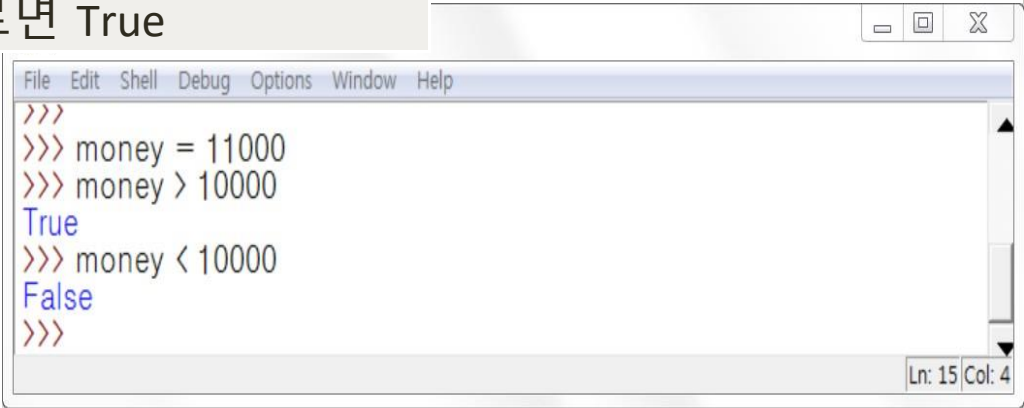


4. 제어문

(조건문과 반복문)

비교 연산자

비교 연산자	의미
<	적으면 True
>	크면 True
<=	같거나 적으면 True
>=	같거나 크면 True
==	같으면 True
!=	다르면 True



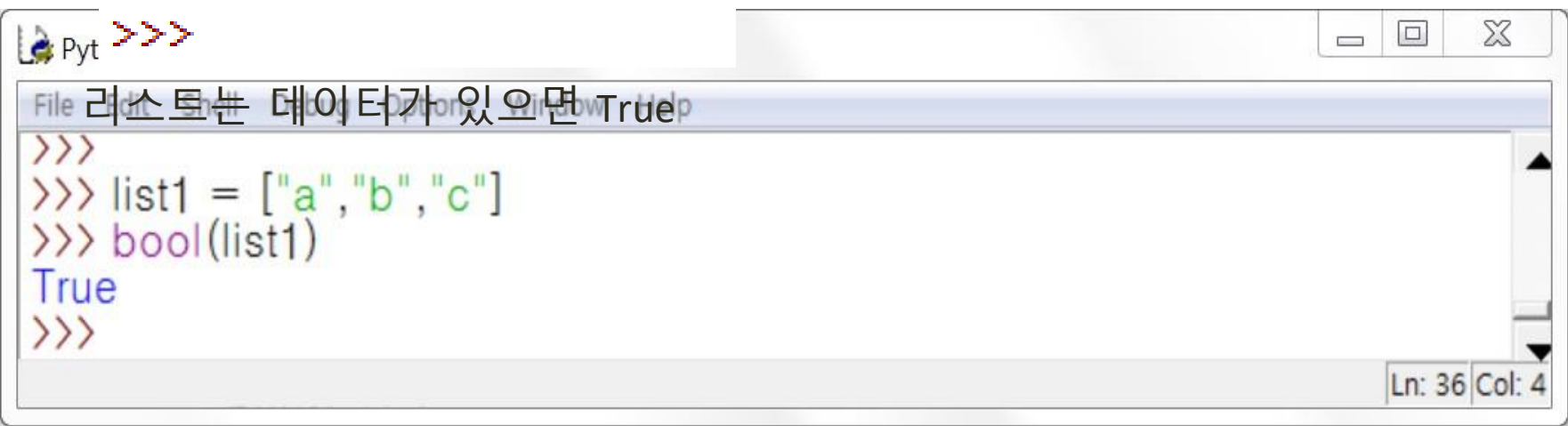
A screenshot of a Python Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The command history shows: >>>, >>> money = 11000, >>> money > 10000, True, >>> money < 10000, False, and >>>. The status bar at the bottom right indicates 'Ln: 15 Col: 4'.

```
>>>
>>> money = 11000
>>> money > 10000
True
>>> money < 10000
False
>>>
```

bool() 함수

bool 함수는 ()안의 결과가 참이면 true 반환

```
>>> money=11000
>>> bool(money>10000)
True
>>> bool( 10 < 11)
True
```



논리 연산자

논리 연산자	의미
and	양쪽 모두 참인 경우만 True
or	양쪽 중 하나라도 참이면 True
not	참이면 false, 거짓면 True

```
>>> money>0 and money <1000
True
>>> money>0 or money <1000
True
>>> not (money>0 or money <1000)
False
```

조건문(if)



리스트에 데이터가 있으므로 True
print(“참”) 수행됨

조건문(if)

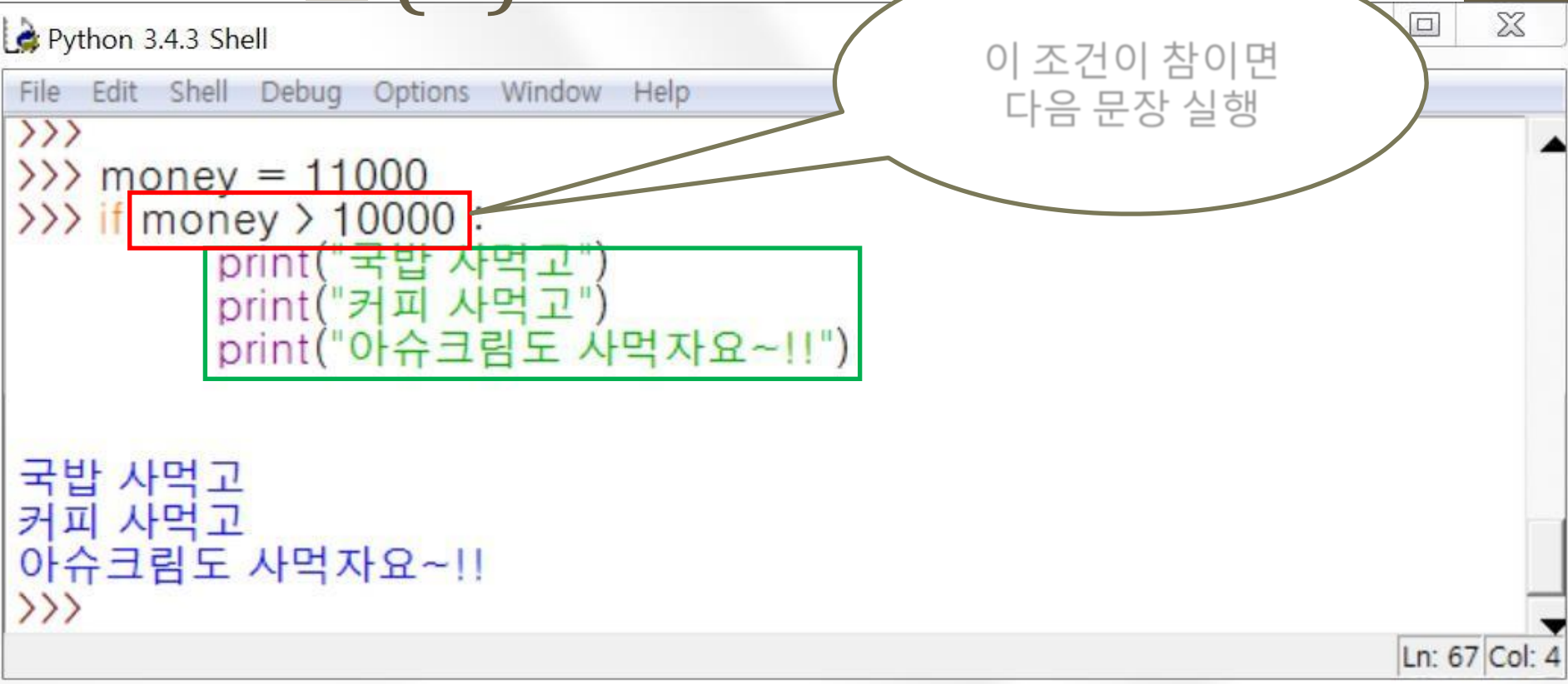
```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>> list2 = [ ]
>>> bool(list2)
False
>>> if list2:
>>>     print("참")
>>>
```

