

Loop 1

국민대 소프트웨어학부

수업목표

- What is a loop
- Counting loops, for loops
- Using a counting loop
- Loop variable names
- Counting by steps
- Counting without numbers
- while loops
- Bailing out of a loop – break and continue

What is a loop

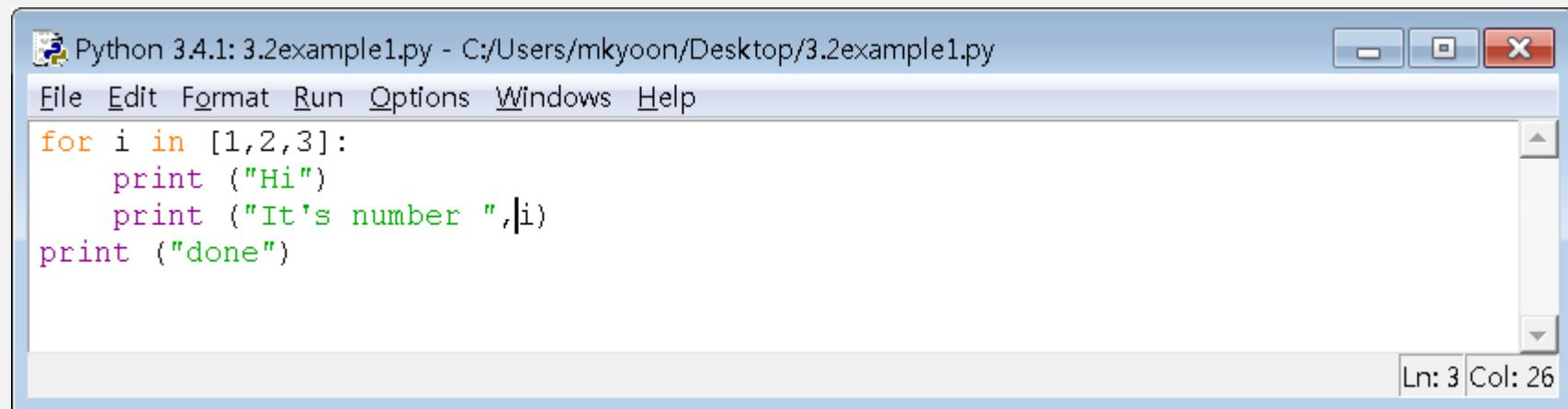
- 컴퓨터가 가장 잘 하는 일
 - Loop: 반복 작업 (계산)
- Counting loops
 - 특정 횟수만큼 반복
 - 100회 반복
- Conditional loops
 - 특정한 조건이 만족될 때까지 반복
 - 정답을 맞추는 경우까지 반복

Counting loops, for loops

- for loops

for 타깃 in 시퀀스

실행문



The screenshot shows a Python 3.4.1 IDE window with the title "Python 3.4.1: 3.2example1.py - C:/Users/mkyoon/Desktop/3.2example1.py". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code editor contains the following Python script:

```
for i in [1,2,3]:  
    print ("Hi")  
    print ("It's number ",i)  
print ("done")
```

The status bar at the bottom right indicates "Ln: 3 Col: 26".

Using a counting loop

- for loops
 - range() 함수
 - range(num1)
 - [0, 1, 2, ..., num1-1]
 - 실제 리스트를 생성하지는 않음. 보충자료에서 자세히 설명.
 - num1번 반복
 - range(num1, num2)
 - [num1, num1+1, num1+2, ..., num2-num1-1]
 - (num2-num1)번 반복
 - range(num1, num2, num3)
 - num1 < num2, 0 < num3
 - [num1, num1+num3, num1+2*num3, ..., num1+k*num3]
 - (num1+k*num3 < num2)까지 반복
 - num1 > num2, 0 > num3
 - [num1, num1+num3, num1+2*num3, ..., num1+k*num3]
 - (num1+k*num3 > num2)까지 반복

Using a counting loop

- for loops
 - range() 함수
 - 파이썬 2에서는 리스트 직접 생성
 - 큰 리스트가 생성되면 비효율적이게 됨
 - 파이썬 3에서는 range 객체 생성
 - 데이터가 요구되는 경우에 하나씩 넘겨줌
 - » 게으른 계산 추가 (lazy evaluation)
 - 메모리를 효율적으로 사용할 수 있기 됨
 - range()를 리스트로 변환하려면 명시적 형변환 사용
 - » list(range(10))
 - 객체에 대한 설명은 강의 후반부에 다름

Using a counting loop

- for loops
 - range() 함수

The screenshot shows a Python 3.5.2 Shell window with the title "Python 3.5.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell window displays the following code and its output:

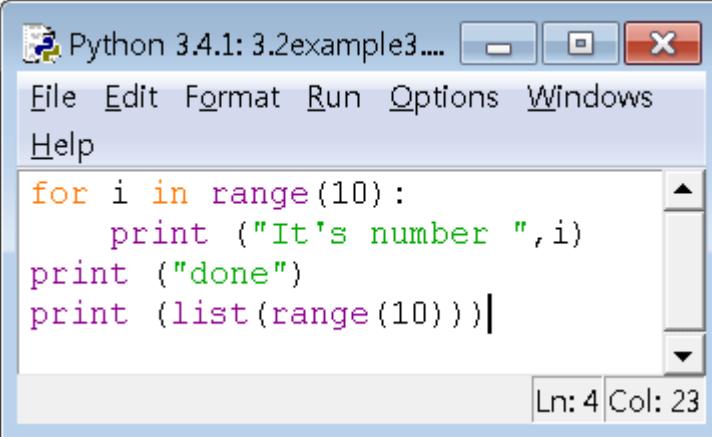
```
Python 3.5.2 |Anaconda 4.2.0 (32-bit)| (default, Jul 5 2016, 11:45:57) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> import sys
>>> a = [1,2]
>>> b = [1,2,3,4,5,6,7,8,9,10]
>>> sys.getsizeof(a)
44
>>> sys.getsizeof(b)
76
>>> sys.getsizeof(range(2))
24
>>> sys.getsizeof(range(10000))
24
>>> range(10)
range(0, 10)
>>> type(range(10))
<class 'range'>
>>> type(a
SyntaxError: invalid syntax
>>> type(a)
<class 'list'>
>>> |
```

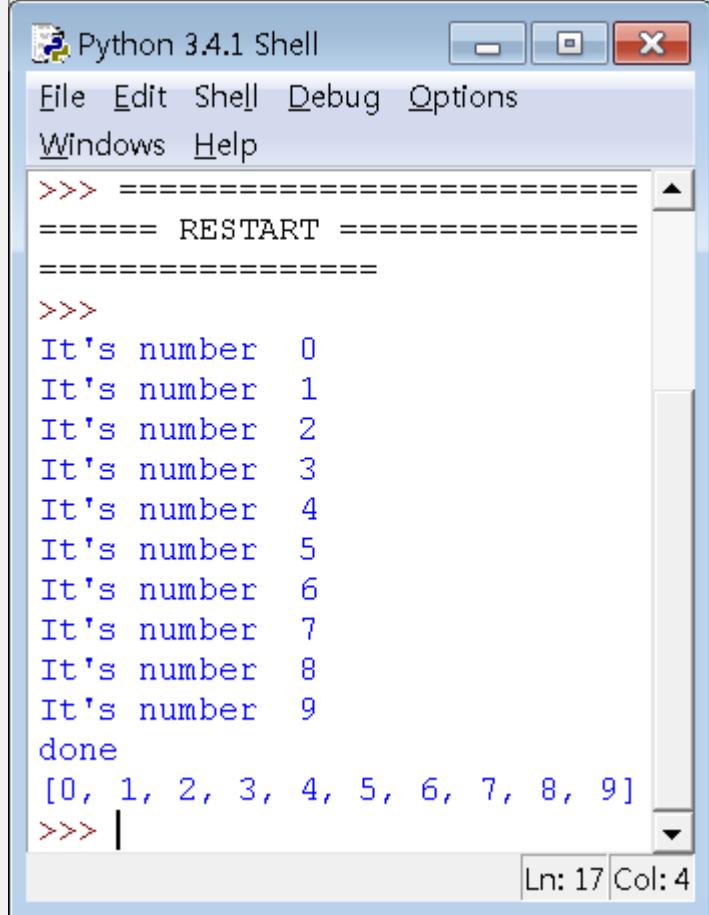
The output shows the memory size of list 'a' (44 bytes), list 'b' (76 bytes), and the range object (24 bytes). It also demonstrates that attempting to print a list without closing parentheses results in a SyntaxError. The status bar at the bottom right indicates "Ln: 22 Col: 4".

