Chapter 1
변수, 자료형, 예약어, 연산자, 숫자형 자료형

2014. 5. 22

BITEC
BioMedical Informatics Training and Education Center
서울대학교병원 유전체정보의학 교육훈련센터

Python 이란 ?

Guido Van Rossum

http://www.python.org/~guido/

http://www.python.org/doc/essays/foreword/

연구실 문이 닫혀있는 1989년 크리스마스 기간에
무료한 시간을 보내기 위하여 새로운 **스크립트 언어**를 만들게 됨
ABC 라는 언어에 기반해서 작성됨

특징
- 명료하고 읽기 쉬운 문법
- 대화형 인터프리터 언어
- Object Oriented Programming(OOP) Language
- high level 동적 데이터 타입 결정 지원
- 메모리 자동 관리
- 다른 언어(C, C++, JAVA, .NET, R, …)와의 확장성
- 다양한 라이브러리 제공
- 무료

성공 사례
Google Groups/Maps, Gmail, YouTube, NASA, Yahoo!, Naver, Daum,
다수의 IT 기관 및 연구 기관

명료하고 읽기 쉬운 문법

C++ (hello.cpp)

```
#include <iostream>
using namespace std;

int main()
{
        cout << "Hello World!" << endl;
        return 0;
}
```

JAVA (Hello.java)

```
public class Hello
{
        public static void main( String[] args )
        {
                System.out.println("Hello world!");
        }
}
```

Python (hello.py)

```
print "Hello world!"
```

Python은 대화형 인터프리터 언어

```
chanchan-ui-Mac-Pro:~ chanchan$ python
Python 2.6.1 (r261:67515, Jun 24 2010, 21:47:49)
[GCC 4.2.1 (Apple Inc. build 5646)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World“
Hello World
>>> for i in range(1,10):
...     print 2*i,
...
2 4 6 8 10 12 14 16 18
>>>
```

실행결과

명령입력

명령입력

실행결과

>>>
프롬프트(prompt) : 명령 입력 대기중임을 나타낸다.

High Level 동적 데이터 타입 결정 지원

```
>>> def add(a, b):
...     return a+b
...
>>> print add(1, 2)
3
>>> print add(BIT', ‘EC')
BITEC
>>> print add([1,2,3], [4,5,6])
[1, 2, 3, 4, 5, 6]
```

숫자 Type

문자 Type

List Type

return a+b 에서
a 와 b 에 맞는 + method 를 자동으로 호출함

메모리 자동 관리

- C나 C++ 같은 언어는 메모리 관리 책임이 개발자에게 달려있음.
- 약간의 수정에도 항상 메모리 관리를 점검해야 함.
  - 의도치 않은 에러 발생 확률 높음.

- Python 에서는 Garbage Collection 기능을 이용하여 메모리 관리를 함.
  - 필요시 자동으로 메모리를 자동할당
  - 사용이 끝나면 자동을 사용 해제

다른 언어와의 확장성

현재는 작동하지 않는 코드입니다.
타이핑 하지 마십시오.

```
> x <- c(5.05, 6.75, 3.21, 2.66)
> y <- c(1.65, 26.5, -5.93, 7.96)
> z <- c(1.65, 2.64, 2.64, 6.95)
> cor(x, y, method="spearman")
[1] 0.4

> cor(x, z, method="spearman")
[1] -0.6324555
```

R cor 함수를 이용해 상관계수 구하기

```
>>> import rpy
>>> x = [5.05, 6.75, 3.21, 2.66]
>>> y = [1.65, 26.5, -5.93, 7.96]
>>> z = [1.65, 2.64, 2.64, 6.95]
>>> print rpy.r.cor(x, y, method="spearman")
0.4

>>> print rpy.r.cor(x, z, method="spearman")
-0.632455532034
```

