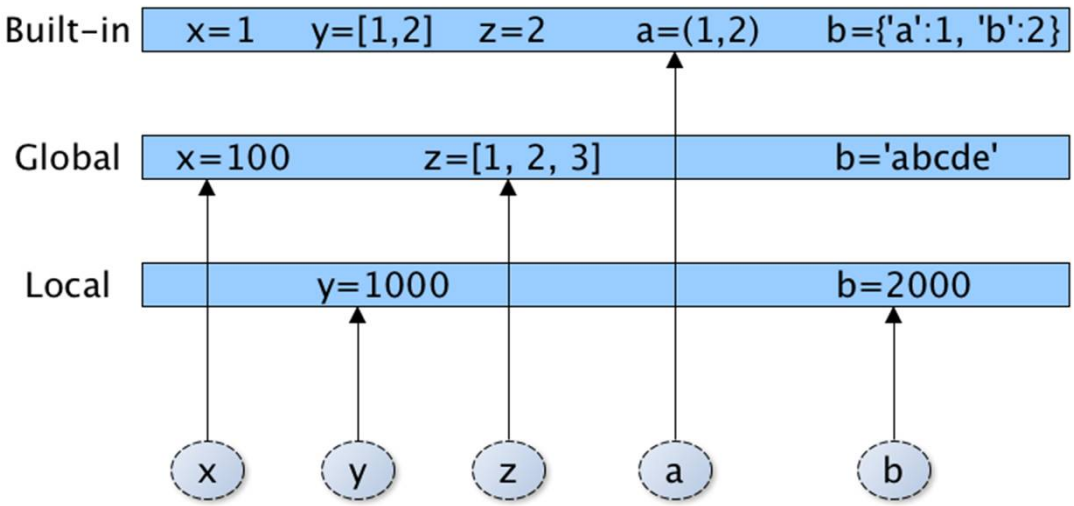
파이썬 프로그래밍

파이썬 모듈

한국기술교육대학교
온라인평생교육원

▣ 이름 공간

•이름 공간 또는 스코프 (Naming Space or Scope): 이름이 존재하는 장소.
파이썬은 실행 시간에 각 이름들을 적절한 이름 공간에 넣어 관리한다.
•이름 공간(스코프)의 종류
- 지역(Local): 각 함수 내부
- 전역(Global): 모듈 (파일) 내부
- 내장(Built-in): 파이썬 언어 자체에서 정의한 영역
•변수가 정의되는 위치에 의해 변수의 스코프가 정해짐
- 파이썬에서 변수의 정의
◦변수가 l-value로 사용될 때
•변수가 r-value로 사용될 때 해당 변수의 값을 찾는 순서 규칙
- L --> G --> B



▪지역(Local) : 일반적으로 각 함수 내부, 클래스 메소드 안. 객체 내부
▪일정한 공간 내에 존재하는 것 = local
▪모듈의 물리적인 단위 = 파일
▪내장(built-in) : 파이썬 언어 자체에서 정의한 영역
▪a = 1 에서 a라고 하는 것이 =의 왼쪽에 존재 시 a는 l - value
▪b = a 를 하게 되면 b는 l-value, a는 r-value
▪a 변수와 b 변수는 동일한 파일 내 존재하는 전역(global) 변수가 됨
▪L → G → B : a 값을 local, global, built-in 순서로 찾음
▪b는 local 영역에서 정의가 되었기 때문에 local 영역 정의 사용

▣ 이름 공간

1. 지역변수와 전역변수

•변수의 스코프는 해당 변수가 l-value로서 정의되는 위치에 따라 달라짐
•변수가 함수 내에서 정의되면 해당 함수의 지역 변수가 된다.

```
# g, h는 전역 변수
g = 10
h = 5

def f(a): # a는 지역 변수
    h = a + 10 # h는 지역, 새로 l-value로 정의했음
    b = h + a + g # b도 지역, g는 r-value이므로 기존 값을 참조 - 전역 변수
    return b

print f(h) # 함수 호출시에 사용되는 변수는 해당 위치의 스코프에서 값을 찾음
           - 전역 변수
print h # 전역 변수 h는 변함 없음
```

```
30
5
```