Using a counting loop

- for loops
 - range(num1)
 - [0, 1, 2, ..., num1-1]
 - num1 번 반복



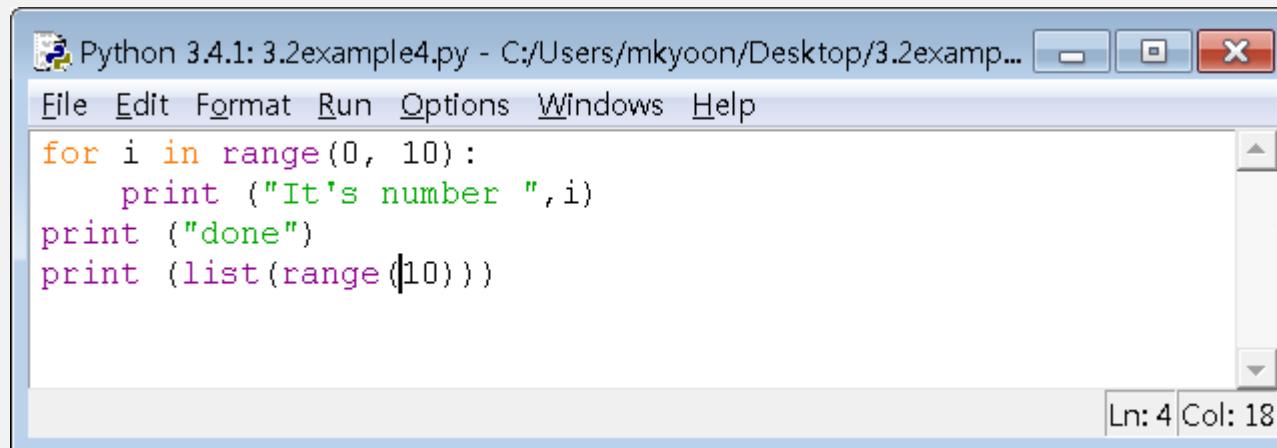
```
Python 3.4.1: 3.2example3.py
File Edit Format Run Options Windows Help
for i in range(10):
    print ("It's number ",i)
print ("done")
print (list(range(10)))|
```



```
Python 3.4.1 Shell
File Edit Shell Debug Options
Windows Help
>>> =====
===== RESTART =====
=====
>>>
It's number 0
It's number 1
It's number 2
It's number 3
It's number 4
It's number 5
It's number 6
It's number 7
It's number 8
It's number 9
done
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> |
Ln: 17 Col: 4
```

Using a counting loop

- for loops
 - range(num1, num2)
 - [num1, num1+1, num1+2, ..., num2-num1]
 - (num2-num1)번 반복



The screenshot shows a Python 3.4.1 IDLE window titled "Python 3.4.1: 3.2example4.py - C:/Users/mkyoon/Desktop/3.2example4.py". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code editor contains the following Python code:

```
for i in range(0, 10):
    print ("It's number ",i)
print ("done")
print (list(range(10)))
```

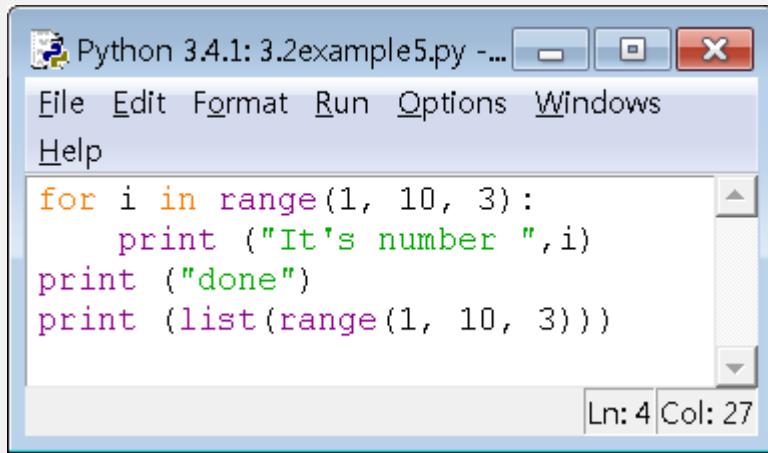
The status bar at the bottom right indicates "Ln: 4 Col: 18".

