파이썬 2.3 간편 참조서

한글판 johnsonj 2003.11.11

목차

- 작업공간 탐험과 관용구 힌트

- 이맥스용 파이썬 모드

서론

버전 2.3
최신 버전은 여기에서 얻을 수 있을 것이다.
에러, 오류, 그리고 제안이 있으면 리차드 그루엣(Richard Gruet) (pqr at rgruet.net)에게 보고 해주기를 바란다.

11 Oct 2003, rev 2
리차드 그루엣(Richard Gruet)이 파이썬 2.3 용으로 개정
11 May 2003, rev 4
리차드 그루엣(Richard Gruet)이 파이썬 2.2 용으로 개정 (안드레이(Andrei)가 스타일 개정)
7 Aug 2001
사이먼 브루닝(Simon Brunning)이 파이썬 2.1 용으로 개정
16 May 2001
리차드 그루엣(Richard Gruet)과 사이먼 브루닝(Simon Brunning)이 파이썬 2.0 용으로 개정
18 Jun 2000
리차드 그루엣(Richard Gruet)이 파이썬 1.5.2 용으로 개정
30 Oct 1995
크리스 호프만(Chris Hoffmann)이 파이썬 1.3 용으로 처음 만듦
색상 코드:

2.3 에 추가된 특징.
2.2 에 추가된 특징.
2.1 에 추가된 특징.
원래 기반으로 하는 곳:

- Python Bestiary, 저자: 켄 만하이머(Ken Manheimer)

- 파이썬 매뉴얼, 저자: 귀도 반 로섬(Guido van Rossum) 그리고 프레드 드레이크(Fred Drake)

- python-mode.el, 저자: 팀 피터스(Tim Peters)

- 그리고 comp.lang.python 의 독자 여러분

- 공식 문서: http://www.python.org/doc/

- 기타 문서& 책들: 파이썬으로 다이빙해 들어가기 , 파이썬 요리책 , 질문과 답변 모음집 , 파이썬으로 생각하기 , 파이썬으로 텍스트 처리하기

- 꾸러미: 파이썬 꾸러미 인덱스 (PyPI) , 파르너서스의 천정 , 소스포지 ("python"으로 탐색할 것) , 오라일리사의 파이썬 개발센터 , 우주선 파이썬

- 위키: 모인모인

팁: 파이썬 인터프리터 안에서, help 또는 help(object)를 사용하자.

요청 선택사항

python[w] [-dEhiOQStuUvVWxX?] [-c command| scriptFile | -] [args]

(pythonw 는 터미널/콘솔을 열지 않는다; python 은 연다)

요청 선택사항

선택사항	효과
-d	해석기 디버깅 정보를 출력한다 (PYTHONDEBUG=x)
-E	(PYTHONPATH 와 같은) 환경 변수를 무시한다
-h	도움말 메시지를 인쇄하고 종료한다 (예전에는 -?)
-i	스크립트를 실행한 후에 상호대화적으로 내부검사한다 (또한 PYTHONINSPECT=x) 그리고 stdin 이 터미날이 아닌 것 같더라도 프롬프트를 강제한다.
-O	생성된 바이트코드를 최적화한다 (또한 PYTHONOPTIMIZE=x). 확인문(Asserts)은 억제된다.
-OO	-O 최적화를 하고 문서화 문자열을 제거한다.
-Q arg	나눗셈 선택사항: -Qold (기본), -Qwarn, -Qwarnall, -Qnew
-S	초기화시에 import site 를 실행하지 말 것.
-t	일관성 없이 탭을 사용하면 경고한다 (-tt: 에러를 제출한다).
-u	버퍼화되지 않은 이진 표준출력과 표준 에러 (PYTHONUNBUFFERED=x).
-U	파이썬이 모든 문자열 기호문자를 유니코드 기호문자로 해석하도록 만든다.
-v	상세한 설명 (수입 서술문을 추적한다) (PYTHONVERBOSE=x).
-V	파이썬 버전 번호를 인쇄하고 종료한다.
-W arg	경고를 제어한다 (인수(arg)는 다음과 같다 action:message:category:module:lineno)
-x	소스에서 첫 줄을 건너뛰어서, 비-유닉스 형태의 #!cmd 를 사용할 수 있도록 허용한다.
~~-X~~	~~클래스 기반의 내장 예외를 불능으로 만든다 (예외를 하위 호환성 있게 관리하기 위해)~~
-c command	명령어를 지정하여 실행한다 (다음 섹션 참조). 이렇게 하면 옵션 목록은 중지된다 (뒤에 따르는 옵션은 명령어에 인수로 건네진다).
scriptFile	실행할 파이썬 파일 이름 (.py). 표준입력으로부터 읽는다.
-	프로그램이 표준입력으로부터 읽는다 (기본; tty 면 상호대화 모드).
args	스크립트나 명령에 건네진다 (in sys.argv[1:])
	스크립트(scriptFile)나 명령어가 없으면, 상호대호 모드로 들어간다.

- 표준 배포본에서 얻을 수 있는 통합환경(IDE): IDLE (tkinter 기반, 이식성있는), Pythonwin (윈도우즈). 기타 무료 IDE 들: BOA constructor, SPE, wxGlade.

- 전형적인 파이썬 모듈 머리부 :

  #!/usr/bin/env python
  # -*- coding: latin1 -*-
  2.3 이후로 파이썬 소스 파일의 인코딩(encoding)을 반드시 첫 두줄중 하나에 다음과 같은 형식으로 선언해야 한다 (기본 값은 7 비트 Ascii 이다) [PEP-0263]:
  # -*- coding: encoding -*-
  표준 인코딩(encodings)은 여기에 정의되어 있다, 예. ISO-8859-1 (이른바 latin1), iso-8859-15 (latin9), UTF-8... 모든 인코딩이 지원되는 것은 아니다, 특히 UTF-16 이 지원되지 않는다.

환경 변수

환경 변수

변수	효과
PYTHONHOME	대안적인 접두사(prefix) 디렉토리 (또는 prefix;exec_prefix). 기본 모듈 탐색 경로는 prefix/lib 를 사용한다.
PYTHONPATH	모듈 파일에 대하여 기본 탐색 경로를 추가한다. 형식은 쉘의 $PATH 와 똑같다: 하나 이상의 디렉토리 경로이름은 (쌍)반점 주위에 공백 없이 ':'이나 ';'으로 분리된다! 윈도우즈에서 첫 탐색 경로는 다음 레지스트리 키이다 : HKEY_LOCAL_MACHINE₩Software₩Python₩PythonCore₩x.y₩PythonPath (기본 값). 어플리케이션의 루트 디렉토리 경로를 기본 문자열 값으로 해서 키를 정의해도 좋다. 다른 방법으로는, .pth 확장자를 가진 텍스트 파일을 파이썬 홈 디렉토리에 만들어서 (한 줄씩) 그 경로를 담아도 된다.
PYTHONSTARTUP	만약 이것이 읽을 수 있는 파일의 이름이라면, 그 파일 안에 들은 파이썬 명령어가 먼저 실행되고 나서 상호대화 모드로 첫 프롬프트가 화면에 표시된다 (기본값 아님).
PYTHONDEBUG	비어 있지 않으면, -d 옵션과 같다
PYTHONINSPECT	비어 있지 않으면, -i 옵션과 같다
PYTHONOPTIMIZE	비어 있지 않으면, -O 옵션과 같다
PYTHONUNBUFFERED	비어 있지 않으면, -u 옵션과 같다
PYTHONVERBOSE	비어 있지 않으면, -v 옵션과 같다
PYTHONCASEOK	비어 있지 않으면, 파일/모듈 이름에 대소문자를 무시한다 (수입한다).

주목할 어휘 개체

키워드

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

- (키워드 목록은 표준 화면에서 얻을 수 있다: keyword)

- 불법적인 토큰 (문자열에서만 유효하다): @ $?

- 서술문은 모두 반드시 하나의 줄이어야 한다. 한 서술문을 여러 줄에 걸쳐서 자르려면 C 전처리기에서와 같이 "₩"를 사용하자.
  예외: (), [], 또는 {} 쌍 안에서는 언제든지 자를 수 있다. 또는 삼중-(겹)따옴표로 둘러싸인 문자열에서도 언제든지 자를 수 있다.

- 반점(";")으로 구분되어 있으면 하나 이상의 서술문이 한 줄에 나타날 수 있다.

- 주석은 "#"으로 시작해서 줄 끝까지 계속된다.

Identifiers

(letter | "_") (letter | digit | "_")*

- 파이썬에서 식별자, 키워드, 속성, 등등은 모두 대소문자-민감하다.

- 특수한 형태: _ident ('from module import *'으로 수입되지 않는다); __ident__ (시스템이 정의한 이름); __ident (클래스-소유를 위해 이름 조작(mangling)).

문자열 기호문자(String literals)

기호문자
"겹 따옴표로 둘러싸인 문자열"
'따옴표로 경계를 짓고 안에 "를 가진 문자열'
'''임베드된 새줄문자 표식과 따옴표(') 표식을 담고 있는 문자열, 삼중 따옴표로 경계를 지을 수 있다.'''
""" 삼중 겹따옴표를 경계표식자로 사용해도 좋다"""
u'유니코드 문자열'
U"또다른 유니코드 문자열"
r'₩가 (기호문자화되어) 유지되는 날 문자열: 정규 표현식과 윈도우즈 경로에 간편하게 사용된다!'
R"또다른 날 문자열" -- 날 문자열은 ₩로 끝나면 안된다
ur'유니코드 날 문자열'
UR"또다른 날 유니코드"

- 문자열을 다음 줄에 계속하려면 줄 끝에 ₩를 사용하자.

- 인접한 문자열들은 결합된다, 예. 'Monty' 'Python'는 'Monty Python'과 똑 같다.

- u'hello' + ' world' --> u'hello world' (강제로 유니코드로 변환된다)

문자열 기호문자 피신(String Literal Escapes)	
피신(Escape)	의미
₩newline	무시된다 (새줄문자 피신)
₩₩	역사선 (₩)
₩e	피신 (ESC)
₩v	수직 탭 (VT)
₩'	따옴표 (')
₩f	한장먹임 (FF)
₩ooo	8 진값 ooo 를 가진 문자
₩"	겹따옴표 (")
₩n	한줄먹임 (LF)
₩a	벨소리 (BEL)
₩r	나르개복귀 (CR)
₩xhh	16 진값 hh 를 가진 문자
₩b	백스페이스 (BS)
₩t	수평 탭 (TAB)
₩uxxxx	16-비트 16 진 값 xxxx 을 가진 문자 (유니코드 전용)
₩Uxxxxxxxx	32-비트 16 진 값 xxxxxxxx 을 가진 문자 (유니코드 전용)
₩N{name}	유니코드 데이터베이스에 이름이 붙은 문자 (유니코드 전용), 예. u'₩N{Greek Small Letter Pi}' <=> u'₩u03c0'. (반대로, unicodedata 모듈에서는, unicodedata.name(u'₩u03c0') == 'GREEK SMALL LETTER PI')
₩AnyOtherChar	그대로 둔다

- NUL 바이트(₩000)는 문자열-끝 표식이 아니다; NUL 은 문자열 안에 임베드해도 된다.

- 문자열 (그리고 터플)은 변경불능이다: 변경이 될 수 없다.

불리언 상수

- True

- False

2.2 에서, True 그리고 False 는 정수로 1 그리고 0 이다. 2.3 이후로, 불리언 상수는 새로운 유형인 bool 유형이다.

숫자

- 10 진 정수: 1234, 123456789O546378940L (또는 l)

- 8 진 정수: 0177, 01777777777777777777L (0 으로 시작)

- 16 진 정수: 0xFF, 0XFFFFffffFFFFFFFFFFL (0x 또는 0X 으로 시작)

- 장정도 정수 (정밀도 무제한): 1234567890123456L (L 이나 l 로 끝남) 또는 long(1234)

- 소수 (double precision): 3.14e-10, .001, 10., 1E3

- 복소수: 1J, 2+3J, 4+5j (J 나 j 로 끝나고, + 는 (소수) 실수 부분과 허수 부분을 가른다)

정수와 긴 정수는 버전 2.2 에서부터 통일되었다 (L 접미사는 이제 더 이상 필요없다)

연속열

- 길이가 0, 1, 2 인 문자열 (위 참조)
  '', '1', "12", 'hello₩n'

- 길이가 0, 1, 2, 등등인 터플:
  () (1,) (1,2) # 괄호는 길이가 > 0 이라면 선택적이다

- 길이가 0, 1, 2, 등등인 리스트:
  [] [1] [1,2]

- 지표화는 0-기반이다. 음의 지표는 (보통) 연속열의 끝으로부터 거꾸로 세는 것을 뜻한다.

- 연속열 조각썰기(slicing) [지표-위치-시작 : 지표-위치-미만 [: 뛸값]]. 기본값으로 시작은 0, 끝은 len(sequence)까지 , 뛸값은 1 이다.

  ```
  a = (0,1,2,3,4,5,6,7)
  a[3] == 3
  a[-1] == 7
  a[2:4] == (2, 3)
  a[1:] == (1, 2, 3, 4, 5, 6, 7)
  a[:3] == (0, 1, 2)
  a[:] == (0,1,2,3,4,5,6,7) # 연속열의 사본을 만든다.
  a[::2] == (0, 2, 4, 6) # 오직 짝수만.
  a[::-1] = (7, 6, 5, 4, 3 , 2, 1, 0) # 순서를 뒤집는다.
  ```

사전 (짝짓기)

사전 길이가 0, 1, 2, 등등인 사전(dict) 유형: {} {1 : 'first'} {1 : 'first', 'next': 'second'}

연산자와 평가 순서

연산자와 평가 순서

최고	연산자	주석
	, [...] {...} `...`	터플, 리스트 & 사전. 생성; 문자열 변환.
	s[i] s[i:j] s.attr f(...)	지표화 & 조각썰기; 속성, 함수 호출
	+x, -x, ~x	단항 연산자
	x**y	거듭제곱
	x*y x/y x%y	곱셈, 나눗셈, 나머지셈
	x+y x-y	덧셈, 뺄셈
	x<<y x>>y	비트 이동
	x&y	비트별 곱
	x^ y	비트별 배타합
	x|y	비트별 합
	x<y x<=y x>y x>=y x==y x!=y x<>y x is y x is not y x in s x not in s	비교, 신분, 구성원
	not x	불리언 부인
	x and y	불리언 곱(and)
	x or y	불리언 합(or)
최저	lambda args: expr	익명 함수

- 교체 이름들은 연산자(operator) 모듈에 정의된다. (예. + 에 대해서 __add__ 그리고 add)

- 대부분의 연산자들은 덮어쓸 수 있다(overridable)

기본 유형과 연산

비교 (어떤 두 유형 사이에 정의된다)

비교

비교	의미	주의
<	보다 작은 (미만)	(1)
<=	작거나 같은 (이하)	
>	보다 큰 (초과)	
>=	보다 크거나 같은 (이상)	
==	같은 (동등)	
!= or <>	같지 않은 (부등)	
is	객체 신분	(2)
is not	부인된 객체 신분	(2)

주의:

- 비교 행위는 특수 메쏘드 __cmp__를 정의하면 주어진 클래스에 대하여 덮어쓸 수 있다.

- (1) X < Y < Z < W 는 C 와는 다르게, 의미가 예상된다

- (2) 객체 값이 아니라, 객체 신분을 비교한다 (즉. id(object)).

None

- None 은 함수에 대한 기본 반환 값으로 사용된다. 내장 객체는 NoneType 을 유형으로 가진다. 앞으로 키워드가 될 가능성이 있다.

- None 과 동등한 입력값은 파이썬을 상호대화적으로 실행할 때는 인쇄되지 않는다.

불리언 연산자

불리언 값과 연산자

값 또는 연산자	동등한	주의
built-in bool(expr)	expr 이 참이면 True, 그렇지 않으면 False 이다.	True, False 참조
None, 수치 0, 빈 연속열과 짝짓기(사전)	거짓으로 간주된다	
기타 모든 값	참으로 간주	
not x	x 가 False 이면 True, 그렇지 않으면 True	
x or y	x 가 false 이면 y, 그렇지 않으면 x	(1)
x and y	x 가 false 이면 x, 그렇지 않으면 y	(1)

주의:

- 진리 테스트는 주어진 클래스에 대하여 특수 메쏘드 __nonzero__를 정의하면 덮어쓸 수 있다.

- (1) 출력결과를 결정하는데 필요할 경우에만 두 번째 인수를 평가한다.

수치 유형

소수, 정수 그리고 긴 정수.

- 소수(float 유형)는 C 의 double 형으로 구현되었다.

- 정수(int 유형)는 C 의 long 형으로 구현되었다 (부호있는 32 비트이고, 최대 값은 sys.maxint 이다)

- 긴 정수(long 유형)는 크기가 무제한이다 (시스템 자원에만 제한된다).

- 정수와 긴 정수는 2.2 배포본에서부터 통일되었다 (L 접미사는 더 이상 필요하지 않다). int()는 OverflowError 를 일으키는 대신에 긴 정수를 돌려준다.

모든 수치 유형에 대한 연산자

모든 수치 유형에 대한 연산자

연산	결과
abs(x)	x 의 절대 값
int(x)	x 는 정수로 변환된다
long(x)	x 는 장정도 정수로 변환된다
float(x)	x 는 부동소수점수로 변환된다
-x	x 는 부인된다
+x	x 는 그대로이다
x + y	x 와 y 의 합
x - y	x 와 y 의 차
x * y	x 와 y 의 곱
x / y	x 를 y 로 소수 나눗셈(true division): 1/2 -> 0.5 (1)
x // y	정수 나눗셈(floor division) 연산자: 1//2 -> 0 (1)
x % y	x / y 의 나머지
divmod(x, y)	터플 (x/y, x%y)
x ** y	x 를 y 제곱한다 (다음 pow(x,y)와 동일)

주의:

- (1) from __future__ import division 로 유효성을 검증하지 않으면 /는 여전히 정수(floor) 나눗셈이다 (1/2 == 0).

- 클래스로 __truediv__와 __floordiv__ 메쏘드를 덮어쓰면 이 연산자들을 재정의할 수 있다.

정수와 긴 정수에 대한 비트 연산자

비트 연산자

연산	결과
~x	x 의 비트들이 거꾸로 된다
x ^ y	x 와 y 의 비트별 배타합
x & y	x 와 y 의 비트별 곱
x | y	x 와 y 의 비트별 합
x << n	x 가 n 비트만큼 왼쪽으로 이동된다
x >> n	x 가 n 비트만큼 오른쪽으로 이동된다

복소수

- 유형은 complex 이며, 머신-수준에서 이중 정밀도로 부동소수점 수의 쌍으로 표현된다.