▪g, h 두 변수는 함수 바깥에 위치하므로 전역 변수
▪a 인자는 f 함수 안에 존재하는 지역(local) 변수
▪h 변수는 f 함수 내에 있으므로 지역변수가 됨
▪지역 변수 안 a는 r-value로 기존에 존재하는 것 활용 → 지역변수a
▪b도 f 함수 내 지역변수
▪h + a + g는 전부 r-value로 기존에 존재하는 것 활용
▪g 값의 경우 로컬 내 존재하지 않으므로 전역 변수 활용
▪h는 같은 전역 변수 내에서 찾아 값 기입
▪지역 변수 내 h와 전역 변수 h는 다른 값

▣ 이름 공간

1. 지역변수와 전역변수

•함수 내부에서 전역 변수를 직접 사용하고자 할 때
- global 키워드 활용

```
h = 5

def f(a):       # a는 지역
    global h    # h 변수를 전역이라고 미리 선언함
    h = a + 10 # h는 l-velue로 정의되더라도 미리 선언된 내용 때문에 전역 변수
    return h

print f(10)
print h         # 전역 변수 h 값이 함수 내에서 변경되었음
```

```
20
20
```

▪f 함수 내 h는 지역변수
▪global h 키워드를 적으면 h가 전역 변수 사용으로 선언됨
▪h = 5 가 사라지고 새로운 값 20이 들어감

▣ 이름 공간

1. 지역변수와 전역변수

•[주의] 동일 함수 내에서 동일한 변수가 지역변수와 전역변수로 동시에 활용될 수 없음
- 함수 내에서 정의되는 변수는 지역 변수로 간주
- 지역 변수로 선언되기 이전에 해당 변수를 사용할 수 없음

```
g = 10

def f():
    a = g       # l-value로 사용되는 g는 전역 변수
    g = 20      # r-value로 정의되는 g는 지역 변수
    return a

print f()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-68-e323361344da> in <module>()
      6     return a
      7
----> 8 print f()

<ipython-input-68-e323361344da> in f()
      2
      3 def f():
----> 4     a = g       # l-value로 사용되는 g는 전역 변수
      5     g = 20      # r-value로 정의되는 g는 지역 변수
      6     return a

UnboundLocalError: local variable 'g' referenced before assignment
```

▪a라는 변수는 함수 내에서 새로 만들어지는 것
▪g는 local 변수 내에 없으므로 전역변수 값 사용
▪global g 와 같은 내용이 없기 때문에 l-value의 g는 새롭게 선언
▪그러면 g는 함수 안에 존재하는 지역변수가 됨
▪g가 지역변수이면서 전역변수이기 때문에 error 발생

▣ 이름 공간

1. 지역변수와 전역변수

```
g = 10

def f():
    global g    # g는 전역 변수로 선언됨
    a = g       # a는 지역 변수, g는 전역 변수
    g = 20      # g는 전역 변수
    return a

print f()
```

```
10
```

▪a=g 가 아래 쓰이면 error 발생하지 않음

▪a=g의 g는 지역변수

▣ 이름 공간

2. 특정 공간의 이름 목록 얻기

•이름(Name) : 변수 (객체) 이름
  - 함수 이름
  - 클래스 이름
  - 모듈 이름
•dir(): 함수가 호출된 스코프에서 정의되어 있는 모든 Name들을 문자열 리스트로 반환한다.
•dir(object): object이 지니고 있는 모든 Name들을 문자열 리스트로 반환한다.

```
l = []
print dir(l)
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__delslice__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getslice__', '__gt__',
'__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
'__setattr__', '__setitem__', '__setslice__', '__sizeof__', '__str__', '__subclasshook__',
'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

▪dir() 내장함수 → 특정개체, 모듈 내 존재하는 이름들을 리스트로 반환
▪파이썬의 모든 것은 object이 될 수 있음
▪object = 함수, 특정객체, 클래스, 인스턴스, 모듈의 이름 등
▪'__'로 시작되는 것들은 특수한 변수 또는 메소드
▪l 에 존재하는 각각의 식별자

▣ 이름 공간

3. 함수의 중첩 영역(Nested Scopes) 지원

•Nested Scope: 함수 안에 정의되어 있는 함수 내부
  - 가장 안쪽의 스코프부터 바깥쪽의 스코프쪽으로 변수를 찾는다.