Loop variable names

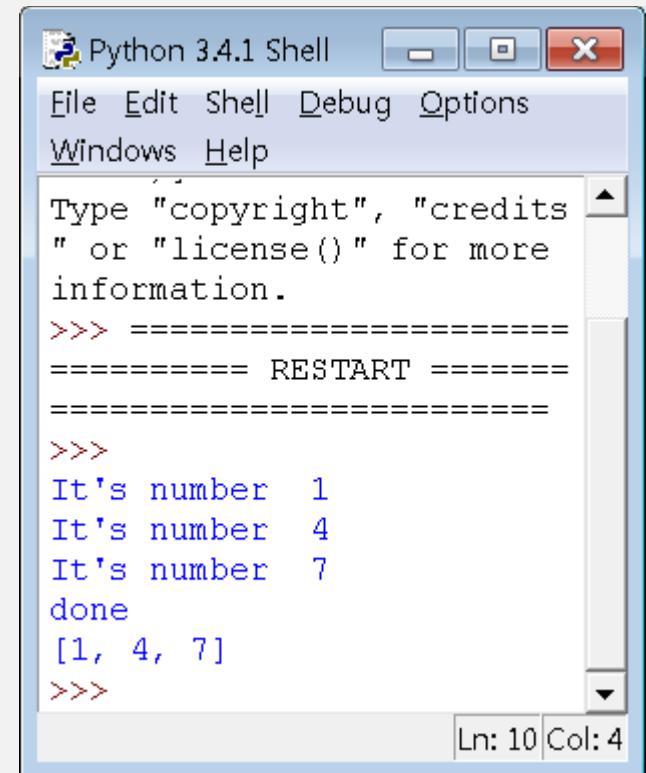
- i, j, k 많이 사용

Counting by steps

- for loops
 - range(num1, num2, num3)
 - num1 < num2, 0 < num3
 - [num1, num1+num3, num1+2*num3, ..., num1+k*num3]
 - (num1+k*num3 < num2)까지 반복



```
for i in range(1, 10, 3):
    print ("It's number ",i)
print ("done")
print (list(range(1, 10, 3)))
```



```
Python 3.4.1 Shell
File Edit Shell Debug Options
Windows Help
Type "copyright", "credits"
" or "license()" for more
information.

>>> =====
===== RESTART =====
=====
>>>
It's number  1
It's number  4
It's number  7
done
[1, 4, 7]
>>>
```

Counting by steps

- for loops
 - range(num1, num2, num3)
 - num1 > num2, 0 > num3
 - [num1, num1+num3, num1+2*num3, ..., num1+k*num3]
 - (num1+k*num3 > num2)까지 반복

```
Python 3.4.1: 3.2example6.py ...
File Edit Format Run Options Windows Help
for i in range(10, 1, -3):
    print ("It's number ",i)
print ("done")
print (list(range(10, 1, -3)))
```

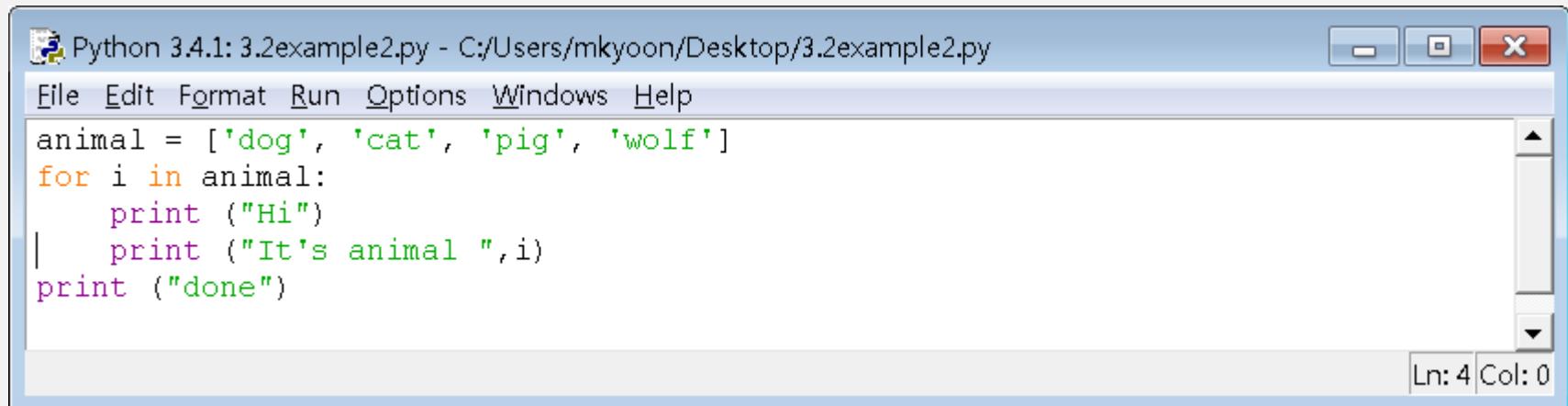
Ln: 4 Col: 26

```
>>>
>>>
>>>
>>> =====
===== RESTART =====
=====
>>>
It's number  10
It's number  7
It's number  4
done
[10, 7, 4]
>>> |
```

Ln: 44 Col: 4

Counting without numbers

- 숫자 없는 List



The screenshot shows a Windows-style application window titled "Python 3.4.1: 3.2example2.py - C:/Users/mkyoon/Desktop/3.2example2.py". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The main code area contains the following Python script:

```
animal = ['dog', 'cat', 'pig', 'wolf']
for i in animal:
    print ("Hi")
    print ("It's animal ",i)
print ("done")
```

The status bar at the bottom right indicates "Ln: 4 Col: 0".

