



**byte of python + PyQt5**

김 승 환

[swkim4610@inha.ac.kr](mailto:swkim4610@inha.ac.kr)

하이테크 401호

# python?

- 파이썬은 배우기 쉽고 강력한 프로그래밍언어이다.
- 간단하게 객체지향적 접근이 가능하다.
- 다양한 플랫폼에서 사용될 수 있는 Interpreter 방식의 RAD(rapid application development) 언어이다.
- 파이썬은 귀도 반 로섬(Guido van Rossum)이 만들었다.
- 파이썬은 고대 신화에 나오는 큰 뱀을 말한다.
- 파이썬은 오픈소스 언어이다.
- 파이썬으로 프로그램을 개발하고 더 빠른 속도를 원하거나 프로그램의 내용을 일부 공개하고 싶지 않을 경우에는 파이썬 코드의 일부분을 C, C++로 작성한 후, 파이썬에서 읽어 사용할 수 있다.
- 파이썬은 방대한 라이브러리를 갖추고 있다.
- 여기에서는 파이썬 2를 다룰 것이다.
- 윈도우 환경은 <https://www.python.org/downloads/>에서 설치가 가능하고 리눅스 환경은 대부분 이미 설치되어 있다.
- <https://www.continuum.io/downloads>에서 아나콘다를 받는 것을 추천한다. 아나콘다는 numpy, PyQt, matplotlib와 같은 필수 패키지와 spyder 라는 좋은 개발환경이 내장되어 있다.
- Life is short, use python

# 파이썬 시작

```
$ python
Python 2.7.6 (default, Feb 23 2014, 16:08:15)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "hello world"
hello world
```

- 파이썬 shell에서 파이썬 스크립트로 명령을 하면 결과가 즉시 출력된다.  
이러한 방식이 Interpreter 방식이다.
- 파이썬 셸에서 나오는 방법은 `exit()`로 나온다.
- 파이썬 프로그래밍은 셸에서 직접하지 않는다. 대부분 편집기를 사용한다.
- 이는 R에서 Rstudio를 사용하는 것과 같은 개념이다.
- 아나콘다를 설치하면 `spyder`라는 편집기를 사용할 수 있다.
- 스파이더 외에 `PyCharm`이라는 편집기도 많이 사용된다.(<https://www.jetbrains.com/pycharm/>)

# 파이썬 시작

파이썬에서 주석은 아래와 같은 형식으로 표현 가능하다. 주석은 가능한 많이 자세하게 하는 습관을 기르는 것이 좋다.

```
print 'hello world' # Note that print is a statement
```

```
# Note that print is a statement  
print 'hello world'
```

숫자형에는 정수형(Integer)과 부동 소수점 숫자형(Float)의 두 가지 종류가 있다.

다른 언어처럼 short, long을 구분하지 않고 모두 담을 수 있다.

1\_hello.py

```
age=20  
name="Swaroop"  
print '{0} was {1} years old when he wrote this book'.format(name, age)  
print name + " is " + str(age) + " years old"  
# 중괄호안의 숫자는 생각가능하다.  
print ' {} was {} years old when he wrote this book '.format(name, age)  
# 소수점 이하 셋째 자리까지 부동 소수점 숫자 표기 (0.333), (___hello___) 출력  
print '{0:.3f} {1:_^11}'.format(1.0/3, "hello")
```

# 파이썬 시작

print문에는 “\n”이 추가되어 있다. 이를 방지하려면 print 문끝에 “,” 을 붙이면 된다.

```
print "a",  
print "b",
```

“What’s your name?” 이라는 문자열을 만들고 싶은데 파이썬에서는 “과 ‘ 이 동일하기 때문에 곤란하다.

이때에는 “What\’s your name?” 라고 표기하면 된다. 만약, 백슬래시를 표현하고자 할 때에는 \\ 로 표현한다.

‘This is the first line\nThis is the second line’ 에서 \n 은 줄바꿈을 의미한다.

또한, 아래와 같이 두 줄로 표현된 문장도 \ 에 의해 한 줄로 표현된다.

"This is the first sentence. \

This is the second sentence.“

"This is the first sentence. This is the second sentence.“

파이썬에서 구문과 변수는 대,소문자를 구분한다. 또한, 파이썬에서 들여쓰기가 중요하다.

같은 블록은 같은 들여쓰기를 해야 한다. 그렇지 않을 경우, 오류를 일으킨다.

# 파이썬 시작

파이썬에서는 +, -, \*, /, \*\*, % 를 사용할 수 있다.

13 / 3 은 4이고, 13.0 / 3 은 4.333333333333333 이다. 13 % 3 은 1 이다.

2\*\*3 = 8 이다.

비트 연산자로 &, |, ^, ~, 논리연산자로 <, >, <=, >=, ==, !=, Boolean 연산자로 not, or, and를 사용할 수 있다.

# 흐름제어(if: elif: else)

이 프로그램은 우리에게 if문 사용과 raw\_input 이라는 함수를 통해 숫자를 입력받는 방법 그리고, 들여쓰기를 하는 방법을 설명하고 있다. If 문 끝 “:” 은 새로운 블록이 시작됨을 알리는 것이고 그 다음에는 들여쓰기를 해야 한다. 들여쓰기를 잘못할 경우, 심각한 오류가 발생한다는 점에 주의해야 한다.

2\_if.py

```
number = 23
guess = int(raw_input('Enter an integer : '))
if guess == number:
    # New block starts here
    print 'Congratulations, you guessed it.'
    print '(but you do not win any prizes!)'
    # New block ends here
elif guess < number:
    # Another block
    print 'No, it is a little higher than that'
    # You can do whatever you want in a block ...
else:
    print 'No, it is a little lower than that'
    # you must have guessed > number to reach here
print 'Done'
```

# 흐름제어(while)

아래는 while문을 사용해 `guess == number`를 만족할 때까지 looping 하는 예제이다.

마지막 `else` 문은 `while` 문의 `else`인 점이 특이하다. 즉, `while` 문을 빠져나갈 때, 실행되는 문장이다.

참고로 파이썬에는 `switch` 문이 없다.

3\_while.py

```
number = 23
running = True

while running:
    guess = int(raw_input('Enter an integer : '))
    if guess == number:
        print 'Congratulations, you guessed it.'
        # this causes the while loop to stop
        running = False
    elif guess < number:
        print 'No, it is a little higher than that.'
    else:
        print 'No, it is a little lower than that.'
else:
    print 'The while loop is over.'
# Do anything else you want to do here
print 'Done'
```



# 흐름제어(for loop)

for 문을 사용해서 i를 1~5 직전까지 증가시키며 looping 하는 예제이다.

for 문에도 else 문을 사용할 수 있다는 점이 특이하다.

수행결과를 보면 1,2,3,4가 나옴에 주의해야 한다.

```
for i in range(1, 5):  
    print i  
else:  
    print 'The for loop is over'
```

while, for 문은 loop 도중에 break문을 사용해 강제로 빠져나올 수 있다.

이 때에는 else 문은 실행되지 않는다.

continue문은 break 처럼 빠져나가는 것이 아니고 다음 루프로 넘어가게 하는 명령이다.

함수는 def 키워드를 통해 아래와 같이 만들 수 있다.

5\_function1.py

```
def say_hello():  
    print "hello world"  
  
say_hello()  
  
def print_max(a,b):  
    if a>b:  
        print a, 'is maximum'  
    elif a==b:  
        print a, 'is equal to', b  
    else:  
        print b, 'is maximum'  
  
print_max(1,2)  
print_max(2,2)
```

# 함수(local/global 변수)

함수 내부에서 정의된 변수는 지역변수로 역할을 하여 함수 밖에서는 인식되지 못한다.

만약, 함수 내부에서 정의된 변수가 함수 밖에서 사용하고 싶다면 함수 내부에서 global 변수로 선언되어야 한다.

단, global 변수는 인수를 통해 전달할 수 없다.