이 조건이 참이면
다음 문장 실행

Ln: 50 Col: 4

리스트에 데이터가 없으므로 False
print("참") 수행 안됨

조건문(if)



A screenshot of a Python 3.4.3 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The code being executed is as follows:

```
>>>  
>>> money = 11000  
>>> if money > 10000:  
    print("국밥 사먹고")  
    print("커피 사먹고")  
    print("아슈크림도 사먹자요~!!")
```

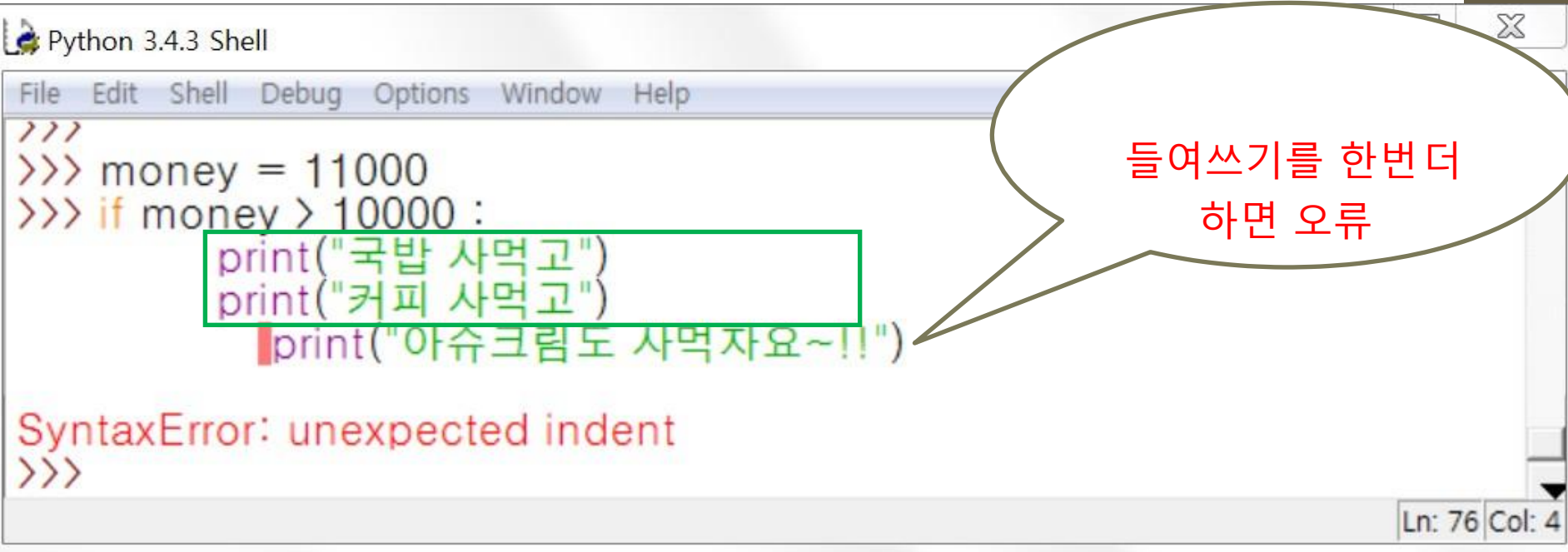
The line `if money > 10000:` is highlighted with a red box. A speech bubble points to this line with the text "이 조건이 참이면 다음 문장 실행" (If this condition is true, execute the next sentence). The output of the code is shown below the code block:

```
국밥 사먹고  
커피 사먹고  
아슈크림도 사먹자요~!!  
>>>
```

The status bar at the bottom right shows 'Ln: 67 Col: 4'.

단, 들여쓰기가 같은 문장들이 한 묶음으로 실행

조건문(if)



```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> money = 11000
>>> if money > 10000 :
    print("국밥 사먹고")
    print("커피 사먹고")
    print("아슈크림도 사먹자요~!!")

SyntaxError: unexpected indent
>>>
```

Ln: 76 Col: 4

들여쓰기를 한번 더
하면 오류

들여쓰기가 같은 문장들이 한 묶음으로 실행.
들여쓰기를 한번 더하면 오류

조건문(if~ else)

거짓이면
다음 문장 실행

이 조건이 참이면
다음 문장 실행

```
Python 3.4.3 Shell
Edit Shell Debug Options Window Help
>>> money = 9000
>>> if money > 10000:
>>>     print("국밥을 먹을 수 있다~!!")
else:
>>>     print("김밥이나 먹어야지..ㅠㅠ")

김밥이나 먹어야지..ㅠㅠ
>>>
```