```
x = 2
def F():
    x = 1
    def G():
        print x
    G()

F()
```

```
1
```

▪아래 x는 위 x와 다른 f 함수의 지역변수
▪G() → 중첩영역을 지원하는 함수
▪함수 안에 print x를 하면 가까운 x를 먼저 찾음

파이썬 프로그래밍

파이썬 모듈

한국기술교육대학교
온라인평생교육원

▣ 모듈의 정의

•모듈: 파이썬 프로그램 파일로서 파이썬 데이터와 함수등을 정의하고 있는 단위
- 서로 연관된 작업을 하는 코드들을 묶어서 독립성을 유지하되 재사용 가능하게 만드는 단위
- 모듈을 사용하는 측에서는 모듈에 정의된 함수나 변수 이름을 사용

•모듈의 종류
- 표준 모듈
  ◦파이썬 언어 패키지 안에 기본적으로 포함된 모듈
  ◦대표적인 표준 모듈 예◦math, string
- 사용자 생성 모듈◦개발자가 직접 정의한 모듈
- 써드 파티 모듈◦다른 업체나 개인이 만들어서 제공하는 모듈

•모듈이 정의되고 저장되는 곳은 파일
- 파일 : 모듈 코드를 저장하는 물리적인 단위
- 모듈 : 논리적으로 하나의 단위로 조직된 코드의 모임

•파이썬 모듈이 정의되는 파일의 확장자: .py

•다른 곳에서 모듈을 사용하게 되면 해당 모듈의 .py는 바이트 코드로 컴파일 되어 .pyc로 존재한다.

▪test.py 자체가 기본적으로 test라고 하는 모듈을 만듦
▪실행코드들만 존재하면 다른 모듈이나 프로그램에서 활용 X
▪함수 f나 class 모듈을 import해서 식별자를 호출해서 이용
▪pyc 파일은 자바의 바이트 코드와 유사
▪pyc가 만들어지면 py가 없더라도 pyc를 활용하여 import가능

▣ 모듈의 정의

2. 사용자 모듈 만들기와 호출하기

```
#File: mymath.py
mypi = 3.14

def add(a, b):
    return a + b

def area(r):
    return mypi * r * r
```

▪mypi 변수, add 함수, area 함수 존재

▪mymath --> 모듈객체

•모듈 이름은 해당 모듈을 정의한 파일 이름에서 .py를 제외한 것
  - 모듈을 불러오는 키워드: import

•모듈에서 정의한 이름 사용하기

```
import mymath

print dir(mymath)     # mymath에 정의된 이름들 확인하기
print mymath.mypi     # mymath 안에 정의된 mypi를 사용한다
print mymath.area(5) # mymath 안에 정의된 area를 사용한다
```

```
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'add', 'area', 'mypi']
3.14
78.5
```

▪dir(mymath) --> mymath안에 들어있는 모든 이름을 리스트로 출력

▪'__'가 없는 식별자는 평범한 식별자

▪모듈명 + 모듈 안 존재하는 member

▣ 모듈의 정의

3. 모듈을 왜 사용하는가?

•함수와 모듈
- 함수: 파일 내에서 일부 코드를 묶는 것
- 모듈: 파일 단위로 코드들을 묶는 것◦비슷하거나 관련된 일을 하는 함수나 상수값들을 모아서 하나의 파일에 저장하고 추후에 재사용하는 단위

•모듈 사용의 이점
- 코드의 재사용
- 프로그램 개발시에 전체 코드들을 여러 모듈 단위로 분리하여 설계함으로써 작업의 효율을 높일 수 있음
- 별도의 이름 공간(스코프)를 제공함으로써 동일한 이름의 여러 함수나 변수들이 각 모듈마다 독립적으로 정의될 수 있다.

•별도 파일 내에 파이썬 코드를 저장할 때 (즉, 모듈을 코딩할 때) 한글 처리
- 파일의 맨 위에 다음 코드를 넣어 준다.◦# -*- coding:utf-8 -*-

•모듈은 하나의 독립된 이름 공간을 확보하면서 정의된다

▪함수와 모듈의 공통점 : 관련된 코드를 한 곳으로 묶는 것
▪동일한 레벨에서 여러 개의 함수가 한 모듈 내 존재 가능
▪모듈 : 파일 하나가 모듈
▪여러 개의 모듈을 기능, 역할 단위로 나누어 관련 내용을 채워 넣는 것
▪분업도 가능
▪a 모듈에 aaa --> b 모듈에 aaa
▪#으로 시작됨 --> 주석
▪주석 안 내용은 코딩이라는 키 값의 값으로 utf-8 을 적어줌
▪코딩이 utf-8 방식으로 저장됨
▪파이썬과 개발자 간의 약속
▪주석에 마음대로 한글 작성 가능

▣ 모듈의 정의

4. 모듈이 지닌 이름들 알아보기

•dir(모듈): 모듈이 지니고 있는 모든 이름들을 리스트로 반환

