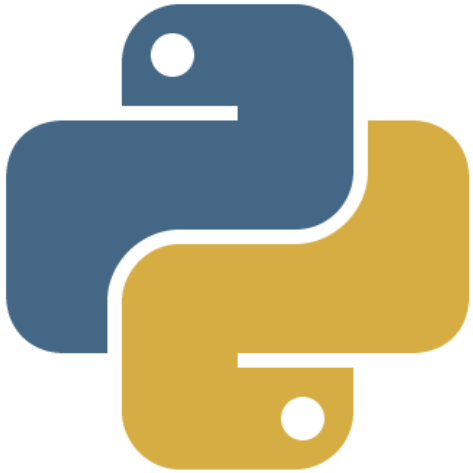
Programming in Python

03 – 프로그램 구문 구조, 파일 입출력



python

2016년 8월, 국민대학교 컴퓨터공학부

Python 예약어

예약어는 프로그램에서 변수 이름으로 사용할 수 없음

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

```
>>> a = 1
>>> repr(a)
'1'
>>> repr = 2
>>> repr
2
>>> del repr
>>> repr
<built-in function repr>
>>> repr(a)
'1'
>>>
```

몇몇 특수한 글자와 밑줄 또한 사용할 수 없음

& ! @ $ * ( ) ? : ; [ ] " < > ' ` | = { } ₩ / . 빈칸

들여쓰기 (Indentation)

Python 프로그램에서 들여쓰기는
단지 코드를 읽기 편하게 하기 위함이 아님

1. 몇 칸을 들여 쓰든 상관없지만,
   같은 수준의 코드 블록의 들여쓰기는 같아야 함
2. 탭과 스페이스는 서로 다른 들여쓰기로 간주됨

들여쓰기를 이용하여 프로그램의 구문 구조를 표현

Python 코드

```
class Lift():
    current_floor = 0

    def getFloor():
        return current_floor

    def moveToFloor(floor_number):
        current_floor = floor_number
```

semicolon
colon
curly braces
def
function return type
function argument type

비교: Java 코드

```
{
    private int current_floor = 0;
    public int getFloor()
    {
        return current_floor;
    }
    public void moveToFloor(int floor_number)
    {
        current_floor = floor_number;
    }
}
```

들여쓰기 (Indentation)

세미콜론 (;) 을 이용하여 한 행에 여러 문장을 쓸 수 있음

```
a = 1; b = 2; print(a + b)
```

하나의 문장을 여러 행에 걸쳐 쓸 수 있음 (괄호 등으로 이루어진 경우)

```
mlist = [111,
         222,
         333]
X = (A + B +
     C + D)
```

역슬래쉬 (\) 를 이용하여 한 문장을 두 행 이상에 걸쳐 쓸 수 있음 (좋은 습관인가?)

```
X = A + B \
    C + D
```

콜론 (:) 으로 끝나는 헤더 뒤, 같은 행에 블록 문장을 쓸 수 있음

```
if x > y: print(x)
```

주석 (Comments)

한 줄짜리 주석:
  # 로 시작하는 행은 주석으로 취급

여러 줄짜리 주석:
  ''' 로 시작하고 ''' 로 끝냄
  (주의: 들여쓰기 수준은 일치해야 함)

```
for i in range(5):
    print i
'''
    This is an example of a multi-line comment.
    Three small quotes are used.
'''
    print i + 1
```

#-*- coding: utf-8 -*- ???