조건문(if~ elif)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///  
>>> grade = 87  
>>>  
>>> if grade >= 90 and grade <= 100:  
    print("A 등급")  
elif grade >= 80 and grade < 90:  
    print("B 등급")  
elif grade >= 70 and grade < 80:  
    print("C 등급")  
elif grade < 70:  
    print("D 등급")
```

B 등급
>>>

이 조건이 참이면
다음 문장(print("A등급")) 실행하고
if 문장 종료.
이 조건 이 거짓이면
다음 elif 의 조건 비교

이 조건이 참이면
다음 문장(print("B등급"))
실행하고 if 문장 종료.
이 조건 이 거짓이면
다음 elif 의 조건 비교

이 조건이 참이면
다음 문장(print("C등급"))
실행하고 if 문장 종료.
이 조건 이 거짓이면
다음 elif 의 조건 비교

이 조건이 참이면
다음 문장(print("D등급"))
실행하고 if 문장 종료. 거
짓이면 그대로 f 문장 종료

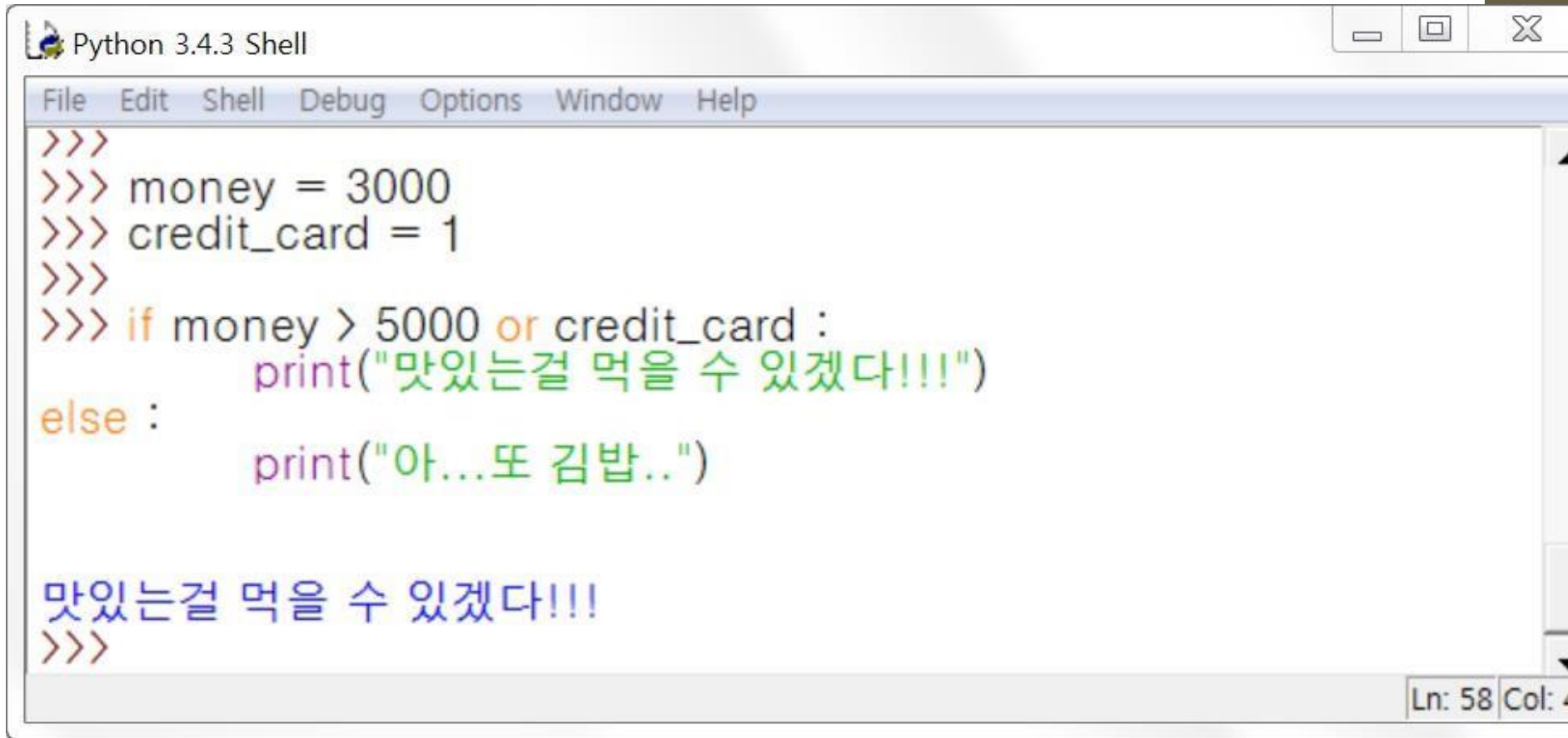
Ln: 28 Col: 4

조건문(if~ else)

```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> money = 6000
>>> card = 1
>>>
>>> if money > 10000 or card :
>>>     print("맛있는거 먹자~~~!")
else :
>>>     print("그냥 김밥이나 먹자....ㅠㅠ")

맛있는거 먹자~~~!
>>>
```

조건문(if~ else)

A screenshot of a Python 3.4.3 Shell window. The window has a title bar with the text 'Python 3.4.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a Python script. The script starts with three lines of assignment: 'money = 3000', 'credit_card = 1', and an empty line. Then it has an 'if' statement: 'if money > 5000 or credit_card :'. The first branch of the 'if' statement is 'print("맛있는걸 먹을 수 있겠다!!!)"'. The 'else' branch is 'print("아...또 김밥..")'. Below the script, the output of the first 'print' statement is shown: '맛있는걸 먹을 수 있겠다!!!'. The prompt '>>>' is visible at the end of the last line of code and at the start of the output line. The status bar at the bottom right shows 'Ln: 58 Col: 4'.

```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> money = 3000
>>> credit_card = 1
>>>
>>> if money > 5000 or credit_card :
>>>     print("맛있는걸 먹을 수 있겠다!!!")
else :
>>>     print("아...또 김밥..")

맛있는걸 먹을 수 있겠다!!!
>>>
```

Ln: 58 Col: 4

조건문(if~ elif)

```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> count = 2
>>> if count == 1 :
>>>     print("count = 하나~")
elif count == 2 :
>>>     print("count = 두울~")
elif count == 3 :
>>>     print("count = 세엳~")

count = 두울~
>>>
```

반복문(for) - 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

>>>
>>> for i in range(1,5) :
 print(i)

1

Ln: 132 Co

1

1. 처음 이 문장을 수행 후
i 저장공간

2. 이 문장 수행

i에 처음 1을 할당하고, 다음 문장(print(i)) 실행 후,
i에 1을 더 한다. 그리고 문장 (print(i)) 을 다시 실행. i가 4가 될 때
까지 반복. i가 5와 같거나 크면 for 문장 종료.

반복문(for) - 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

>>>
>>> for i in range(1,5) :
 print(i)

1
2

Ln: 132 Co

3. 문장을 수행 후 (i 값1 증가)
i 저장공간

4. 문장 반복 수행. (i 출력)

2

i에 처음 1을 할당하고, 다음 문장(print(i)) 실행 후,
i에 1을 더 한다. 그리고 문장 (print(i)) 을 다시 실행. i가 4가 될 때
까지 반복. i가 5와 같거나 크면 for 문장 종료.

반복문(for) - 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

>>>
>>> for i in range(1,5) :
 print(i)

1
2
3

Ln: 132 Co

5. 문장을 수행 후 (i 값1 증가)
i 저장공간

6. 문장 반복 수행. (i 출력)

3

i에 처음 1을 할당하고, 다음 문장(print(i)) 실행 후,
i에 1을 더 한다. 그리고 문장 (print(i)) 을 다시 실행. i가 4가 될 때
까지 반복. i가 5와 같거나 크면 for 문장 종료.

반복문(for) - 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>>
>>> for i in range(1,5) :
    print(i)

1
2
3
4
>>>
```