Python 에서 rpy 모듈을 사용하여
R cor 함수를 호출

풍부한 라이브러리
예) BioPython 을 이용해 PubMed 문헌 search/retrieve

```
#!/usr/bin/python
from Bio import Entrez
from Bio import Medline

handle = Entrez.esearch( db="pubmed", term="Type 2 Diabetes korean ", retmax=3 )
record = Entrez.read( handle )
idlist = record["IdList"]

handle = Entrez.efetch(db="pubmed", id=idlist, rettype="medline", retmode="text")
records = Medline.parse(handle)

for record in records:
    print "PubMed ID : ", record.get("PMID", "?")
    print "TITLE: ", record.get("TI", "?")
    print "ABSTRACT: ", record.get("AB", "?")
```

BioPython 모듈 불러오기

키워드
'Type 2 Diabetes korean' 으로
PubMed 문헌 검색 후
PubMed ID 3개만 가져옴

가져온 PubMed ID 를 이용하여
MedLine Text 를 가져옴

PubMed ID, Title, Abstract 를 출력

총 12 줄 짜리 프로그램

결과

PubMed ID : 23091317
TITLE: Association between Total Bilirubin and Hemoglobin A1c in Korean Type 2 Diabetic Patients.
ABSTRACT: Recent studies have shown that bilirubin is negatively associated…생략

PubMed ID : 23091316
TITLE: Association between Nutrient Intake and Obesity in Type 2 Dia
Program: A Cross-Sectional Study.
ABSTRACT: The aim of the study was to assess the association between usual dietary nutrient … 생략

현재는 작동하지 않는 코드입니다.
타이핑 하지 마십시오.

PubMed ID : 23076296
TITLE: The prevalence and risk factors of vertebral fractures in Korean patients with type 2 diabetes.
ABSTRACT: Although type 2 diabetes mellitus (T2DM) has been associated with an increase in fracture risk, there is no data regarding the prevalence of vertebral fractures or its risk factors for patients with T2DM in.. 생략

간단한 실습

```
>>> 2 + 4
6
>>> ( 4 + 6 ) * 2
20
>>> 11 / 3
3
>>> 11 / 3.
3.6666666666666665
>>> 11 % 3
2
>>> divmod(11, 3)
(3, 2)
>>> a, b = divmod(11, 3)
>>> a
3
>>> b
2
>>> 3.4e4
34000.0
>>> 3.4e-4
0.00034000000000000002
>>> 2**3
8
```

정수 / 정수 = 정수
정수 / 실수 = 실수
실수 / 실수 = 실수

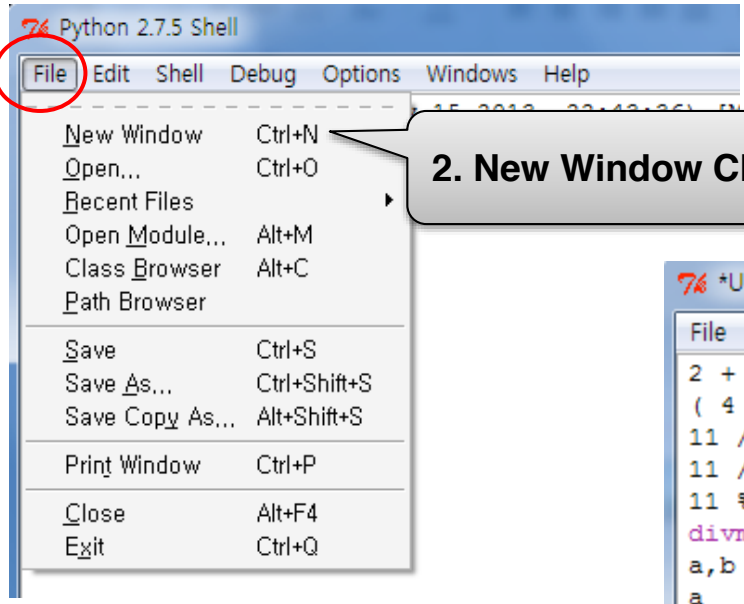
정수/정수 형에서 제대로 된 계산결과를 얻기 위해
3. 이라고 표시하여 3이라는 숫자를 실수형으로 인식

% 나머지 연산자
divmod(a, b) a를 b로 나눴을 때 몫과 나머지를 계산해주는 함수(function)