## 6\_functionLocal.py

```
x=50
def func(x):
    print "x is", x
    x=2
    print "changed local x to", x

func(x)
# x=50
print "x is", x
```

## 7\_functionGlobal.py

```
x=50
def func():
    global x
    print "x is", x
    x=2
    print "changed local x to", x

func()
# x=2
print "x is", x
```

## 8\_functionDefault.py

```
def say(message, times=1):
    print message*times

# times는 default 값으로 1으로
# 갖는다.

say("Hello")
say("python",5)
```

3번째 예제에서 기본값이 있는 매개변수는 위치가 마지막에 있어야 한다.

즉, `def say(times=1, message)`는 문법오류가 발생한다.

# 함수(VarArgs)

매개변수의 수가 가변적일 때, VarArgs를 사용한다.

9\_functionVarArg.py

```
def total(initial=5, *numbers, **keywords):  
    count = initial  
    for i in numbers:  
        count += i  
    for i in keywords:  
        count += keywords[i]  
    return count  
  
# 결과는 모든 값을 합친 166이다.  
print total(10,1,2,3,vegetables=50, fruits=100)
```

이 프로그램에서 \*numbers는 1,2,3을 numbers[0], numbers[1], numbers[2]으로 받고,  
\*\*keywords는 vegetables=50, fruits=100 의 값 50, 100을 keywords[0], keywords[1]로 받는다.

# Algorithm Training Part 1

1. 1에서 100까지 정수더하기, 짝수만 더하기, 홀수만 더하기
2. 구구단 출력하기
3. 10진수를 2진수로 바꾸기

# 함수(DocString)

아래는 DocString을 사용하는 예이다. DocString은 ''' 으로 정의되고 `.__doc__` 메소드를 통해 사용할 수 있다.  
단, DocString의 시작은 대문자이고 끝은 마침표 “.” 으로 끝나야 한다.

10\_functionDocString.py

```
def printMax(x,y):  
    ''' Print the maximum of two numbers.  
    the two values must be integers.'''  
    if x>y:  
        print x, 'is maximum'  
    else:  
        print y, 'is maximum'  
  
printMax(3,5)  
print printMax.__doc__
```

5 is maximum

Print the maximum of two numbers.

the two values must be integers.

# 모듈 import

아래는 sys 라는 표준라이브러리를 불러와 사용하는 방법이다. 이처럼, 파이썬에서는 라이브러리를 import하여 사용가능하다. 라이브러리는 반드시 sys.path 중 한 곳에 위치해야 한다.

11\_moduleSys.py

```
import sys
print("The command line arguments are:")
for i in sys.argv:
    print i
print "\nThe PYTHONPATH is", sys.path, "\n"
```

The command line arguments are:

D:/Lecture/Python/src/moduleSys.py

The PYTHONPATH is ['D:\\Lecture\\Python\\src', 'D:\\Lecture\\Python',  
'C:\\Python27\\python27.zip', 'C:\\Python27\\DLLs', 'C:\\Python27\\lib', 'C:\\Python27\\lib\\plat-  
win', 'C:\\Python27\\lib\\lib-tk', 'C:\\Python27', 'C:\\Python27\\lib\\site-packages']

# 모듈 import

수학함수는 math 모듈에 있다.

```
import math  
print math.sqrt(16)
```

파이썬에서 모듈은 확장자가 .py인 파일로 모듈을 저장하고 이를 import 하는 형식으로 사용한다.  
아래는 myModule.py 의 내용이고 그 다음은 myModule.py 를 import 해서 실행하는 예이다.

12\_mymodule.py

```
def sayHi():  
    print "Hi, this is mymodule speaking"  
__version__ = "0.1"
```

```
import myModule  
myModule.sayHi()
```

모듈파일에는 여러 함수를 저장할 수 있다.



# list

아래는 리스트의 사용법을 보여주는 예이다.

## 13\_listTest.py

```
# This is my shopping list
shoplist = ['apple', 'mango', 'carrot', 'banana']
print 'I have', len(shoplist), 'items to purchase.'
print 'These items are:',
for item in shoplist:
    print item,
print '\nI also have to buy rice.'
shoplist.append('rice')
print 'My shopping list is now', shoplist
print 'I will sort my list now'
shoplist.sort()
print 'Sorted shopping list is', shoplist
print 'The first item I will buy is', shoplist[0]
olditem = shoplist[0]
del shoplist[0]
print 'I bought the', olditem
print 'My shopping list is now', shoplist
```

```
I have 4 items to purchase.
These items are: apple mango carrot banana
I also have to buy rice.
My shopping list is now ['apple', 'mango', 'carrot', 'banana', 'rice']
I will sort my list now
Sorted shopping list is ['apple', 'banana', 'carrot', 'mango', 'rice']
The first item I will buy is apple
I bought the apple
My shopping list is now ['banana', 'carrot', 'mango', 'rice']
```

# tuple

아래는 튜플 사용법을 보여주는 예이다. 튜플은 리스트와 다르게 () 안에 정의한다.

튜플은 값의 수정, 삭제가 불가능하다는 것이 리스트와는 다르다.

프로그램에서 데이터를 수정할 필요성이 있다면 튜플을 사용할 수 없다. 그러므로, 튜플은 인덱스가 중요한 정보가 된다.

## 14\_tupleTest.py

```
zoo = ( ' python ', ' elephant ', ' penguin ' )  
print ' Number of animals in the zoo is ', len(zoo)  
new_zoo = ' monkey ', ' camel ', zoo  
print ' Number of cages in the new zoo is ', len(new_zoo)  
print ' All animals in new zoo are ', new_zoo  
print ' Animals brought from old zoo are ', new_zoo[2]  
print ' Last animal brought from old zoo is ', new_zoo[2][2]  
print 'Number of animals in the new zoo is', \  
len(new_zoo)-1+len(new_zoo[2])
```

```
Number of animals in the zoo is 3  
Number of cages in the new zoo is 3  
All animals in new zoo are ('monkey', 'camel', ('python', 'elephant', 'penguin'))  
Animals brought from old zoo are ('python', 'elephant', 'penguin')  
Last animal brought from old zoo is penguin  
Number of animals in the new zoo is 5
```

크기가 1인 튜플을 정의하려면 singleton = (2 , ) 와 같이 정의해야 한다.

# dictionary

딕셔너리는 사전처럼 단어:뜻의 형태로 표현되는 자료구조 {} 안에 정의한다. 아래와 같은 방식으로 추가 삭제가 가능하다.

15\_dicTest.py

```
ab = { 'Swaroop' : 'swaroop@swaroopch.com',  
       'Larry' : 'larry@wall.org',  
       'Matsumoto' : 'matz@ruby-lang.org',  
       'Spammer' : 'spammer@hotmail.com'  
}  
print "Swaroop's address is", ab['Swaroop']  
# Deleting a key-value pair  
del ab['Spammer']  
print '\nThere are {} contacts in the address-book\n'.format(len(ab))  
for name, address in ab.items():  
    print 'Contact {} at {}'.format(name, address)  
# Adding a key-value pair  
ab['Guido'] = 'guido@python.org'  
if 'Guido' in ab:  
    print "\nGuido's address is", ab['Guido']
```

```
Swaroop's address is swaroop@swaroopch.com  
There are 3 contacts in the address-book  
Contact Swaroop at swaroop@swaroopch.com  
Contact Matsumoto at matz@ruby-lang.org  
Contact Larry at larry@wall.org  
Guido's address is guido@python.org
```

# 함수(tuple 반환)

아래는 평균과 표준편차를 구하는 함수를 만들어 평균과 표준편차 두 개의 값을 return하는 예이다.  
return문에서 ()를 사용하면 tuple로 반환되고, []사용하면 리스트로 반환된다.

16\_funcTuple.py

```
import math
def meanStd(x):
    sum=0.
    sum2=0.
    n=len(x)
    for value in x:
        sum+=value
        sum2+=value**2
    mean=sum/n
    std=math.sqrt((sum2 - n*mean**2)/(n-1))
    return (mean,std)

x=range(1,11)
mean=meanStd(x)[0]
std=meanStd(x)[1]
print mean, std
```

# sequence

sequence는 index를 통해 index 연산을 수행한다. 예를 들어, 리스트 안에 특정 원소를 지정할 수 있다.