```
import string
print dir(string)
```

```
['Formatter', 'Template', '_TemplateMetaclass', '__builtins__', '__doc__', '__file__',
'__name__', '__package__', '_float', '_idmap', '_idmapL', '_int', '_long', '_multimap',
'_re', 'ascii_letters', 'ascii_lowercase', 'ascii_uppercase', 'atof', 'atof_error', 'atoi',
'atoi_error', 'atol', 'atol_error', 'capitalize', 'capwords', 'center', 'count', 'digits',
'expandtabs', 'find', 'hexdigits', 'index', 'index_error', 'join', 'joinfields', 'letters',
'ljust', 'lower', 'lowercase', 'lstrip', 'maketrans', 'octdigits', 'printable', 'punctuation',
'replace', 'rfind', 'rindex', 'rjust', 'rsplit', 'rstrip', 'split', 'splitfields', 'strip', 'swapcase',
'translate', 'upper', 'uppercase', 'whitespace', 'zfill']
```

▪string --> 파이썬에서 같이 설치되는 표준 모듈

▣ 모듈의 정의

5. 이름 공간을 제공하는 다른 예들

•독립된 이름 공간(스코프)을 제공하는 것들
- 모듈
- 클래스
- 객체
- 함수

•string 모듈 이름 공간에 변수 a를 생성한다.
- 표준 모듈에 사용자가 정의하는 이름을 생성하는 것은 비추천
- 단치 모듈 자체가 독립적인 이름 공간을 제공한다는 것을 알려줌

```
import string
string.a = 1
print string.a
```

```
1
```

▪string 모듈 바깥에서 새로운 변수를 정의하여 삽입 가능
▪표준모듈에 새로운 변수를 정의하여 삽입하는 것은 추천 X

▣ 모듈의 정의

5. 이름 공간을 제공하는 다른 예들

•클래스도 독립적인 이름 공간

```
class C:            # 클래스도 독립적인 이름 공간
    a = 2           # 클래스 이름 공간 내에 변수 선언
    pass            # 클래스 정의

c = C()             # 클래스 인스턴스 객체 생성
c.a = 1             # 클래스에서 생성된 인스턴스 객체도 별도의 이름 공간
print c.a
print c.__class__.a
print C.a
```

```
1
2
2
```

▪class 정의 방식 : 'class' 키워드+ class 이름+ 콜론(:)+ 아래 내용 기입
▪ a = 2의 a가 class C의 변수
▪c = C ( ) --> 생성자 호출
▪생성자가 호출되면 해당 클래스의 인스턴스가 반환됨
▪c.a = 1 --> 인스턴스 c 안에 a가 존재하고 a 값은 1
▪클래스와 인스턴스는 이름 공간이 독립적
▪함수도 독립적인 이름공간을 제공

▣ 모듈의 정의

5. 이름 공간을 제공하는 다른 예들

•함수도 독립적인 이름 공간
 - 다만 함수 내에서 선언된 로컬 변수는 함수 외부에서 접근할 수 없다.

```
x = 10    # 현재 모듈 내부에 정의되는 이름
def f():
    a = 1
    b = 2 # 현재 모듈에 정의되는 함수 f 내에 이름 a,b를 정의하고있다.
            함수도 독립적인 이름 공간
f.c = 1
print f.c
print
print f.a
```

```
1


---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-6-1110995cdbd5> in <module>()
      6 print f.c
      7 print
----> 8 print f.a