- 실수 값과 허수 값은 복소수 z 에서 z.real 와 z.imag 속성을 통하여 열람할 수 있다.

수치 예외

TypeError
: 수치 연산을 비-숫자에 적용하면 일어난다

OverflowError
수치가 경계를 초과함
ZeroDivisionError
나눗셈이나 나머지셈에서 두 번째 인수가 0 이면 일어난다.

모든 연속열 유형에 대한 연산 (리스트, 터플, 문자열)

모든 연속열 유형에 대한 연산

연산	결과	주의
x in s	s 의 항목 하나가 x 와 같으면 True, 그렇지 않으면 False	(3)
x not in s	s 의 항목 하나가 x 와 같으면 False, 그렇지 않으면 True	(3)
s + t	s 와 t 의 결합	
s * n, n*s	s 의 사본을 n 개 결합한다	
s[i]	s 의 i'번째 항목, 시작은 0 에서부터	(1)
s[i: j] s[i: j: step]	s 에서 i 이상부터 j 미만까지의 조각. 선택적인 뜀(step) 값, 음수도 가능 (기본: 1).	(1), (2)
len(s)	s 의 길이	
min(s)	s 에서 가장 작은 항목	
max(s)	s 에서 가장 큰 항목	

주의:

- (1) i 나 j 가 음수이면, 지표는 문자열의 끝에서 상대적이다. 즉, len(s)+i 또는 len(s)+j 가 대치된다. 그러나 -0 은 여전히 0 임을 주의하자.

- (2) s 에서, i 부터 j 까지의 조각은 i <= k < j 인 지표 k 를 가지는 항목의 연속열로 정의된다. i 나 j 가 len(s)보다 크면, len(s)를 사용한다. i 가 생략되면, len(s)를 사용한다. i 가 j 보다 크거나 같으면, 그 조각은 비어 있다.

- 문자열에 대하여: 2.3 이전에는, x 가 반드시 한개짜리 문자열이어야 한다; 2.3 이후로, x in s 는 x 가 s 의 하부문자열이면 참(True)이다.

변경가능 연속열에 대한 연산 (list 유형)

변경가능 연속열에 대한 연산

연산	결과	주의
s[i] =x	s 에서 항목 i 가 x 로 교체된다	
s[i:j [:step]] = t	s 에서 i 부터 j 까지의 조각이 t 로 교체된다	
del s[i:j[:step]]	다음과 동일 s[i:j] = []	
s.append(x)	다음과 동일 s[len(s) : len(s)] = [x]	
s.extend(x)	다음과 동일 s[len(s):len(s)]= x	(5)
s.count(x)	s 에서 [i] == x 인 i 의 개수를 돌려준다	
s.index(x[, start[, stop]])	s[i]== x 인 가장 작은 i 를 돌려준다. 시작(start)과 중지(stop)는 리스트의 일부분으로 탐색을 제한한다.	(1)
s.insert(i, x)	다음과 동일 s[i:i] = [x] if i >= 0. i == -1 이면 가장 마지막 요소 앞에 삽입된다.	
s.remove(x)	다음과 동일 del s[s.index(x)]	(1)
s.pop([i])	다음과 동일 x = s[i]; del s[i]; return x	(4)
s.reverse()	s 의 항목을 제자리에서 역정렬한다	(3)
s.sort([cmpFct])	s 의 항목을 제자리에서 정렬한다	(2), (3)

주의:

- (1) s 에서 x 가 발견되지 않으면 (즉, 범위를 벗어나면) ValueError 예외를 일으킨다.

- (2) sort() 메쏘드는 선택적인 인수를 하나 취해 2 개의 인수(리스트 항목)를 비교하는 함수를 지정한다. 이 함수는 첫 인수가 두번째 인수보다 더 작거나, 같거나, 또는 더 큰가에 따라서 -1, 0, 또는 1 을 반환해야 한다. 이 때문에 정렬 처리 속도가 상당히 저하되니 주의하자.

- (3) sort() 메쏘드와 reverse() 메쏘드는 커다란 리스트를 정렬하거나 역정렬할 때 공간을 아끼기 위해 적절한 곳에서 리스트를 변형한다. 이런 부작용을 알려주기 위해 정렬된 리스트 또는 역정렬된 리스트를 반환하지 않는다.

- (4)리스트를 제외하고 pop() 메쏘드는 변경가능한 연속열 유형에서 지원되지 않는다. 선택적 인수 i 는 기본값이 -1 이며, 그래서 기본으로 마지막 항목이 제거되어 반환된다.

- (5) x 가 리스트 객체가 아니면 TypeError 을 일으킨다.

짝짓기/사전에 대한 연산 (type dict)

짝짓기 연산

연산	결과	주의
len(d)	d 에 있는 항목의 개수	
dict(**kw)	사전을 만든다. 초기화 리스트로 키워드 리스트 kw 를 사용하자.	
d.fromkeys(iterable, value=None)	반복자(iterator)에 의해서 제공된 키로 사전을 만드는 클래스 메쏘드, 모든 값들은 value 로 설정된다.	
d[k]	키 k 를 가진 d 의 항목	(1)
d[k] = x	d[k]를 x 로 설정한다	
del d[k]	d 로부터 d[k]를 제거한다	(1)
d.clear()	모든 항목을 d 로부터 제거한다	
d.copy()	d 를 얕게 복사	
d.has_key(k) k in d	d 가 키 k 를 가지면 True, 그렇지 않으면 False	
d.items()	d 에서 (키, 항목) 쌍의 리스트로 구성된 사본을 돌려준다.	(2)
d.keys()	d 에서 키의 리스트로 구성된 사본을 돌려준다	(2)
d1.update(d2)	for k, v in d2.items(): d1[k] = v	(3)
d.values()	d 에서 값의 리스트로 구성된 사본을 돌려준다	(2)
d.get(k, defaultval)	d 에서 키 k 를 가진 항목	(4)
d.setdefault(k[,defaultval])	k 가 d 안에 있으면 d[k], 그렇지 않으면 defaultval (설정도 한다)	(5)
d.iteritems()	(키, 값) 쌍에 대하여 반복자(iterator)를 돌려준다.	
d.iterkeys()	짝짓기의 키(keys)에 대하여 반복자(iterator)를 돌려준다.	
d.itervalues()	짝짓기의 값(values)에 대하여 반복자(iterator)를 돌려준다.	
d.pop(k[, default])	키 k 를 제거하고 그에 상응하는 값을 돌려준다. 키가 발견되지 않으면, 주어진 기본값(default) 반환되고, 없으면 KeyError 가 일어난다.	
d.popitem()	d 로부터 임의의 (키, 값) 쌍을 제거해서 돌려준다.	

주의:

- 키를 받아들일 수 없으면 TypeError 가 일어난다

- (1) 키 k 가 짝짓기지도에 없으면 KeyError 가 일어난다

- (2) 키와 값은 무작위 순서로 나열된다

- (3) d2 는 d1 와 같은 유형이어야 한다

- (4) k 가 짝짓기지도에 없더라도 절대로 예외를 일으키지 않는다, 대신에 defaultval 를 돌려준다. defaultval 는 선택적이다, 기본값이 제공되지 않고 k 가 짝짓기지도에 없으면, None 이 반환된다.

- (5) k 가 짝짓기지도에 없더라도 절대로 예외를 일으키지 않는다, 대신에 defaultVal 을 반환하고, k 를 defaultVal 값에 짝짓는다. defaultVal 는 선택적이다. 기본값이 제공되지 않고 k 가 짝짓기지도에 없으면 , None 이 반환되고 짝짓기지도에 추가된다.

문자열 연산 (type str)

다음 문자열 메쏘드들은 string 모듈에서 사용할 수 있는 함수들을 (완전하게는 아니지만) 대부분 대체한다.
str 과 unicode 유형은 공통 기반 클래스로 basestring 를 공유한다.

문자열 연산

연산	결과	주의
s.capitalize()	s 의 첫 문자만 대문자로 만들어 그 사본을 돌려준다.	
s.center(width)	s 를 width 길이의 문자열에 중앙정렬하여 그 사본을 돌려준다.	(1)
s.count(sub[,start[,end]])	문자열 s 에서 sub 라는 하부문자열의 출현 횟수를 돌려준다.	(2)
s.encode([encoding [,errors]])	s 를 인코드된 버전으로 돌려준다. 기본 인코딩은 현재 기본 문자열 인코딩이다.	(3)
s.endswith(suffix [,start [,end]])	s 가 지정된 suffix 로 끝나면 True 를 반환, 그렇지 않으면 false 를 돌려준다.	(2)
s.expandtabs([tabsize])	s 에서 모든 탭 문자들을 공백으로 확장하여 그 사본을 돌려준다.	(4)
s.find(sub[,start[,end]])	문자열 s 에서 하부문자열 sub 가 발견된 그 하한 지표를 돌려준다. sub 가 발견되지 않으면 -1 을 돌려준다.	(2)
s.index(sub[,start[,end]])	find()와 비슷하지만, 하부문자열이 발견되지 않으면 ValueError 를 일으킨다.	(2)
s.isalnum()	s 의 모든 문자들이 영문자숫자이면, True 를 돌려준다. 그렇지 않으면 False.	(5)
s.isalpha()	s 의 모든 문자들이 영문자이면, True 를 돌려준다. 그렇지 않으면 False.	(5)
s.isdigit()	s 의 모든 문자들이 숫자 문자들이면, True 를 돌려준다. 그렇지 않으면 False.	(5)
s.islower()	s 의 모든 문자들이 소문자이면, True 를 돌려준다. 그렇지 않으면 False.	(6)
s.isspace()	s 의 모든 문자들이 흰공백 문자들이면, True 를 돌려준다. 그렇지 않으면 False.	(5)
s.istitle()	s 의 모든 문자들이 제목격변환된 문자열이면, True 를 돌려준다. 그렇지 않으면 False.	(7)
s.isupper()	s 의 모든 문자들이 대문자이면, True 를 돌려준다. 그렇지 않으면 False.	(6)
separator.join(seq)	연속열 seq 에 있는 문자열들을 결합하여 돌려준다, 문자열 가름자 (separator)로 구분된다, 예: '', ''.join(['A', 'B', 'C']) -> "A, B, C"	
s.ljust(width)	s 를 너비(width) 길이의 문자열에서 왼쪽으로 정렬하여 돌려준다.	(1), (8)
s.lower()	s 를 소문자로 변환한 사본을 돌려준다.	
s.lstrip([chars])	s 에서 이끄는 문자들(chars(기본값: 흰공백))을 제거한 사본을 돌려준다.	
s.replace(old, new[, maxsplit])	s 에서 하부문자열 old 를 모두 new 로 교체하여 사본을 돌려준다.	(9)
s.rfind(sub[, start[, end]])	s 에서 하부문자열 sub 가 발견된 제일 높은 지표를 돌려준다.. sub 가 발견되지 않으면 -1 을 돌려준다.	(2)
s.rindex(sub[, start[, end]])	rfind()와 비슷하지만, 하부문자열이 발견되지 않으면 ValueError 를 일으킨다.	(2)
s.rjust(width)	s 를 너비(width) 길이의 문자열에서 오른쪽으로 정렬하여 돌려준다.	(1), (8)
s.rstrip([chars])	s 에서 따르는 문자들(chars(기본값: 흰공백)을 제거한 사본을 돌려준다.	
s.split([separator[, maxsplit]])	s 에 있는 단어들의 리스트를 돌려준다, 가름자(separator)를 구분 문자열로 사용한다.	(10)
s.splitlines([keepends])	s 에 있는 줄들의 리스트를 돌려준다, 줄 경계에서 자른다.	(11)
s.startswith(prefix [, start[, end]])	s 가 지정된 접두사(prefix)로 시작하면 True 를 돌려준다, 그렇지 않으면 False 를 돌려준다. 시작(start)과 끝(end)에 음수가 사용되어도 좋다.	(2)
s.strip([chars])	s 에서 따르는 그리고 이끄는 문자들(chars(기본값: 흰공백))을 제거한 사본을 돌려준다.	
s.swapcase()	s 에서 대문자는 소문자로 대문자는 소문자로 변환하여 돌려준다.	
s.title()	s 에서 제목격의 사본을 돌려준다, 즉. 첫 문자는 대문자로 시작하고, 나머지는 소문자로 격변환된 단어들.	
s.translate(table [, deletechars])	s 를 변환 테이블(table)을 통하여 짝짓기 한 사본을 돌려준다.	(12)
s.upper()	s 를 대문자로 변환시킨 사본을 돌려준다.	
s.zfill(width)	너비(width) 길이의 문자열에서 왼쪽을 0 으로 채운 수치 문자열을 돌려준다.	

Notes:

- (1) 빈곳은 공백을 사용하여 채운다.

- (2) 선택적인 인수 start 가 주어지며, 하부문자열 s[start:]가 처리된다. 선택적 인수 start 와 end 가 공급되면, 하부문자열 s[start:end]가 처리된다.

- (3) 다른 에러 처리 구도를 설정하려면 선택적인 인수 errors 를 줄 수도 있다. errors 에 대한 기본값은 엄격한('strict')이다, 인코딩 에러는 ValueError 를 일으킨다는 뜻이다. 다른 가능한 값은 'ignore'와 'replace'가 있다.

- (4) 선택적인 인수인 탭크기(tabsize)가 주어지지 않으면, 탭 크기는 8 문자로 간주된다.

- (5) 문자열 s 에 적어도 한 개의 문자가 없으면 False 를 돌려준다.

- (6) 문자열 s 에 격변환된 문자가 하나도 없으면 False 를 돌려준다.

- (7) 제목격변환된 문자열이란 대문자 다음에 소문자변환된 문자들이 따르고 대문자변환된 문자 다음에 소문자가 따르는 단어(들)로 이루어진 문자열이다.

- (8) 너비(width)가 len(s)보다 작으면 s 가 반환된다.

- (9) 선택적인 인수 maxsplit 이 주어지면, 최대 maxsplit 개수 만큼의 출현만 교체된다.

- (10) sep 이 지정되지 않거나 None 이면, 흰공백 문자열은 모두 가름자이다. maxsplit 이 주어지면, 최대 maxsplit 만큼 분리된다.

- (11) keepends 가 주어지고 참이 아닌 한 줄 가름은 결과 리스트에 포함되지 않는다.

- (12) 테이블(table)은 길이가 256 인 문자열이어야 한다. 선택적인 인수인 제거문자들(deletechars)에 나타나는 모든 문자들은 변환 전에 제거된다.

% 연산자로 하는 문자열 형식화

formatString % args --> 문자열로 평가된다

- formatString 은 C 의 printf 포맷 코드를 사용한다 : %, c, s, i, d, u, o, x, X, e, E, f, g, G, r (세부사항은 아래 참조).

- 정수 인수로 실제 너비와 정밀도를 제공하려면 너비(Width)와 정밀도(precision)는 *가 될 수 있다.

- 표식 문자들로서 -, + , blank, # 그리고 0 을 이해한다 (아래 참조).

- %s 는 어떤 유형의 인수든지 문자열로 변환할 것이다. (str() 함수를 사용한다)

- 인수(args)는 한개이거나 터플일 수 있다

  '%s has %03d quote types.' % ('Python', 2) == 'Python has 002 quote types.'

- 오른쪽은 짝짓기(mapping)가 될 수도 있다:

  a = '%(lang)s has %(c)03d quote types.' % {'c':2, 'lang':'Python}

  (vars() 함수를 오른쪽에 사용하면 아주 간편하다)

형식 코드

코드	의미
d	부호있는 십진 정수.
i	부호있는 십진 정수.
o	부호없는 8 진수.
u	부호없는 10 진수.
x	부호없는 16 진수 (소문자).
X	부호없는 16 진수 (대문자).
e	부동 소수점 지수 형식 (소문자).
E	부동 소수점 지수 형식 (대문자).
f	부동 소수점 10 진 형식.
F	부동 소수점 10 진 형식.
g	지수가 -4 정밀도 이하이면 "e"와 동일, 그렇지 않으면 "f".
G	지수가 -4 정밀도 이하이면 "E"와 동일, 그렇지 않으면, "F".
c	한개의 문자 (정수나 혹은 한개짜리 문자열을 받는다).
r	문자열 (파이썬 객체를 repr()을 사용하여 변환한다).
s	문자열 (파이썬 객체를 str()을 사용하여 변환한다).
%	어떤 인수도 변환되지 않는다, 그 결과에 "%" 문자가 담긴다. (완전한 지정은 %%이다.)

변환 표식 문자

표식	의미
#	값 변환은 ``대안적 형태''를 사용한다 (역주: 8 진법과 16 진법에 대하여, 각각 앞에 '0'과 ','0X'나 '0x'를 붙임).
0	변환은 0 으로 채워진다.
-	변환된 값은 왼쪽 정렬된다 ("-"를 덮어쓴다).
	(공백 하나) 양수 앞에는 반드시 공백 하나(또는 빈문자열)를 남겨두어야 한다. 부호있는 변환에 의해서 생성된다.
+	부호 문자 ("+" 또는 "-")는 변환보다 먼저 처리된다 ("공백(space)" 표식을 덮어쓴다).

파일 객체

(유형은 file 이다). 내장 함수 open()으로 만들어지거나 [비추천됨] 또는 file()로 만들어진다. 다른 모듈의 함수로 역시 만들어도 된다.
유니코드 파일 이름은 이제 파일 이름을 받거나 반환하는 모든 함수에서 지원된다 (open, os.listdir, 등등...).