## 17\_sequenceTest.py

```
shoplist = ['apple', 'mango', 'carrot', 'banana']
name = 'swaroop'
# Indexing or 'Subscription' operation #
print 'Item 0 is', shoplist[0]
print 'Item 1 is', shoplist[1]
print 'Item 2 is', shoplist[2]
print 'Item 3 is', shoplist[3]
print 'Item -1 is', shoplist[-1]
print 'Item -2 is', shoplist[-2]
print 'Character 0 is', name[0]
# Slicing on a list #
print 'Item 1 to 3 is', shoplist[1:3]
print 'Item 2 to end is', shoplist[2:]
print 'Item 1 to -1 is', shoplist[1:-1]
print 'Item start to end is', shoplist[:]
print 'Item all but step is 2', shoplist[::2], "\n"
# Slicing on a string #
print 'characters 1 to 3 is', name[1:3]
print 'characters 2 to end is', name[2:]
print 'characters 1 to -1 is', name[1:-1]
print 'characters start to end is', name[:]
```

```
Item 0 is apple
Item 1 is mango
Item 2 is carrot
Item 3 is banana
Item -1 is banana
Item -2 is carrot
Character 0 is s
Item 1 to 3 is ['mango', 'carrot']
Item 2 to end is ['carrot', 'banana']
Item 1 to -1 is ['mango', 'carrot']
Item start to end is ['apple', 'mango', 'carrot',
'banana']
Item all but step is 2 ['apple', 'carrot']
characters 1 to 3 is wa
characters 2 to end is aroop
characters 1 to -1 is waroo
characters start to end is swaroop
```

-1은 뒤에서 첫번째, 1:3의 의미는 첫번째에서 3번째 원소 바로 직전까지라는 의미이다.

집합은 원소의 순서가 없고 중복을 표현할 수 없다.

하지만, 집합간의 합집합, 교집합, 차집합 등의 연산과 집합에 원소가 있는지 등을 판단할 수 있는 기능을 가지고 있다.

18\_setTest.py

```
bri = set(['brazil', 'russia', 'india'])
print 'india' in bri
print 'usa' in bri
bric = bri.copy()
bric.add('china')
print bric.issuperset(bri)
bri.remove('russia')
print bri & bric # OR bri.intersection(bric)
```

```
True
False
True
set(['brazil', 'india'])
```

# reference

아래의 프로그램에서 `mylist=shoplist`의 원소를 모두 받는 것이 아니고 주소값을 받는다. 그러므로, `shoplist`와 `mylist`는 주소값을 저장하고 있고 이 주소값은 서로 같다. `mylist=shoplist[:]`는 원소의 값을 물리적으로 복사하는 것이다.

## 19\_referenceTest.py

```
print 'Simple Assignment'
shoplist = ['apple', 'mango', 'carrot', 'banana']
# mylist is just another name pointing to the same object!
mylist = shoplist
# I purchased the first item, so I remove it from the list
del shoplist[0]
print 'shoplist is', shoplist
print 'mylist is', mylist
# Notice that both shoplist and mylist both print
# the same list without the 'apple' confirming that
# they point to the same object
print 'Copy by making a full slice'
# Make a copy by doing a full slice
mylist = shoplist[:]
# Remove first item
del mylist[0]
print 'shoplist is', shoplist
print 'mylist is', mylist
# Notice that now the two lists are different
```

```
Simple Assignment
shoplist is ['mango', 'carrot', 'banana']
mylist is ['mango', 'carrot', 'banana']
Copy by making a full slice
shoplist is ['mango', 'carrot', 'banana']
mylist is ['carrot', 'banana']
```

# string

"Life is too short, You need Python" "a" "123" 과 같이 "" 사이에 있는 문자가 스티링이다.

food = "Python's favorite food is perl" 도 올바른 표현이다. 문자열 안에 Escape Code \n, \t 등을 삽입할 수 있다.

두 개의 문자열을 합치고 할 때에는 + 연산자를 사용할 수 있다.

문자열은 sequence 연산이 가능하다. 즉, a = "Life is too short, You need Python" 에서

a[0]='L' 이고, a[3]='e', a[-1]='n', a[0:4]='Life' 가 가능하다. 문자열을 프린트 하는 방법은 아래와 같다.

```
a = "Life is too short, You need Python"
print a[0], a[3], a[-1], a[0:4]
weight=80
print"weight=", format(weight,"d")
print"weight= {0}".format(weight)
print "{0} {1} {2}".format(12,34,56)
print "{} / {} = {:.2f}".format(5,2,5/2.)
```



# string

아래는 String 객체에 대한 다양한 메소드를 보여주고 있다.

## 20\_stringTest.py

```
# This is a string object
name = 'Swaroop'
if name.startswith('Swa'):
    print 'Yes, the string starts with "Swa"'
if 'a' in name:
    print 'Yes, it contains the string "a"'
if name.find('war') != -1:
    print 'Yes, it contains the string "war"'
delimiter = '_*_'
mylist = ['Brazil', 'Russia', 'India', 'China']
print delimiter.join(mylist)
```

```
Yes, the string starts with "Swa"
Yes, it contains the string "a"
Yes, it contains the string "war"
Brazil_*_Russia_*_India_*_China
```

# Algorithm Training Part 2

1. 200 미만의 피보나치 수열 구하기
2. 피보나치 수열에 의해 사슴수 별 치킨 수 추정하기
3.  $n$ 과  $r$ 을 입력받아  $nCr$  계산하기
4. 수치적분/몬테칼로적분하기
5. Caesar 암호화, 복호화(ascii 코드표 이용)

# 객체지향프로그래밍

객체지향 프로그래밍은 우리가 프로그램 하고자 하는 대상을 몇 개의 객체로 표현하고 이를 객체간의 관계를 만들어가는 방식의 프로그래밍 방법이다. 클래스는 객체의 일반화된 형태로 생성자, 필드, 메소드로 이루어져 있다. 객체는 클래스에서 생성자를 통해 초기 모양으로 생성하고 필드를 수정해가면서 메소드를 사용할 수 있다.

아래의 프로그램은 Person이라는 클래스를 만들었다. 이 클래스는 say\_hi()라는 메소드를 가지고 있다.

p=Person() 은 Person 클래스에서 인스턴스를 생성해 p라는 객체에 저장하라는 명령이고, p.sayhi()는 p 객체의 메소드를 실행하는 명령이다.

우측은 Swaroop라는 이름을 가진 사람을 생성하는 예이다. self는 클래스를 나타내는 지시어이다.

\_\_init\_\_ 메소드는 객체가 생성될 때, 자동으로 실행되는 메소드이고, name은 메소드 내부의 Local 변수이다.

## 21\_classTest1.py

```
class Person:
    def say_hi(self):
        print('Hello, how are you?')
p = Person()
p.say_hi()
```

## 22\_classExercise.py

```
class Person:
    def __init__(self, name):
        self.name = name
    def say_hi(self):
        print 'Hello, my name is', self.name

p = Person('Swaroop')
p.say_hi()
```

# 클래스변수와 객체변수

population은 클래스 변수로 Robot에 소속되어 있다. R2-D2, C-3PO는 droid1, droid2 이름의 객체로 각각 객체 변수 self.name을 가지고 있다. @classmethod 명령으로 how\_many()가 클래스메소드임을 선언한다. 결과는 아래와 같다.

```
(Initializing R2-D2)
Greetings, my masters call me R2-D2.
We have 1 robots.
(Initializing C-3PO)
Greetings, my masters call me C-3PO.
We have 2 robots.
Robots can do some work here.
Robots have finished their work. So
let's destroy them.
R2-D2 is being destroyed!
There are still 1 robots working.
C-3PO is being destroyed!
C-3PO was the last one.
We have 0 robots.
```

```
class Robot:
    population = 0
    def __init__(self, name):
        self.name = name
        print "(Initializing {})".format(self.name)
        Robot.population += 1
    def die(self):
        print "{} is being destroyed!".format(self.name)
        Robot.population -= 1
        if Robot.population == 0:
            print "{} was the last one.".format(self.name)
        else:
            print "There are still {:d} robots working.".format(Robot.population)
    def say_hi(self):
        print "Greetings, my masters call me {}.".format(self.name)
    @classmethod
    def how_many(cls):
        print "We have {:d} robots.".format(cls.population)
# class end
droid1 = Robot("R2-D2")
droid1.say_hi()
Robot.how_many() # Call class method
droid2 = Robot("C-3PO")
droid2.say_hi()
Robot.how_many()
print "\nRobots can do some work here.\n"
print "Robots have finished their work. So let's destroy them."
droid1.die()
droid2.die()
Robot.how_many()
```

23\_robotClass.py

# 상속

상속은 슈퍼클래스와 서브클래스의 개념으로 서브클래스는 슈퍼클래스로부터 상속받아 만든다. 이때, 서브클래스는 슈퍼클래스의 속성과 메소드를 그대로 사용할 수 있다.