AttributeError: 'function' object has no attribute 'a'
```

▪x = 10의 x는 global 변수 해당
▪a, b는 지역 변수
▪정의가 끝난 F 함수에 F 함수 바깥에서 새 변수 삽입 가능
▪일반적인 함수가 가지고 있는 지역변수 a, b는 바깥에서 코딩 X

파이썬 프로그래밍

파이썬 모듈

한국기술교육대학교
온라인평생교육원

▣ 모듈 검색 경로

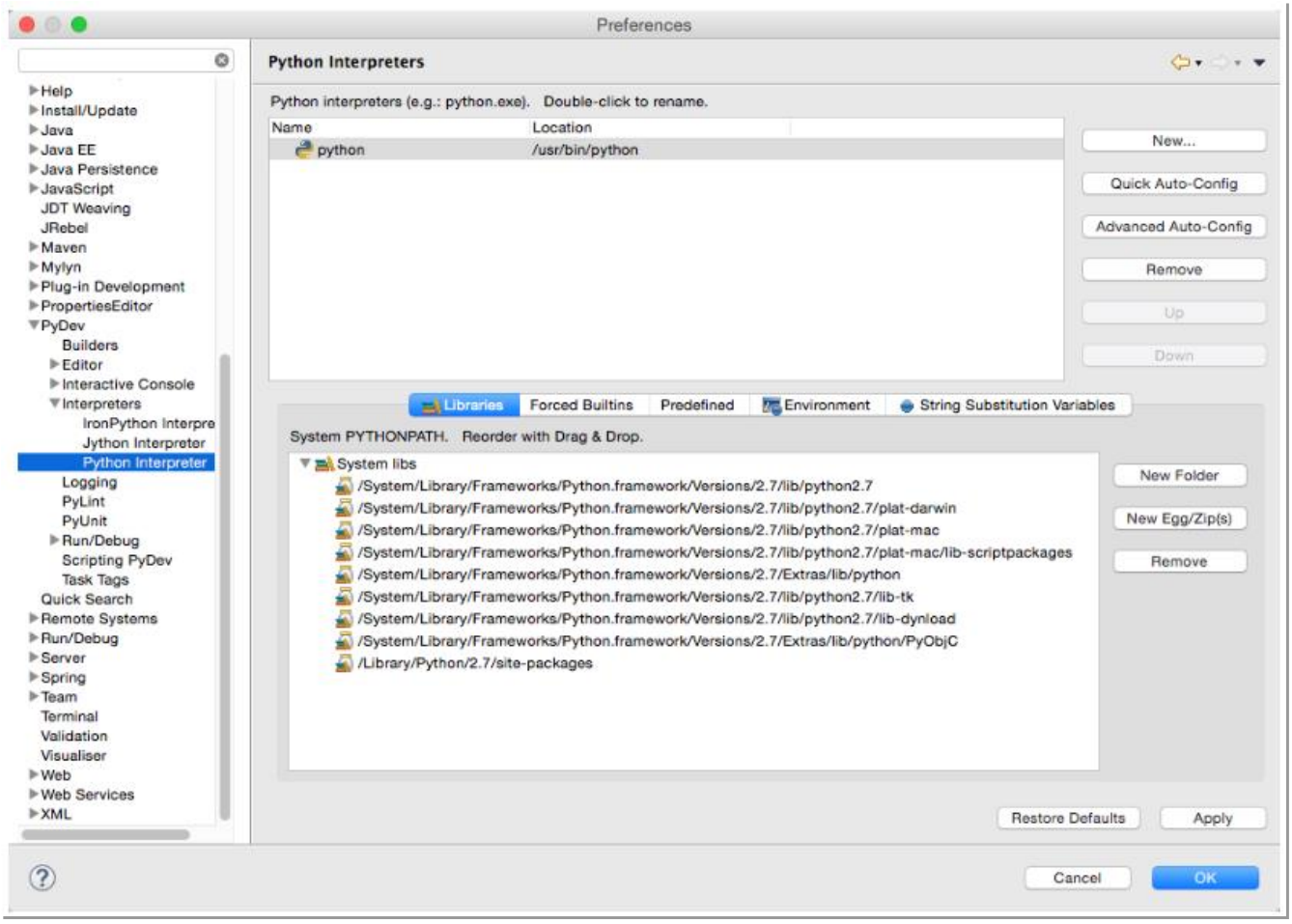
1. PYTHONPATH 환경 변수 설정하기

•파이썬이 모듈을 찾는 순서
  1) 이미 메모리에 로딩되어진 것
  2) 현재 디렉토리 (ipython에서 pwd 명령으로 확인 가능)
  3) PYTHONPATH 환경 변수에 기술된 디렉토리 목록을 차례로 탐색
  4) 표준 라이브러리 디렉토리들
    ◦sys 모듈 임포트 후 sys.path 로 확인 가능
    ◦sys.path.append(), sys.path.remove()로 추가, 삭제 가능
•PYTHONPATH 환경 변수 설정 방법
  - 윈도우
    ◦제어판₩시스템 및 보안₩시스템₩고급 시스템 설정₩환경 변수
    ◦새로 만들기: 변수 이름 - PYTHONPATH, 변수 값 - C:₩Users₩yhhan₩mypythonlib
    ◦python 새로 시작하기 (cmd창 새로 시작한 후)
  - MAC이나 리눅스◦~/.bash_profile 혹은 ~/.profile 파일에 다음 라인 추가
    ◦export PYTHONPATH=/Users/yhhan/mypythonlib
    ◦터미널 창에서 다음 명령어 수행
      ◦source ~/.bash_profile
  - 이클립스에서 PYTHONPATH 설정

▪한 번이라도 import 한 경우 메모리에 모듈이 존재함
▪값은 스스로 한 디렉토리에 정해 놓고 파이썬 관련 자료 넣기
▪source .profile --.profile 내에 있는 모든 것 다시 수행
▪이클립스에서 파이썬 등록 시 모듈 정의 및 검색 가능한 폴더 확인
▪각각의 모듈들이 나름대로 분류가 되어 있음
▪sys.path의 리턴 값이 리스트이기 때문에 append 함수 호출 가능
▪append ('폴더 내용') --> 이 폴더도 리스트에 포함

▣ 모듈 검색 경로

1. PYTHONPATH 환경 변수 설정하기



▣ 모듈 검색 경로

1. PYTHONPATH 환경 변수 설정하기

•코드 내에서 모듈 검색 경로 확인하기