파일 객체에 대한 연산

파일 연산

연산	결과
f.close()	파일 f 가 닫힌다.
f.fileno()	파일 f 에 대하여 fileno (fd)를 얻는다.
f.flush()	파일 f 의 내부 버퍼를 강제로 저장한다.
f.isatty()	파일 f 가 tty-류의 장치에 연결되어 있으면 1, 그렇지 않으면 0 이다.
f.read([size])	파일 f 로부터 최대 size 바이트만큼을 읽어서 문자열 객체로 돌려준다. size 가 생략되면, 파일끝(EOF)까지 읽는다.
f.readline()	한 줄을 모두 파일 f 로부터 읽는다.
f.readlines()	파일끝(EOF)까지 readline()으로 모두 읽어서, 읽은 줄들을 리스트로 돌려준다.
f.xreadlines()	전체 파일을 메모리로 읽어들이지 않고 한줄씩 파일을 읽어서 연속열-류의 객체를 돌려준다. 2.2 부터, 다음을 사용하자: for line in f (아래 참조).
for line in f: do something...	파일에서 줄들을 (readline 을 사용하여) 순회한다
f.seek(offset[, whence=0])	파일 f 의 위치를 설정한다. "stdio 의 fseek()"와 비슷함. whence == 0 이면 절대적인 지표화 사용. whence == 1 이면 현재 위치로부터 상대적인 거리 사용. whence == 2 이면 파일 끝으로부터 상대적인 거리 사용.
f.tell()	파일 f 의 현재 위치를 돌려준다 (바이트 단위 거리).
f.write(str)	문자열을 파일 f 에 쓴다.
f.writelines(list)	문자열 리스트를 파일 f 에 쓴다.

파일 예외

EOFError
: 읽는 도중 파일-의-끝(End-of-file)에 도착했다. (f 가 tty 라면 여러번 일어날 수 있다.).

IOError
: 기타 I/O-관련 I/O 연산 실패

집합들

sets 모듈에는 새로운 set 데이터 유형 구현이 정의되어 있다. 변경가능 집합에는 Set 클래스를 통하여, 변경불능 집합에는 ImmutableSet 를 통하여 정의된다. 집합은 유일한 (중복 없는) 요소들의 순서없는 모음집이다. 요소들은 반드시 해쉬가능해야 한다.

주요 집합 연산

연산	결과
Set/ImmutableSet([iterable=None])	주어진 반복가능자(iterable)로부터 Set 또는 ImmutableSet 을 만든다 (기본: 공집합), 예. Set([1,2,3]).
elt in S	요소 elt 가 집합 S 에 속하면 True 이다.
S.add(elt)	요소 elt 를 집합 S 에 추가한다. (그 요소가 아직 존재하지 않으면).
S.remove(elt)	요소 elt 를 집합 S 로부터 제거한다 (그 요소가 존재하면). 요소가 발견되지 않으면 KeyError 가 일어난다.
S1.intersection(S2)	집합 S1 와 집합 S2 의 교집합을 돌려준다. 다음도 주목 S1&S2.
S1.union(S2)	집합 S1 와 집합 S2 의 합집합을 돌려준다. 다음도 주목 S1| S2.
S1.symmetric_difference(S2)	집합 S1 와 집합 S2 의 대칭적 차집합을 돌려준다. 다음도 주목 S1^ S2.
S1.issubset(S2)	집합 S1 이 집합 S2 의 자집합이면 True 이다.
S1.issuperset(S2)	집합 S1 이 집합 S2 의 모집합이면 True 이다.

고급 유형

- 자세한 사항은 매뉴얼 참조 -

- Module 객체
- Class 객체
- Class instance 객체
- Type 객체 (다음 모듈 참조: types)
- File 객체 (위 참조)
- Slice 객체
- Ellipsis 객체, 확장된 조각썰기 표기법에서 사용된다 (유일하다, 이름은 Ellipsis 이다)
- Null 객체 (유일하다, 이름은 None 이다)
- XRange 객체
- Callable 유형:
    - 사용자-정의 (파이썬으로 작성):
        - 사용자-정의 함수 객체
        - 사용자-정의 메쏘드 객체
    - 내장 (C 로 작성):
        - 내장 함수 객체
        - 내장 메쏘드 객체

- 내부 유형:
    - Code 객체 (바이트-컴파일된 실행가능 파이썬 코드: bytecode)
    - Frame 객체 (실행 프레임)
    - Traceback 객체 (예외의 스택 추적)

서술문

서술문	결과
pass	Null 서술문
del name[, name]*	name(들)을 객체로부터 푼다. 객체는 더 이상 참조되지 않을 때만 간접적으로 (그리고 자동적으로) 지워질 것이다.
print[>> fileobject,] [s1 [, s2]* [,]	sys.stdout 에 쓰거나, 주어진다면 fileobject 에 쓴다. 인수들 사이에는 공백을 둔다. 서술문이 쉼표(comma)로 끝나지 않는한 끝에는 새줄문자를 둔다. 상호대화적으로 실행될 때는, Print 는 필요하지 않은데, 그냥 표현식을 타자하면, 그 값이 None 이 아닌 한 그의 값을 인쇄해 준다.
exec x [in globals [, locals]]	제공된 이름공간에서 x 를 실행한다. 기본은 현재 이름공간이다. x 는 문자열, 파일 객체나 함수 객체가 될 수 있다.
callable(value,... [id=value] , [*args], [**kw])	함수 callable 을 매개변수를 가지고 호출한다. 매개변수는 이름으로 건네질 수 있으며 또는 함수가 기본값들을 정의하고 있으면 생략될 수 있다. 예. callable 이 다음과 같이 정의되어 있다면: "def callable(p1=1, p2=2)" "callable()" <=> "callable(1, 2)" "callable(10)" <=> "callable(10, 2)" "callable(p2=99)" <=> "callable(1, 99)" *args 는 위치지정 인수의 터플이다. **kw 는 키워드 인수의 사전이다.
yield expression	(오직 발생자(generator) 함수의 몸체 안에서 그리고 try..finally 시도 밖에서만 사용된다). 평가된 표현식(expression)을 "돌려준다".

할당 연산자

할당 연산자		
연산자	결과	주의
a = b	기본 할당 - 객체 b 를 라벨 a 에 할당한다	(1)
a += b	대략 다음과 동일 a = a + b	(2)
a -= b	대략 다음과 동일 a = a - b	(2)
a *= b	대략 다음과 동일 a = a * b	(2)
a /= b	대략 다음과 동일 a = a / b	(2)
a //= b	대략 다음과 동일 a = a // b	(2)
a %= b	대략 다음과 동일 a = a % b	(2)
a **= b	대략 다음과 동일 a = a ** b	(2)
a &= b	대략 다음과 동일 a = a & b	(2)
a |= b	대략 다음과 동일 a = a | b	(2)
a ^ = b	대략 다음과 동일 a = a ^ b	(2)
a >>= b	대략 다음과 동일 a = a >> b	(2)
a <<= b	대략 다음과 동일 a = a << b	(2)

주의:

- (1) 터플, 리스트, 그리고 문자열을 꺼낼 수 있다:

  first, second = a[0:2]
  [f, s] = range(2)
  c1,c2,c3='abc' 팁: x,y = y,x 는 x 와 y 를 바꾼다.

- (2) 정확하게 같지는 않다 - a 는 한 번만 평가된다. 또, 가능하면, 적절한 곳에서 연산이 수행된다 - a 는 교체되기 보다 변경된다.

제어 흐름 서술문

서술문	결과

if condition:
  suite
[elif condition: suite]
*
[else:
  suite]
	일반적인 if/else if/else 서술문

while condition:
  suite
[else:
  suite]
	일반적인 while 서술문. else 모듬(suite)은 회돌이가 탈출(break)로 종료하지 않는 한, 그 회돌이가 종료한 후에 실행된다.

for element in sequence:
  suite
[else:
  suite]
	연속열(sequence)에 대하여 반복하면서, 각 요소를 요소(element)에 할당한다. 수 많은 항목을 반복하려면, 내장된 범위(range) 함수를 사용하자. else 모듬(suite)은 회돌이가 탈출(break)로 종료하지 않는 한 끝에서 실행된다.

break	즉시 for 회돌이나 while 회돌이를 종료한다.
continue	즉시 for 회돌이나 while 회돌이의 다음 반복을 수행한다.
return [result]	함수(또는 메쏘드)를 종료하고 결과(result)를 돌려준다 (하나 이상의 값을 반환하려면 터플(tuple)을 사용하자). 결과가 주어지지 않으면, None 을 돌려준다.

예외(Exception) 서술문

서술문	결과
assert expr[, message]	표현식(expr)을 평가한다. 거짓이면, 메시지로 AssertionError 예외를 일으킨다. 2.3 이전에, __debug__가 0 이면 금지된다.

try:
  suite1
[except [exception [, value]:
  suite2]+
[else:
  suite3]
	suite1 에 있는 서술문이 실행된다. 예외가 발생하면, except 절들을 뒤져서 일치하는 예외(exception)를 찾는다. except 에 일치하거나 일어나면, 그 절에 있는 suite 를 실행한다. 예외가 일어나지 않으면, suite1 다음에 else 절에 있는 suite 가 실행된다. exception 에 값이 있으면, 변수 value 에 배정된다. exception 는 또한 예외들이 담긴 터플(tuple)이 될 수 있다, 예. except (KeyError, NameError), val: print val.

try:
  suite1
finally:
  suite2
	suite1 에 있는 서술문들이 실행된다. 예외가 없으면, suite2 를 실행한다 (suite1 이 return,break 또는 continue 서술문을 가지고 종료하더라도). 예외가 일어났다면, suite2 를 실행하고나서, 즉시 예외를 다시 일으킨다.

raise exceptionInstance	Exception 로부터 파생된 클래스의 실체를 일으킨다. (선호되는 raise 형식).
raise exceptionClass [,value [, traceback]]	선택적인 값 value 를 가지고 주어진 클래스 exceptionClass 의 예외를 일으킨다. 역추적(traceback) 인수는 예외의 역추적을 인쇄할 때 역추적 객체를 지정하여 사용한다.
raise	인수 없이 raise 서술문이 사용되면 현재 함수에서 가장 마지막으로 일어난 예외를 다시 일으킨다.

- 예외는 예외 클래스(exception class)의 실체(instance)이다 (2.0 이전이면, 단순한 문자열(string)이어도 괜찮다).

- 예외 클래스는 미리 정의된 클래스로부터 파생되어야 한다: Exception, 예:

```
class TextException(Exception): pass
try:
    if bad:
        raise TextException()
except Exception:
    print 'Oops'
    # 이는 인쇄된다
    # TextException 이 Exception 의 하부클래스이기 때문이다
```

- 처리되지 못한 예외에 대하여 에러 메시지가 출력되면, 클래스 이름이 인쇄되고, 다음에 쌍점과 공백이, 그리고 마지막으로 내장 함수 str()을 사용하여 문자열로 변환된 문자열이 인쇄된다.

- 모든 내장 예외 클래스는 StandardError 로부터 파생되고, 이는 Exception 로부터 파생된다.

이름 공간 서술문

수입된 모듈 파일은 파이썬 경로(sys.path)에 나열된 디렉토리에 있어야 한다. 2.3 이후로, zip 파일에 존재할 수도 있다 [e.g. sys.path.insert(0, "theZipFile.zip")].

Packages (>1.5): 꾸러미(package)는 모듈(들)과 특수한 초기화 모듈 __init__.py (비어있어도 됨)를 포함하는 디렉토리에 짝지어진 이름 공간이다.
Packages/directories 는 내포되어도 좋다. [package.[package...].module.symbol 를 통하여 모듈의 심볼에 접근하면 된다.
[1.51: 매킨토시 & 윈도우즈에서, 모듈 파일 이름의 대소문자 구분은 이제 수입(import) 서술문에서 사용된 대소문자와 격이 일치해야 한다]

이름 공간 서술문

서술문	결과
import module1 [as name1] [, module2]*	모듈을 수입한다. 모듈의 구성원들이 참조되려면 반드시 [package.]module 이름으로 자격이 부여되어야 한다, 예.: import sys; print sys.argv import package1.subpackage.module package1.subpackage.module.foo() module1 이 공급되면 name1 로 이름이 바뀐다.
from module import name1 [as othername1] [, name2]*	현재 이름공간에 있는 모듈(module)로부터 이름을 수입한다. from sys import argv; print argv from package1 import module; module.foo() from package1.module import foo; foo() name1 이 주어지면 othername1 로 이름이 바뀐다.
from module import *	module 에서 모든 이름을 수입한다, 단 "_"로 시작하는 이름은 제외한다 되도록이면 사용하지 말 것, 이름 충돌을 주의하자! from sys import *; print argv from package.module import *; print x 모듈의 최상위에서만 적법하다. 모듈(module)에서 __all__ 속성이 정의되면, __all__에 나열된 이름들만 수입된다. NB: "from package import *"는 오직 꾸러미의 __init__.py 파일에 정의된 심볼만 수입한다. 꾸러미의 모듈들에 있는 심볼들이 아니다!
global name1 [, name2]	이름은 지역영역이 아니라 (보통 함수에서만 의미가 있다) 전역 영역에서 온다 (보통 모듈로부터 온다는 의미이다). 예. global 서술문이 없는 함수에서, "x"는 지금까지 함수나 모듈에서 사용된 적이 없는 이름으로 간주된다: - "x"를 읽으려고 시도하면 -> NameError - "x"에 쓰려고 시도하면 -> 함수에 지역적으로 "x"를 만든다 "x"가 함수에 정의되지 않고, 모듈에 있다면: - "x"를 읽으려고 시도하면, 모듈에서 값을 얻는다 - "x"에 쓰려고 시도하면, 함수에 지역적으로 "x"를 만든다 그러나 주목하자. "x[0]=3"은 "x"에 대한 탐색을 시작하고, 지역에서 "x"가 없으면 전역에서 "x"를 사용할 것이다.

함수 정의

def func_id ([param_list]):
    suite
함수 객체를 만들어서 이름 func_id 에 묶는다.
param_list ::= [id [, id]*]
id ::= value | id = value | *id | **id
인수들은 값(value)으로 건네진다. 그리하여 변경가능(mutable) 객체를 표현하는 인수들만 변경할 수 있다 (입출력(inout) 매개변수). 하나 이상의 값을 반환하려면 터플(tuple)을 사용하자.

예제:
 def test (p1, p2 = 1+ 1, *rest, **keywords):
매개변수에 "="가 있으면 기본값이 있다 (v 는 함수가 정의될 때 평가된다).
리스트에 "*id"가 있으면 id 에는 (C 의 vararg 처럼) 함수에 건네지는 남은 모든 인수들이 담긴 터플이 할당된다. 리스트에 "**id"가 있으면 id 에는 키워드로 건네지는 나머지 모든 인수들이 담긴 사전이 할당된다.

클래스 정의

class <class_id> [(<super_class1> [,<super_class2>]*)]:
  <suite>
클래스 객체를 만들고 거기에다 <class_id>이라는 이름을 할당한다.
<suite>에는 지역적으로 클래스 메쏘드를 정의("defs")하고 클래스 속성을 할당하여도 된다.

예제:
class my_class (class1, class_list[3]): ...
"class1" 그리고 어떻게 평가되든지 "class_list[3]" 클래스 객체로부터 상속을 받아 클래스 객체를 만든다. 새로운 클래스 객체를 "my_class"라는 이름에 할당한다.

- 클래스 메쏘드에 건네지는 첫 인수는 언제나 실체 객체이며, 관례상 'self'라고 부른다.
- 특수 메쏘드 __init__()은 실체가 만들어질 때 호출된다.
- 특수 메쏘드 __del__()은 객체에 더 이상 참조가 없을 때 호출된다.
- 클래스 객체를 호출하면 "(calling)" 실체가 만들어진다. 인수를 가질 수도 있다. (그리하여 instance=apply(aClassObject, args...)는 실체를 만든다!)
- 현재의 구현 상태(역주:2.1)에서는, 내장 클래스를 하부클래스화할 수 없다. 그러나 "감쌀(wrap) 수"는 있다, UserDict & UserList 모듈을 참조하고, 아래에서 __getattr__()을 참조하자.

예제:
```
class c (c_parent):
    def __init__(self, name):
        self.name = name
    def print_name(self):
        print "I'm", self.name
    def call_parent(self):
        c_parent.print_name(self)

instance = c('tom')
print instance.name
'tom'
instance.print_name()
"I'm tom"
```

부모의 메쏘드에 직접 접근해서 "self"를 명시적으로 건네면 부모의 상위 클래스를 호출할 수 있다. (위의 예제에서 "call_parent" 참조).
기타 많은 특수 메쏘드들을 사용하여 수치 연산자, 연속열, 짝짓기, 지표화, 등등...에 구현할 수 있다...

유형 / 클래스의 통일

기본 유형들 int, float, str, list, tuple, dict 그리고 file 은 이제 (2.2) 기본 클래스 (object)로부터 파생된 클래스(classes) 처럼 행위하고, 하부클래스화(subclassed) 해도 좋다:

```
x = int(2) # 내장된 강제형변환 함수(cast function)는 이제 기본 유형을 위한 구성자(constructor)이다
y = 3  # <=> int(3) (숫자기호는 새로운 기본 유형의 실체이다)
print type(x), type(y) # 둘 다 정수형(int, int)

assert isinstance(x, int) # 다음 isinstance(x, types.IntType)를 대체한다

assert issubclass(int, object) # 기본 유형은 기본 클래스 'object'로부터 파생된다.
s = "hello" # <=> str("hello")
assert isinstance(s, str)

f = 2.3  # <=> float(2.3)
class MyInt(int): pass    # 기본 유형을 하부클래스화해도 좋다
x,y = MyInt(1), MyInt("2")
print x, y, x+y # => 1,2,3
class MyList(list): pass
l = MyList("hello")
print l # ['h', 'e', 'l', 'l', 'o']
```

새로운-스타일의 클래스는 객체(object)를 확장한다. 예전-스타일의 클래스는 그렇지 않다.

문서화 문자열

모듬에서 문자열 기호문자를 첫 서술문으로 두면 모듈, 클래스, 그리고 함수가 문서화 된다. 모듈, 클래스 또는 함수에서 '__doc__' 속성을 얻으면 그 문서화를 열람할 수 있다.

예제:
```
class C:
    "C 에 대한 설명"
        def __init__(self):
            "구성자에 대한 설명"
            # 등등.

c.__doc__ == "C 에 대한 설명".
c.__init__.__doc__ == "구성자에 대한 설명"
```

반복자

- 반복자(iterator)는 모음집(collection)의 요소를 열거한다. 단 한개의 메쏘드 next()를 가지는 객체로서 다음 요소를 반환하거나 StopIteration 을 일으킨다.

- 객체(obj)에 대한 반복자는 새로운 내장 함수 iter(obj)를 통하여 얻을 수 있는데, 이 함수는 obj.__class__.__iter__()를 호출한다.

- __iter__()와 next()를 구현하면, 모음집은 그 스스로 반복자일 수 있다.

- 내장 모음집 (리스트, 터플, 문자열, 사전)은 __iter__()를 구현한다; 사전 (짝짓기)는 자신의 키를 열거한다; 파일은 자신의 줄을 열거한다.