SchoolMember는 슈퍼클래스이고, Teacher, Student는 서브클래스이다. SchoolMember에 정의된 이름과 나이 그리고 tell 메소드는 서브 클래스에서 상속되어 사용됨을 알 수 있다.

(Initialized Teacher: Mrs. Shrividya)

(Initialized Student: Swaroop)

Name:"Mrs. Shrividya" Age:"40" Salary: "30000"

Name:"Swaroop" Age:"25" Marks: "75"

## 24\_subclassTest.py

```
class SchoolMember:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def tell(self):
        print 'Name: "{}" Age: "{}"'.format(self.name, self.age),

class Teacher(SchoolMember):
    def __init__(self, name, age, salary):
        SchoolMember.__init__(self, name, age)
        self.salary = salary
        print '(Initialized Teacher: {})' .format(self.name)
    def tell(self):
        SchoolMember.tell(self)
        print 'Salary: "{:d}"' .format(self.salary)

class Student(SchoolMember):
    def __init__(self, name, age, marks):
        SchoolMember.__init__(self, name, age)
        self.marks = marks
        print '(Initialized Student: {})' .format(self.name)
    def tell(self):
        SchoolMember.tell(self)
        print 'Marks: "{:d}"' .format(self.marks)

t = Teacher('Mrs. Shrividya', 40, 30000)
s = Student('Swaroop', 25, 75)
print
members = [t, s]
for member in members:
    member.tell()
```

# 객체지향 연습-1

좌측 프로그램을 객체지향 방식으로 수정해보자. 클래스 이름은 fibonacci이고 생성인자는 n을 가진다.  
수열을 만들기 위한 메소드로 getSeries() 를 만들었다.

```
# Fibonacci Series

numEnd=100
f=[]
f.append(0)
f.append(1)

i=2
fVal=0
while(fVal<=100):
    fVal=f[i-2]+f[i-1]
    f.append(fVal)
    i+=1

print f
```

```
class fibonacci:
    def __init__(self, n):
        self.n = n
    def getSeries(self):
        f=[]
        f.append(0)
        f.append(1)
        i=2
        fVal=0
        while(fVal<=self.n):
            fVal=f[i-2]+f[i-1]
            f.append(fVal)
            i+=1
        return f
# 100 이하의 피보나치수열을 만들어서 fSeries에 저장한다.
fSeries= fibonacci(100)
print fSeries.getSeries()
```

# 객체지향 연습-2

피보나치수열과 최소제곱법을 수행하는 클래스를 만들어 사람수에 따른 치킨 수를 구해보자.

## 25\_mylib.py

```
class lm:
```

```
    """ lm 클래스는 n개의 원소를 갖는 두개의 리스트쌍에 대해  
    최소제곱법의 해인 a, b를 구해 리스트로 반환한다,
```

```
    Example:
```

```
        lmRes=lm(x, y)
```

```
        a=lmRes.getCoef()[0]
```

```
        b=lmRes.getCoef()[1]
```

```
    """
```

```
    def __init__(self,x,y):
```

```
        self.x=x
```

```
        self.y=y
```

```
    def getCoef(self):
```

```
        n=len(self.x)
```

```
        sumx=0.
```

```
        sumy=0.
```

```
        sum2=0.
```

```
        sumxy=0.
```

```
        for i in range(0,n):
```

```
            sumx+=self.x[i]
```

```
            sumy+=self.y[i]
```

```
            sum2+=self.x[i]*self.x[i]
```

```
            sumxy+=self.x[i]*self.y[i]
```

```
        xbar=sumx/n
```

```
        ybar=sumy/n
```

```
        b=(sumxy-n*xbar*ybar)/(sum2-n*xbar*xbar)
```

```
        a=ybar-b*xbar
```

```
        return [a,b]
```

# 객체지향 연습-2

Fibonacci class와 lm class를 myLib.py로 저장하고 이를 import해서 사람수에 따른 치킨수를 구하는 코드이다.

## 26\_chickenTest.py

```
import myLib
print myLib.fibonacci.__doc__
f=myLib.fibonacci(100)
cntMan= f.getSeries()
cntChicken=cntMan[1:]
cntMan=cntMan[0:-1]

print myLib.lm.__doc__
lmRes=myLib.lm(cntMan, cntChicken)
a=lmRes.getCoef()[0]
b=lmRes.getCoef()[1]

nPeople=int(raw_input('Enter a number of peoples who want to eat chicken : '))
nChicken=a+b*nPeople

print "Count of Chickens = ", nChicken
```



# 입출력(input)

`raw_input`은 사용자의 입력을 받는 함수로 입력받은 값이 좌측에 저장된다.

`reverse` 함수는 입력된 텍스트를 거꾸로 배열하는 함수이다. `palindrome`은 “기러기“ 처럼 텍스트와 거꾸로 정렬된 텍스트가 같은 문자열을 의미한다.

27\_palindromeTest.py

```
def reverse(text):  
    return text[::-1]  
  
def is_palindrome(text):  
    return text == reverse(text)  
  
something = raw_input("Enter text: ")  
  
if is_palindrome(something):  
    print "Yes, it is a palindrome"  
else:  
    print "No, it is not a palindrome"
```

숙제:

“Rise to vote, sir.” 은 `palindrome`이지만  
위 예제 프로그램은 이것은 `palindrome`이  
아니라고 판단할 것이다.

위 프로그램을 고쳐서 이러한 문자열들을  
`palindrome` 으로 인식할 수 있는 프로그램으로  
수정해 보자.

# 입출력(file I/O)

이 예제는 파일을 쓰기모드로 Open 하고, write 메소드로 텍스트를 저장한 다음 다시 읽기 모드로 읽어보는 예이다.

28\_ioTest.py

```
poem = '''\
Programming is fun
When the work is done
if you wanna make your work also fun:
use Python!
'''

# Open for 'w'riting
f = open('d:/poem.txt', 'w')
# Write text to file
f.write(poem)
# Close the file
f.close()

# If no mode is specified,
# 'r'ead mode is assumed by default
f = open('d:/poem.txt')
while True:
    line = f.readline()
    # Zero length indicates EOF
    if len(line) == 0:
        break
    print line,
f.close()
```

# 입출력(pickle)

pickle 모듈은 파이썬 객체를 파일로 저장했다가 다시 불러 사용할 수 있도록 도와준다.

import 명령으로 pickle 모듈을 불러오고 리스트를 저장할 파일을 정한 다음 'wb' 모드로 엽니다.

여기서, wb는 write binary를 나타낸다. 이후, pickle.dump 메소드로 리스트를 파일에 write한다.

## 29\_pickleTest.py

```
import pickle
# The name of the file where we will store the object
shoplistfile = 'd:/shoplist.data'
# The list of things to buy
shoplist = ['apple', 'mango', 'carrot']
# Write to the file
f = open(shoplistfile, 'wb')
# Dump the object to a file
pickle.dump(shoplist, f)
f.close()
# Destroy the shoplist variable
del shoplist
# Read back from the storage
f = open(shoplistfile, 'rb')
# Load the object from the file
storedlist = pickle.load(f)
print storedlist
```

# 입출력(한글사용)

컴퓨터에서 영어 이외의 문자를 표현하기 위해서는 Encoding을 해야 한다.