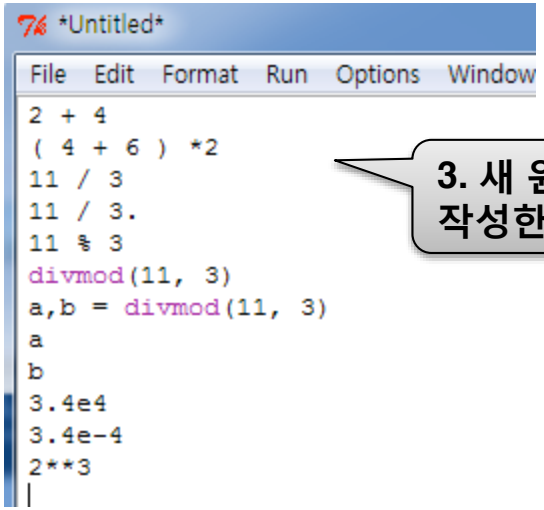
3.4e4 : 3.4 곱하기 10의 4승
3.4e-4 : 3.4 곱하기 10의 -4승
23 : 2의 3승**

작성한 프로그램 저장

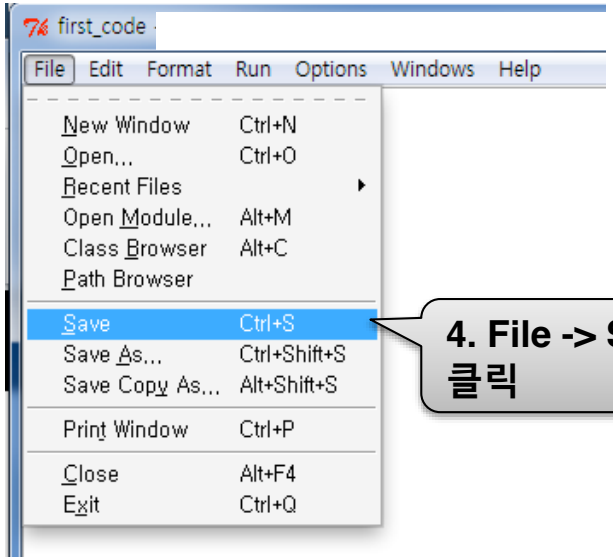
1. File Click



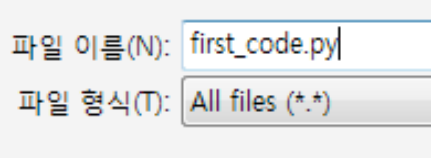
2. New Window Click



3. 새 윈도우 창에 앞에서 작성한 계산을 입력



4. File -> Save 클릭



5. first_code.py 라고 타이핑 후 저장

주의 : 저장 시 저장 경로는 현재 사용자가 쓰기 권한이 있는 디렉토리여야 합니다.

작성한 프로그램 실행

first_code.py -

File Edit Format Run Options Windows Help

```
2 + 4
( 4 + 6 ) *2
11 / 3
11 / 3.
11 % 3
divmod(11, 3)
a,b = divmod(11, 3)
a
b
3.4e4
3.4e-4
2**3
```

Python Shell
Check Module Alt+X
Run Module F5

1. Run 선택 후 Run Module 로 실행

Python 2.7.5 Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>>
```

2. 아무런 결과도 나오지 않는다???

first_code.py -

File Edit Format Run Options Windows Help

```
print 2 + 4
print ( 4 + 6 ) *2
print 11 / 3
print 11 / 3.
print 11 % 3
print divmod(11, 3)
a,b = divmod(11, 3)
print a
print b
print 3.4e4
print 3.4e-4
print 2**3
```

3. 출력하고 하는 부분에 print 구문을 추가 이는 idle 뿐 아니라 다른 에디터 프로그램을 사용하여 출력할 시에도 동일함

Python 2.7.5 Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>> ================================ RESTART ================================
>>>
6
20
3
3.66666666667
2
(3, 2)
3
2
34000.0
0.00034
8
>>>
```

4. Bingo !!!

Indentation (들여쓰기) (1/2)

```
>>>a = 1
>>>  a = 1

  File "<pyshell#5>", line 1
    a = 1
   ^
IndentationError: unexpected indent
```

한 칸 공란이 있음

Indentation(들여쓰기) : 명령 입력 시 첫 컬럼부터 시작해야 한다. 그렇지 않으면 indentation error 가 나게 된다.