- 파이썬은 회돌이(loop)를 해야할 때마다 묵시적으로 반복자를 사용한다:

    - for elt in collection:

    - if elt in collection:

    - 터플을 할당할 때: x,y,z= collection

발생자

- 발생자(generator)는 두 번의 호출사이에서 그의 상태를 유지하고 각 요청마다 새로운 값을 생산하는 함수이다. 그 값들은 (한 번에 하나씩) 새로운 키워드 yield 를 사용하여 반환되며, 값의 끝을 알리기 위해 return 이 사용되거나 StopIteration()이 일어난다.

- 전형적인 사용법은 ID, 이름, 또는 일련의 숫자들의 생산에 사용된다.

- 발생자를 사용하려면: 발생자 함수(generator function)를 호출하여 발생자 객체를 확보한 다음, generator.next()를 호출하여 StopIteration 이 일어날 때까지 다음 값을 확보한다.

- 2.2 에서, 이 특징을 사용하려면 서술문으로 가능하도록 만들 필요가 있었다: from __future__ import generators (2.3+ 이후로 불필요하다)

예제:

```
def genID(initialValue=0):
  v = initialValue
  while v < initialValue +  1000:
    yield "ID_%05d" % v
    v + = 1
  return    # 또는: StopIteration()을 일으킨다

generator = genID() # 발생자를 만든다
for i in range(10): # 값을 10 개 만든다
  print generator.next()
```

기술자/속성 접근

- 기술자(Descriptors)는 클래스의 속성과 메쏘드를 기술하는 객체로서, __get__, __set__, 등등의 메쏘드를 통하여 조작이 가능하다... 파이썬은 이제 투명하게 기술자들을 사용하여 데이터 멤버에 접근한다.

- 특수한 기술자를 사용하면 이제 다음과 같이 정의가 가능하다.:

  - 정적 메쏘드(Static methods) : C 에서의 static 과 같다 => staticmethod(f)를 사용하면 메쏘드 f(x)를 정적(static)으로 만들 수 있다.

  - 클래스 메쏘드(Class methods): static 과 같지만 클래스를 첫 인수로 받는다 => f = classmethod(f)를 사용하면 메쏘드 f(theClass, x)를 클래스 메쏘드로 만들 수 있다.

  - Properties (in Delphi/Corba/JavaBean style). 특성(property)은 새로운 내장 유형 property 의 실체로서, 속성에 대하여 기술자(descriptor) 프로토콜을 구현한다 => 클래스의 안쪽이나 바깥에 속성을 정의하려면 다음과 같이 사용할 것 propertyName = property(getter=None, setter=None,deleter=None,description=None).

  - 슬롯(Slots). 새로운 스타일의 클래스는 클래스 속성 __slots__을 정의하여 할당가능한 (assignable) 속성 이름의 리스트를 제약할 수 있다. 타자실수(typos)를 예방하기 위함. (보통은 파이썬에서 탐지되지 않고 새로운 속성이 생성된다), 예. __slots__ = ('x', 'y') 주의: 최근의 토론에 의하면, 슬롯의 실제 목적은 여전히 불명확해 보이며 (최적화 때문인가?), 아마도 사용을 삼가해야 할 것 같다.

Misc

lambda [param_list]: returnedExpr
익명 함수를 만든다.
반환된 표현식(returnedExpr)은 서술문이 아니라, 표현식어야 한다 (예, "if xx:...", "print xxx", 등등 불가) 그리하여 새줄문자(newlines)가 포함되면 안된다. 주로 사용되는 곳은 filter(), map(), reduce() functions, 그리고 GUI 역호출이다.

리스트 통합(List comprehension) : 역주 - 통합이란 내포하는 능력(The power of including)

```
result = [expression for item1 in sequence1  [if condition1]
          [for item2 in sequence2 ... for itemN in sequenceN]
         ]
is equivalent to:
result = []
for item1 in sequence1:
   for item2 in sequence2:
      ...
      for itemN in sequenceN:
         if (condition1) and further conditions:
            result.append(expression)
```

내포된 영역(Nested scopes)
2.2 에서, 내포된 영역은 더 이상 from __future__ import nested_scopes 지시어로 특별하게 설정가능하도록 만들 필요가 없으며, 현재는 기본으로 존재한다.

내장 함수

내장 함수

함수	결과
__import__(name[, globals [,locals[,from list]]])	주어진 맥락안에서 모듈을 수입한다 (자세한 것은 라이브러리 레퍼런스 참조)
abs(x)	숫자 x 의 절대값을 돌려준다.
apply(f, args[, keywords])	args 인수와 선택적인 키워드를 가지고 함수/메쏘드 f 를 호출한다.
buffer(object[, offset[, size]])	object 의 조각으로부터 Buffer 를 돌려준다, 이는 버퍼 호출 인터페이스를 지원해야 한다 (String, array, buffer).
callable(x)	x 가 호출가능하면 1 이고, 그렇지 않으면 0 이다.
chr(i)	그의 ASCII 코드가 정수 i 인 한-문자 문자열을 돌려준다.

classmethod(function)	함수(function)에 대한 클래스 메쏘드를 돌려준다. 클래스 메쏘드는 마치 실체 메쏘드가 그 실체를 받는 것처럼, 클래스를 묵시적인 첫 인수로 받는다. 클래스 메쏘드를 선언하려면, 다음 관용구를 사용하자:
	
	class C:
	  def f(cls, arg1, arg2, ...): ...
	  f = classmethod(f)
	
	그리고 클래스에 대하여 C.f()로 호출하거나 실체에 대하여 C().f()로 호출하자. 실체는 자신의 클래스가 없으면 무시된다. 클래스 메쏘드가 파생된 클래스에 대하여 호출되면, 그 파생된 클래스 객체가 묵시적으로 첫 인수로 건네진다.

cmp(x,y)	x < y 이면 -1, x == y 이면 0, x > y 이면 + 1 을 돌려준다
coerce(x,y)	두 수치(numeric) 인수를 공통 유형으로 변환하여 터플로 돌려준다.
compile(string, filename, kind[, flags[, dont_inherit]])	string 을 코드 객체로 컴파일한다. filename 은 에러 메시지에 사용되고, 아무 문자열이나 된다. 보통은 그 코드가 읽힌 파일이며, 또는 파일에서 읽지 않았다면 문자열('<string>')이다. kind 는 문자열(string)이단일 서술문이면 'single'이 될 수 있다. 또는 kind 가 'eval'이 되면 None 말고 다른 어떤 것으로 평가되는 표현식 서술문의 출력결과를 인쇄한다. 또는 모듈로 컴파일하려면 'exec'이 될 수 있다. 새로운 인수 flags 와 dont_inherit 은 future 서술문들에 관련된다.
complex(real[, image])	복소수(complex) 객체를 만든다 (또는 J 나 j 를 접미사로 사용해도 된다. 예. 1+ 3J).
delattr(obj, name)	객체 obj 에서 name 이라는 이름의 속성을 제거한다 <=> del obj.name
dict([mapping-or-sequence])	선택적 인수로 초기화된 새로운 사전을 반환환다 (또는 인수가 없으면 빈 사전을 돌려준다). 인수는 (키,값) 상의 연속열이면 (혹은 반복가능한 어떤 것이라도) 된다.
dir([object])	인수가 없으면, 현재 지역 심볼 테이블에 있는 이름들의 리스트를 돌려준다. 모듈, 클래스, 또는 클래스 실체 객체가 인수(arg)로 주어지면, 그의 속성(attr.) 사전에 있는 이름들의 리스트를 돌려준다.
divmod(a,b)	다음 터플을 돌려준다 (a/b, a%b)
enumerate(iterable)	iterable 의 (index, value) 쌍을 반환하는 반복자, 예. List(enumerate ('Py')) -> [(0, 'P'), (1, 'y')].
eval(s[, globals[, locals]])	문자열 s 를 (선택적인) globals, locals 맥락에서 평가한다. s 는 절대로 NUL 이나 새줄문자(newlines)를 가지면 안된다. s 는 코드 객체가 될 수도 있다. 예제: x = 1; incr_x = eval('x + 1')
execfile(file[, globals [,locals]])	import 와는 다르게, 새로운 모듈을 만들지 않고 파일을 실행한다.

내장 함수	
file(filename[,mode [,bufsize]])	파일을 하나 열고 새로운 파일(file) 객체를 돌려준다. open 을 대체한다. • filename 은 열릴 파일 이름이다 • mode 는 그 파일이 어떤 상태로 열릴 것인지 나타낸다: o 'r' 읽기 o 'w' 쓰기 (기존 파일을 잘라낸다) o 'a' 추가 o '+' (이전에 어떤 모드이든지 추가된다) 갱신을 위해 파일을 연다 ('w+'는 파일을 잘나냄을 주의) o 'b' (이전에 어떤 모드이든지 추가된다) 파일을 이진 모드로 연다 o 'U' (또는 'rU')는 파일을 범용 새줄문자 모드 (Universal Newline mode)로 읽기 위해 연다: EOL 의 변형들(CR, LF, CR+LF)은 모두 단 하나의 LF ('₩n')로 변환된다. • 버퍼크기(bufsize)는 버퍼를 사용하지 않으면(unbuffered) 0, 줄단위-버퍼에는 1, 시스템-기본은 음수이며, 다른 것이면 (대략) 주어진 크기이다.
filter(function,sequence)	연속열(sequence)에서 함수(function)가 참을 반환하는 요소들로부터 리스트를 구성한다. 함수(function)는 매개변수를 한개 취한다.
float(x)	숫자나 문자열을 부동소수점 수로 변환한다.
getattr(object,name [,default]))	객체(object)로부터 name 이라고 부르는 속성을 얻는다. 예, getattr(x, 'f') <=> x.f). 발견되지 않으면 AttributeError 를 일으킨다. 또는 기본값(default)이 지정되어 있으면 그것을 돌려준다.
globals()	현재 전역 변수들이 담긴 사전을 돌려준다.
hasattr(object, name)	객체(object)에 name 이라고 부르는 속성이 있으면 참을 돌려준다.
hash(object)	객체의 해쉬값을 돌려준다 (해쉬값을 가지고 있으면).
help([object])	내장 도움말 시스템을 요청한다. 인수가 없으면 -> 상호대화 도움말 상태로 들어간다; 객체(object)가 문자열이면 (모듈, 함수, 클래스, 메쏘드, 키워드, 또는 문서 주제의 이름(name)), 도움말 페이지가 콘솔에 인쇄된다; 그렇지 않으면 객체(object)에 대한 도움말 페이지가 만들어진다.
hex(x)	숫자 x 를 16 진수 문자열로 변환한다.
id(object)	object 에 대하여 유일한 정수 식별자를 돌려준다.

내장 함수	
input([prompt])	주어진다면 prompt 를 인쇄한다. 입력을 읽어서 평가한다(evaluates). readline 모듈을 사용할 수 있으면 줄 편집/히스토리를 사용한다.
int(x[, base])	숫자나 문자열을 평범한 정수로 변환한다. 선택적인 밑수(base) 매개변수에 변환할 진법을 지정하여 문자열 값으로 변환한다.
intern(aString)	aString 을 내부화된 문자열 테이블에 넣고 그 문자열을 돌려준다. 2.3 이전에는, 내부화된 문자열은 '불멸'이었다 (쓰레기 수집이 되지 않음). 2.3+ 에서부터는 적용되지 않는다.
isinstance(obj, classInfo)	obj 가 class 의 실체이면 참을 돌려준다. classInfo 또는 classInfo 유형(type) 객체 (클래스정보(classInfo)는 클래스나 유형의 터플(tuple)이 될 수도 있다). If issubclass(A,B) then isinstance(x,A) => isinstance(x,B)
issubclass(class1, class2)	class1 이 class2 으로부터 파생되었다면 참을 돌려준다 (또는 class1 이 class2 와 같으면(is)).
iter(obj[,sentinel])	객체(obj)에 대한 반복자(iterator)를 돌려준다. 경계표식(sentinel)이 없으면, 객체(obj)는 __iter__() 또는 __getitem__()를 구현한 모음집이어야 한다. 경계표식(sentinel)이 있으면, 객체(obj)가 인수 없이 호출될 것이다; 반환된 값이 경계표식(sentinel)과 같다면, 반복멈춤(StopIteration)이 일어날 것이고, 그렇지 않으면 그 값이 반환될 것이다. 반복자(Iterators) 참조.
len(obj)	객체의 길이(항목의 개수)를 돌려준다 (연속열, 사전, 또는 __len__을 구현한 실체들).
list(sequence)	연속열(sequence)을 리스트로 변환한다. 이미 리스트라면 그의 사본을 돌려준다.
locals()	현재 지역 변수들을 담은 사전을 돌려준다.
long(x[, base])	숫자나 문자열을 긴 정수로 변환한다. 선택적인 밑수(base) 매개변수에 변환할 진법을 지정하여 문자열 값을 변환한다.
map(function, list, ...)	함수(function)를 리스트(list)의 모든 항목에 적용하고 그 결과 리스트를 돌려준다. 추가적인 인수들이 건네지면, 함수(function)는 그 개수만큼의 인수를 취해야 한다. 그리고 그 인수들은 각 호출 때마다 함수(function)에 건네진다.
max(seq[, args...])	인수로 seq 한 개만 주어지면, (문자열, 터플, 또는 리스트같은) 비어있지 않은 연속열에서 가장 큰(largest) 항목을 돌려준다. 인수가 여러개이면, 가장 큰 인수를 돌려준다.
min(seq[, args...])	인수로 seq 하나이면, (문자열, 터플, 또는 리스트 같은) 비어있지 않은 연속열에서 가장 작은(smallest) 항목을 돌려준다. 인수가 여러개이면, 가장 작은 인수를 돌려준다.
oct(x)	숫자를 8 진 문자열로 변환한다.
open(filename [, mode='r', [bufsize=implementation dependent]])	새로운 파일 객체를 돌려준다. 새로운 함수 file()의 별칭.
ord(c)	정수 c 의 ASCII 값을 돌려준다 (길이 1 인 문자열). 유니코드 문자와 작동한다.
pow(x, y [, z])	x 를 y 제곱하여 돌려준다 [modulo z]. 또 ** 연산자 참조.
range(start [,end [, step]])	정수로 이루어진 리스트를 돌려준다(시작이상(>=) 끝미만(<)). 인수가 1 개면, 0..arg-1 리스트 인수가 2 개면, start..end-1 리스트 인수가 3 개면, step 만큼 건너뛴 start..end 리스트
raw_input([prompt])	프롬프트(prompt) 주어지면 인쇄하고, 문자열을 표준 입력으로부터 읽는다(₩n 이 따르지 않음). 또 input() 참조.
reduce(f, list [, init])	이진 함수 f 를 리스트(list)의 항목들에 적용하여서 그 리스트를 단 한 개의 값으로 축소한다. 초기값(init)이 주어지면, 리스트(list)에 "미리 고려된다".

내장 함수	
reload(module)	이미 적재된 모듈을 다시 해석하고 다시 초기화한다. 상호대화 모드에서 모듈을 고친다음 재적재하고 싶을 때 쓸모가 있다. 모듈이 구문적으로 옳지만 초기화에 에러가 있다면, 한 번 이상 그 모듈을 적재하고 나서 reload()를 호출해야 한다.
repr(object)	한 객체의 인쇄가능한 그리고 가능하면 평가가능한 표현을 담은 문자열을 돌려준다. <=> `object` (역인용부호를 사용하여). 클래스는 재정의 가능하다 (__repr__). 또 str() 참조
round(x, n=0)	부동소수점 값 x 를 10 진 소수점 이후로 n 자리수만큼 버리고 돌려준다.
setattr(object, name, value)	이는 getattr()의 대응짝이다. setattr(o, 'foobar', 3) <=> o.foobar = 3. 존재하지 않으면 속성들을 만든다!
slice([start,] stop[, step])	범위를 표현하는 조각 객체(slice object)를 돌려준다, 다음 R/O 속성을 가진다: start, stop, step.

staticmethod(function)	함수(function)에 대한 정적 메쏘드를 돌려준다. 정적 메쏘드는 묵시적인 첫 인수를 받아들이지 않는다. 정적 메쏘드를 선언하려면, 다음 관용구를 사용하자:
	
	class C:
	  def f(arg1, arg2, ...): ...
	  f = staticmethod(f)
	
	그리고 나서 클래스에 대하여 C.f()을 호출하거나 실체에 대하여 C().f()를 호출하자. 실체는 자신의 클래스가 없다면 무시된다.

str(object)	한 객체의 깔끔하게 인쇄가능한 표현을 담은 문자열을 돌려준다. 클래스 오버라이드 가능 (__str__). 참조 repr().
sum(iterable[, start=0])	숫자로된 연속열의 합을 돌려준다 (문자열이 아님), 매개변수의 값을 더한다. 연속열이 비어 있으면 start 를 돌려준다.

super(type[, object-or-type])	유형(type)의 상위클래스를 돌려준다. 두 번째 인수가 생략되면 반환된 상위 객체는 풀린다(unbound). 두 번째 인수가 객체라면 , isinstance(obj, type)는 반드시 참이어야 한다 두 번째 인수가 유형이라면, issubclass(type2, type)는 반드시 참이어야 한다. 전형적인 사용법:
	class C(B):
	  def meth(self, arg):
	  super(C, self).meth(arg)

tuple([sequence])	빈 터플이나 연속열(sequence)과 똑 같은 요소들을 가지는 터플을 돌려준다. 연속열(sequence)은 연속열, 반복을 지원하는 그릇(container) 또는 반복자 객체가 될 수 있다. 연속열(sequence)이 이미 터플이면, (그의 사본이 아니라) 자기 자신을 돌려준다.
type(obj)	객체(obj)의 유형을 표현하는 유형 객체(type object)를 돌려준다 [모듈 types 참조]. 예제: import types if type(x) == types.StringType: print 'It is a string'. NB: 대신에 다음과 같이 사용하는 것이 더 좋다: if isinstance(x, types.StringType)...
unichr(code)	주어진 코드(code)로 1 문자 길이의 유니코드 문자열을 돌려준다.
unicode(string[, encoding[,error]]])	8-bit 문자열으로부터 유니코드 문자열을 만든다, 주어진 인코딩 이름과 에러 처리법('strict', 'ignore',또는 'replace')을 사용한다. __unicode__() 메쏘드를 제공하는 객체들에 대해서는, 인수 없이 이 메쏘드를 호출하여 유니코드 문자열을 만든다.
vars([object])	인수가 없으면, 현재 지역 심볼 테이블에 상응하는 사전을 돌려준다. 모듈, 클래스나 클래스 객체가 인수로 주어지면, 그 객체의 심볼 테이블에 상응하는 사전을 돌려준다. "%" 문자열 형식화 연산자와 함께 사용하면 쓸모가 많다.
xrange(start [, end [, step]])	range()와 비슷하지만, 실제로 전체 리스트를 한꺼번에 저장하지 않는다. 범위는 크고 메모리는 작을 때 "for" 회돌이에서 사용하면 좋다.
zip(seq1[, seq2,...])	각 인수 연속열에서 n 번째 요소가 담긴 터플의 리스트를 돌려준다.