실습

- 숫자 출력
 - 1) 0부터 9까지 출력하시오.
 - 2) 1부터 10까지 출력하시오.
 - 3) 1부터 10까지 짝수만 출력하시오.

실습

- 숫자 출력
 - 1) 0부터 9까지 출력하시오.
 - 2) 1부터 10까지 출력하시오.
 - 3) 1부터 10까지 짝수만 출력하시오.

```
print("# 0부터 9까지 출력")
for i in range(10):
    print(i, end = ' ')
print()
print("# 1부터 10까지 출력")
for i in range(1, 11):
    print(i, end = ' ')
print()
print("# 2부터 10까지 짝수만 출력")
for i in range(2, 11, 2):
    print(i, end = ' ')
```

실습

- 소수판별

- 자연수 p를 입력 받고, p가 소수(1과 자신만으로 나누어지는 수)인지를 판별하는 프로그램을 작성하시오.
 - 예) 96818971 는 소수인가?
 - 계산 시간을 측정 하시오.

```
# 시간과 관련된 기능을 가져옵니다.
import time

n = int(input("자연수 >"))

# 소수를 구하기전 시간을 저장 합니다.
start_time = time.time()

# whatever you want

# 걸린 시간을 출력합니다.
print("걸린 시간 :", time.time() - start_time)
```

실습

- 소수판별

- 자연수 p를 입력 받고, p가 소수(1과 자신만으로 나누어지는 수)인지를 판별하는 프로그램을 작성하시오.
 - 예) 96818971 는 소수인가?
 - 계산 시간을 측정 하시오.

```
# 시간과 관련된 기능을 가져옵니다.
import time

n = int(input("자연수 >"))

# 소수를 구하기전 시간을 저장 합니다.
start_time = time.time()
for i in range(2, n) :
    if n % i == 0 :
        print("{N}은 {I}로 나누어 떨어집니다.".format(N = n, I = i))
        break

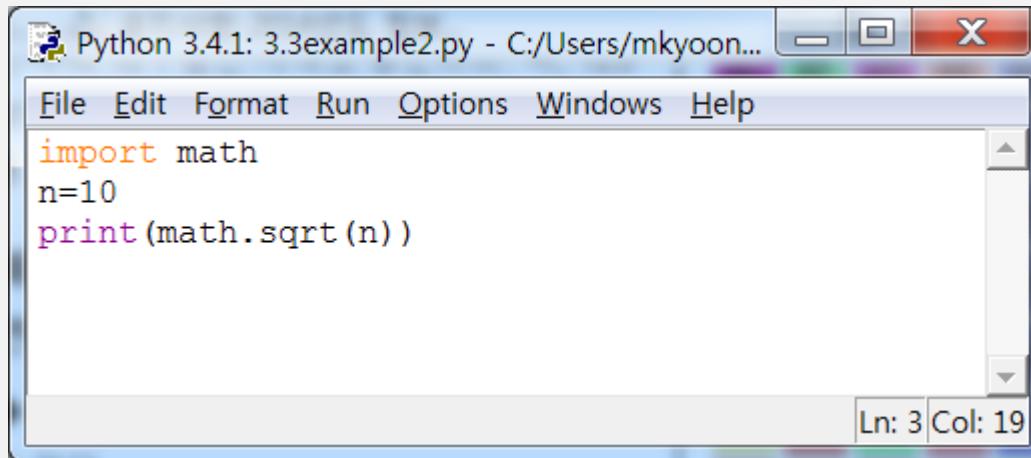
else :
    print("{N}은 소수입니다.".format(N = n))

# 걸린 시간을 출력합니다.
print("걸린 시간 :", time.time() - start_time)
```

실습

- 소수판별

- 자연수 p 를 입력 받고, p 가 소수(1과 자신만으로 나누어지는 수)인지를 판별하는 프로그램을 작성하시오.
 - 예) 96818971 는 소수인가?
 - 반복 횟수를 $p/2$ 보다 적게 하시오.
 - 힌트: python 제곱근 함수 사용법



The screenshot shows a Python 3.4.1 IDLE window titled "Python 3.4.1: 3.3example2.py - C:/Users/mkyoon...". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code editor contains the following Python code:

```
import math
n=10
print(math.sqrt(n))
```

The status bar at the bottom right indicates "Ln: 3 Col: 19".