Python 은 들여쓰기를 엄격하게 지키는 언어이다.

Indentation (들여쓰기) (2/2)

```
import MySQLdb
db= MySQLdb.connect(host="localhost", user="python-test", passwd="python",db="python-test")
    cursor = db.cursor();stmt = "SELECT * from books"
cursor.execute(stmt);rows = cursor.fetchall ()
for row in rows: print "Row: “;
for col in row : print "Column: %s" % (col); print "End of Row"
print "Number of rows returned: %d" % cursor.rowcount
cursor.close(); db.close()
```

Indentation 을 지키지 않으면 Syntax error 를 일으킬 뿐 아니라 소스 코드를 읽기조차 힘들어진다.

```
import MySQLdb

db= MySQLdb.connect(host="localhost", user="python-test", passwd="python", db="python-test")

cursor = db.cursor()
stmt = "SELECT * from books"
cursor.execute(stmt)

rows = cursor.fetchall ()
for row in rows:
    print "Row: "
    for col in row :
        print "Column: %s" % (col)
    print "End of Row"
print "Number of rows returned: %d" % cursor.rowcount

cursor.close()

db.close()
```

현재는 작동하지 않는 코드입니다. 타이핑 하지 마십시오.

Syntax Error vs. Logical Error

- **Syntax Error**
  - 스펠링이나 문법이 틀려서 나는 에러
  - 실행 시 출력되는 에러 메시지를 이용하여 에러 검출이 가능

- **Logical Error**
  - Syntax Error 은 전혀 없으나 논리상 오류가 나는 에러 종류
  - 찾아내기 쉽지 않은 에러 형태.

```
>>> prin "Hello Python"
  File "<stdin>", line 1
    prin "Hello Python"
                  ^
SyntaxError: invalid syntax
```

예)Syntax error

```
>>> result = ''
>>> grade = 95
>>> if grade >= 90:
...     result = 'F'
...
>>> print result
F
```

예) Logical error

변수 (variable)

알려진 또는 알려지지 않은 어떤 값(value)을 담는 저장공간(Storage location) 과 그에 상응하는 심볼릭 이름(symbolic name)을 의미

```
>>> a = 1
>>> string_test = 'Python'
```

variable	Value
a	1
string_test	Python

'=' 치환 연산자 (assignment operator)
오른쪽 항의 값을 왼쪽으로 치환

string_test = 'Python'

Python 이란 값(value)을
string_test 란 변수(variable)로 치환

```
>>> a = 1
>>> a = a + 1
>>> a
2
>>>
```

a 라는 변수에 1이라는 값을 할당한다.

a 라는 변수에 1이라는 값을 더한 후 결과를 다시 a 라는 변수에 할당한다.

변수선언

- 첫 문자가 underscore(_) 또는 영문자
- 두 번째 문자부터는 영문자, 숫자, underscore 가 올 수 있음.
- 크기에는 제한이 없음
- [a-zA-Z_][a-zA-Z-0-9_]*
- 사용 가능한 예
  - a, input_str, MyVar, _private_string, …
- 변수명이 될 수 없는 예
  - 5public, @email, %var, ….
- 변수명은 대소문자를 구분.
  - 예) Email 과 email 은 다른 변수

```
>>> @email = 'chanchan@snu.ac.kr'
SyntaxError: invalid syntax
>>> email = 'chanchan@snu.ac.kr‘

>>> Email = 'test@test.com'
>>> email = 'chanchan@snu.ac.kr'
>>> Email
'test@test.com'
>>> email
'chanchan@snu.ac.kr'
```

변수선언 규칙에 맞지 않는다.

변수명은 대소문자를 구분한다.

예약어는 변수명으로 사용할 수 없다.

- Python 에서 이미 쓰고 있는 예약어(reserved words or keywords)는 변수명으로 사용할 수 없다.

