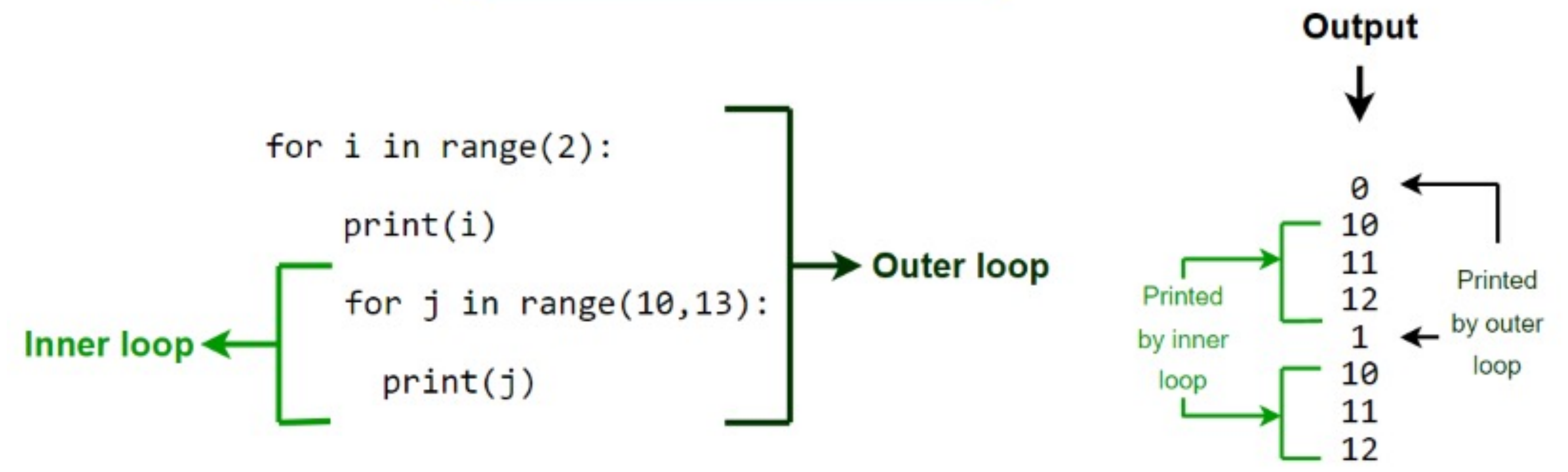
Nested For Loops (cont.)

- `Break`
  - exits *the innermost loop* in which it is closed
  - 가장 안쪽 *loop* 빠져나옴



Nested For Loops (cont.)