실습

- 소수판별
 - [증명] n 에 대한 소수판별은 \sqrt{n} 이하의 정수로만 나누면 된다.
 - “ n 이 소수가 아니면, n 은 \sqrt{n} 이하의 숫자들을 약수로 갖는다.” 증명
 - \sqrt{n} 이하의 숫자를 대입했을 때 나누어 떨어짐이 확인됨
 - $n=a*b$ (소수가 아니라고 가정)
 - a 와 b 가 모두 \sqrt{n} 보다 큰 숫자라고 가정하면, $a*b$ 는 n 보다 큰 수가 됨 → 모순 발생 → a 와 b 가 모두 \sqrt{n} 보다 큰 숫자는 아니다가 증명됨 → 적어도 하나는 \sqrt{n} 이하의 수임이 증명됨 → 2부터 \sqrt{n} 까지만 나누어 보면 됨

실습

- 소수판별
 - 기존 코드에 비해서 얼마나 빨라 졌나요?

```
# 시간과 관련된 기능과 수학과 관련된 기능을 가져옵니다.  
import time, math  
  
n = int(input("자연수 >"))  
  
# 소수를 구하기전 시간을 저장합니다.  
start_time = time.time()  
for i in range(2, int(math.sqrt(n)) + 1) :  
    if n % i == 0 :  
        print("{N}은 {I}로 나누어 떨어집니다.".format(N = n, I = i))  
        break  
  
    else :  
        print("{N}은 소수입니다.".format(N = n))  
  
# 걸린 시간을 출력합니다.  
print("걸린 시간 :", time.time() - start_time)
```

While loops

- while loops

```
while <불 표현식> :  
    <코드>  
else :  
    <코드>
```

The screenshot shows a window titled "Python 3.4.1: 3.2example7.py - C:/Users/mkyoon/Desktop/3.2example7.py". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code editor displays the following Python script:

```
# Listing_8-8.py  
# Copyright Warren Sande, 2009  
# Released under MIT license http://www.opensource.org/licenses/mit-license.php  
# Version 61 -----  
  
# A conditional or while loop  
print ("Type 3 to continue, anything else to quit.")  
someInput = input()  
while someInput == '3':                      # loop condition  
    print ("Thank you for the 3. Very kind of you.")      # body  
    print ("Type 3 to continue, anything else to quit.")      # of the  
    someInput = input()                                     # loop  
print ("That's not 3, so I'm quitting now.")
```

The status bar at the bottom right indicates "Ln: 6 Col: 0".

While loops

- 실습
 - 은행 이율이 5%일 때, 100만원으로 200만원을 만들려면 몇 년을 저금 해야 하는지 while 문으로 계산해보자.

```
year = 0
balance = 1000000

while balance < 2000000 :
    year = year + 1
    balance = int(balance * 1.05)

print("{}년 저금 하시면 {}원이 됩니다.".format(year, balance))
```

While loops

- 실습
 - 제공받은 컴퓨터가 10초 동안 몇 번의 덧셈 연산을 반복하는지 실험 해보자.
 - 이 숫자가 컴퓨터의 계산 능력을 실제로 반영하는가? 아니라면 이러한 측정 방식의 문제는 무엇인가?

```
# 시간과 관련된 기능을 가져옵니다.  
import time  
  
# n(초)를 설정합니다.  
n = 10  
  
cnt = 0  
target_time = time.time() + n  
  
while time.time() < target_time :  
    cnt += 1  
  
print("{N}초 동안 {CNT}만큼 반복하였습니다.".format(N = n, CNT = cnt))
```

Bailing out of a loop – break and continue

- for loop 이나 while loop이 끝나기 전에 빠져 나오고 싶을 때
 - continue
 - 현재 반복 중지. 다음 반복 시행

Python 3.4.1: 3.2example8.py - C:/U... File Edit Format Run Options Windows Help

```
for i in range (1, 6):
    print ()
    print ('i =', i, end=" ")
    print ('Hello, how', end=" ")
    if i == 3:
        continue
    print ('r u?')
```

Ln: 7 Col: 18

Python 3.4.1 Shell File Edit Shell Debug Options Windows Help

```
ormation.
>>> =====
===== RESTART =====
=====
>>>

i = 1 Hello, how r u?

i = 2 Hello, how r u?

i = 3 Hello, how
i = 4 Hello, how r u?

i = 5 Hello, how r u?
>>> |
```

Ln: 14 Col: 4

Bailing out of a loop – break and continue

- for loop 이나 while loop이 끝나기 전에 빠져 나오고 싶을 때
 - break
 - 현재 반복문 중지

```
Python 3.4.1: 3.2example9.py - C:/U... File Edit Format Run Options Windows Help
for i in range (1, 6):
    print ()
    print ('i =', i, end=" ")
    print ('Hello, how', end=" ")
    if i == 3:
        break
    print ('r u?')
Ln: 6 Col: 13
```

```
Python 3.4.1 Shell File Edit Shell Debug Options Windows Help
>>> =====
===== RESTART =====
=====
>>>
i = 1 Hello, how r u?

i = 2 Hello, how r u?

i = 3 Hello, how
>>>
>>>
>>>
>>>
>>> |
Ln: 26 Col: 4
```

Loop else

- 파이썬은 loop 문에서 else 지원
 - Conditional statement가 false일 때 else문 수행
 - 마지막에 수행

The screenshot shows the Python 3.4.1 Shell window with the title "Python 3.4.1: 3.2example10.py - C:/...". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code editor contains the following Python script:

```
for i in range (1, 6):
    print ()
    print ('i =', i, end=" ")
    print ('Hello, how', end=" ")
#    if i == 3:
#        break
    print ('r u?')
else: print ('else')
```

The status bar at the bottom indicates "Ln: 6 Col: 1".