Ln: 132 Co

7. 문장을 수행 후 (i 값1 증가)
i 저장공간

8. 문장 반복 수행. (i 출력)

4

i에 처음 1을 할당하고, 다음 문장(print(i)) 실행 후,
i에 1을 더 한다. 그리고 문장 (print(i)) 을 다시 실행. i가 4가 될 때
까지 반복. i가 5와 같거나 크면 for 문장 종료.

반복문(for) - 예제2

```
>>> sum = 0
>>> for x in range(1,6) :
    sum += x
```

1. 이 문장 수행 후 **sum** 0

[day04]

x에 처음 1을 할당하고, 다음 문장($\text{sum} += x$) 실행 후,
x에 1을 더 한다. 그리고 문장($\text{sum} += x$) 을 다시 실행.
x가 5가 될 때 까지 반복.

x가 6과 같거나 크면 for 문장 종료.

반복문(for) - 예제2

```
>>> sum = 0
>>> for x in range(1, 6) :
    sum += x
```

2. 이 문장을 수행 후 x	1
3. 이 문장 수행 후 sum (sum = sum + x)	1

[day04]

x에 처음 1을 할당하고, 다음 문장($\text{sum} += x$) 실행 후,
x에 1을 더 한다. 그리고 문장($\text{sum} += x$) 을 다시 실행.
x가 5가 될 때 까지 반복.

x가 6과 같거나 크면 for 문장 종료.

반복문(for) - 예제2

```
>>> sum = 0
>>> for x in range(1, 6) :
    sum += x
```

4. 이 문장을 수행 후 **x** 2

5. 이 문장 수행 후 **sum**
(sum = sum + x) 3

[day04]

x에 처음 1을 할당하고, 다음 문장($\text{sum} += x$) 실행 후,
x에 1을 더 한다. 그리고 문장($\text{sum} += x$) 을 다시 실행.
x가 5가 될 때 까지 반복.

x가 6과 같거나 크면 for 문장 종료.

반복문(for) - 예제2

```
>>> sum = 0
>>> for x in range(1, 6) :
    sum += x
```

6. 이 문장을 수행 후 x	3
7. 이 문장 수행 후 sum (sum = sum + x)	6

[day04]

x에 처음 1을 할당하고, 다음 문장($\text{sum} += x$) 실행 후,
x에 1을 더 한다. 그리고 문장($\text{sum} += x$) 을 다시 실행.
x가 5가 될 때 까지 반복.

x가 6과 같거나 크면 for 문장 종료.

반복문(for) - 예제2

```
>>> sum = 0
>>> for x in range(1, 6) :
    sum += x
```

8. 이 문장을 수행 후 **x** 4

9. 이 문장 수행 후 **sum**
(sum = sum + x) 10

[day04]

x에 처음 1을 할당하고, 다음 문장($\text{sum} += x$) 실행 후,
x에 1을 더 한다. 그리고 문장($\text{sum} += x$) 을 다시 실행.
x가 5가 될 때 까지 반복.

x가 6과 같거나 크면 for 문장 종료.

반복문(for) - 예제2

```
>>> sum = 0
>>> for x in range(1, 6) :
    sum += x
```

10 이 문장을 수행 후 x	5
11 이 문장 수행 후 sum (sum = sum + x)	15

[day04]

x에 처음 1을 할당하고, 다음 문장($\text{sum} += x$) 실행 후,
x에 1을 더 한다. 그리고 문장($\text{sum} += x$) 을 다시 실행.
x가 5가 될 때 까지 반복.

x가 6과 같거나 크면 for 문장 종료.

반복문(for) - 예제2

12 이 문장을 수행 후 **x** 6

sum 15

```
>>> sum = 0
>>> for x in range(1,6) :
    sum += x
```

```
>>> print(sum)
15
```

13.

[day04]

x에 처음 1을 할당하고, 다음 문장($\text{sum} += x$) 실행 후,
x에 1을 더 한다. 그리고 문장($\text{sum} += x$) 을 다시 실행.
x가 5가 될 때 까지 반복.

x가 6과 같거나 크면 for 문장 종료.

반복문(for) - 예제3

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> sum = 0
>>> for x in range(1,1001) :
    sum += x

>>> print(sum)
500500
>>>
```

Ln: 195 Col: 4

x에 처음 1을 할당하고, 다음 문장(sum+=x) 실행 후, x에 1을 더 한다.
그리고 문장(sum+=x)을 다시 실행. x가 1000가 될 때 까지 반복.
x가 10001과 같거나 크면 for 문장 종료.

반복문(for) - 예제4

```
>>> for x in range (1, 5, 2) :  
      print (x)
```

2씩 증가

1
3

[day04]

x에 처음 1을 할당하고, 다음 문장(print(x)) 실행 후, x에 2을 더 한다.
그리고 문장 (print(x)) 을 다시 실행. x가 5와 같거나 크면 for 문장
종료.

반복문(for) - 예제 5 (리스트)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

>>>
>>> list1 = ["원", "투", "차차차"]
>>> for i in list1 :
 print(i)

원

1. 이 문장을 수행 후 i

2.

원

Ln: 99 Col: 4

반복문(for) - 예제 5 (리스트)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>>  
>>> list1 = ["원", "투", "차차차"]  
>>> for i in list1 :  
    print(i)
```

Ln: 99 Col: 4

3. 이 문장을 수행 후 i

4.

투

반복문(for) - 예제 5 (리스트)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>>  
>>> list1 = ["원", "투", "차차차"]  
>>> for i in list1 :  
    print(i)
```

원
투
차차차

5. 이 문장을 수행 후 i
6.

차차차

Ln: 99 Col: 4

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

1. 이 문장을 수행 후 **i**

2.

안

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

3. 이 문장을 수행 후 **i**

4.

안녕

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

5. 이 문장을 수행 후 **i**

6.

하

안녕하시렵니까

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

7. 이 문장을 수행 후 **i**

8.

시

안녕하시

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

9. 이 문장을 수행 후 **i**

10.

안녕하시렵

렵

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

11. 이 문장을 수행 후 i

니

12.

안녕하시렵니

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

13. 이 문장을 수행 후 i

14.

까

안녕하시렵니까

반복문(for) - 예제 6 (문자열)

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> str1 = "안녕하시렵니까?"
>>> for i in str1 :
    print(i)
```

Ln: 116 Col: 4

15. 이 문장을 수행 후 i ?