내장 예외 클래스

Exception
모든 예외의 어머니. exception.args 은 그 구성자에게 건네어진 인수들이 담긴 터플이다.

- StopIteration
  반복자의 next() 메쏘드에 의하여 발생하여 더 이상 값들이 없다는 신호를 보내다.

- SystemExit
  sys.exit()시

- Warning
  경고에 대한 바탕 클래스 (모듈 warning 참조)

  - UserWarning
    사용자 코드에 의하여 만들어진 경고.

  - PendingDeprecationWarning
    앞으로 권고되지 않는 코드에 관한 경고.

  - DeprecationWarning
    비추천되는 코드에 관한 경고.

  - SyntaxWarning
    모호한 구문에 관한 경고.

  - RuntimeWarning
    모호한 실행시간 행위에 관한 경고.

- StandardError
  모든 내장 예외에 대한 기본 클래스; Exception 루트 클래스로부터 파생된다.

  - ArithmeticError
    산술연산 에러에 대한 기본 클래스.

    - FloatingPointError
      부동 소수점 연산이 실패할 때.

    - OverflowError
      과도하게 거대한 산술 연산 때.

        - ZeroDivisionError
          2 번째 인수로 0 을 가지고 나누기나 나머지 연산을 할 때.

    - AssertionError
      assert 서술문이 실패할 때.

    - AttributeError
      속성 참조 실패나 할당 실패 때

- EnvironmentError [1.5.2 에서 도입]
  파이썬 밖에서 에러가 있을 때; 에러 인수 터플은 다음과 같다 (errno, errMsg...)

    - IOError [changed in 1.5.2]
      I/O-관련 연산 실패.

    - OSError [new in 1.5.2]
      os 모듈의 os.error 예외에서 사용됨.

        - WindowsError
          윈도우즈-종속적 에러가 발생할 때나 에러 번호가 errno 값에 상응하지 않을 때.

- EOFError
  input() 또는 raw_input()에 의해 직접 파일-의-끝 도달

- ImportError
  모듈이나 이름을 발견하지 못해 수입(import) 실패시.

- KeyboardInterrupt
  사용자가 인터럽트 키를 입력했을 때 (보통 `CTRL-C')

- LookupError
  IndexError, KeyError 에 대한 바탕 클래스

    - IndexError
      연속열의 아래첨자 범위를 벗어날 때

    - KeyError
      존재하지 않는 짝짓기 (사전) 키를 참조할 때

- MemoryError
  회복가능한 메모리 소진시

- NameError
  지역이나 전역 (자격없는) 이름의 발견에 실패시.

    - UnboundLocalError
      할당되지 않은 지역 변수를 참조시.

- ReferenceError
  약한 참조 프록시를 통하여 쓰레기-수집된 객체에 접근을 시도할 때.

- RuntimeError
  쓸모없는 다목적(catch-all) 에러; 대신에 적당한 에러를 정의하자.

    - NotImplementedError [1.5.2 에서 처음 도입]
      메쏘드가 구현되지 않았을 때.

- SyntaxError
  해석기가 구문 에러를 맞을 때

    - IndentationError
      해석기가 들여쓰기 구문 에러를 맞을 때

    - TabError
      해석기가 들여쓰기 구문 에러를 맞을 때

- SystemError
  치명적이지 않은 인터프리터 에러 발생시 - 버그이므로 - 보고하자

- TypeError
  부적절한 유형을 내장 연산자나 함수에 건넬 때.

- ValueError
  인수를 TypeError 가 제대로 다루지 못할 때 나는 에러.

    - UnicodeError
      유니코드-관련 인코딩이나 디코딩 에러시.

클래스에서의 표준 메쏘드 & 연산자 재정의

표준 메쏘드 & 연산자는 특수한 메쏘드 '__method__'에 짝지워져 있으며 그리하여 재정의가 가능하다 (대부분은 사용자-정의된 클래스에서), 예:

```
class C:
    def __init__(self, v): self.value = v
    def __add__(self, r): return self.value + r

a = C(3) # C.__init__(a, 3)로 호출하는 것과 비슷
a + 4    # a.__add__(4)와 동등
```

클래스에 대한 특수 메쏘드

메쏘드	설명
__init__(self, args)	실체 초기화 (구성시)
__del__(self)	객체가 사망할 때 (참조횟수가 0 이 될 때) 호출된다
__repr__(self)	repr() 그리고 `...` 변환
__str__(self)	str() 그리고 print 서술문
__cmp__(self,other)	self 와 other 를 비교해서 <0, 0, 또는 >0 을 돌려준다. >, <, == 등등을 구현한다...
__lt__(self, other)	self < other 비교에 호출된다. 무엇이든 돌려줄 수 있으며, 또는 예외를 일으킬 수 있다.
__le__(self, other)	self <= other 비교에 호출된다. 무엇이든 돌려줄 수 있으며, 또는 예외를 일으킬 수 있다.
__gt__(self, other)	self > other 비교에 호출된다. 무엇이든 돌려줄 수 있으며, 또는 예외를 일으킬 수 있다.
__ge__(self, other)	self >= other 비교에 호출된다. 무엇이든 돌려줄 수 있으며, 또는 예외를 일으킬 수 있다.
__eq__(self, other)	self == other 비교에 호출된다. 무엇이든 돌려줄 수 있으며, 또는 예외를 일으킬 수 있다.
__ne__(self, other)	self != other 비교 (그리고 self <> other 비교)에 호출된다. 무엇이든 돌려줄 수 있으며, 또는 예외를 일으킬 수 있다.
__hash__(self)	32 비트 해쉬 코드를 계산한다; hash() 그리고 사전 연산
__nonzero__(self)	이 메쏘드가 정의되어 있지 않으면, 진리 값 테스트에 대하여 0 또는 1 을 돌려준다. 정의되어 있으면 __len__()이 호출된다; 그렇지 않으면 모든 클래스 실체는 "참"으로 간주된다.
__getattr__(self,name)	속성 참조가 name 을 찾지 못할 때 호출된다. __getattribute__ 참조.
__getattribute__(self, name)	__getattr__과 동일하지만. name 이 접근될 때마다 언제나 호출된다.
__setattr__(self, name, value)	속성을 설정할 때 호출된다 (안에다, "self.name = value"라고 사용하지 말 것, 대신에 "self.__dict__[name] = value"라고 사용하자)
__delattr__(self, name)	속성을 제거하기 위해 호출된다 <name>.
__call__(self, *args, **kwargs)	실체가 함수로 호출될 때 호출된다: obj(arg1, arg2, ...)는 obj.__call__(arg1, arg2, ...)을 짧게 쓴 것이다.

연산자

연산자(operator) 모듈에 있는 목록 참조. 연산자 함수 이름은, 이끄는 & 따르는 '__'가 있는 것과 없는 것 2 가지 변종으로 제공된다 (예. __add__ 또는 add).

수치 연산 특수 메쏘드	
연산자	특수 메쏘드
self + other	__add__(self, other)
self - other	__sub__(self, other)
self * other	__mul__(self, other)
self / other	__future__.division 이 활성화 되어 있으면 __div__(self, other) 또는 __truediv__(self,other)이다.
self // other	__floordiv__(self, other)
self % other	__mod__(self, other)
divmod (self,other)	__divmod__(self, other)
self ** other	__pow__(self, other)
self & other	__and__(self, other)
self ^ other	__xor__(self, other)
self | other	__or__(self, other)
self << other	__lshift__(self, other)
self >> other	__rshift__(self, other)
nonzero(self)	__nonzero__(self) (불리언 테스트에서 사용된다)
-self	__neg__(self)
+ self	__pos__(self)
abs(self)	__abs__(self)
~self	__invert__(self) (bitwise)
self += other	__iadd__(self, other)
self -= other	__isub__(self, other)
self *= other	__imul__(self, other)
self /= other	__future__.division 이 작용하면, __idiv__(self, other) 또는 __itruediv__(self,other)이다.
self //= other	__ifloordiv__(self, other)
self %= other	__imod__(self, other)
self **= other	__ipow__(self, other)
self &= other	__iand__(self, other)
self ^ = other	__ixor__(self, other)
self |= other	__ior__(self, other)
self <<= other	__ilshift__(self, other)
self >>= other	__irshift__(self, other)

변환	
내장 함수	특수 메쏘드
int(self)	__int__(self)
long(self)	__long__(self)
float(self)	__float__(self)
complex(self)	__complex__(self)
oct(self)	__oct__(self)
hex(self)	__hex__(self)
coerce(self, other)	__coerce__(self, other)

모든 이진 연산자에 대하여 동등물은 오른-쪽에 존재한다; 연산자의 오른쪽에 클래스 실체가 있으면 호출된다:

- a + 3 calls __add__(a, 3)

- 3 + a calls __radd__(a, 3)

그릇(containers)에 대한 특수 연산		
연산	특수 메쏘드	주의
모든 연속열 그리고 짝짓기 :		
len(self)	__len__(self)	객체의 길이, >= 0. Length 0 == false
self[k]	__getitem__(self, k)	지표 위치에서 요소를 얻는다 /key k (지표는 0 에서 시작한다). 또는, k 가 조각 객체(slice object)이면, 조각을 돌려준다.
self[k] = value	__setitem__(self, k, value)	k 지표/키/조각 위치에 요소를 설정한다 .
del self[k]	__delitem__(self, k)	k 지표/키/조각 요소를 삭제한다.
elt in self elt not in self	__contains__(self, elt) not __contains__(self, elt)	연속열을 통한 표준 반복보다 효율적이다.
iter(self)	__iter__(self)	요소들에 대하여 반복자를 돌려준다 (짝짓기용 키 <=> self.iterkeys()). 참조 반복자(iterators).
연속열, 일반 메쏘드, 등등:		
self[i:j]	__getslice__(self, i, j)	2.0 이후로 비추천 된다, 조각(slice) 객체를 매개변수로 한 __getitem__로 대체됨.
self[i:j] = seq	__setslice__(self, i, j,seq)	2.0 이후로 비추천 된다, 조각(slice) 객체를 매개변수로 한 __setitem__으로 대체됨.
del self[i:j]	__delslice__(self, i, j)	다음가 동일 self[i:j] = [] - 2.0 이후로 비추천 된다, 조각(slice) 객체를 매개변수로 한 __delitem__으로 대체됨.
self * n	__repeat__(self, n)	
self + other	__concat__(self, other)	
짝짓기(Mappings), 일반 메쏘드, 등등:		
hash(self)	__hash__(self)	객체 self 를 해쉬한 값은 사전의 키로 사용된다

유형(Type)에 대한 특수한 정보 상태 속성:

팁: 살아있는 객체의 내부를 보려면 inspect 모듈을 사용하자.

리스트 & 사전	
속성	의미
~~__methods__~~ ~~=~~	~~(list, R/O): 객체의 메쏘드 이름 리스트~~ 비추천됨, 대신 dir()을 사용하자

모듈	
속성	의미
__doc__	(string/None, R/O): 문서화 문자열 (<=> __dict__['__doc__'])
__name__	(string, R/O): 모듈 이름 (또한 __dict__['__name__'])
__dict__	(dict, R/O): 모듈의 이름 공간
__file__	(string/undefined, R/O): .pyc, .pyo 또는 .pyd 파일의 경로이름 (파이썬에 정적으로 연결된 모듈들로 (역주:내장 모듈로) 정의되어 있지 않은)
__path__	(list/undefined, R/W): 꾸러미를 찾을 디렉토리 경로의 리스트 (꾸러미 전용).

클래스

속성	의미
__doc__	(string/None, R/W): 문서화 문자열 (<=> __dict__['__doc__'])
__name__	(string, R/W): 클래스 이름 (또한 __dict__['__name__'])
__bases__	(tuple, R/W): 부모 클래스
__dict__	(dict, R/W): 속성 (클래스 이름 공간)

실체

속성	의미
__class__	(class, R/W): 실체의 클래스
__dict__	(dict, R/W): 속성

사용자 정의 함수

속성	의미
__doc__	(string/None, R/W): 문서화 문자열
__name__	(string, R/O): 함수 이름
func_doc	(R/W): __doc__과 같다
func_name	(R/O): __name__과 같다
func_defaults	(tuple/None, R/W): 있다면 기본 인수 값
func_code	(code, R/W): 컴파일된 함수 몸체를 표현하는 코드 객체
func_globals	(dict, R/O): 함수 전역 변수를 담은 사전을 참조한다

사용자-정의 메쏘드

속성	의미
__doc__	(string/None, R/O): 문서화 문자열
__name__	(string, R/O): 메쏘드 이름 (im_func.__name__과 동일)
im_class	(class, R/O): 메쏘드를 정의하는 클래스 (기본 클래스일 수도 있다)
im_self	(instance/None, R/O): 목표 실체 객체 (발견되지 않으면 None)
im_func	(function, R/O): 함수 객체

내장 함수 & 메쏘드

속성	의미
__doc__	(string/None, R/O): 문서화 문자열
__name__	(string, R/O): 함수 이름
__self__	[메쏘드 전용] 목표 객체
~~__members__~~	~~속성 이름 목록: ['__doc__','__name__','__self__'])~~ 비추천됨, 대신에 dir()을 사용하자

코드

속성	의미
co_name	(string, R/O): 함수 이름
co_argcount	(int, R/0): 위치지정 인수의 개수
co_nlocals	(int, R/O): (인수를 포함한) 지역 변수의 개수
co_varnames	(tuple, R/O): 지역 변수들의 이름 (인수로 시작하는)
co_code	(string, R/O): 바이트코드 지시어의 연속열
co_consts	(tuple, R/O): 바이트코드가 사용하는 기호문자, 첫 기호문자는 함수 이름이다 (또는 None 이다)
co_names	(tuple, R/O): 바이트코드에 의해서 사용되는 이름들
co_filename	(string, R/O): 코드가 컴파일된 파일이름
co_firstlineno	(int, R/O): 함수의 첫 줄 번호
co_lnotab	(string, R/O): 바이트코드 거리를 줄 번호에 인코딩한 문자열.
co_stacksize	(int, R/O): 필요한 스택 크기 (지역 변수 포함)
co_flags	(int, R/O): 파이썬용 표식에서 함수가 "*arg" 구문을 사용하면 2 번 비트가 설정되고, '**keywords' 구문을 사용하면 3 번 비트가 설정된다.

프레임

속성	의미
f_back	(frame/None, R/O): 앞 스택 프레임 (호출자 쪽으로)
f_code	(code, R/O): 이 프레임에서 실행중인 코드 객체
f_locals	(dict, R/O): 지역 변수
f_globals	(dict, R/O): 전역 변수
f_builtins	(dict, R/O): 내장 (내부적인) 이름
f_restricted	(int, R/O): 함수가 제한 모드에서 실행중인지 나타내는 표식
f_lineno	(int, R/O): 현재 줄 번호
f_lasti	(int, R/O): 바로 전에 실행된 지령어의 (바이트 코드에 대한) 지표
f_trace	(function/None, R/W): 각 소스 줄이 시작할 때 호출된 갈고리를 디버그한다
f_exc_type	(Type/None, R/W): 최근의 예외 유형
f_exc_value	(any, R/W): 최근의 예외 값
f_exc_traceback	(traceback/None, R/W): 최근의 예외 역호출

역추적

속성	의미
tb_next	(frame/None, R/O): 스택 추적에서 다음 수준 (예외가 발생한 곳의 프레임 쪽으로)
tb_frame	(frame, R/O): 현재 수준의 실행 프레임
tb_lineno	(int, R/O): 예외가 발생한 곳의 줄 번호
tb_lasti	(int, R/O): 바로 전에 실행된 지령어의 (바이트 코드에 대한) 지표

조각썰기

속성	의미
start	(any/None, R/O): 하한, 포함
stop	(any/None, R/O): 상한, 배제
step	(any/None, R/O): 뜀 값

복소수

속성	의미
real	(float, R/O): 실수부분
imag	(float, R/O): 허수부분

xranges

속성	의미
tolist	(내장-메쏘드, R/O): ?