The screenshot shows the Python 3.4.1 Shell window with the title "Python 3.4.1 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The shell area displays the output of the script:

```
=====
RESTART
=====
>>>

i = 1 Hello, how r u?

i = 2 Hello, how r u?

i = 3 Hello, how r u?

i = 4 Hello, how r u?

i = 5 Hello, how r u?
else
>>> |
```

The status bar at the bottom indicates "Ln: 39 Col: 4".

Loop else

- 파이썬은 loop 문에서 else 지원
 - Conditional statement가 false일 때 else문 수행
 - 마지막에 수행
 - 단, break를 만나면 else문 수행하지 않음

```
Python 3.4.1: 3.2example10.py - C:/...  
File Edit Format Run Options Windows Help  
for i in range (1, 6):  
    print ()  
    print ('i =', i, end=" ")  
    print ('Hello, how', end=" ")  
    if i == 3:  
        break  
    print ('r u?')  
else: print ('else')  
Ln: 6 Col: 0
```

```
Python 3.4.1 Shell  
File Edit Shell Debug Options  
Windows Help  
>>> =====  
===== RESTART =====  
=====  
>>>  
  
i = 1 Hello, how r u?  
  
i = 2 Hello, how r u?  
  
i = 3 Hello, how  
>>>  
>>>  
>>>  
>>>  
Ln: 51 Col: 4
```

실습

- 유클리드 알고리즘
 - 최대공약수 구하기
 - $40(=2^3 \cdot 5)$ 와 $24(=2^3 \cdot 3)$ 의 최대 공약수 = $8(=2^3 \cdot 2)$
 - 정수 a 와 b 의 최대 공약수 ($a \geq b > 0$)를 $G(a,b)$ 라고 하면,
 - if $b == 0$
 - $G(a,b) = a,$
 - else: $G(a,b) = G(b, a \% b)$