16.

안녕하시렵니까?

반복문(for) - 예제 7

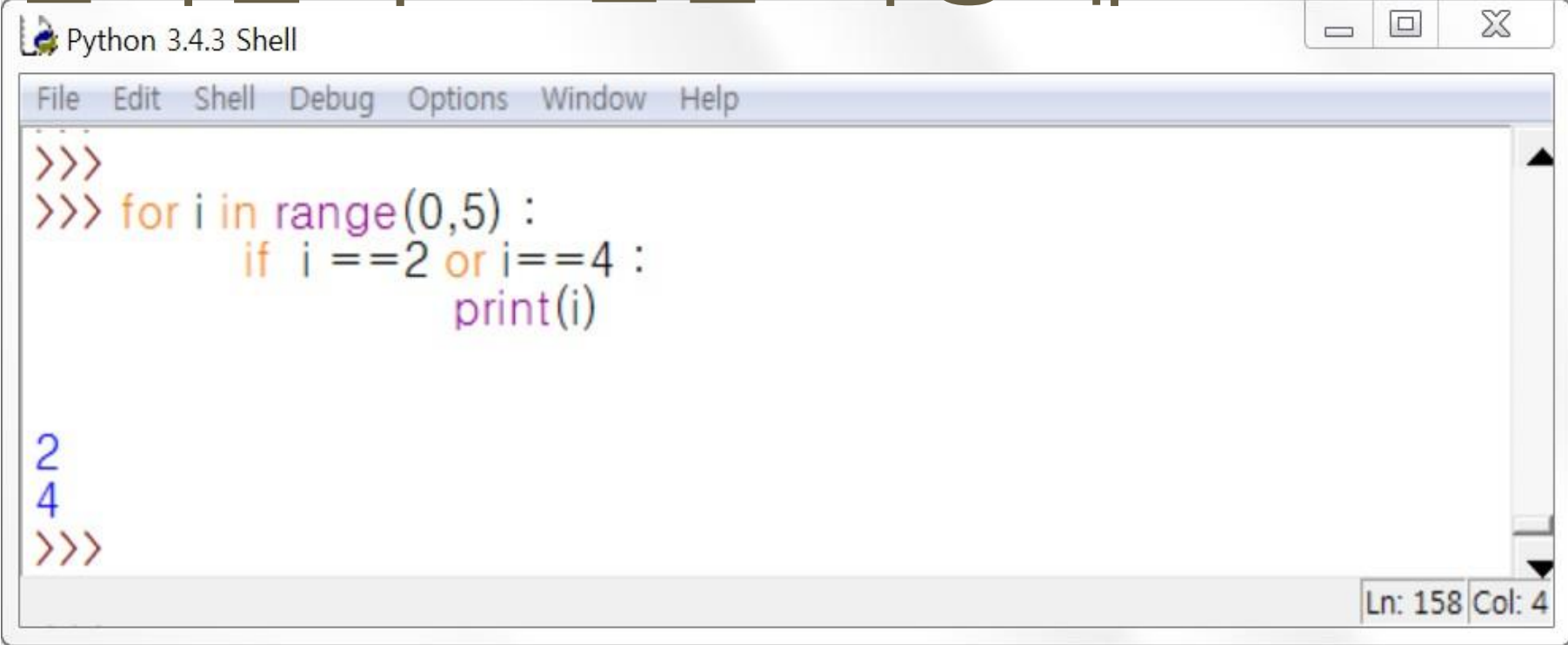
Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> for x in (1,2,3,4,5) :  
        print(x)  
  
1  
2  
3  
4  
5  
>>>
```

Ln: 134 Col: 4

반복문과 조건문 사용예



A screenshot of a Python 3.4.3 Shell window. The window has a title bar with the text 'Python 3.4.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains the following Python code:

```
>>>  
>>> for i in range(0,5):  
    if i == 2 or i == 4:  
        print(i)  
  
2  
4  
>>>
```

The output of the code is shown as two lines: '2' and '4'. The status bar at the bottom right of the window displays 'Ln: 158 Col: 4'.

반복문과 조건문 사용예

```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>> messages = '''
안녕하세요 %s 님 방문해 주셔서 감사합니다.
행복한 여행 되세요~ '''
>>> name = ['서진수','김희연','서채원','서주원']
>>> for x in name :
    print(messages % x)
    print('*' * 40)

안녕하세요 서진수 님 방문해 주셔서 감사합니다.
행복한 여행 되세요~
*****

안녕하세요 김희연 님 방문해 주셔서 감사합니다.
행복한 여행 되세요~
*****

안녕하세요 서채원 님 방문해 주셔서 감사합니다.
```


반복문과 조건문 사용예

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> for x in range(0,5) :  
    print('%s 번 학생 손드세요~' % x)
```

```
0 번 학생 손드세요~  
1 번 학생 손드세요~  
2 번 학생 손드세요~  
3 번 학생 손드세요~  
4 번 학생 손드세요~  
>>>
```

Ln: 145 Col: 0

반복문(while) 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///
>>> a = 0
>>> while a < 5:
>>>     print(a)
>>>     a += 1 # a = a + 1 과 같은 의미입니다
>>>
```

Ln: 170 Col: 4

이 조건이 참인 동안
다음 문장을 반복 수행

1. 이 문장을 수행 후 a 0

반복문(while) 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///  
>>> a = 0  
>>> while a < 5:  
    print(a)  
    a += 1
```

a = a + 1 과 같은 의미입니다

0

Ln: 170 Col: 4

2. True

4. 이 문장을 수행 후 a

1

반복문(while) 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///  
>>> a = 0  
>>> while a < 5:  
    print(a)  
    a += 1
```

a = a + 1 과 같은 의미입니다

0
1

5. True

7. 이 문장을 수행 후 a

2

Ln: 170 Col: 4

반복문(while) 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///  
>>> a = 0  
>>> while a < 5:  
    print(a)  
    a += 1
```

a = a + 1 과 같은 의미입니다

0
1
2

Ln: 170 Col: 4

8. True

10. 이 문장을 수행 후 a

3

반복문(while) 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///
>>> a = 0
>>> while a < 5:
>>>     print(a)
>>>     a += 1
```

a = a + 1 과 같은 의미입니다

0
1
2
3

11. True

13. 이 문장을 수행 후 a

4

Ln: 170 Col: 4

반복문(while) 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///  
>>> a = 0  
>>> while a < 5:  
    print(a)  
    a += 1
```

a = a + 1 과 같은 의미입니다

0
1
2
3
4

Ln: 170 Col: 4

14. True

16. 이 문장을 수행 후 a

5

반복문(while) 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///  
>>> a = 0  
>>> while a < 5:  
    print(a)  
    a += 1 # a = a + 1 과 같은 의미입니다
```

17. false

0
1
2
3
4

Ln: 170 Col: 4

반복문(while) - 예제2