중요한 모듈

Sys 시스템-종속적 매개변수와 함수. [자세한 문서]

sys 변수

변수	내용
argv	파이썬 스크립트에 건네어진 명령어 줄 인수의 리스트. sys.argv[0]는 그 스크립트 이름이다.
builtin_module_names	파이썬에 연결되어 있는 C 로 작성된 모든 모듈의 이름을 담은 문자열 리스트.
byteorder	고유의 바이트 순서, 큰값 종료형('big') 또는 작은값 종료형('little')이다.
check_interval	쓰레드 전환이나 신호에 대한 점검 빈도 (가상 기계 명령어 개수로 측정된다)
copyright	파이썬에 관련된 복사권이 담긴 문자열.
exec_prefix prefix	플랫폼-의존적인 파이썬 파일이 설치되어 있는 루트 디렉토리, 예. 'C:₩₩Python23', '/usr'.
executable	파이썬 실행 이진 파일의 이름 (예. 'C:₩₩Python23₩₩python.exe', '/usr/bin/python')
exitfunc	사용자가 매개변수 없는 함수에 설정할 수 있다. 파이썬이 종료하기 전에 호출된다.
last_type, last_value, last_traceback	예외가 처리되지 못할 때만 설정되고 파이썬은 에러를 인쇄한다. 디버거가 사용한다.
maxint	최대 양의 정수 값. 2.2 이후로 정수와 긴정수가 통일되었다, 그리하여 정수는 한계가 없다.
maxunicode	유니코드 문자에 대하여 지원되는 가장 큰 코드 포인트.
modules	이미 적재된 모듈 사전.
path	외부 모듈에 대한 경로를 탐색한다. 프로그램에 의해서 변경될 수 있다. sys.path[0] == 현재 실행되는 스크립트의 디렉토리.
platform	현재 플랫폼, 예. "sunos5", "win32"
ps1, ps2	상호대화 모드에서 사용되는 프롬프트, 보통 ">>>" 그리고 "..."
stdin, stdout, stderr	I/O 에 사용되는 파일 객체. 여기에다 새로운 파일 객체를 할당하면 방향전환할 수 있다 (또는 어떤 객체라도 된다: stdout/stderr 에 대해서는 write(string) 메쏘드를 가진 객체, 또는 stdin 에 대해서는 readline() 메쏘드를 가진 객체이면 된다). __stdin__,__stdout__ 그리고 __stderr__이 기본값이다.
version	파이썬에 관한 버전 정보가 담긴 문자열.
version_info	파이썬 버전 정보가 담긴 터플 - (major, minor, micro, level, serial).
winver	윈도우즈에서 레지스트리 키를 만드는데 사용되는 버전 번호 (예. '2.2').

sys 함수

함수	결과
displayhook	상호대화 모드에서 제출된 명령어의 출력을 화면에 표시하는데 사용되는 함수 - 기본은 내장 repr()이다. __displayhook__이 원래 값이다.
excepthook	사용자 정의 함수에 설정될 수 있다. 이 함수에 잡지 못한 예외를 모두 건넨다. __excepthook__이 원래 값이다.
exit(n)	상태 n 을 가지고 종료한다 (정상은 0 이고 OK 를 의미한다). SystemExit 예외를 일으킨다 (그래서 프로그램에서 잡을 수도 무시할 수도 있다)
getrefcount(object)	객체의 참조 회수를 돌려준다. 일반적으로 object 가 임시 참조되기 때문에 1 이상으로 예상된다.
setcheckinterval(interval)	파이썬의 쓰레드 전환 간격을 설정한다 (바이트코드 명령어의 개수로 설정되고, 기본은: 2.2 이후로 10 이고, 2.3 이후로 100 이다).
settrace(func)	추적 함수를 설정한다: 코드의 각 줄이 종료하기 전에 호출된다.
setprofile(func)	수행성능 프로파일을 위해 윤곽 함수를 설정한다.
exc_info()	현재 처리되고 있는 예외에 관한 정보; 다음과 같은 터플이다 (exc_type, exc_value, exc_traceback). 경고: 역추적 반환 값을 예외를 처리하고 있는 함수의 지역 변수에 할당하면 순환참조가 야기된다.
setdefaultencoding(encoding)	기본 유니코드 인코딩을 변환한다 - 기본값은 7-비트 ASCII 이다.
getrecursionlimit()	최대 재귀 깊이를 열람한다.
setrecursionlimit()	최대 재귀 깊이를 설정한다 (기본값 1000).

os

기타 운영 체제 인터페이스. [자세한 문서]

어떤 운영체제-종속적 모듈(nt, mac, posix...)이든지 현재 환경에 알맞게 "동의어"가 설정된다. 이 모듈은 가능하면 posix 모듈을 사용한다.
(램버그(M.A. Lemburg)의 유틸리티 platform.py 참조 (현재 2.3+ 에 포함됨)

os 변수

변수	의미
이름	O/S-종속적 모듈 이름 (예. "posix", "mac", "nt")
path	경로 조작을 위한 O/S-종속적 모듈. 유닉스에서, os.path.split() <=> posixpath.split()
curdir	현재 디렉토리를 표현하는데 사용되는 문자열 (예 '.')
pardir	부모 디렉토리를 표현하는데 사용되는 문자열 (예 '..')
sep	디렉토리를 가르는데 사용되는 문자열 ('/' 또는 '₩'). 팁: 이식성 있는 경로를 만들려면 os.path.join()을 사용하자.
altsep	적용가능하다면 교체가능한 가름자 (그렇지 않으면 None)
pathsep	탐색 경로 구성요소를 가르는데 사용되는 문자열 ($PATH 에서와 같이), 예. 윈도우즈에서는 ';' .
linesep	텍스트(text) 파일에 사용되는 줄 가름자, 즉 유닉스에서는 '₩n', 도스/윈에서 '₩r₩n', 맥에서 '₩r'.

os 함수

함수	결과
makedirs(path[, mode=0777])	재귀적인 디렉토리 생성 (필요한 매개 디렉토리를 만든다); 실패하면 os.error 가 일어난다.
removedirs(path)	재귀적인 디렉토리 삭제 (빈 매개 디렉토리를 삭제한다); 디렉토리가 비어있지 않으면 실패한다 (os.error).
renames(old, new)	재귀적인 디렉토리나 파일 이름짓기; 실패하면 os.error 가 일어난다.

posix

Posix OS 인터페이스. [상세한 문서] 이 모듈을 직접 수입하지 말 것, 대신에 os 를 수입하자! (다음 모듈 참조: 파일 복사 & 삭제 함수를 위해서는 shutil 참조)

posix 변수

변수	의미
environ	환경 변수들을 담은 사전, 예. posix.environ['HOME'].
error	POSIX-관련 에러시에 일어나는 에러. 상응하는 값은 에러번호(errno) 코드와 perror() 문자열의 터플이다.

posix 함수

함수	결과
chdir(path)	현재 디렉토리를 경로(path)로 바꾼다.
chmod(path, mode)	경로(path)의 모드를 숫자형 모드(mode)로 바꾼다
close(fd)	posix.open 으로 열린 파일기술자 fd 를 닫는다.
_exit(n)	즉시 종료한다, 메모리 청소 없음, SystemExit 없음, 등등... 자손 프로세스를 종료하려면 이 함수를 사용해야 함.
execv(p, args)	args 를 가지고 p 를 "실행한다"
getcwd()	현재 작업 디렉토리를 표현하는 문자열을 돌려준다.
getcwdu()	현재 작업 디렉토리를 표현하는 유니코드 문자열을 돌려준다.
getpid()	현재 프로세스 신분번호(id)를 돌려준다.
fork()	C 의 fork()와 비슷. 0 을 자손에게 돌려주고, 자손의 프로세스 신분번호(pid)가 부모에게 반환된다 [윈도우즈 불가].
kill(pid, signal)	C 의 kill 과 비슷 [윈도우즈 불가].
listdir(path)	경로(path) 디렉토리에 담긴 항목들의 (기본)이름을 나열한다, '.'와 '..'는 제외된다. 경로(path)가 유니코드 문자열이면, 반환된 문자열도 유니코드다.
lseek(fd, pos, how)	파일(fd)에서 현재 위치를 pos 에 설정한다, 파일의 처음에서 (how=0), 현재 위치까지 (how=1), 또는 파일의 끝까지 (how=2) 상대적인 거리로 표현된다.
mkdir(path[, mode])	경로(path)라는 이름의 디렉토리를 숫자형 모드(mode)로 만든다 (기본값 0777).
open(file, flags, mode)	C 의 open()과 비슷. 파일 기술자를 돌려준다. 이런 저 수준 함수보다는 파일 객체 함수를 사용하자.
pipe()	파이프를 만든다. 파일 기술자 한 쌍(r, w)을 돌려준다 [윈도우즈 불가].
popen(command, mode='r', bufSize=0)	명령어(command)에 또는 명령어로부터 파이프를 연다. 결과는 'r'이나 'w'인 모드(mode)에 맞추어 쓸 파일 객체이다. 명령어 출력을 잡을 수 있고 ('r' 모드), 또는 입력을 명령어에 먹일 수 있다 ('w' 모드).
remove(path)	unlink 참조.
rename(old, new)	파일이나 디렉토리를 제거하거나 old 에서 new 로 이름을 바꾼다. [목표 이름이 이미 존재하면 에러이다]
renames(old, new)	재귀적인 디렉토리 또는 파일 이름바꾸기 함수. rename()과 비슷하다, 단 다른점은 새로운 경로이름을 유효하게 만드는데 필요한 매개 디렉토리의 생성이 먼저 시도된다는 점이다. 이름을 바꾼 후에는, 예전 이름에서 가장 오른쪽 경로 부분에 상응하는 디렉토리가 removedirs()을 사용하여 말끔히 다듬어진다.
rmdir(path)	빈 디렉토리(path)를 제거한다
read(fd, n)	n 바이트를 파일 기술자(fd)로부터 읽어서 문자열로 돌려준다.
stat(path)	다음 순서로 반환: st_mode, st_ino, st_dev, st_nlink, st_uid,st_gid, st_size, st_atime, st_mtime, st_ctime. [윈도우즈에서 다음 값은 더미이다 : st_ino, st_uid, st_gid]
system(command)	문자열 command 를 하부쉘에서 실행한다. 하부쉘의 종료 상태를 돌려준다 (보통 0 은 OK 이다).
times()	축적된 CPU 시간을 초단위로 돌려준다 (사용자, 시스템, 자손의 사용자, 자손의 시스템, 실제 경과 시간) [윈도우즈에서 마지막 3 개는 불가].
unlink(path)	(디렉토리 아님) 파일 경로(path)를 끊는다 ("삭제한다"). 다음과 동일: remove.
utime(path, (aTime, mTime))	파일의 접근시간 & 변경시간을 주어진 터플값으로 설정한다.
wait()	자손 프로세서의 완료를 기다린다. pid, exit_status 로 이루어진 터플을 돌려준다 [윈도우즈 불가].
waitpid(pid, options)	프로세스 pid 가 완료되기를 기다린다. pid, exit_status 로 이루어진 터플을 돌려준다 [윈도우즈 불가].
write(fd, str)	문자열(str)을 파일 fd 에 쓴다. 씌여진 바이트를 정수 개수로 돌려준다.

posixpath

Posix 경로이름 연산.
이 모듈을 직접 수입하지 말 것, 대신에 os 를 수입하고 이 모듈을 os.path 로 참조하자. (예. os.path.exists(p))!

posixpath 함수

함수	결과
abspath(p)	현재 작업 디렉토리를 고려하여, 경로 p 에 대하여 절대적인 경로를 돌려준다.
commonprefix(list)	가장 긴 경로 접두사를 돌려준다(문자단위로 취한다). 이는 리스트 안에 있는 모드 경로들의 접두사이다 (리스트가(list)가 비어있으면 '').
dirname/basename (p)	경로 p 에서 디렉토리와 이름 부분. split 참조.
exists(p)	문자열 p 가 기존의 경로(파일이나 디렉토리)이면 True.
expanduser(p)	p 에서 "~"를 확장한 문자열 사본을 돌려준다.
expandvars(p)	p 에서 환경 변수를 확대하여 그 사본을 돌려준다. [윈도우즈: 대소문자 중요; 반드시 유닉스의: $var 표기법을 사용할 것, %var% 불가]
getmtime(filepath)	파일경로(filepath)의 마지막 변경 시간을 돌려준다 (기원(70.01.01.00:00) 이후로 초단위 정수 개수).
getatime(filepath)	파일경로(filepath)의 마지막 접근 시간을 돌려준다 (기원(70.01.01.00:00) 이후로 초단위 정수 개수).
getsize(filepath)	파일경로(filepath)의 크기를 바이트 단위로 돌려준다. 파일이 존재하지 않거나 접근이 불가능 하면 os.error.
isabs(p)	문자열 p 가 절대 경로이면 True.
isdir(p)	문자열 p 가 절대 디렉로리이면 True.
islink(p)	문자열 p 가 심볼릭 링크이면 True.
ismount(p)	문자열 p 가 마운팅 포인트이면 True [윈도우즈에서는 모든 디렉토리에 대하여 True].
join(p[,q[,...]])	지능적으로 여러 경로 구성요소를 연결한다.
split(p)	p 를 (head, tail)로 자른다. tail 은 마지막 경로이름 구성요소이고 head 는 그에 앞서는 모든 것이다. <=> (dirname(p), basename(p))
splitdrive(p)	경로 p 를 ('drive:', tail) 쌍으로 가른다 [윈도우즈]
splitext(p)	(root, ext)로 가른다. root 의 마지막 구성요소에는 점이 포함되지 않으며 ext 는 비어있거나 점으로 시작한다.
walk(p, visit, arg)	p 를 루트로 하는 트리에서 각 디렉토리를 재귀적으로 방문하면서 다음 인수들(arg, dirname, names)을 가지고 함수 visit 을 호출한다 (p 가 디렉토리이면 자신도 포함된다). 인수 dirname 은 방문된 디렉토리를 지정하고, 인수 names 는 그 디렉토리에 있는 파일들을 나열한다. visit 함수는 names 을 변경하여, 예를 들어, 트리의 특정 부분을 방문하지 않기 위해서 dirname 아래에 방문된 디렉토리 집합에 영향을 줄 수 있다.

shutil

고-수준 파일 연산 (복사, 삭제). [자세한 문서]

주요 shutil 함수

함수	결과
copy(src, dest)	파일 src 의 내용물을 파일 dest 에 복사한다. 파일 허가권은 유지된다.
copytree(src, dest[, symlinks])	재귀적으로 src 를 루트로 하는 전체 디렉토리 트리를 dest 에 복사한다 (이미 존재하고 있으면 안된다). 만약 symlinks 가 참이면 src 의 링크들은 그대로 dest 에도 유지된다.
move(src, dest)	재귀적으로 파일이나 디렉토리를 새로운 위치로 옮긴다.
rmtree(path[, ignore_errors[, onerror]])	전체 디렉토리 트리를 삭제한다, ignore_errors 가 참이면 에러를 무시한다, 또는 onerror(func, path, sys.exc_info())가 공급되면 func (오류처리 함수)와 path (관련 파일)를 인수로 하여 호출한다.

(다음 역시 참조: copyfile, copymode, copystat, copy2)

time

시간 접근과 변환. [자세한 문서]

변수	
변수	의미
altzone	기본 자오선의 서쪽으로 지역 DST 시간대의 부호있는 초단위 거리.
daylight	일광절약(DST) 시간대가 지정되면 0 이 아니다.

함수	
함수	결과
time()	UTC 시간을 표현하는 소수를 기원 이후로 초단위로 돌려준다.
gmtime(secs), localtime(secs)	시간을 표현하는 터플을 돌려준다 : (year aaaa, month(1-12), day(1-31), hour(0-23), minute(0-59), second(0-59), weekday(0-6, 0 은 월요일), Julian day(1-366), 일광시간(daylight) 표식(-1,0 또는 1)).
asctime(timeTuple),	다음 형태로 24-개의 문자열을 돌려준다: 'Sun Jun 20 23:21:05 1993'.
strftime(format, timeTuple)	시간을 표현하는 형식화된 문자열을 돌려준다. 형식은 아래 테이블 참조.
mktime(tuple)	localtime()의 반대. 소수를 돌려준다.
strptime(string[, format])	시간을 표현하는 형식 문자열을 해석하여, gmtime()의 형태로 터플을 돌려준다.
sleep(secs)	secs 초 동안 실행을 멈춘다. secs 는 소수일 수 있다.

다음도 참조: clock, ctime.

strftime()으로 포맷하기	
지시어	의미
%a	로케일 약자화된 요일 이름.
%A	로케일의 완전한 요일 이름.
%b	로케일 약자화된 달 이름.
%B	로케일의 완전한 달 이름.
%c	로케일에 맞는 날짜와 시간 표현.
%d	달에서 10 진수로 표현된 날짜 [01,31].
%H	십진수로 나타낸 시간 (24-시간제) [00,23].
%I	십진수로 나타낸 시간 (12-시간제) [01,12].
%j	해에서 십진수로 나타낸 날짜 [001,366].
%m	십진수로 나타낸 달 [01,12].
%M	10 진수로 나타낸 분 [00,59].
%p	로케일에서 AM 이나 PM 의 동등물.
%S	십진수로 나타낸 초 [00,61]. 주의, 61 !
%U	해에서 주 번호를 10 진수 번호로 돌려준다 (일요일이 주의 첫째 날이다) [00,53]. 새해에서 첫 일요일 앞에 있는 날짜들은 0 번 주에 있는 것으로 간주된다.
%w	십진수로 나타낸 요일 [0(월요일),6].
%W	십진수로 나타낸 해에서의 주 번호 (주에서 첫 날을 월요일로 한다) [00,53]. 새해에서 첫 일요일 앞에 있는 날짜들은 0 번 주에 있는 것으로 간주된다.
%x	로케일에 맞는 데이터 표현.
%X	로케일에 맞는 시간 표현.
%y	십진수로 나타낸 세기 없는 해 [00,99].
%Y	십진수로 나타낸 세기 있는 해.
%Z	시간대 이름 (또는 시간대가 존재하지 않으면 문자가 존재하지 않는다).
%%	기호문자 "%" 문자.

문자열

일반적 문자열 연산. [자세한 문서]
파이썬 2.0 부로, (모두는 아니지만) string 모듈에서 제공되던 기능이 대부분 내장 문자열 메쏘드로 대체되었다 - 자세한 것은 문자열 연산을 참조.

문자열(string) 변수

변수	의미
digits	다음 문자열 '0123456789'.
hexdigits, octdigits	적법한 16 진 & 8 진 자리 문자열.
letters, uppercase, lowercase, whitespace	적절한 문자들을 담고 있는 문자열.
ascii_letters, ascii_lowercase, ascii_uppercase	위와 같다, 현재 로케일(locale)을 고려한다.
index_error	하부문자열이 발견되지 않으면 index()에 의해서 예외가 일어난다.

문자열(string) 함수

함수	결과
expandtabs(s, tabSize)	문자열 s 에서 탭을 확대한 사본을 돌려준다.
find/rfind(s, sub[, start=0[, end=0])	s 에서 하부 문자열 sub 가 발견된 하한/상한 지표를 돌려준다. sub 는 s[start:end]에 온전하게 담겨 있어야 한다. sub 가 발견되지 않으면 -1 을 돌려준다.
ljust/rjust/center(s, width)	문자열 s 의 사본을 돌려준다; 주어진 너비의 범위에서 왼쪽/오른쪽/중앙 정렬하고 빈곳은 스페이스로 채운다. s 는 잘려지지 않는다.
lower/upper(s)	s 를 대문자/소문자로 변환한 사본을 돌려준다.
split(s[, sep=whitespace[, maxsplit=0]])	문자열 s 에서 단어들을 담은 리스트를 돌려준다, 문자열 sep 를 가름자로 사용한다.
join(words[, sep=' '])	가름자를 사이에 두고 단어들을 리스트나 터플로 결합한다; split 의 반대.
replace(s, old, new[, maxsplit=0]	문자열 s 에서 나타난 하부문자열 old 를 모두 new 로 대치하여 그 사본을 돌려준다. maxsplit 이 지정되면 최대 maxsplit 개수 만큼의 교체로 제한된다.
strip(s[, chars=None])	문자열 s 에서 이끄는 문자와 따르는 문자(chars)를 제거하고 (사본을) 돌려준다 (기본값: 흰공백(whitespace)). 참조: lstrip, rstrip.

re (sre)

정규 표현식 연산. [자세한 문서]

유니코드 문자열을 처리한다. 새 모듈 sre 에 구현되었고, re 는 이제 그냥 호환을 위한 전방 모듈에 불과하다.
패턴은 문자열로 지정된다. 팁: 역사선을 기호화하려면 날(raw) 문자열을 사용하자 (예. r'₩w*').