- Example #1

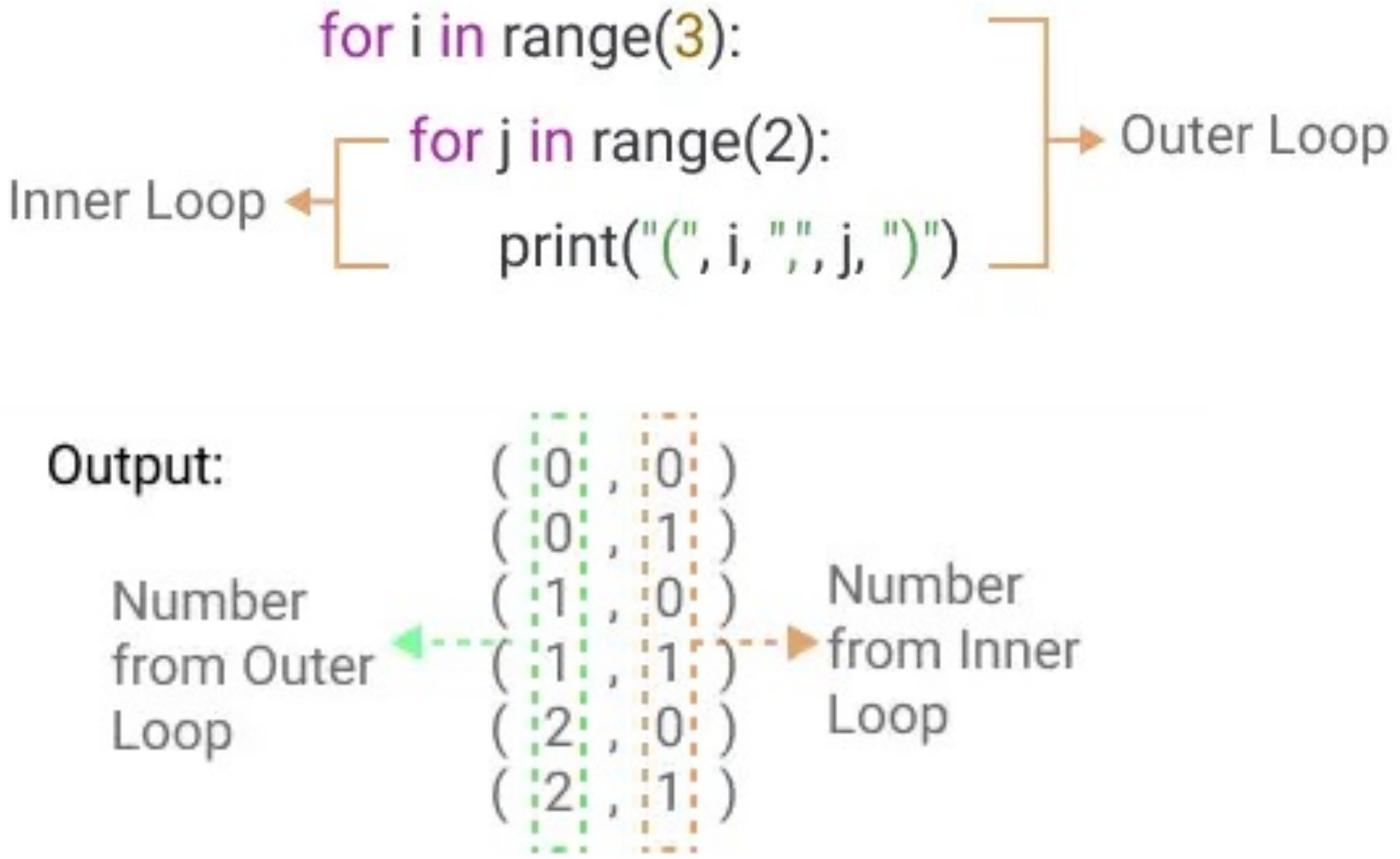


```
for i in range(2):
    print(i)
    for j in range(10,13):
      print(j)
```

```
0
10
11
12
1
10
11
12
```

Nested For Loops (cont.)

- Example #2



```
for i in range(3):
    for j in range(2):
        print("(", i, ",", j, ")")
```

Output:

```
( 0 , 0 )
( 0 , 1 )
( 1 , 0 )
( 1 , 1 )
( 2 , 0 )
( 2 , 1 )
```

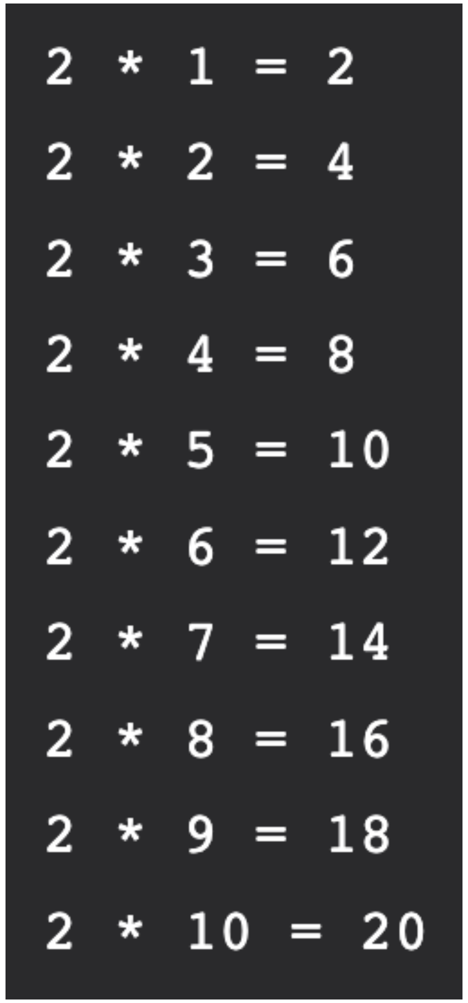
Nested For Loops (cont.)