영문자는 1바이트  $2^8$ 개에 모든 문자의 표현이 가능하지만, 1 바이트로 표현이 불가능한 언어도 많이 있기 때문에 전 세계적으로 UniCode를 사용한다. UniCode의 종류로 utf-8은 ANSI 부분은 똑같이 1바이트, 유럽문자는 2바이트, 아시아문자는 3바이트 등으로 가변 표기하는 방법이다.

ANSI를 개조해서 영문은 1바이트, 한글은 2바이트로 변형한 것이 euc-kr이다. 문제는 윈도우에서는 euc-kr을 사용하고 리눅스, 유닉스, 오픈소스 환경에서는 utf-8을 사용해 윈도우에서 오픈소스 프로그램을 활용할 때, utf-8로 변환하는 것이 필요한 경우가 많다.

파이썬에서 한글로 파일을 쓰고, 읽기 위해서는 맨 앞줄에 `# encoding=utf-8` 으로 선언한 후, u” “ 형식으로 한글을 사용하면 된다.

## 30\_unicodeTest.py

```
# encoding=utf-8
import io
f = io.open("d:/abc.txt", "w", encoding="utf-8")
f.write(u"한글 테스트")
f.close()
text = io.open("d:/abc.txt", encoding="utf-8").read()
print text
```

# Exception

예외처리는 프로그램에서 있어야 할 파일이 없을 때, 데이터베이스가 꺼져 데이터를 불러올 수 없을 때 등 예기치 않은 오류가 발생할 때, 시스템 오류 대신 사용자 프로그램으로 오류를 처리하는 방식이다.

아래의 코드는 입력에서 ctrl+D를 누르면 EOFError, ctrl+c를 누르면 KeyboardInterrupt 오류가 발생하는데 그 때 수행해야할 print 문을 기술한 예이다.

31\_exceptionTest.py

```
try:
    text = raw_input('Enter something --> ')
except EOFError:
    print 'Why did you do an EOF on me?'
except KeyboardInterrupt:
    print 'You cancelled the operation.'
else:
    print 'You entered {}'.format(text)
```

# DB Access

아래는 외부에 있는 MySQL DB에 접속해 SQL를 수행하는 예제이다.

MySQL 사용을 위해서는 MySQL-python-1.2.3.win-amd64-py2.7.exe 를 인터넷에서 받아 설치해야 한다.

만약, 한글이 있다면 .connect 에 charset='utf8' 옵션을 줘야 한다.

```
import MySQLdb
db=MySQLdb.connect(host="165.246.38.237",user="scscStudent",passwd="1234",db="Iris",port=8888)
cur=db.cursor()
cur.execute("select * from IrisData")
for value in cur:
    print value[0], value[1], value[2],value[3],value[4],value[5]
```

Iris DB는 setosa, versicolor, virginica 세 종류의 꽃에 대한 SepalLength, SepalWidth, PetalLength, PetalWidth를 측정한 자료이다. 꽃의 종류는 마지막 컬럼에 있다.

꽃의 종류에 따라 4가지 Feature variable의 특징이 다른 점을 이용하여 새로운 꽃에 대한 종을 판별하는 것이 거의 가능하다.

위의 프로그램 코드를 활용하여 Species별 SepalLength, SepalWidth, PetalLength, PetalWidth의 평균과 표준편차를 구하고 표준화하는 프로그램을 작성하시오.(객체지향프로그래밍으로 구현하시오)

아래는 openweathermap.org에 접속해 서울의 날씨를 가져오는 예이다.

urllib 모듈을 통해 url을 오픈한다. urllib는 파이썬 내장 모듈이다.

아래의 결과에서 원하는 값을 아래와 같이 json 모듈을 이용하여 parsing해서 사용하면 된다

32\_urlTest.py

```
import urllib, json
f=urllib.urlopen("http://api.openweathermap.org/data/2.5/weather?id=1835848&APPID=2ae88b3ec4b4537d15165fca377a3ca9")
s=json.loads(f.read())

print s
print s['name'], s['coord']['lat'], s['coord']['lon'], s['main']['temp'], s['main']['humidity'], s['wind']['speed']
```

```
{"coord":{"lon":126.98,"lat":37.57},"weather":[{"id":801,"main":"Clouds","description":"few clouds","icon":"02d"}],"base":"cmc stations","main":{"temp":303.03,"pressure":1000,"humidity":40,"temp_min":301.15,"temp_max":306.15},"wind":{"speed":2.1,"deg":330},"rain":{"clouds":{"all":20},"dt":1468477200,"sys":{"type":3,"id":8519,"message":0.0028,"country":"KR","sunrise":1468441332,"sunset":1468493606},"id":1835848,"name":"Seoul","cod":200}
```

Seoul 37.57 126.98 297.32 71 2.11

rPy2 패키지를 사용하면 파이썬에서 R을 사용할 수 있다. 설치과정 및 예제는 아래와 같다.

1. <http://www.lfd.uci.edu/~gohlke/pythonlibs/#rpy2> 에서 rPy2 패키지를 다운받는다.

(rpy2-2.7.8-cp27-none-win\_amd64.whl)

2. dos 창에서 pip install rpy2-2.7.8-cp27-none-win\_amd64.whl 명령을 통해 rPy2 패키지를 설치한다.

이 명령을 위해서는 pip.exe가 있는 위치를 윈도우 path에 추가해야한다.( c:/anaconda2/scripts)

2. 윈도우 환경변수 R\_HOME과 R\_USER를 설정한다.

(R\_HOME:C:\Program Files\R\R-3.2.2, R\_USER:swkim)

```
from rpy2.robjects import r
r('x <- rnorm(100)')
r('y <- x + rnorm(100,sd=0.5)')
r('plot(x,y)')
r('lmout <- lm(y~x)')
coef=r('lmout$coefficients')
print coef[0], coef[1]
```

```
from rpy2.robjects import r
x=range(1,10)
print x
r.assign('x',x)
r('x<-unlist(x)')
r('mu<-mean(x)')
r("setwd('d:/')")
r("save.image()")
mu=r('mu')
print mu
```



pyQT는 파이썬에서 GUI 프로그램을 도와주는 툴로 아나콘다에서는 이미 설치되어 있다.  
아나콘다를 사용하지 않는 경우, 아래의 주소에서 설치할 수 있다.

<https://riverbankcomputing.com/software/pyqt/download>

파이썬 버전과 컴퓨터 비트 수에 따라 적절한 버전을 받아야 한다.

GUI 프로그램의 개발 단계는 UI Design → Event Listener 정의로 이루어진다.

GUI는 Sequential Program과 다르게 사용자의 마우스 클릭, 키보드 입력 등 사용자의 행동을 이벤트로 정의하고 이벤트가 발생할 경우, 어떤 Action을 수행해야 하는지를 프로그래밍한다.

# HelloGUI

아래의 프로그램은 Hello World!를 출력하는 GUI 프로그램이다.

먼저, QtWidgets에 있는 QApplication, QWidget, QLabel을 import 하고, window()라는 함수를 정의한 다음 window()를 실행하는 방식이다. w는 윈도우를 나타내는 객체로 (100,100) 위치에 가로로 200, 세로로 50의 크기를 가진다.

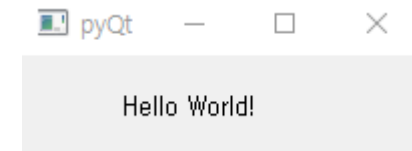
b는 QLabel이라는 위젯에 “Hello World”라고 쓰고 위치를 50,20 위치에 출력하는 예제이다.

/Anaconda2/Lib/site-packages/PyQt5/QtWidgets.pyd 파일은 C언어를 컴파일해서 만든 동적 라이브러리이다.