```
sum=0
x=1
while x < 5 :
    sum+=x
    x+=1
```

이 조건이 참
인 동안
다음 문장을
반복 수행

- 1. 이 문장을 수행 후 sum 0
- 2. 이 문장을 수행 후 x 1

반복문(while) - 예제2

```
sum=0
x=1
while x < 5 :
```

```
    sum+=x
    x+=1
```

이 조건이 참
인 동안
다음 문장을
반복 수행

3. True

4. 이 문장을 수행 후 sum

1

5. 이 문장을 수행 후 x

2

반복문(while) - 예제2

```
sum=0
```

```
x=1
```

```
while x < 5 :
```

```
    sum+=x
```

```
    x+=1
```

이 조건이 참
인 동안
다음 문장을
반복 수행

6. True

7. 이 문장을 수행 후 sum

3

8. 이 문장을 수행 후 x

3

반복문(while) - 예제2

```
sum=0
```

```
x=1
```

```
while x < 5 :
```

```
    sum+=x
```

```
    x+=1
```

이 조건이 참
인 동안
다음 문장을
반복 수행

9. True

10 이 문장을 수행 후 sum

6

11 이 문장을 수행 후 x

4

반복문(while) - 예제2

```
sum=0
```

```
x=1
```

```
while x < 5 :
```

```
    sum+=x
```

```
    x+=1
```

이 조건이 참
인 동안
다음 문장을
반복 수행

12. True

13 이 문장을 수행 후 sum

10

14 이 문장을 수행 후 x

5

반복문(while) - 예제2

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

15. false

sum	10
x	5

```
>>>
10
>>> |
```

반복문(while) - 예제3

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> sum = 0
>>> x = 1
>>> while x < 1001:
    sum += x
    x += 1

>>> print(sum)
500500
>>>
```

Ln: 212 Col: 4

이 조건이 참인
동안
다음 문장을 반
복 수행

break 문 - 예제1

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
///
>>> a = ["원", "투", "차차차", "쓰리", "포", "차차차"]
>>>
>>> for i in a:
    if i == "쓰리":
        break
    print(i)
```

원

투

차차차

>>>

이 조건이 참일때

다음 문장(break)을 수행

break가 수행되면

반복 문장을 끝냄

Ln: 194 Col: 4

break 문 - 예제2

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
>>> a = 10
>>> while a > 0 :
    print(a)
    a -= 1
    if a == 6:
        break
```

10
9
8
7
>>>

이 조건이 참일때
다음 문장(break)을 수행

break가 수행되면
반복 문장을 끝냄

Ln: 62 Col: 4

continue 문 - 예제 1

```
for i in range (0,4) :  
    if (i%2)==0 :  
        continue  
    print (i)
```

이 조건이 참일때
다음 문장(continue)을 수행
continue 가 수행되면
for 문장으로

continue 문 - 예제 1

```
for i in range (0,4) :  
    if (i%2)==0 :  
        continue  
    print(i)
```

1. 이 문장을 수행 후 i

2 True

0

continue 문 - 예제 1

4. 이 문장을 수행 후 **i** 1

```
for i in range (0,4) :  
    if (i%2)==0 :  
        continue  
    print(i)
```

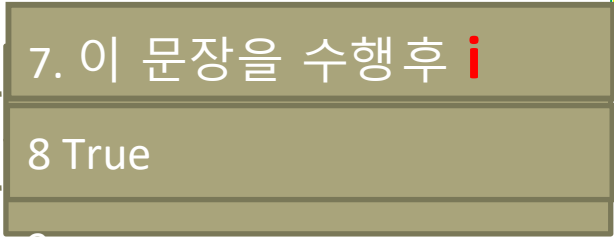
5 False

6

```
>>>  
1
```

continue 문 - 예제 1

```
for i in range (0,4) :  
    if (i%2)==0 :  
        continue  
    print(i)
```



2

```
>>>  
1
```

continue 문 - 예제 1

10. 이 문장을 수행 후 **i** 3

```
for i in range (0,4) :  
    if (i%2)==0 :  
        continue  
    print(i)
```

11 False

12

```
>>>  
1  
3  
>>>
```

continue 문 - 예제 1

13. 이 문장을 수행 후 **i** 4

```
for i in range (0,4) :  
    if (i%2)==0 :  
        continue  
    print(i)
```

```
>>>  
1  
3  
>>>
```

continue 문 - 예제2

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

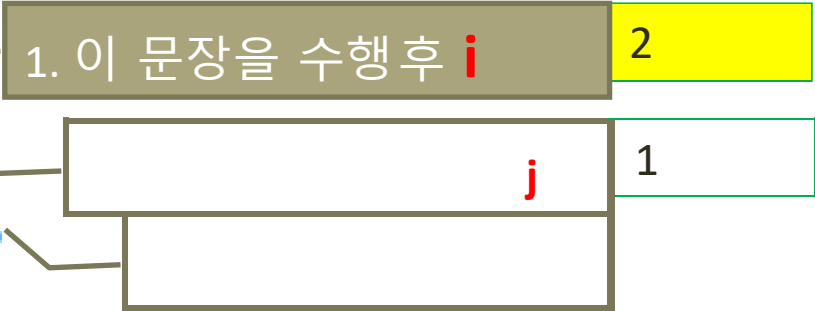
```
>>> foods = ["치킨", "피자", "깐풍기", "버섯", "돈까스"]
>>>
>>> for i in foods :
        if i == "버섯" :
            continue
        print(i)
```

치킨
피자
깐풍기
돈까스
>>>

Ln: 254 Col: 4

반복문의 중첩 - 예제1

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



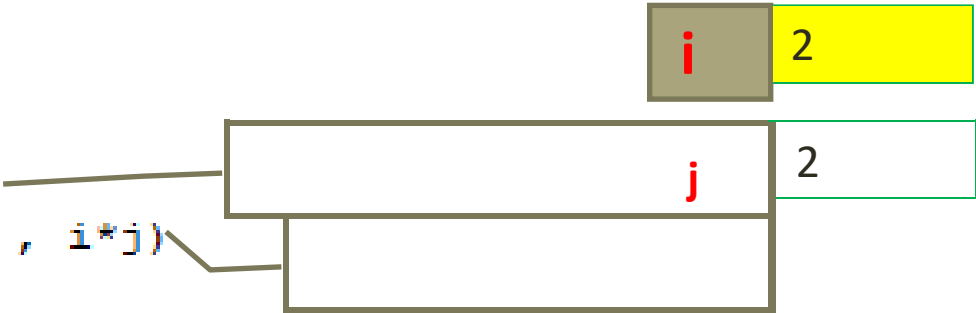
[day04]