Python 에서 사용하는 예약어

and	del	from	not	while
as	elif	global	or	with
assert	else	if	pass	yield
break	except	import	print	class
exec	in	raise	continue	
is	return	def	finally	
for	lambda	try		

```
>>> from = 10
SyntaxError: invalid syntax
```

```
# 예약어 확인하기
>>> import keyword
>>> keyword.kwlist
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally',
'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try',
'while', 'with', 'yield']
>>>
```

import : 가져오기/들여오기 – 다른 사람이 만든 또는 본인이 만든 데이터/프로그램을 읽어들이는 행위

내장함수 (built-in functions) 이름을 변수명으로 사용하면 생기는 문제

텍스트	Built-in Functions			
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

http://docs.python.org/2/library/functions.html

```
>>> divmod
<built-in function divmod>
>>> divmod(10,3)
(3, 1)
>>> divmod = 'Division and Modulus'
>>> divmod(10, 3)
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    divmod(10, 3)
TypeError: 'str' object is not callable
>>>
```

divmod 는 문자형 변수로 바뀌었고 더 이상 함수가 아니다.

문자형 변수에 함수형태의 값을 입력하기 때문에 에러가 발생한다.

변수명의 좋은 예와 나쁜 예

```
>>> a = 160.4
>>> aa = 71.1
>>> print a, aa
160.4 71.1
>>>

>>> height = 160.4
>>> weight = 71.1
>>> print height, weight
160.4 71.1
```

```
>>> x = 90
>>> xx = 77
>>> xxx = 93
>>> y = x + xx + xxx
>>> y / 3.
86.66666666666667

>>> english = 90
>>> mathematics = 77
>>> computer = 93
>>> total = english + mathematics + computer
>>> total / 3.
86.66666666666667
```

잘 명명한 변수명은 코드를 다시 들여다 보는 본인 뿐 아니라 코드를 보는 다른 사람들도 행복하게 만듭니다.

코드에 주석 달기

- 주석(comment)은 # 문자를 이용하여 달 수 있음.
- 주석 처리된 부분은 프로그램 실행 시 영향을 미치지 않음.

```
>>> # The score of english
>>> english = 90

>>> # The score of math.
>>> mathematics = 77

>>> # The score of computer
>>> computer = 93

>>> # The total score of english, math and computer
>>> total = english + mathematics + computer

>>> # Average score
>>> total / 3.
86.66666666666667

>>> total / 3. # Average Score total/3.
86.66666666666667
```

Total / 3. 까지의 결과만 나오고 # 이후의 내용은 무시함

자료형 (Data Type)

- 프로그램에서 표현 또는 저장하는 데이터의 유형.
- 컴퓨터로 표현할 수 있는 자료의 종류
- Python 에서는 다른 언어에서 제공하지 않는 고수준의 자료형이 존재

자료형	설명	예
수치형(Numbers)	정수, long형 정수, 실수, 복소수등을 표현	123, 12345L, 1.43, 5+4j
문자열(Strings)	문자들의 모임	‘BITEC’, “Python”, “”””BITEC Python””””
리스트(Lists)	순서를 가지는 python 객체의 집합	[‘BITEC’, ‘Python’]
사전(Dictionary)	순서를 가지지 않는 객체의 집합. 키(key)와 값(value)으로 구성됨	{ ‘BITEC’: 1, ‘Python’:2 }
튜플(Tuples)	순서를 가지는 python 객체의 집합. 순서변경이 안됨	(‘BITEC’, ‘Python’)
파일(Files)	파일에 자료를 입출력하기 위한 객체	fp = open(‘BITEC.txt’)

```
>>> variable = 1
>>> type(variable)
<type 'int'>

>>> variable = 3.141592
>>> type(variable)
<type 'float'>

>>> variable = "BITEC"
>>> type(variable)
<type 'str'>

>>> variable = ['BITEC', 'Python']
>>> type(variable)
<type 'list'>
```

type 함수는 객체의 자료형을 알려준다.