## 1\_HelloGUI.py

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel
def window():
    app = QApplication(sys.argv)
    w = QWidget()
    b = QLabel(w)
    b.setText("Hello World ~~~!")
    w.setGeometry(100, 100, 200, 50)
    b.move(50, 20)
    w.setWindowTitle("pyQt")
    w.show()
    sys.exit(app.exec_())

window()
```



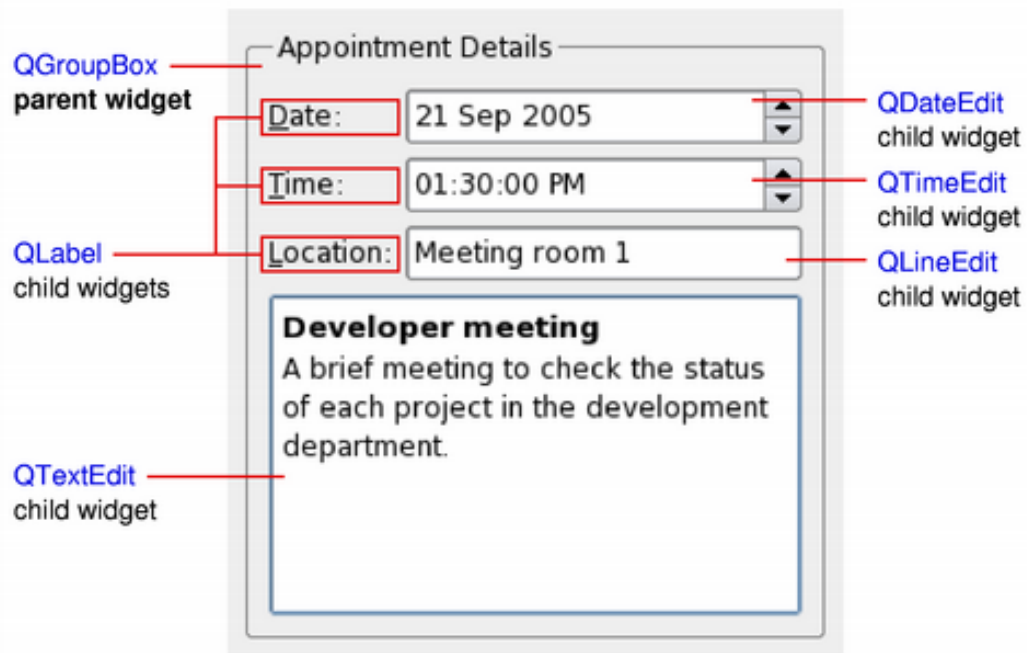
# Widget

GUI 프로그램은 윈도우안에 아래와 같이 위젯을 배치하고 이들 간의 관계를 만들어 주는 방식이다.

위젯은 종류에 따라 QLabel, QDateTime, QTextEdit 등이 제공된다.

윈도우는 QMainWindow나 QDialog 클래스를 이용하여 객체로 만들 수 있다.

사실, 윈도우 역시 위젯이다. 위젯 중 최상위 위젯이다.



# QMainWindow

class MyWindow는 QMainWindow를 상속받아 만든 객체이고 MyWindow를 생성할 때, 슈퍼클래스인 QMainWindow의 생성자를 실행한다.

## 2\_QMainWindow.py

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication

class MyWindow(QMainWindow):
    def __init__(self):
        super(MyWindow, self).__init__()
        self.setWindowTitle("QMainWindow")
        self.setGeometry(300, 300, 300, 400)

app = QApplication(sys.argv)
myWindow = MyWindow()
myWindow.show()
app.exec_()
```

# QMainWindowEvent

아래는 이전 프로그램에 btn1이라는 QPushButton을 생성하고 QMessageBox를 이용해 “clicked”라는 메시지를 출력하는 예제이다.

3\_QMainWindowEvent.py

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication

class MyWindow(QMainWindow):
    def __init__(self):
        super(MyWindow, self).__init__()
        self.setWindowTitle(" QMainWindow Event")
        self.setGeometry(300, 300, 300, 400)
        btn1 = QPushButton(" Click me ", self)
        btn1.move(20, 20)
        btn1.clicked.connect(self.btn1_clicked)
    def btn1_clicked(self):
        QMessageBox.about(self, "message", "clicked")

app = QApplication(sys.argv)
myWindow = MyWindow()
myWindow.show()
app.exec_()
```

# QDialogEvent

이 프로그램은 다이얼로그 윈도우(win)에 b1, b2의 QPushButton을 만들고 b1.clicked.connect(b1\_clicked) 명령으로 b1이 클릭되었을 때, b1\_clicked() 함수를 호출하도록 하는 프로그램이다.

## 4\_buttonClick.py

```
import sys
from PyQt5.QtWidgets import QApplication, QDialog, QPushButton
def window():
    app = QApplication(sys.argv)
    win = QDialog()
    b1= QPushButton(win)
    b1.setText("Button1")
    b1.move(50,20)
    b1.clicked.connect(b1_clicked)
    b2=QPushButton(win)
    b2.setText("Button2")
    b2.move(50,50)
    win.setGeometry(100,100,200,100)
    win.setWindowTitle("PyQt")
    win.show()
    sys.exit(app.exec_())
```

```
def b1_clicked():
    print "Button 1 clicked"
def b2_clicked():
    print "Button 2 clicked"