- Quiz #1

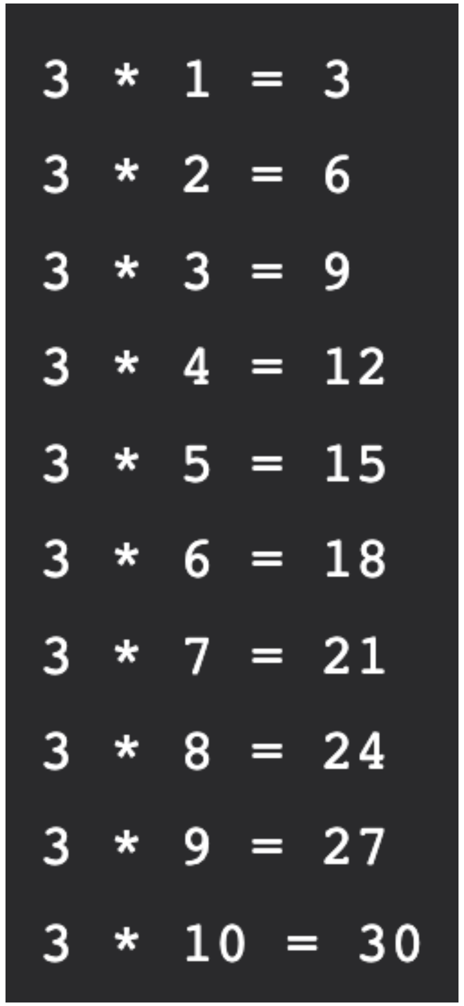
```
for i in range(2, 4):
    for j in range(1, 11):
        print(i, "*", j, "=", i*j)
    print()
```

Nested For Loops (cont.)

- Quiz #2



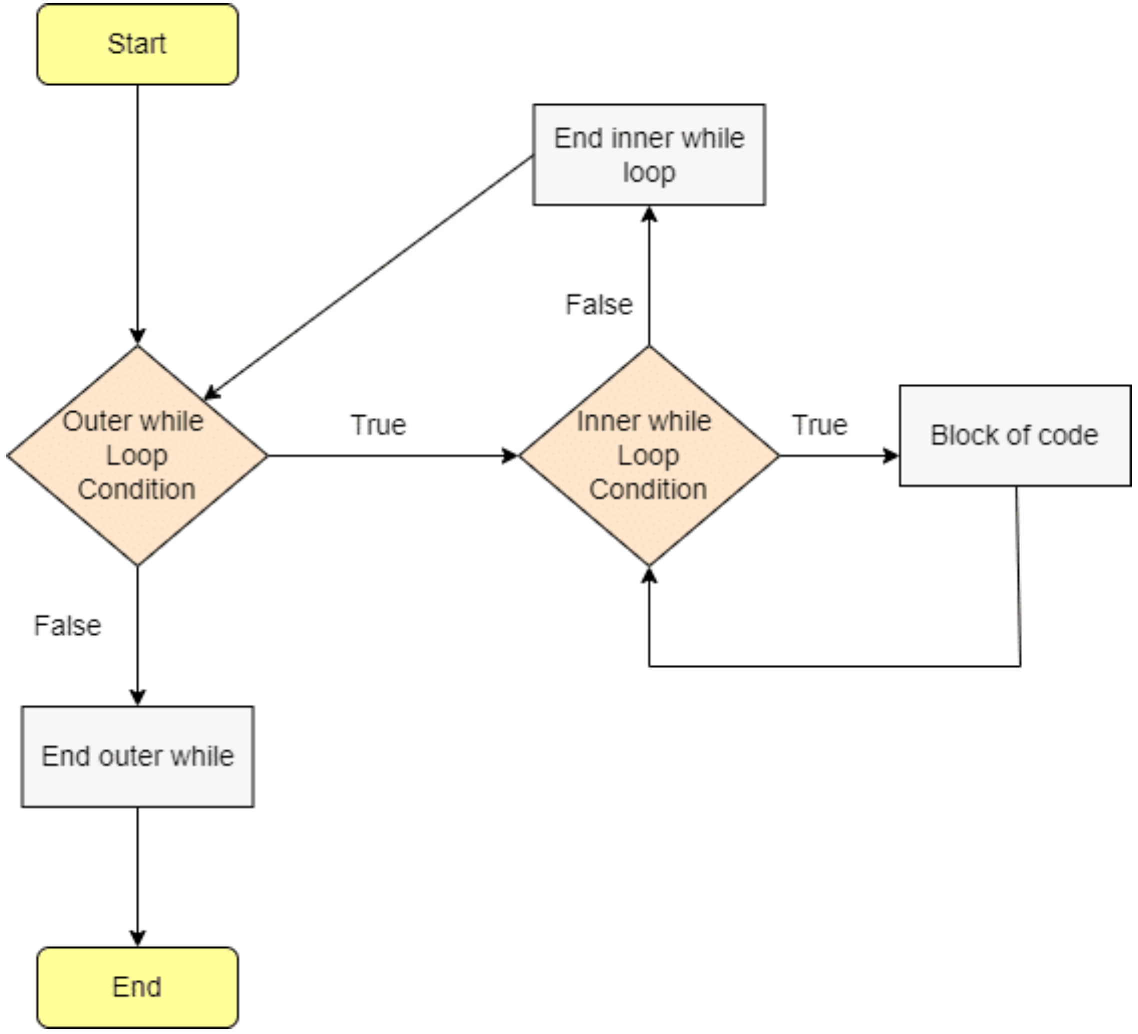
```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```