2 X 1 = 2

반복문의 중첩 - 예제1

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

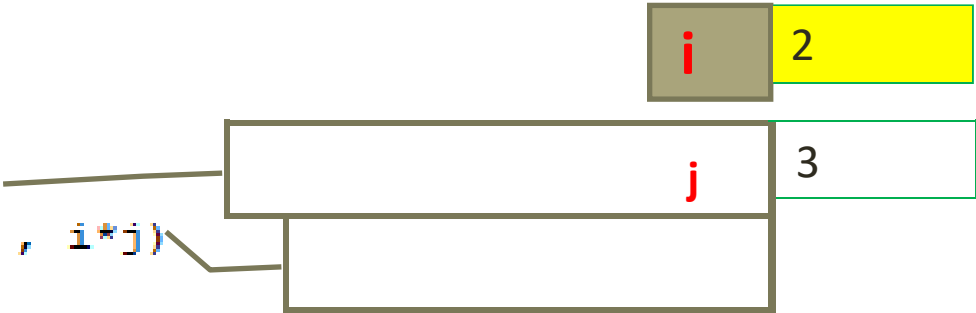
2 X 1 = 2
2 X 2 = 4



반복문의 중첩 - 예제1

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

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6



반복문의 중첩 - 예제1

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

i	2
---	---

8. 이 문장을 수행후 j	4
----------------	---

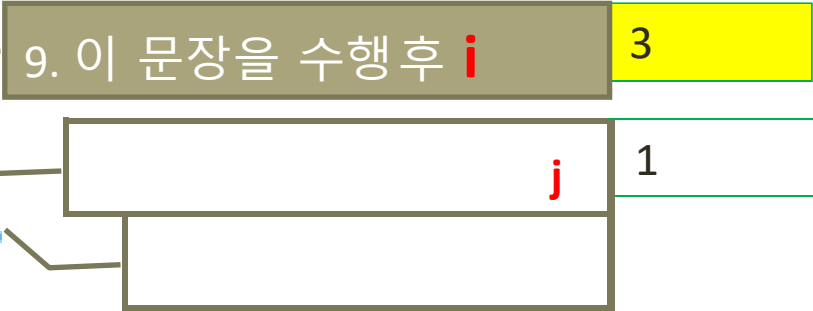
[day04]

$$\begin{array}{l} 2 \times 1 = 2 \\ 2 \times 2 = 4 \\ 2 \times 3 = 6 \end{array}$$

반복문의 중첩 - 예제1

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

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
3 X 1 = 3



반복문의 중첩 - 예제1

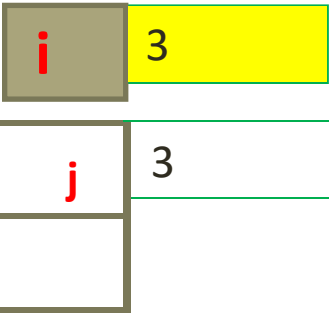
```
for i in range(2,4):  
    for j in range(1,4) :  
        print(i,"X" , j , "=" , i*j)
```

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
3 X 1 = 3
3 X 2 = 6



반복문의 중첩 - 예제1

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



2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9

반복문의 중첩 - 예제1

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

i

3

16. 이 문장을 수행후 j

4

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9

반복문의 중첩 - 예제1

17. 이 문장을 수행 후 **i**

4

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

```
2 X 1 = 2  
2 X 2 = 4  
2 X 3 = 6  
3 X 1 = 3  
3 X 2 = 6  
3 X 3 = 9
```

반복문의 중첩 - 예제2

```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>> i = 2
>>> while i < 5 :
>>>     j = 1
>>>     while j < 10 :
>>>         print(i,"x",j,"=",i*j)
>>>         j += 1
>>>     i += 1

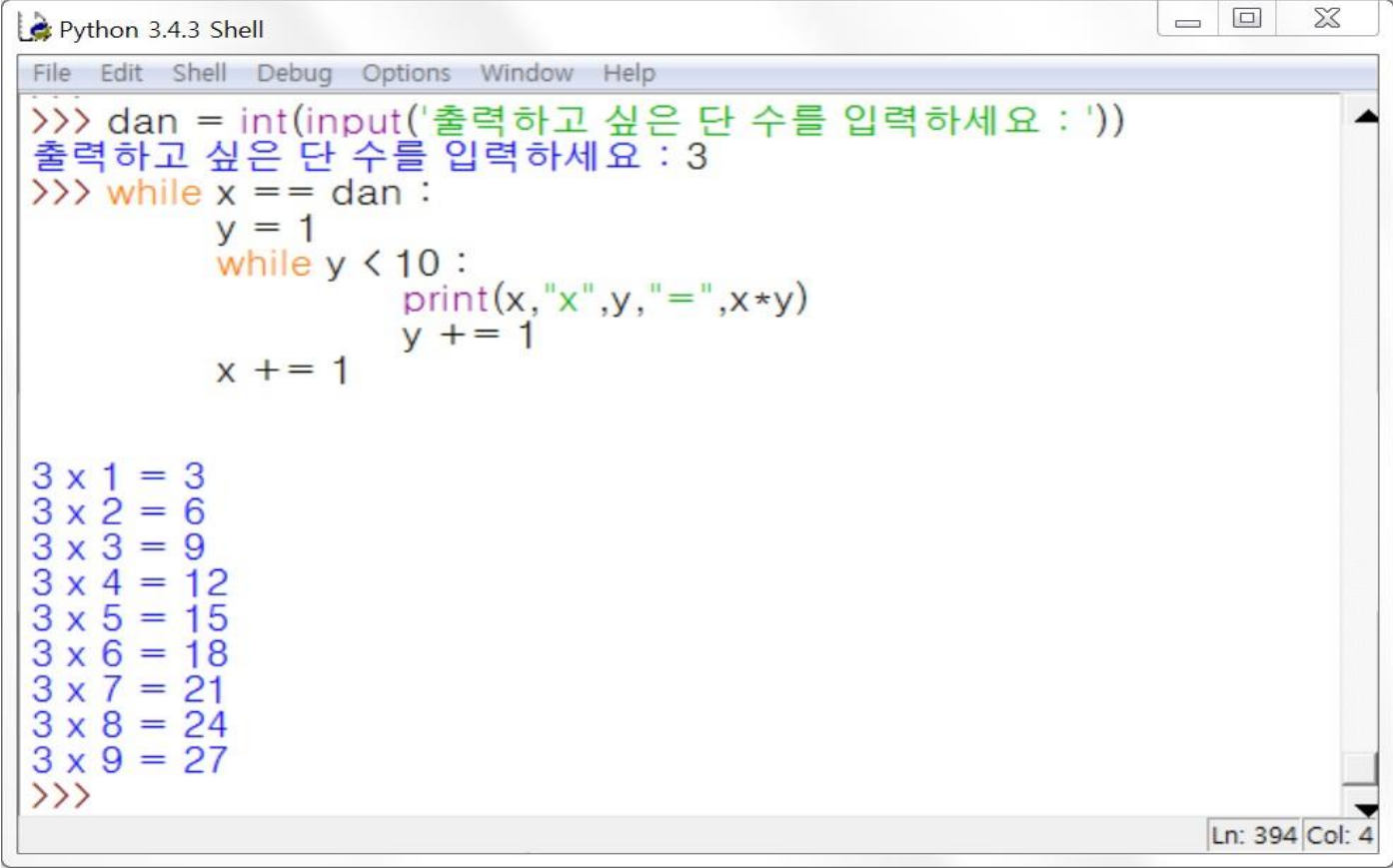
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
>>>
```