window()
```

# ui Designer

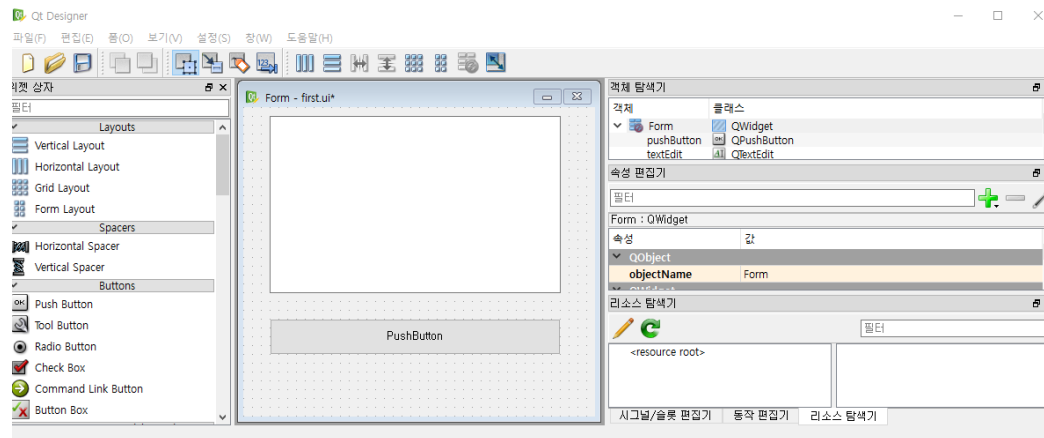
GUI 프로그래밍에서 화면을 디자인하는 것을 코드로 할 수도 있지만, 디자이너를 사용하면 좀 더 직관적으로 화면에서 파워포인트 작업 하듯이 수행할 수도 있다.

디자이너는 Anaconda2\Lib\site-packages\PyQt4와 같이 자신의 파이썬 설치 폴더에 designer.exe 파일이 있는데 이를 실행하면 QT 디자이너가 실행된다. QT 디자이너를 사용해 자신이 원하는 프레임을 디자인하고 저장하면 확장자가 ui인 파일이 생성된다. ui 파일은 xml 형식의 자료이다.

확장자가 ui 인 디자인 파일이 완성되면 \*.ui를 파이썬 코드로 바꿔서 사용하는 방법과 \*.ui를 파이썬 프로그램에서 읽어서 사용하는 방법이 있다.

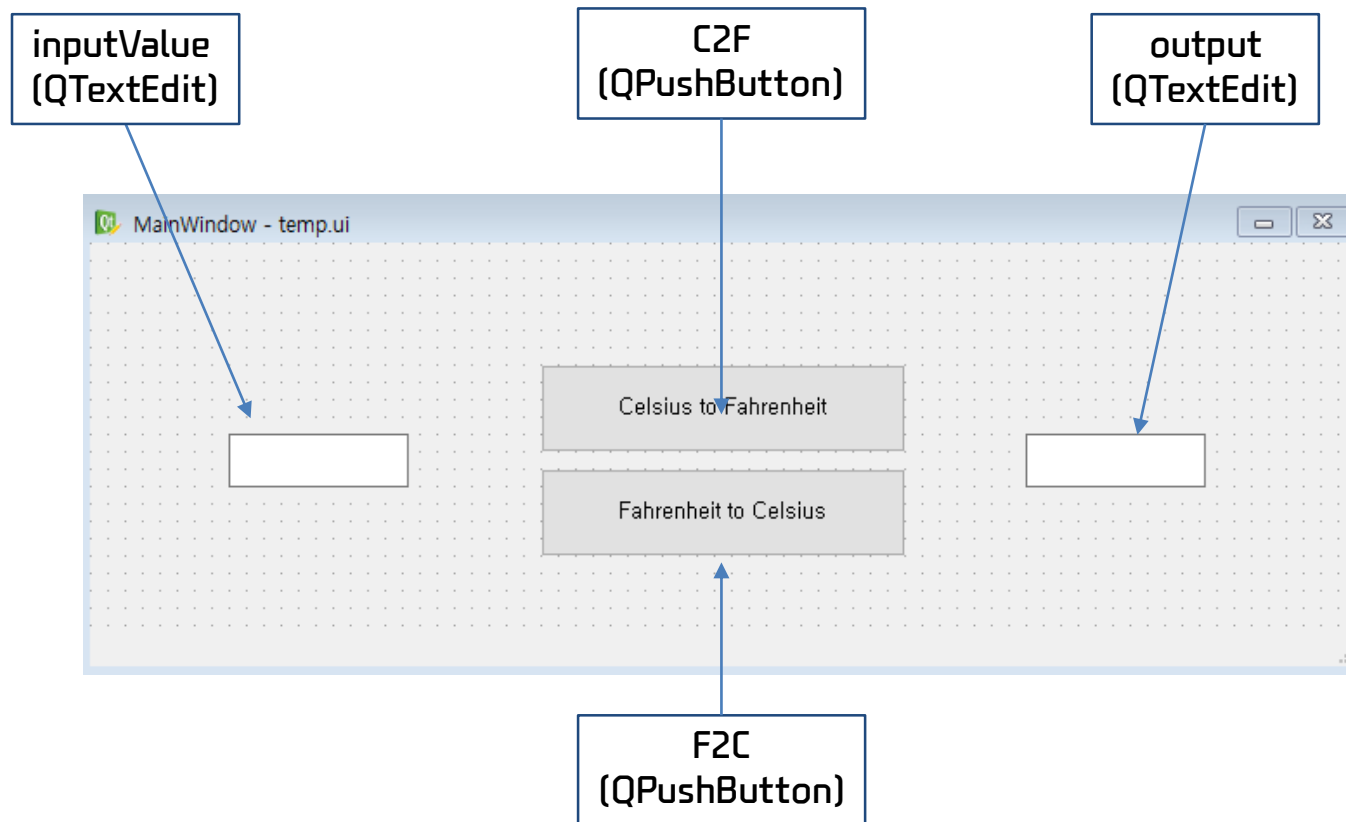
리눅스에서는 `sudo apt-get install qt4-designer` 명령으로 바이너리를 설치하면 된다.

바이너리는 `/usr/lib/x86_64-linux-gnu/qt4/bin/designer` 이다.



# ui Designer

이제 ui 디자이너를 사용하여 아래와 같이 섭씨를 화씨로, 화씨를 섭씨로 변환하는 프로그램을 만들어 보자.





# ui Designer

5\_temp.py

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.uic import loadUiType

form_class=loadUiType("temp.ui")[0]
class MyWindowClass(QMainWindow, form_class):
    def __init__(self, parent=None):
        QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.C2F.clicked.connect(self.C2F_clicked)
        self.F2C.clicked.connect(self.F2C_clicked)
    def C2F_clicked(self):
        cel=float(self.inputValue.toPlainText())
        fahr=cel*9/5.0+32
        self.output.setText(str(fahr))
    def F2C_clicked(self):
        fahr=float(self.inputValue.toPlainText())
        cel=(fahr-32)/9.0*5
        self.output.setText(str(cel))
app=QApplication(sys.argv)
myWindow=MyWindowClass(None)
myWindow.show()
app.exec_()
```

uic 모듈은 여러 기능을 하는데 대표적으로 Qt Designer를 통해서 저장했던 \*.ui 파일들을 파이썬의 클래스에 불러들이는 기능이다. 리턴값은 두 개가 있는데 ui에 대한 레이아웃정보는 [0]번째에 있다. \_\_init\_\_ 메소드에서 C2F, F2C가 클릭되면 각각 C2F\_clicked, F2C\_clicked 를 실행하도록 정의하고, 각각의 함수에서 이벤트 발생 시, 해야 할 일을 프로그램 하였다.

# SimpleCalc

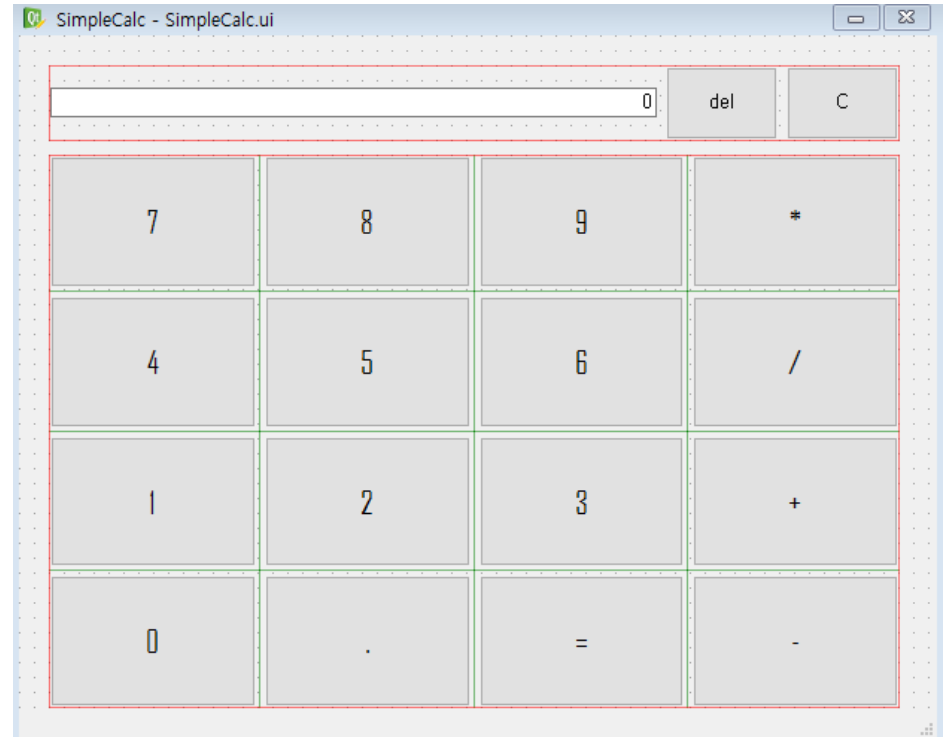
qt Designer를 이용해서 우측과 같은 계산기 ui를 만들어 SimpleCalc.ui로 저장하자.

각 숫자는 b0, b1, ..., b9으로  
del은 bDel, C는 bClear, 결과창은 QLineEdit 클래스의  
result 객체이고 \*,/,+,-는 각각 bMult, bDivide,  
bPlus, bMinus, .은 bDot의 이름을  
부여하였다.

상단은 horizontal Layout으로 정렬하고,  
하단은 grid Layout으로 정렬하였다.

이프로그램은 간단한 계산기로 12+3 과 같이  
좌측, 우측에 숫자와 가운데 부분에 연산자가  
있는 계산만 수행이 가능하다.

이 계산기에서 Simple을 제거하는 것은 여러분의 몫이다.