```
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

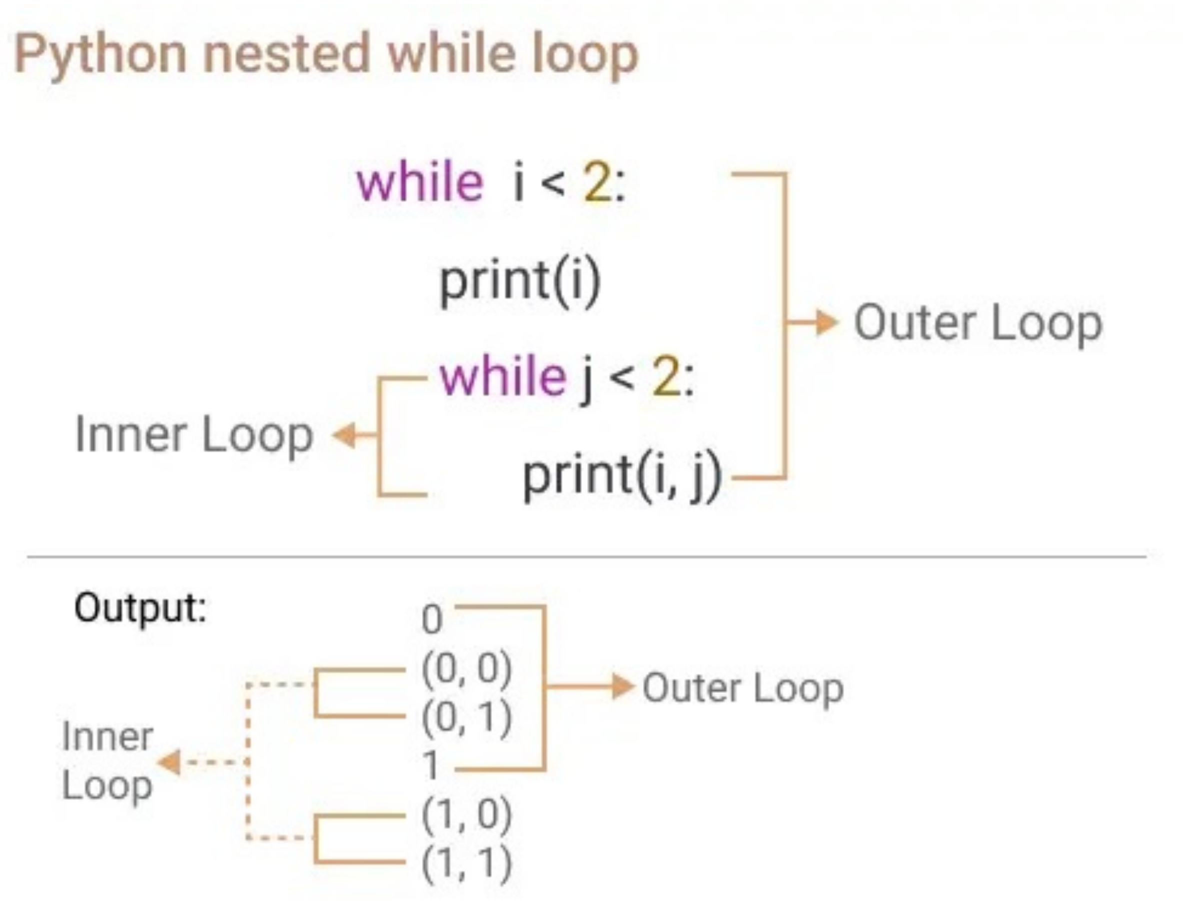
Nested While Loops

```
while (condition 1):
    block of code
    while (condition 2):
        block of code
```



Nested While Loops (cont.)

- Example #1:



Python nested while loop

```
while i < 2:
    print(i)
    while j < 2:
        print(i, j)
```

Output:

```
0
(0, 0)
(0, 1)
1
(1, 0)
(1, 1)
```

Nested While Loops (cont.)

- Quiz #1

```
i = 1
while i <= 3:
    print("Outer Loop: ", i, "time -----")
    j = 1
    while j <= 2:
        print("Inner Loop:", j)
        j += 1
    i += 1
```

Nested While Loops (cont.)

- Quiz #1 (answer):