int : 정수형
float : 실수형
str : 문자형
list : 리스트형

```
>>> variable = { 'BITEC':1, 'Python':2 }
>>> type(variable)
<type 'dict'>

>>> variable = ('BITEC', 'Python')
>>> type(variable)
<type 'tuple'>

>>> fp = open("C:\BITEC_Python\ch1\hello.py")
>>> type(fp)
<type 'file'>
```

dict : 딕셔너리형
tuple : 튜플형
file : 파일형

수치 자료형 (1/2)

- 수치자료형
  - 정수형(int), 실수형(float), 롱형(long), 복소수형(complex)이 있다.
- 정수형 상수
  - 수치 표현 범위 : -2,147,483,648~2,147,483,647(32bit인 경우)

```
# 32bit OS
>>> maxint = 2**31 - 1
>>> maxint
2147483647L
>>> -maxint - 1
-2147483648L

>>> import sys
>>> sys.maxint
2147483647
```

```
# 64bit OS
>>> import sys
>>> sys.maxint
9223372036854775807
>>> 2**63 -1
9223372036854775807L
```

32/64bit OS에 따라 최대 표현 범위가 다르다.

- 롱형 상수
  - 정수형으로 표현할 수 없는 큰 수는 자동으로 롱형 정수로 표현
  - 롱형상수는 정수수치 마지막에 'l', 혹은 'L'을 붙여서 표현
  - 유효자리 : 메모리가 허용하는 한 무한대.

```
#32bit OS
>>> 12345678987654321
12345678987654321L
```

정수형의 최대 표현범위를 넘기면 자동으로 L 이 붙는다.

```
# 32bit OS
>>> print 12345678987654321
12345678987654321
```

print 문을 사용하면 L 이 붙지 않는다

```
#64bit OS
>>> 1234567890987654321
1234567890987654321
```

정수형의 최대 표현범위를 넘지 않았으므로 L 이 붙지 않는다.

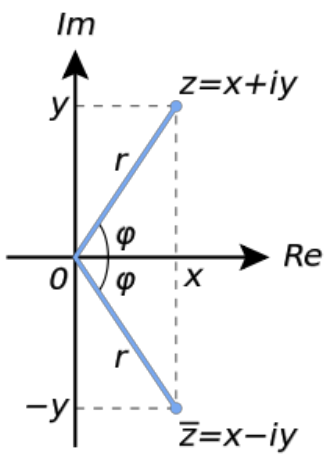
수치 자료형 (2/2)

- 실수형
  - 소수점을 포함하거나 e나 E로 지수를 포함
  - 범위 : 10의 -308승~ + 308승

```
>>> a = 4.5
>>> b = 3.5e3
>>> c = -0.3e-4
>>> print a, b, c
4.5 3500.0 -3e-05

>>> import sys
>>> sys.float_info.max
1.7976931348623157e+308
```

- 복소수형
  - 실수부 + 허수부 로 나뉨
  - 허수부에는 ‘j’나 ‘J’를 붙여야 함




```
>>> c = 5 + 8j
>>> d = 2 - 6j
>>> c + d
(7+2j)
>>> c.real
5.0
>>> c.imag
8.0
>>> c.conjugate()
(5-8j)
```

real : 실수부

imag : 허수부

conjugate(): 켤레복소수

math 모듈

```
>>> import math
>>> math.sqrt(16)
4.0

>>> math.exp(3)
20.085536923187668

>>> math.log(10)
2.3025850929940459

>>> math.log10(10)
1.0
>>> math.pi
3.1415926535897931

>>> math.e
2.7182818284590451

>>> math.fabs(-11.5)
11.5

>>> round(3.54)
4.0
```

math 모듈 불러오기

Built-in 함수

math.sqrt() : 제곱근
math.exp() : 지수
math.log() : 자연 로그 (base 가 e)
math.log10() : 자연 로그 (base가 10)
math.pi : pi 값
math.e : e 값
math.fabs() : 절대값

round : 반올림 함수(내장형 함수임)

수치 연산자

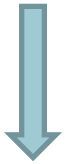
- +(덧셈), -(뺄셈), *(곱셈)
- /(나눗셈), **(지수승), %(나머지)