정규 표현식 구문

형태	설명
.	어떤 문자에도 일치한다 (DOTALL 표식이 지정되면 새줄도 포함한다).
^	문자열의 처음에 일치한다 (여러줄(MULTILINE) 모드에서는 줄마다).
$	문자열의 끝에 일치한다 (여러줄(MULTILINE) 모드에서는 줄마다).
*	앞의 정규 표현식을 0 회 이상 반복 (가능한 많이).
+	앞의 정규 표현식을 1 회 이상 반복 (가능한 많이).
?	앞의 정규 표현식을 0 회 또는 1 회만 반복.
*?, +?, ??	*, + 그리고 ?와 동일하지만 가능한 적게 문자와 일치한다.
{m,n}	앞의 정규 표현식을 m 회에서 n 회까지 반복.
{m,n}?	동일, 가능한 적게 일치 시도.
[ ]	문자 집합을 정의한다: 예. '[a-zA-Z]'는 모든 문자들에 일치한다 (참조 ₩w ₩S).
[^]	여 문자 집합을 정의한다: 문자가 집합에 없으면 일치한다.
₩	특수 문자들 '*?+ &$|()'을 피신시키고 특수한 연속열을 도입한다 (아래 참조). 파이썬의 문자열 규칙 때문에, 패턴 문자열에서 '₩₩'이나 r'₩'로 쓴다.
₩₩	기호문자 '₩'에 일치한다; 파이썬의 문자열 규칙에 의해, 패턴 문자열에 '₩₩₩₩'로 쓰거나, 더 좋게는 날 문자열을 다음과 같이 사용한다: r'₩₩'.
|	대안을 지정한다: 'foo|bar'는 'foo' 또는 'bar'에 일치한다.
(...)	() 안의 어떤 RE 에도 일치한다. 그룹을 구분한다.
(?:...)	동일. 하지만 그룹을 구분하지 않는다.
(?=...)	...가 다음에 일치해야만 일치한다. 하지만 문자열을 하나도 소비하지 않는다 예. 'Isaac (?=Asimov)'는 'Asimov'가 다음에 올 때만 'Isaac'에 일치한다.
(?!...)	...가 다음에 일치하지 않으면 일치한다. (?=...)의 부정 표현이다.
(?P<name>...)	() 안의 어떤 RE 에도 일치한다. 이름붙은 그룹을 확정한다, (예. r'(?Pid[a-zA-Z_]₩w*)'는 그룹 이름 id 를 정의한다).
(?P=name)	name 이라는 이름의 바로 앞의 그룹에서 일치한 텍스트에 일치한다.
(?#...)	주석; 무시된다.
(?letter)	letter 는 'i','L', 'm', 's', 'x'중의 하나이다. 전체 정규 표현식에 대하여 상응하는 표식을 설정한다 (re.I, re.L, re.M, re.S, re.X).

특수한 연속열

연속열	설명
number	같은 번호를 가진 그룹(group)의 내용에 일치한다; 그룹은 1 부터 번호가 매겨진다.
₩A	문자열의 처음에만 일치한다.
₩b	단어(word)의 앞이나 뒤에 빈 문자열: '₩bis₩b'는 'is'에는 일치하지만, 'his'에는 일치하지 않는다.
₩B	단어의 앞이나 뒤가 빈 문자열이 아니다.
₩d	10 진 숫자 (<=> [0-9]).
₩D	비-10 진 숫자 (<=> [^ 0-9]).
₩s	흰공백 문자 (<=> [₩t₩n₩r₩f₩v]).
₩S	비-흰공백 문자 (<=> [^ ₩t₩n₩r₩f₩v]).
₩w	영문자숫자 문자 (LOCALE 표식에 따른다).
₩W	비-영문자숫자 문자 (LOCALE 표식에 따른다).
₩Z	문자열의 끝에만 일치.

변수

변수	의미
error	패턴 문자열이 유효한 regexp 가 아니면 예외.

함수

함수	결과
compile(pattern [,flags=0])	RE 패턴 문자열을 정규 표현식 객체로 컴파일한다. 표식문자 (| 로 조합 가능): I 나 IGNORECASE 또는 (?i) 대소문자 구분 일치 L 이나 LOCALE 또는 (?L) ₩w, ₩W, ₩b, ₩B 를 현재 로케일에 의존하게 만든다 M 이나 MULTILINE 또는 (?m) 모든 새로운 라인에 일치한다. 뿐만 아니라 온전한 문자열의 처음/끝에 도 일치한다. S 나 DOTALL 또는 (?s) '.'는 모든 문자에 일치하며, 새줄(newline)도 포함된다. X or VERBOSE or (?x) 문자 집합 바깥쪽의 흰공백을 무시한다
escape(string)	문자열(string)에서 모든 비-영문자숫자를 역사선화하여 그 사본을 돌려준다.
match(pattern, string[, flags])	0 개 이상의 문자가 문자열(string)의 시작에서 RE 패턴 문자열과 일치하면 상응하는 MatchObject 실체를 돌려준다. 일치하지 않으면 None 을 돌려준다.
search(pattern, string[, flags])	문자열(string)을 뒤져서 패턴(pattern)에 일치하는 위치를 찾으면, 상응하는 MatchObject 실체를 돌려준다. 일치하지 않으면 None 을 돌려준다.
split(pattern, string[, maxsplit=0])	문자열(string)에 패턴(pattern)이 출현하면 가른다. capturing ()이 패턴에 사용되면, 패턴의 출현이나 하부패턴도 역시 반환된다.
findall(pattern, string)	패턴(pattern)에서 중복되지 않은 일치들을 담은 리스트를 돌려준다. 또는 패턴이 1 그룹이상이면 터플의 리스트나 그룹의 리스트를 돌려준다.
sub(pattern, repl, string[, count=0])	문자열(string)에서 (문자열 또는 RE 객체의) 패턴(pattern)이 (최대 count 개수 만큼) 제일왼쪽에서 중첩되지 않고 출현할 때 마다 repl 로 바꾸어 교체함으로써 얻은 문자열을 돌려준다; repl 은 문자열이거나 또는 함수가 될 수 있다. 함수는 단 한개의 MatchObj 인수로 호출되는데, 반드시 교체 문자열을 반환해야 한다.
subn(pattern, repl, string[, count=0])	sub()와 동일, 하지만 터플을 돌려준다 (newString, numberOfSubsMade).

정규 표현식 객체

RE 객체는 compile 함수에 의해서 반환된다.

re 객체 속성

속성	기술
표식	표식은 RE 객체가 컴파일될 때 사용되거나, 아무것도 제공되지 않으면 0 이다.
groupindex	패턴 사전 {그룹 이름: 그룹 번호}.
pattern	RE 객체가 컴파일되는 패턴 문자열.

re 객체 메쏘드

메쏘드	결과
match(string[, pos][, endpos])	0 개 이상의 문자가 문자열의 처음에서 이 정규 표현식에 일치하면, 그에 상응하는 MatchObject 실체를 돌려준다. 그렇지 않으면 None 을 돌려준다; 이것은 0-길이 일치와는 다르다는 것을 주의하자. 선택적인 두 번째 매개변수 pos 는 문자열에서 탐색이 시작될 지표를 지정한다; 기본값은 0 이다. 이는 문자열 조각썰기와 완전하게 같지는 않다; 다음의 '^ ' 패턴 문자는 문자열의 진짜 시작에서 일치하고 새줄문자 바로 앞에서 일치하며, 오히려 탐색이 시작될 지표에 반드시 일치하지는 않는다. 두번째 매개변수 endpos 는 얼마나 멀리 문자열이 탐색될 것인지를 제한한다; 마치 문자열이 endpos 만큼의 문자 길이처럼 되어서, 일치를 위해 pos 에서 endpos 까지만 탐색된다.
search(string[, pos][, endpos])	문자열을 뒤져서 이 정규 표현식이 일치되는 위치를 찾는다. 그리고 상응하는 MatchObject 실체를 돌려준다. 그렇지 않으면 None 을 돌려준다; 이는 문자열의 어떤 위치에서 0-길이 일치를 찾는 것과는 다르다는 것을 주의하자. 선택적인 pos 와 endpos 매개변수는 match() 메쏘드와 똑 같은 의미를 가진다.
split(string[, maxsplit=0])	split() 함수와 동일하다. 컴파일된 패턴을 사용한다.
findall(string)	findall() 함수와 동일하다. 컴파일된 패턴을 사용한다.
sub(repl, string[, count=0])	sub() 함수와 동일하다. 컴파일된 패턴을 사용한다.
subn(repl, string[, count=0])	subn() 함수와 동일하다. 컴파일된 패턴을 사용한다.

일치 객체(Match Objects)

일치 객체는 match & search 함수가 돌려준다.

일치 객체 속성

속성	설명
pos	search 함수나 match 함수에 건네지는 pos 값; 문자열에서 RE 엔진이 탐색을 시작한 지표.
endpos	search 함수나 match 함수에 건네지는 endpos 값; 문자열에서 RE 엔진이 탐색을 멈출 지표.
re	자신의 일치(match) 또는 탐색(search)함수가 이런 MatchObj 실체를 산출하는 RE 객체
string	match()나 search()에 건네지는 문자열.

일치 객체 함수

함수	결과
group([g1, g2, ...])	일치된 하나 이상의 그룹을 돌려준다. 인수가 하나이면, 결과는 문자열이다; 인수가 여러개이면, 결과는 인수당 하나씩 항목을 가진 터플이다. gi 이 0 이면, 반환값은 전체 일치 문자열이다; 1 <= gi <= 99 이면, 그룹 #gi 에 일치하는 문자열을 돌려준다 (또는 그런 그룹이 없으면 None 이다); gi 는 또한 그룹 이름(name) 될 수도 있다.
groups()	일치된 모든 그룹을 터플로 돌려준다; 일치에 참여하지 못한 그룹은 값으로 None 을 갖는다. len(tuple)== 1 이면 터플 대신에 문자열을 돌려준다.
start(group), end(group)	그룹에 의하여 일치된 하부문자열의 처음 & 끝 지표를 돌려준다 (그룹은 존재하지만 일치에 기여하지 못하면 None 이다).
span(group)	2-터플을 돌려준다 (start(group), end(group)); 그룹이 일치에 기여하지 못하면 (None, None)이 될 수 있다.

math

격렬하게 숫자를 부수어 보고 싶으면, 수치처리 파이썬 그리고 파이썬과 과학적 컴퓨팅 페이지를 참조하자. [자세한 문서]

상수	
이름	값
pi	3.1415926535897931
e	2.7182818284590451

함수	
이름	결과
acos(x)	(호도로 측정된) x 의 아크 코사인을 돌려준다.
asin(x)	(호도로 측정된) x 의 아크 사인을 돌려준다.
atan(x)	(호도로 측정된) x 의 아크 탄젠트를 돌려준다.
atan2(x, y)	(호도로 측정된) y/x 의 아크 탄젠트를 돌려준다. atan(y/x)와는 다르게, x 와 y 의 부호가 모두 고려된다.
ceil(x)	x 의 천정값(ceiling)을 소수로 돌려준다. x 보다 큰 가장 작은 정수값이다.
cos(x)	(호도로 측정된) x 의 코사인을 돌려준다.
cosh(x)	x 의 쌍곡 코사인을 돌려준다.
degrees(x)	각 x 를 호도에서 각도로 변환한다.
exp(x)	e 를 x 제곱하여 돌려준다.
fabs(x)	소수 x 의 절대값을 돌려준다.
floor(x)	x 의 바닥값(floor)을 실수로 돌려준다. x 보다 작은 가장 큰 정수 값이다.
fmod(x, y)	플랫폼 C 에 따라 fmod(x, y)를 돌려준다. x % y 는 다를 수 있다.
frexp(x)	(m, e)의 쌍으로, x 의 가수와 지수를 돌려준다. m 은 소수이고 e 는 정수이다, 예를 들면 x = m * 2.**e. x 가 0 이면, m 과 e 는 모두 0 이다. 그렇지 않으면 abs(m)는 0.5 이상 1.0 미만이다.
hypot(x, y)	유클리드 거리 sqrt(x*x + y*y)를 돌려준다.
ldexp(x, i)	x * (2**i)
log(x[, base])	base 를 밑수로 하여 x 의 로그를 돌려준다. 밑수가 지정되지 않으면, (밑수가 e 인) x 의 자연 로그를 돌려준다.
log10(x)	밑수를 10 으로 하는 x 의 로그를 돌려준다.
modf(x)	x 의 소수 부분과 정수 부분을 돌려준다. 두 결과 모두 x 의 부호를 유지한다. 정수부분은 소수로 반환된다.
pow(x, y)	x**y 를 돌려준다 (x 를 y 제곱). y=2 라면, x*x 를 사용하는 것이 더 효율적이다.
radians(x)	각 x 를 각도에서 호도로 변환한다.
sin(x)	(호도로 측정된) x 의 아크 사인을 돌려준다.
sinh(x)	x 의 쌍곡 사인을 돌려준다.
sqrt(x)	x 의 제곱근을 돌려준다.
tan(x)	(호도로 측정된) x 의 탄젠트를 돌려준다.
tanh(x)	x 의 쌍곡 탄젠트를 돌려준다.

getopt

명령어 줄 선택사항을 위한 해석기. [Full doc]

이는 파이썬 2.3 이 될때까지 표준 해석기였다, 이제 optparse 로 대체되었다.
[참조: 리차드 그루엣(Richard Gruet)의 간단한 해석기 getargs.py (부끄러움을 무릅쓰고 선전)]

함수:
getopt(list, optstr) -- C 와 비슷. <optstr>은 찾을 선택 문자열이다.
선택사항에서 인수를 취하면 문자 뒤에 ':'를 놓자. 예

```
# 요청이 다음과 같이 "python test.py -c hi -a arg1 arg2"라면
  opts, args =  getopt.getopt(sys.argv[1:], 'ab:c:')
# 옵션(opts)은 다음과 같이 되고
  [('-c', 'hi'), ('-a', '')]
# 인수는(args) 다음이 된다
  ['arg1', 'arg2']
```

기본 배포되는 모듈과 꾸러미 목록

내장함수들과 파이썬의 Lib 디렉토리 안에 있는 모듈. 다음 Lib/site-packages 하부디렉토리에는 플랫폼-종속적인 꾸러미와 모듈이 담긴다.
[파이썬 NT 배포본은 다른 배포본들과 약간 다를 수 있다]

표준 라이브러리 모듈	
연산	결과
aifc	AIFF-C 와 AIFF 파일을 해석하는 것들.
anydbm	모든 dbm 복제품들에 대한 총괄 인터페이스 (dbhash, gdbm, dbm, dumbdbm).
asynchat	채팅-스타일(명령/응답)의 프로토콜을 지원하는 클래스.
asyncore	비동기 소켓 서비스 클라이언트와 서버를 위한 기본 기반구조.
atexit	파이썬 종료시에 호출되는 함수들을 등록한다.
audiodev	오디어 장치를 조작하기 위한 클래스들 (현재는 Sun 과 SGI 만 지원).
base64	RFC-1521 에 맞추어 base64 전송 인코딩 으로/으로부터 변환.
BaseHTTPServer	HTTP 서버 기본 클래스
~~Bastion~~	~~"요새화" 유틸리티 (실제 변수들에 대한 접근을 통제).~~
bdb	총괄 파이썬 디버거 기본 클래스.
bsddb	(선택적이다) 개선된 BSD 데이터베이스 인터페이스 [꾸러미]
binhex	매킨토시 binhex 압축/압축풀기.
bisect	Bisection 알고리즘.
bz2	BZ2 압축.
calendar	달력 인쇄 함수들.
cgi	공통 통로 인터페이스 (CGI)를 감싼다.
CGIHTTPServer	CGI-이해 HTTP 서버.
cmd	라인-지향적 명령어 인터프리터를 구축하는 총괄 클래스.
~~cmp~~	~~효율적으로 파일을 비교한다. 불리언 값만 결과로 나온다.~~
~~cmpcache~~	~~똑 같다. 그러나 속도를 위해 'stat' 결과를 캐쉬한다.~~
code	파이썬의 상호대화 인터프리터를 에뮬레이터하는데 필요한 유틸리티들.
codecs	기존의 유니코드 인코딩을 찾고 새로운 인코딩을 등록한다.
codeop	불완전할 가능성이 있는 파이썬 소스 코드를 컴파일하는 유틸리티들.
colorsys	RGB 와 기타 컬러 시스템 사이의 변환 함수들.
commands	os.popen 을 통하여 쉘 명령어를 실행한다 [유닉스 전용].
compileall	한 디렉토리 안에 존재하는 모든 .py 파일을 강제로 "컴파일한다".
ConfigParser	설정 파일 해석기 (윈도우즈의 .ini 파일과 아주 비슷함).
Cookie	HTTP 상태 (쿠키) 관리.
copy	총괄 얕은 복사와 깊은 복사 연산.
copy_reg	pickle/cPickle 모듈에 확장성을 제공하는 도움자.
csv	쉼표로-분리된 파일을 읽는 도구들 (비슷한 변종도 마찬가지이다).
datetime	개선된 날짜/시간 유형들 (date, time, datetime, timedelta).
dbhash	bsdhash.hashopen 에 대한 (g)dbm-호환 인터페이스.
difflib	연속열들을 비교하고, 한 연속열에서 다른 연속열로 변환하는데 필요한 변경사항들을 계산하는 도구.
dircache	한 디렉토리 안에 존재하는 파일들을 cache 를 사용하여 정렬한 파일 목록.
~~dircmp~~	~~디렉토리 차이를 식별할 도구를 만들 클래스를 정의한다.~~
dis	바이트코드 역어셈블러.
distutils	패키지 설치 시스템.
distutils.command.register	파이썬 꾸러미 인덱스(PyPI)에 모듈을 등록한다. 이 명령어 플러그인은 distutil 스크립트에 등록 명령어를 추가한다.
distutils.debug	
distutils.emxccompiler	
distutils.log	