# SimpleCalc

6\_simpleCalc.py

```
import sys
from PyQt5.QtWidgets import QMainWindow, QApplication
from PyQt5.uic import loadUiType
form_class=loadUiType("SimpleCalc.ui")[0] # SimpleCalc.ui에서 객체 정보를 읽어온다.
# QMainWindow 클래스에서 상속받은 CalcClass 생성
class CalcClass(QMainWindow, form_class):
    def __init__(self, parent=None):
        QMainWindow.__init__(self,parent) # QMainWindow 클래스 객체 정보를 가져옴
        self.setupUi(self)
        nums = [self.b0, self.b1, self.b2, self.b3, self.b4, self.b5, self.b6, self.b7, self.b8, self.b9]
        # b0 ~ b9은 CalcClass global 객체이므로 self.b0로 표현
        for number in nums:
            number.clicked.connect(self.Nums) # 숫자가 입력되면 Nums 함수 실행
        self.bDel.clicked.connect(self.bDelClick)
        self.bClear.clicked.connect(self.bClearClick)
        self.bRun.clicked.connect(self.bRunClick)
        self.bPlus.clicked.connect(self.bPlusClick)
        self.bMinus.clicked.connect(self.bMinusClick)
        self.bMult.clicked.connect(self.bMultClick)
        self.bDivide.clicked.connect(self.bDivideClick)
        self.bDot.clicked.connect(self.bDotClick)
```

# SimpleCalc

```
def nums(self):
    global num
    sender = self.sender() # SimpleCalc 클래스의 sender 메소드를 통해 입력값을 전달받는다.
    newNum = int(sender.text())
    setNum = str(newNum)
    if self.result.text() == "0": # 입력값을 result 객체에 전달한다.(현재값이 "0"이면 0을 무시)
        self.result.setText(setNum)
    else:
        self.result.setText(self.result.text()+setNum)

def bDotClick(self):
    self.result.setText(self.result.text()+".")
def bPlusClick(self):
    self.result.setText(self.result.text()+"+")
def bMinusClick(self):
    self.result.setText(self.result.text()+"-")
def bDivideClick(self):
    self.result.setText(self.result.text()+"/")
def bMultClick(self):
    self.result.setText(self.result.text()+"*")
def bDelClick(self): # del이 클릭되면 result의 텍스트에서 마지막 부분을 잘라낸다.
    n=len(self.result.text())
    self.result.setText(self.result.text()[0:n-1])
def bClearClick(self):
    self.result.setText("0") # C가 클릭되면 result의 텍스트를 "0"으로 세팅
```

# SimpleCalc

```
def bRunClick(self):
    A=self.result.text()
    for i in range(0,len(A)):
        # result를 한문자씩 읽으면서 숫자가 아니면 연산자라는 가정하에 연산자의 위치를 찾는다.
        if str(A[i]) == ".": continue
        if str(A[i]).isdigit() == False:
            # 연산자를 중심으로 왼쪽은 leftNum, 오른쪽은 rightNum에 숫자를 문자로 받아 실수로 변환
            leftNum=float(str(A[0:i]))
            rightNum=float(str(A[i+1:len(A)]))
            if str(A[i]) == "+": self.result.setText(str(leftNum+rightNum))
            if str(A[i]) == "-": self.result.setText(str(leftNum-rightNum))
            if str(A[i]) == "*": self.result.setText(str(leftNum*rightNum))
            if str(A[i]) == "/": self.result.setText(str(leftNum/rightNum))

app=QApplication(sys.argv)
myWindow=CalcClass(None)
myWindow.show()
app.exec_()
```

# 실행파일 만들기

파이썬은 Interpreter 방식의 언어이기 때문에 소스코드를 하나하나 실행한다.

이렇게 개발된 파이썬 프로그램을 배포하기 위해서는 배포판을 만들어야 하는데 방법은 여러 가지가 있다.

여기에서는 pyInstaller를 사용해보기로 한다.

pyInstaller를 설치하기 위해 시작화면에 Anaconda 메뉴에서 Anaconda Prompt를 실행하고 아래와 같이 명령하면 pyInstaller가 설치된다.

```
> pip install pyinstaller
```

이제, 여러분이 실행파일을 만들고자 하는 py 파일이 존재하는 폴더로 이동해서 아래와 같이 명령한다.

```
> pyinstaller --onefile --noconsole SimpleCalc.py
```

위의 명령이 실행되면 py 파일이 존재하는 폴더 밑에 dist라는 폴더가 생성되고 그 곳에 SimpleCalc.exe가 생성된다.

SimpleCalc.ui 등 파일이 실행되는데 필요한 파일을 dist 폴더로 복사해주면 실행될 것이다.

배포는 dist 폴더 밑에 있는 파일을 배포하면 된다.

# 도전과제

이제, 여러분이 스스로 작품을 만들어볼 차례이다. 주제는 수도쿠이다.

수도쿠는 1~9까지의 숫자를 9\*9 행렬에 배열하는데 각 행과 열은 1~9까지의 숫자가 중복되어 나타날 수 없고 또 3\*3 행렬 9개에서도 1~9까지의 숫자가 중복되어 나타나지 않도록 숫자를 배열하는 게임이다.

프로그램이 시작되면 각 셀이  $1 - 0.7 = 0.3$ 의 확률로 빈칸이 만들어진다. 사용자는 이 빈칸에 스도쿠 규칙에 적합한 1~9까지의 숫자를 채우고 Finish 버튼을 누르면 게임이 종료되는 것이다. Shuffle 버튼은 초기배열 값을 난수를 통해 재 생성하는 버튼이다. Shuffle 버튼은 가운데 확률값을 가지고 빈칸을 만드므로 0.7 보다 큰 값을 가지면 빈칸의 수가 작아져 난이도가 쉬워지고, 작은 값을 가지면 빈칸의 수가 많아져 난이도가 증가한다.

여러분은 먼저, QT Designer를 사용하여 우측 UI를 만들고 Sudoku.ui라는 이름으로 저장한다.



# 도전과제

아래는 81개의 버튼 Label 값이다. 버튼값을 이렇게 만드는 이유는 정답이 존재하는 수도쿠 배열을 만들기 위한 작전으로 먼저, 수도쿠 룰에 적합한 배열을 만들고 그 배열의 값을 랜덤하게 섞어 문제를 만들어 내기 위함이다.

123456789  
456789123  
789123456  
231897564  
564231897  
897564231  
312645978  
645978312  
978312645

즉, 1~9사이의 수에서 9개의 수를 비복원 추출한다. 예: [5, 2, 3, 9, 7, 4, 8, 1, 6]

그 다음으로 위의 배열에서  $1 \rightarrow 5$ ,  $2 \rightarrow 2$ ,  $3 \rightarrow 3$ ,  $4 \rightarrow 9$  와 같은 방식으로 숫자를 바꾼 다음 0.3의 확률로 각 셀 중에 빈칸을 만들어 문제를 출제하는 것이다. Shuffle 버튼을 누를 때 마다 새로운 문제가 출제된다.



# 도전과제

사용자는 원하는 버튼을 마우스로 클릭하여 어떤 셀에 값을 입력할 것인지 알려주고 키보드로 숫자를 입력한다.

그러면 그 숫자는 붉은 색으로 입력된다. 이렇게 모든 셀을 다 입력하고 Finish 버튼을 누르면 프로그램은

각 열에 숫자가 1~9까지 유일한가? 각 행의 숫자가 1~9까지 유일한가? 3\*3 행렬의 값이 1~9까지 유일한가? 의 규칙을 검사하여 오류가 있는지 아니면 성공적으로 게임을 끝냈는지 결과를 출력하면 된다.



Sudoku

Shuffle

0.7

Finish

5	2	3	9	7	4	8	1	6
9	7	4	8	1	6	5	2	3
8	1	6	5	2	3	9	7	4
2	3	5	1	6	8	7	4	9
7	4	9	2	3	5	1	6	8
1	6	8	7	4	9	2	3	5
3	5	2	4	9	7	6	8	1
4	9	7	6	8	1	3	5	2
6	8	1	3	5	2	4	9	7

버튼을 클릭하고 1~9사이의 정수를 입력하세요. Finish를 누르면 채점 결과를 알려드립니다.

!!! ~~~Congratulation~~~ !!!

수고많았습니다.

이제 파이썬 프로그래머가

되어 보세요~~~