실습

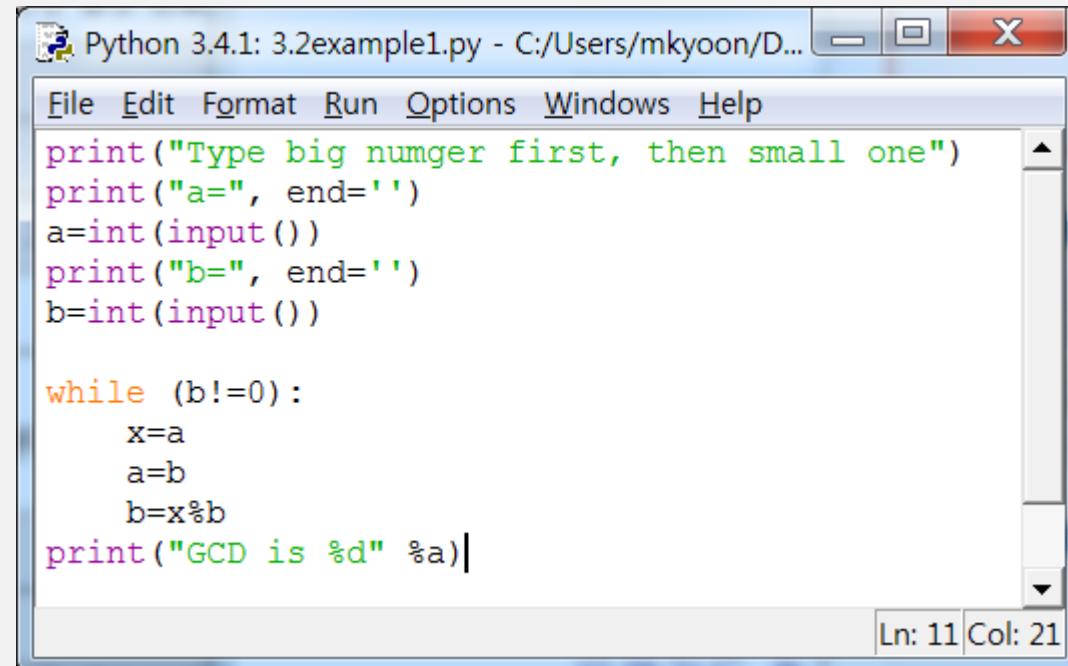
- 유클리드 알고리즘

- 증명

1. $G(a,b)=k$ 라고 하자 $\rightarrow a=m*k, b=n*k$ (m 과 n 은 서로 소)
2. $a=b*q+r$ (quotient : 몫, residue: 나머지) 라고 하면,
 - 파이썬 문법으로는, $r=a \% b, q=a//b$ 성립
3. $m*k=n*k*q+r \rightarrow r=(m-n*q)*k \rightarrow r$ 은 k 의 배수. 따라서 k 는 b 와 r 의 공약수
4. k 가 b 와 r 의 최대공약수임을 증명해야 함!
 - 위에서 $b=n*k$ 이며 $r=(m-n*q)*k$ 이므로, k 가 b 와 r 의 최대공약수임을 증명하기 위해서는 n 과 $(m-n*q)$ 가 서로 소임을 증명하면 됨
 - 귀류법 사용
 - 1) n 과 $(m-n*q)$ 가 서로 소가 아니라고 가정하자. 두 수의 최대공약수를 G 라고 하면, $n=x*G, m-n*q=y*G$ (x 와 y 는 서로 소)
 - 2) 그런데, $m-n*q=m-x*G*q=y*G \rightarrow m=G*(x*q+y)$ $\rightarrow m$ 과 n 이 G 를 공통으로 갖게됨 \rightarrow 1.에서 m 과 n 은 서로 소라고 했던 부분과 모순 발생
 - 3) 따라서, 1)의 가정은 틀렸다 $\rightarrow n$ 과 $(m-n*q)$ 는 서로 소이다!
 - 4) 따라서, k 는 b 와 r 의 최대공약수이다. 증명 끝.

실습

- 유클리드 알고리즘
 - 정수 a와 b의 최대 공약수 ($a \geq b > 0$)를 G(a,b)라고 하면,
 1. if $b==0$
 - 종료. a가 최대공약수
 2. $a=b$, $b=a \% b$
 3. 1,2반복



The screenshot shows a Windows-style application window titled "Python 3.4.1: 3.2example1.py - C:/Users/mkyoon/D...". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code area contains the following Python script:

```
print("Type big number first, then small one")
print("a=", end=' ')
a=int(input())
print("b=", end=' ')
b=int(input())

while (b!=0):
    x=a
    a=b
    b=x%b
print("GCD is %d" %a)|
```

The status bar at the bottom right indicates "Ln: 11 Col: 21".

실습

- 유클리드 알고리즘
 - 다음을 이용해서 알고리즘을 개선하시오
 - `while(b)`
 - $a, b = b, a \% b$

숙제

- **요일 구하기**

- 사용자로부터 날짜(연,월,일) 정보를 입력 받아서 해당 날짜에 해당하는 요일(월~일)을 출력하는 프로그램을 작성하시오.
 - 힌트: 1년 1월 1일은 월요일이다.

숙제

- 요일 구하기
 - 예) 2015년 3월 3일 → 화요일
 - 1년 1월 1일은 월요일이므로, 2015년 3월 3일까지 몇 일이 지났는지 알아내면(이 값을 n 이라고 하자), $(n \% 7)$ 의 값을 구하면 요일을 알아낼 수 있다
 - $(n \% 7) == 0 \rightarrow$ 월요일
 - $(n \% 7) == 1 \rightarrow$ 화요일
 - $(n \% 7) == 2 \rightarrow$ 수요일
 - $(n \% 7) == 3 \rightarrow$ 목요일
 - $(n \% 7) == 4 \rightarrow$ 금요일
 - $(n \% 7) == 5 \rightarrow$ 토요일
 - $(n \% 7) == 6 \rightarrow$ 일요일

숙제

- 요일 구하기

- 1년 1월 1일~2015년 3월 3일까지 날짜수 = (1년 날짜수+2년 날짜수+...+2014년 날짜수) + 2015년 3월 3일까지의 날짜수
 - 연도별 날짜수는 윤년에 따라 변함
 - 윤년일때는 366일
 - 윤년이 아닐때는 365일
 - 2015년 3월 3일까지의 날짜수는 달과 윤년에 따라 변함
 - 1월: 31일
 - 2월: 28일(윤년이 아닐때), 29일(윤년일때)
 - 3월: 31일
 - 4월: 30일
 - 5월: 31일
 - 6월: 30일
 - 7월: 31일
 - 8월: 31일
 - 9월: 30일
 - 10월: 31일
 - 11월: 30일
 - 12월: 31일