- 정수형을 다른 정수형과 연산하면 결과는 언제나 정수형
- 정수형과 다른 실수형을 연산하면 결과는 언제나 실수형

```
>>> 4 + 6    # 정수 + 정수
10           # 결과는 정수
>>> 4 + 6.0  # 정수 + 실수
10.0         # 결과는 실수
>>> 6 / 4.0  # 정수 / 실수
1.5          # 결과는 실수
>>> 6 / 4    # 정수 / 정수
1            # 결과는 정수
```

수치 연산자의 우선순위

우선순위 높음



우선순위 낮음

연산자	설명	결합 순서
()	괄호	왼쪽 → 오른쪽
+, -	단항 연산자(예. -5, +4)	오른쪽 → 왼쪽
**	지수	오른쪽 → 왼쪽
*, /, %	곱하기, 나누기, 나머지	왼쪽 → 오른쪽
+, -	더하기, 빼기	왼쪽 → 오른쪽

```
>>> 3 * -2
-6
>>> 2 + 3 * 4
14
>>> ( 2 + 3 ) * 4
20
```

```
>>> 4 / 2 * 2
4
>>> 4 / ( 2 * 2 )
1
>>> 2 ** 3 ** 4
2417851639229258349412352L
>>> ( 2 ** 3 ) ** 4
4096
```

순서가 혼동된다면 괄호로 묶어주는 것이 가장 안전하다!

관계 연산자

❖ 값을 서로 비교하는 연산자

관계 연산자	의미
>	크다
<	작다
>=	크거나 같다
<=	같거나 작다
==	같다
!=	같지 않다

텍스트

```
>>> 8 == 9
False
>>> 8 != 9
True
>>> 3 > 6
False
```

```
>>> 5 <= 6
True
>>> a = 2
>>> b = 3
>>> a > b
False
>>> 0 < a < b
True
```

0 < a and a < b
와 동일한 표현

논리 연산자

- 진리값(True, False)을 피연산자로 취하여 논리값을 계산하는 연산자

우선순위 높음

우선순위 낮음

논리 연산자	의미
not x	x가 거짓이면 True, 아니면 False
x and y	x 와 y 모두 참이면 True 아니면 False
x or y	x 또는 y 둘 중 하나라도 참이면 True, 모두 거짓이면 False

```
>>> a = 10
>>> b = 20

>>> a > 10 and b < 50
False
>>> a >= 10 and b < 50
True

>>> a > 6 or b < 0
True
>>> a > 20 or b < 10
False
>>> ( a > 20 or b < 50 ) and a > 100
False
```

```
>>> not 1
False
>>> not 0
True
```

내부적으로 True 는 1을 False 는 0 의 값을 가짐

x	y	x and y
True	True	True
True	False	False
False	True	False
False	False	False

x	y	x or y
True	True	True
True	False	True
False	True	True
False	False	False

Homework

제출 기한 : 2014년 5월 29일(목) 자정
E-mail : chanchan@snu.ac.kr, 박찬희
이메일 제목 맨앞에 [BITEC 성함] 을 명시할것.
예) [BITEC 홍길동] 1강 Homework 입니다.
파일 형식 : ms-word 또는 한글 파일 또는 pdf 파일
파일명 : 성함.pdf, 성함.hwp, 성함.doc
프로그래밍 하는 부분은 반드시 소스코드를 기입해서 제출해야 합니다

1. 아래 문자 중 변수명으로 사용할 수 없는 것들을 찾아 이유를 기술하시오
   1. patient, _secret_text, %_percent, 1st_author, ranges, range, pass

2. (6 < 10) and (7 != 5) or (10 == 9) 의 결과가 무엇이고 왜 그렇게 나왔는지 설명하시오

3. BMI(Body Mass Index) 는 아래와 같이 계산할 수 있습니다.

$$BMI = \frac{mass\ (kg\ )}{(\ height\ (meter))^2}$$

   1. mass, height 라는 변수에 몸무게가 89.3kg 이고 키가 172cm 인 경우를 입력하고 BMI 변수에 계산 결과를 담은 후 출력하시오 (단위 주의!!!)