```
import sys
print sys.path
```

```
['', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.
app/Contents/lib/python27.zip', '/Applications/Canopy.app/appdata/canopy-1.4.1.
1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7', '/Applications/Canopy.app
/appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/
plat-darwin', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.macosx-x86_64/
Canopy.app/Contents/lib/python2.7/plat-mac', '/Applications/Canopy.app/appdata/
canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/plat-mac/
lib-scriptpackages', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.
macosx-x86_64/Canopy.app/Contents/lib/python2.7/lib-tk', '/Applications/Canopy.app
/appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/
lib-old', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.macosx-x86_64/
Canopy.app/Contents/lib/python2.7/lib-dynload', '/Users/yhhan/Library/Enthought/
Canopy_64bit/User/lib/python2.7/site-packages', '/Users/yhhan/Library/Enthought/
Canopy_64bit/User/lib/python2.7/site-packages/PIL', '/Applications/Canopy.app/
appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/
site-packages', '/Users/yhhan/Library/Enthought/Canopy_64bit/User/lib/python2.7/
site-packages/IPython/extensions', '~/mypythonlib']
```

▪많은 폴더에 각각의 모듈들이 저장될 수 있고 검색 가능

▣ 모듈 검색 경로

2. 모듈의 검색 경로 동적으로 바꾸기

```
import sys

sys.path.append('~/mypythonlib')
print sys.path
```

```
['', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.
app/Contents/lib/python27.zip', '/Applications/Canopy.app/appdata/canopy-1.4.1.
1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7', '/Applications/Canopy.app
/appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/
plat-darwin', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.macosx-x86_64/
Canopy.app/Contents/lib/python2.7/plat-mac', '/Applications/Canopy.app/appdata/
canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/plat-mac/
lib-scriptpackages', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.
macosx-x86_64/Canopy.app/Contents/lib/python2.7/lib-tk', '/Applications/Canopy.app
/appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/
lib-old', '/Applications/Canopy.app/appdata/canopy-1.4.1.1975.macosx-x86_64/
Canopy.app/Contents/lib/python2.7/lib-dynload', '/Users/yhhan/Library/Enthought/
Canopy_64bit/User/lib/python2.7/site-packages', '/Users/yhhan/Library/Enthought/
Canopy_64bit/User/lib/python2.7/site-packages/PIL', '/Applications/Canopy.app/
appdata/canopy-1.4.1.1975.macosx-x86_64/Canopy.app/Contents/lib/python2.7/
site-packages', '/Users/yhhan/Library/Enthought/Canopy_64bit/User/lib/python2.7/
site-packages/IPython/extensions', '~/mypythonlib']
```