표준 라이브러리 모듈	
doctest	문서화문자열(docstrings)에 임베드된 예제들을 실행하는데 기반을 둔 유닛 테스트 작업틀.
DocXMLRPCServer	스스로-문서화하는 XML-RPC 서버의 생성, pydoc 을 사용하여 HTML API 문서를 바로바로 만들어 낸다.
~~dospath~~	~~DOS 경로이름에 대한 일반 연산들.~~
dumbdbm	느리고 둔하지만 간단한 dbm 복제품.
~~dump~~	~~변수를 재구성한 파이썬 코드를 인쇄한다.~~
dummy_thread	
dummy_threading	쓰레드가 지원되면 사용하고 지원되지 않더라도 여전히 파이썬 버전 위에서 실행되는 코드를 더 쉽게 작성할 수 있게 해주는 도움자. 이 더미 모듈들은 그냥 단순하게 쓰레드들을 순서대로 실행한다.
encodings	새로운 코덱들: idna (IDNA strings), koi8_u (Ukranian), palmos (PalmOS 3.5), punycode (Punycode IDNA codec), string_escape (파이썬 문자열 피신 코덱: 인쇄-불능 문자들을 w/ 파이썬-스타일 문자열 피신으로 대체한다).
exceptions	내장 예외 계층도에 기반한 클래스.
filecmp	파일과 디렉토리 비교.
fileinput	모든 표준 입력 파일들에 대하여 빠르게 회돌이를 작성하는 도움자 클래스.
~~find~~	~~패턴에 일치하는 파일 디렉토리 계층도를 찾는다.~~
fnmatch	쉘 패턴에 일치하는 파일이름.
formatter	총괄 출력 형식화.
fpformat	총괄 부동 소수점 형식화 함수들.
ftplib	FTP 클라이언트 클래스. RFC 959 에 기반.
gc	쓰레기 수집을 수행하고, GC 디버그 상태를 획득하며, 그리고 GC 매개변수들을 조율한다.
getopt	표준 명령어 줄 처리. 또 optparse 참조.
getpass	현재 사용자 이름이나 암호를 얻는 유틸리티들.
gettext	국제화와 지역화 지원.
glob	파일이름 "globbing" 유틸리티.
gopherlib	고퍼 프로토콜 클라이언트 인터페이스.
~~grep~~	~~'grep' 유틸리티들.~~
gzip	zip 형식으로 압축된 파일을 읽고 & 쓴다.
heapq	히프 큐 (선점 규) 도움자들.
hmac	HMAC (메시지 인증을 위한 키-해쉬).
hotshot.stones	핫샷 프로파일러 하에서 파이스톤 벤치마크를 실행하는 도움자.
htmlentitydefs	HTML 문자 개체 참조.
htmllib	HTML2 해석 유틸리티들
HTMLParser	간단한 HTML 그리고 XHTML 해석기.
httplib	HTTP1 클라이언트 클래스.
idlelib	IDLE 개발 환경 (꾸러미) 지원 라이브러리.
ihooks	"수입" 메커니즘을 가로채는 갈고리.
imaplib	IMAP4 클라이언트. RFC 2060 에 기반.
imghdr	첫 몇 바이트에 근거하여 이미지 파일을 인식한다.
imputil	맞춤재단된 수입 갈고리를 만드는 방법을 제공한다.
inspect	살아있는 파이썬 객체들에 대한 정보를 획득한다.
itertools	반복자 그리고 게으른 연속열들과 작동하는 도구들.
keyword	파이썬 키워드 목록.
~~knee~~	~~계층적인 모듈 수입을 파이썬으로 재구현.~~
linecache	파일로부터 줄들을 캐쉬한다.
~~linuxaudiodev~~	~~리눅스 /dev/audio 지원.~~. 다음 ossaudiodev (Linux)으로 대체됨.
locale	현재 로케일 설정에 맞추어 숫자 형식화를 지원한다.
logging	(꾸러미) 구조화된 로그인 log4j 스타일을 지원하는 도구.
macpath	매킨토시를 위한 경로이름 (또른 관련) 연산들 [Mac].
macurl2path	경로이름과 URL 사이를 변환하는데 사용되는 맥 종속적 모듈 [Mac].

표준 라이브러리 모듈	
mailbox	유닉스 스타일, MMDF 스타일, 그리고 MH 스타일 편지함을 지원하는 클래스들.
mailcap	메일캡(Mailcap) 파일 처리 (RFC 1524).
markupbase	HTML 과 XHTML 에서 문서 유형 선언의 탐지를 공유한다.
mhlib	MH (편지함) 인터페이스.
mimetools	MIME-읽기나 MIME-쓰기 프로그램에서 사용되는 여러 도구들.
mimetypes	파일의 MIME 유형을 짐작한다.
MimeWriter	총괄 MIME 작성기. 2.3 배포본 이후로 비추천된다. 대신에 email 꾸러미를 사용하자.
mimify	전자우편 메시지를 MIME 으로 변환하거나 그 반대로 변환한다(역주: 2.3 이후로 비추천, email 로 교체).
mmap	메모리-짝짓기된 파일에 대한 인터페이스 - 변경가능한 문자열처럼 행위한다.
modulefinder	실제로 프로그램을 실행하지 않고서 주어진 파이썬 프로그램이 사용하는 모듈이 무엇인지 찾아내는 도구들.
multifile	여러부분(multipart) 메시지에서 각 부분에 readline()-스타일로 접근하는 인터페이스.
mutex	상호 배제 -- sched 모듈에 사용. 표준(std) 모듈, threading, 그리고 glock 참조.
netrc	netrc 파일 포맷을 해석하고 캡슐화한다.
nntplib	NNTP 클라이언트 클래스. RFC 977 에 기반.
ntpath	윈도우즈 경로이름에 대한 공통 처리.
nturl2path	NT 경로이름을 URL 파일로 그리고 그 반대로 변환한다.
olddifflib	예전 버전의 difflib (객체들 사이의 델타(deltas)를 계산하는데 사용되는 도움자)?
optparse	개선된 명령어-줄 옵션 해석 라이브러리 (getopt 참조).
os	어느 시스템인가에 따라 Mac, DOS, NT, 또는 Posix 를 위한 OS 루틴들.
os2emxpath	OS/2 EMX 를 위한 os.path 지원.
~~packmail~~	~~스스로 풀리는 쉘 압축파일을 만든다.~~
pdb	파이썬 디버거
pickle	파이썬 객체를 절이기(저장하고 복구한다)(더 빠른 C 구현이 내장 모듈에 존재한다: cPickle).
pickletools	절임객체(pickles)들을 분석하고 분해(disassemble)하는 도구들.
pipes	변환 파이프라인 주형틀.
pkgutil	주어진 꾸러미에 대하여 모듈 탐색 경로를 확장하는 도구들.
platform	바탕 플랫폼에 관한 정보를 획득한다.
~~poly~~	~~폴리노미얼.~~
popen2	파이프로 명령어를 stdin, stdout, 그리고 선택적으로 stderr 에 연결시킨다.
poplib	POP3 클라이언트 클래스.
posixfile	POSIX 에서 확장된 파일 연산.
posixpath	POSIX 경로이름에 대한 일반 연산들.
pprint	리스트, 터플 & 사전을 재귀적으로 예쁘게 출력한다.
pre	정규 표현식(RE) 지원 - re 참조.
profile	파이썬 코드를 프로파일하기 위한 클래스.
pstats	프로파일된 파이썬 코드에 대한 보고서를 인쇄하기 위한 클래스.
pty	의사 터미널 유틸리티.
py_compile	.py 파일을 .pyc 파일로 "컴파일"하는 루틴.
pyclbr	파이썬 파일을 해석해서 클래스와 메쏘드를 열람한다.
pydoc	상호대화적으로 사용하기 위해 파이썬 문서를 HTML 이나 텍스트로 만든다.
pyexpat	Expat XML 해석기에 대한 인터페이스.
~~PyUnit~~	JUnit 에 영감을 받은 ~~유닛 테스트 작업틀.~~ unittest 참조.
Queue	여러-생산자, 여러-소비자 큐.
quopri	인용부호화되어-인쇄가능한 전송 인코딩 으로/으로부터 RFC 1521 에 맞게 변환.

표준 라이브러리 모듈	
rand	C 의 rand()와 호환성을 원한다면 사용하지 말 것.
random	무작위 변수 생성기 (역주: 2.1 이후로 whrandom 을 대체).
re	정규 표현식
readline	GNU readline 인터페이스 [Unix].
reconvert	구형("regex") 정규 표현식을 신형 구문("re")으로 변환한다.
regex_syntax	regex.set_syntax()용 표식.
regexp	모듈 "regexp"를 위하여 "regex"를 사용하여 하위 호환성 유지.
regsub	Regexp-기반의 가르고 대치하기, 폐기된 regex 모듈 사용.
repr	repr()을 다시 만든 것이다. 하지만 최대 크기에 제한이 있다.
~~rexec~~	~~제한된 실행 편의기능들 ("safe" exec, eval, 등등).~~
rfc822	RFC-8222 메일 머리부를 해석한다.
rlcompleter	GNU readline 2.0 을 위한 자동 단어 완성.
robotparser	robot.txt 파일을 해석한다. 웹 스파이더에 쓸모가 있다.
sched	일반적으로 쓸모있는 이벤트 일정관리기 클래스.
sets	사전에 기반한 집합(Set) 데이터유형 구현 (Sets 참조).
sgmllib	SGML 용 해석기, 파생된 클래스를 정적 DTD 로 사용.
shelve	절여진(pickled) 객체들이 담긴 선반(shelv)을 관리한다.
shlex	간단한 쉘-류의 구문을 위한 어휘 분석기 클래스.
shutil	파일과 디렉토리 트리를 복사하는데 사용되는 유틸리티 함수들.
SimpleHTTPServer	간단한 HTTP 서버.
SimpleXMLRPCServer	간단한 XML-RPC 서버
site	제삼자 꾸러미들을 위하여 sys.path 에 모듈 탐색 경로를 추가한다.
smtpd	RFC 2821 smtp 프록시.
smtplib	SMTP/ESMTP 클라이언트 클래스.
sndhdr	소리를 인식하는데 도움을 주는 여러 루틴들.
socket	소켓 연산과 관련 함수 몇가지. 이제 settimeout(t)를 통하여 시간초과를 지원한다. 또한 윈도우즈에서는 SSL 을 지원한다.
SocketServer	총괄 소켓 서버 클래스.
sre	정규 표현식(RE) 지원. re 참조.
stat	os 의 결과를 번역하는데 사용되는 상수들/함수들.
statcache	파일에 관한 stat() 정보의 캐쉬를 관리한다.
statvfs	(존재한다면) os.statvfs() 그리고 os.fstatvfs()가 반환한 statvfs 구조체(struct)를 번역하는데 사용되는 상수들.
string	문자열 연산 모음집 (Strings 참조).
stringprep	유니코드 문자열의 규범화와 조작.
StringIO	문자열 버퍼를 읽고/쓰는 파일-류의 객체들 (더 빨라진 C 구현이 내장 모듈에 존재한다: cStringIO).
sunau	Sun 과 NeXT 오디어 파일을 해석하는 것.
sunaudio	sun audio 헤더를 번역한다.
symbol	파이썬 문법의 비-터미날 심볼 (from "graminit.h").
symtable	컴파일러의 내부 심볼 테이블에 대한 인터페이스.
tabnanny	파이썬 소스에서 애매한 들여쓰기를 점검한다.
tarfile	TAR 아카이브를 만들고 쓰는 도구들.
telnetlib	TELNET 클라이언트 클래스. RFC 854 에 기반함.
tempfile	임시 파일과 파일이름.
textwrap	텍스트 문단을 감싸는 도구들.
threading	제안된 새로운 쓰레드 모듈, 자바 쓰레드 모델의 하부모듬을 흉내낸다.
threading_api	(쓰레드 모듈의 문서).
timeit	벤치마크 도구.
toaiff	"임의의" 소리 파일을 AIFF (애플사와 SGI 사의 오디어 포맷)로 변환.
token	토큰 상수들 ("token.h"에서 가져옴).
tokenize	파이썬 소스를 위한 토큰생성기.
traceback	파이썬 스택 추적에 관한 정보를 추출하고, 포맷해서 인쇄한다.
trace	프로그램이나 함수의 실행을 추적하는 도구들.

표준 라이브러리 모듈	
tty	터미널 유틸리티들 [Unix].
turtle	로고메이션(LogoMation)-류의 그래픽.
types	표준 인터프리터에 모든 유형의 심볼들에 대한 이름을 정의한다.
tzparse	시간대 규격을 해석한다.
unicodedata	유니코드 특성들에 대한 인터페이스.
unittest	파이썬 유닛 테스트 작업틀, 에릭 감마(Erich Gamma)와 켄트 벡(Kent Beck)의 JUnit 에 바탕.
urllib	임의의 URL 을 연다.
urllib2	다양한 프로토콜을 사용하여 URL 을 여는데 사용되는 확장가능 라이브러리.
urlparse	(절대적인 그리고 상대적인) URL 을 해석한다.
user	사용자-지정한 맞춤재단 코드를 실행하도록 허용하는 갈고리.
UserDict	내장 사전 클래스의 하부클래스화를 허용하는 감싸개 (새로운-스타일의 클래스로 쓸모가 없어짐. 파이썬 2.2 에서부터, dict 는 하부클래스화가 가능하다).
UserList	내장 리스트 클래스의 하부클래스화를 허용하는 감싸개 (새로운-스타일의 클래스로 쓸모가 없어짐. 파이썬 2.2 이후로, list 는 하부클래스화가 가능하다)
UserString	내장 문자열 클래스의 하부클래스화를 허용하는 감싸개 (새로운-스타일의 클래스로 쓸모가 없어짐. 파이썬 2.2 이후로, str 은 하부클래스화가 가능하다).
~~util~~	~~다른 곳에는 맞지 않는 몇가지 쓸모있는 함수들!!~~
uu	UUencode 와 UUdecode 함수의 구현.
warnings	경고 서브시스템의 파이썬 부분. 경고를 제출하고, 원하지 않는 경고를 여과한다.
wave	WAVE 파일을 해석하는 것.
weakref	파이썬을 위한 약한 참조 지원. 또 프록시 객체의 생성을 허용한다.
webbrowser	플랫폼 독립적 URL 시동기.
~~whatsound~~	~~소리 파일을 인식하는데 도움을 주는 여러 루틴.~~
whichdb	db 파일을 여는데 어느 db 패키지가 사용되는지 짐작한다.
whrandom	Wichmann-Hill 무작위 수 생성기 (역주: 2.1 이후로 비추천, 대신 random 사용 권장).
xdrlib	Sun XDR (외부 데이터 표현(eXternal Data Representation))의 구현물 (하부모듬).
xmllib	XML 을 위한 해석기, 파생된 클래스를 정적 DTD 로 사용함.
xml.dom	DOM(문서 객체 모델)을 사용하여 XML 을 처리하는 클래스. 2.3: 새로 추가된 모듈들; expatbuilder, minicompat, NodeFilter, xmlbuilder.
xml.sax	SAX API 를 사용하여 XML 을 처리하는 클래스.
xmlrpclib	파이썬용 XML-RPC 클라이언트 인터페이스.
xreadlines	연속열-류의 객체를 제공하여 전체 파일을 메모리로 읽어들이지 않고 파일을 한줄씩 읽는다. 2.3 버전 이후로 권장되지 않는다. 대신에 for line in file 을 사용하라.
zipfile	PK 압축된 파일을 읽고 & 쓴다.
zipimport	ZIP 보관소 수입기.
~~zmod~~	~~난해한 수학적 개념의 데모.~~

작업공간 탐험과 관용구 힌트

```
dir(<module>)                    모듈(<module>)의 함수, 변수를 나열한다
dir()                          객체 키를 얻는다, 기본은 지역 이름 공간이다
if __name__ == '__main__': main()   스크립트로서 실행중일 때 main 을 호출한다
map(None, lst1, lst2, ...)       리스트를 합병한다
b = a[:]                       연속열 구조의 사본을 만든다
_                              (밑줄) 상호대화 모드에서, 가장 마지막으로 인쇄된 값이다
```

이맥스용 파이썬 모드

(개정 안됨, 최신이 아닐 가능성 있음 - 어떤 공헌도 환영한다 -)

Type C-c ? 파이썬 모드에서 광범위한 도움을 얻고 싶을 때.
들여쓰기
주로 새로운 코드를 입력하기 위함:
TAB 줄을 적절하게 들어쓰기
LFD 새로운 줄을 입력하고, 들여쓴다
DEL 들여쓰기를 축소하거나, 한개의 문자를 삭제한다
주로 기존의 코드를 다시 들여쓰기 위함:
C-c : 파일 내용으로 파이썬-들여쓰기-간격(py-indent-offset from)을 추측한다; 지역적으로 변경된다
C-u C-c : 위와 같지만, 전역적으로 변경된다
C-c TAB 상황에 맞게 구역을 다시들여쓰기 한다
C-c < 구역을 파이썬 들여쓰기 만큼 왼쪽으로 옮긴다
C-c > 구역을 파이썬 들여쓰기 만큼 오른쪽으로 옮긴다
표식설정 & 코드 구역의 조작
C-c C-b 줄 블록을 표식설정한다
M-C-h 가장 작은 def 경계에 표식설정한다
C-u M-C-h 가장 작은 class 경계에 표식설정한다
C-c # 코드 구역을 주석처리한다
C-u C-c # 코드 구역에서 주석을 걷어낸다
MOVING POINT
C-c C-p 이전 지점의 서술문으로 이동
C-c C-n 다음 지점의 서술문으로 이동
C-c C-u 현재 블럭의 처음으로 이동
M-C-a def 의 시작으로 이동
C-u M-C-a class 의 시작으로 이동
M-C-e def 의 끝으로 이동
C-u M-C-e class 의 끝으로 이동
파이썬 코드 실행
C-c C-c 전체 버퍼를 파이썬에게 보낸다
C-c | 현재 구역을 보낸다
C-c ! 파이썬 창을 시작한다; 이것은 다음의 명령어들이 사용한다
잇달아서 C-c C-c 또는 C-c | 명령어
변수
py-indent-offset 들여쓰기 증가
py-block-comment-prefix py-comment-region 이 사용하는 주석 문자열
py-python-command 파이썬을 요청하는 쉘 명령어
py-scroll-process-buffer 역주: 폐기됨
py-temp-directory (필요하면) 임시 파일에 사용되는 디렉토리
py-beep-if-tab-change 탭-너비가 변경되면 벨을 울린다

파이썬 2.3 간편 참조서
by johnsonj 2003.11.11