반복문의 중첩- 예제3

```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> dan = int(input('출력하고 싶은 단 수를 입력하세요 : '))
출력하고 싶은 단 수를 입력하세요 : 3
>>> for x in range(dan , dan+1) :
        for y in range(1,10) :
            print(x,"x",y,"=",x*y)
        print

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
<built-in function print>
```

반복문의 중첩- 예제4

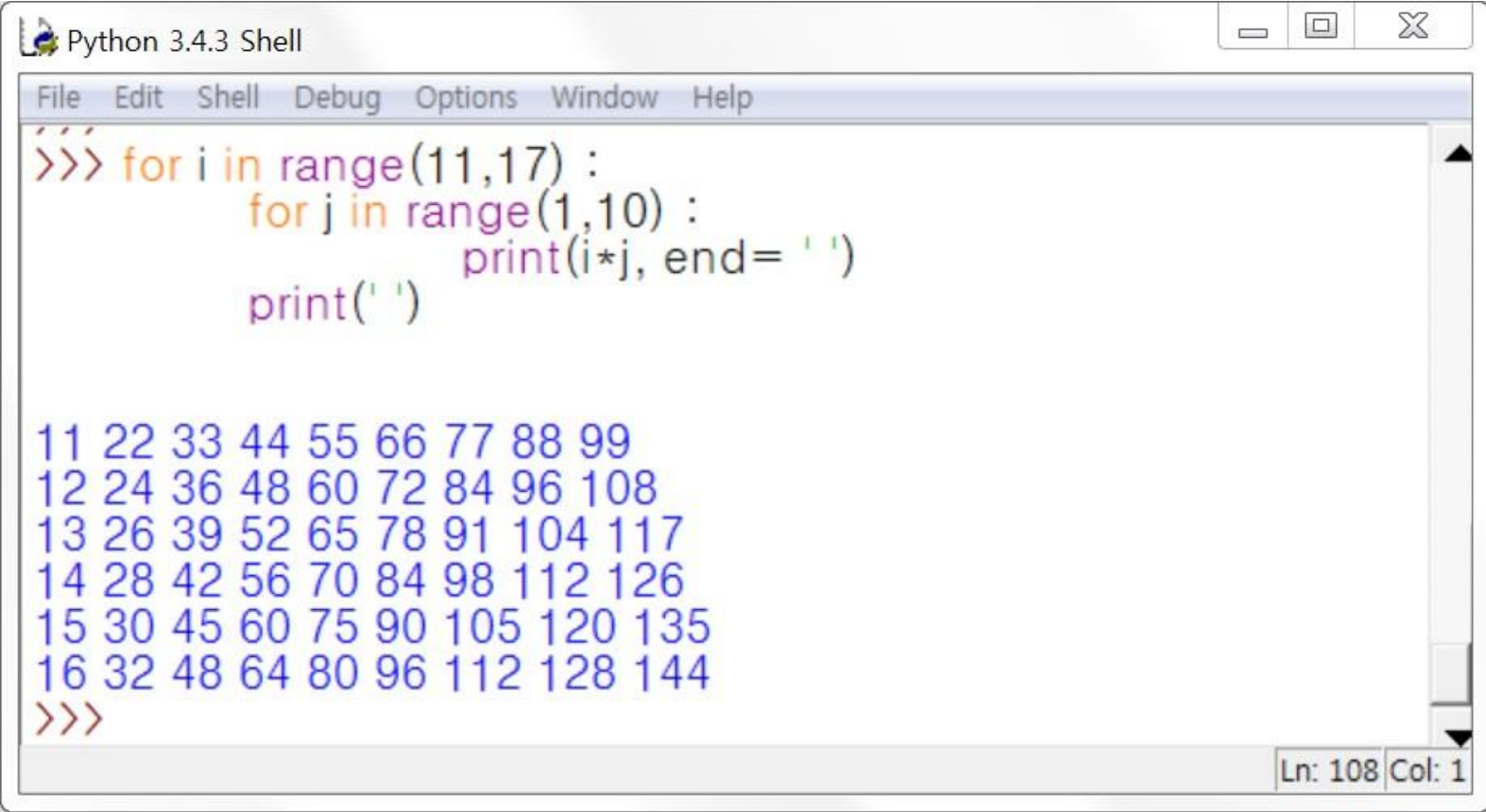


```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
>>> dan = int(input('출력하고 싶은 단 수를 입력하세요 : '))
출력하고 싶은 단 수를 입력하세요 : 3
>>> while x == dan :
    y = 1
    while y < 10 :
        print(x,"x",y,"=",x*y)
        y += 1
    x += 1

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
>>>
```

Ln: 394 Col: 4

반복문의 중첩- 예제5

A screenshot of a Python 3.4.3 Shell window. The window has a title bar with the text 'Python 3.4.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a Python script with a nested loop. The first loop is 'for i in range(11,17):' and the second loop is 'for j in range(1,10):'. Inside the second loop, there is a 'print(i*j, end= ' ')" statement. After the loops, there is a 'print(' ')" statement. The output of the script is a 6x10 grid of numbers, where each row corresponds to a value of i and each column to a value of j. The numbers are: Row 1: 11 22 33 44 55 66 77 88 99; Row 2: 12 24 36 48 60 72 84 96 108; Row 3: 13 26 39 52 65 78 91 104 117; Row 4: 14 28 42 56 70 84 98 112 126; Row 5: 15 30 45 60 75 90 105 120 135; Row 6: 16 32 48 64 80 96 112 128 144. The prompt '>>>>' is visible at the start of the code and at the end of the output. The status bar at the bottom right shows 'Ln: 108 Col: 1'.