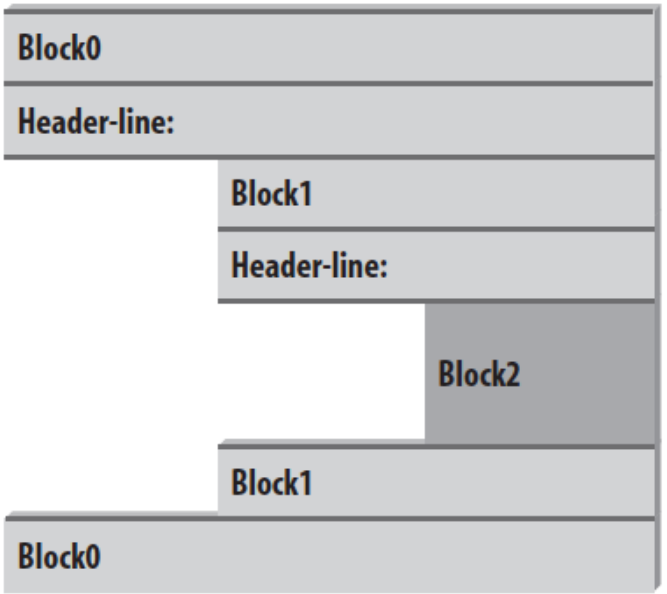
```
#-*- coding: utf-8 -*-

'''
    여기부터는 Lift class 입니다.
    영국에서처럼 우리나라 1층을 0층으로.
'''

class Lift():
        # 엘리베이터는 0층에서 시작
        current_floor = 0

        # 엘리베이터를 찾는 메서드
        def getFloor():
                return current_floor

        # 엘리베이터를 움직이는 메서드
        def moveToFloor(self, floor_number):
                self.current_floor = floor_number
```

조건문 (Conditional Statements)

```
if <test1>:
    <statements1>
elif <test2>:
    <statements2>
else:
    <statements3>
```

```
>>> if 1:
...     print('true')
...
true
>>> if not 1:
...     print('true')
... else:
...     print('false')
...
false
```

```
x = 1
if x:
    y = 2
    if y:
        print('block2')
    print('block1')
print('block0')
```



조건식 (Truth Tests)

조건식	의미
X and Y	X 와 Y 가 둘 다 참일 때에 참
X or Y	X 와 Y 중 한 쪽 이상이 참일 때에 참
not X	X 가 참이 아닐 때에 참

```
>>> 2 or 3                  # True 인 경우 or 의 왼쪽
2                           # object 를 취함
>>> 3 or 2
3
>>> [] or 3                 # False 인 경우에는 or 의 오른쪽
3                           # object 를 취함
>>> [] or {}
{}
>>> if []: print 'x'        # False
...
>>> if {}: print 'x'        # False
...
>>> if not []: print 'y'    # True
...
y
```

```
>>> 2 and 3                 # True 인 경우 and 의 오른쪽
3                           # object 를 취함
>>> 3 and 2
2
>>> [] and {}               # and 의 왼쪽이 false 인 경우
[]
>>> 3 and []                # and 의 왼쪽이 true 인 경우
[]
```

세 항을 가지는 if 문 (Ternary if's)

```
if X:
    A = Y
else:
    A = Z
```

=

```
A = Y if X else Z
```

```
>>> A = 't' if 'spam' else 'f'
>>> A
't'
>>> A = 't' if '' else 'f'
>>> A
'f'
```

```
A = ((X and Y) or Z)
```

```
A = [Z, Y][bool(X)]
```

```
>>> ['f', 't'][bool('')]
'f'
>>> ['f', 't'][bool('spam')]
't'
```

while 순환문 (while Loops)

```
while <test>:
        <statements1>
else:
        <statements2>
```

```
>>> while True:
...         print('Type ctrl-C to stop me!')
```

```
>>> x = 'spam'
>>> while x:
...         print(x, end=' ')
...         x = x[1:]
...
spam pam am m
```

```
>>> a = 0; b = 10
>>> while a < b:
...         print(a, end=' ')
...         a += 1
...
0 1 2 3 4 5 6 7 8 9
```

do-while loop?
: Python 에는 존재하지 않는 구문 구조

```
while True:
        ··· loop body ···
        if exitTest(): break
```

순환문 제어 문장들

문장	의미
break	가장 가까운 순환문을 빠져나감 (순환문 블록의 이후 문장들은 실행되지 않음)
continue	가장 가까운 순환문의 처음으로 돌아감 (순환문의 다음 반복 회차를 실행)
pass	아무 일도 하지 않음 (왜 필요할까?)
Loop else block	순환문이 정상적으로 실행을 마친 후 실행됨 (즉, break 를 만나지 않은 경우)

```
while True:
    name = input('Enter name: ')
    if name == 'stop': break
    age = input('Enter age: ')
    print('Hello', name, '=>', int(age))
```

```
x = 10
while x:
    x = x - 1
    if x % 2 != 0: continue
    print(x, end=' ')
```

```
x = y // 2
while x > 1:
    if y % x == 0:
        print(y, 'has factor', x)
        break
    x -= 1
else:
    print(y, 'is prime')
```

for 순환문 (for Loops)

```
for <target> in <object>:
        <statements>
else:
        <statements>
```

```
>>> for x in ["spam", "eggs", "ham"]:
...         print(x, end=' ')
...
spam eggs ham
```

```
>>> sum = 0
>>> for x in [1, 2, 3, 4]:
...         sum = sum + x
...
>>> sum
10
>>> prod = 1
>>> for item in [1, 2, 3, 4]:
...         prod *= item
...
>>> prod
24
```

```
>>> S = "lumberjack"
>>> T = ("and", "I'm", "okay")
>>> for x in S: print(x, end=' ')
...
l u m b e r j a c k
>>> for x in T: print(x, end=' ')
...
and I'm okay
```

```
>>> T = [(1, 2), (3, 4), (5, 6)]
>>> for (a, b) in T:
...         print(a, b)
...
1 2
3 4
5 6
```

Dictionaries?
Key, value pairs?

range

```
>>> list(range(5))
[0, 1, 2, 3, 4]

>>> list(range(2, 5))
[2, 3, 4]

>>> list(range(0, 10, 2))
[0, 2, 4, 6, 8]

>>> list(range(1, 10, 2))
???
```

```
>>> L = [1, 2, 3, 4, 5]

>>> for x in L:
...         x += 1
...
>>> L
[1, 2, 3, 4, 5]
>>> x
6

>>> for i in range(len(L)):
...         L[i] += 1
...
>>> L
[2, 3, 4, 5, 6]
```

파일 (Files)

```
>>> f = open('data.txt', 'w')          # 새로운 파일을 쓰기 모드로 생성
>>> f.write('Hello\n')                 # 문자열 (바이트의 나열) 을 파일에 기록
6
>>> f.write('world\n')                 # 또다른 문자열을 기록 (뒤에 이어서 쓰여짐)
6
>>> f.close()                          # 파일 닫기 시점에 버퍼 내용을 디스크에 기록
>>> f
<closed file 'data.txt', mode 'w' at 0x…>

>>> f = open('data.txt')               # 디폴트는 읽기 모드로 파일을 열게 됨 ('r' 을 지정한 것과 동일)
>>> text = f.read()                    # 파일 내용 전체를 문자열로 읽어들임
>>> text
'Hello\nworld\n'
>>> print(text)
Hello
world

>>> text.split()                       # 파일의 내용은 언제나 텍스트 문자열
['Hello', 'world']
```

Quiz

- Python 에서 코드 블록을 지정하는 방법은 무엇인가?
  - C/C++, Java 등의 프로그래밍 언어와 큰 차이점이 있다.

- while 순환문과 for 순환문의 가장 큰 차이점은 무엇인가?
  - 어떤 때 while 을, 어떤 때 for 를 이용할 것인가?

- C/C++ 의 do - while 순환문과 동일한 구조의 순환문을 Python 으로 작성하면?
  - Python 에는 do - while 구문이 없다.

- 3 개의 항을 가지는 if 문을 이용하여, 짝수 ('even') 와 홀수 ('odd') 를 구별하는 문장을 작성하면?
  - X 가 짝수인 경우 s 에 'even' 을, 홀수인 경우에 'odd' 를 대입하는 한 줄짜리 문장

Exercise

(1) 구구단을 출력하는 프로그램을 작성해 본다.

A. 한 줄에 세 개씩 출력하시오.

2 X 2 = 4, 2 X 3 = 6, 2 X 4 = 8

2 X 5 = 10, 2 X 6 = 12, 2 X 7 = 14

…

B. 한 단이 끝나면 빈 줄을 삽입하시오.

C. 자리수가 달라도 잘 정렬되도록 하려면?

Q & A