Python

Modules

6. Modules

What are you saying?

- 파이썬 인터프리터를 종료하고 다시 키면, 기존에 사용한 변수나 함수들을 사용할 수 없다
- 더 커다란 프로그램을 작성 할 때, 텍스트 에디터를 사용하며 파일 이름을 입력하며 사용한다.
- 이를 script를 작성했다고 한다.
- 복잡하고 큰 프로젝트를 작업 할 때, 이를 기능 별로 분류하고 관리하고 싶으실텐데…?
- 파이썬은 이를 지원한다!
- Module 이라고 부르고, 이는 다른 모듈에 import 될 수 있다!

6. Modules

Module

- .py 확장자를 사용
- 모듈의 이름은 global variable '__name__'로 알 수 있다.

```
# Fibonacci numbers module

def fib(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

```
>>> import fibo
```

```
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21,
>>> fibo.__name__
'fibo'
```

```
>>> fib = fibo.fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

6.1 More on Module

- 모듈에는 사용가능한 함수와 그 설명이 포함되어 있다
- 일반적으로 설명은 모듈의 도입에 작성한다

- 처음 import 할 때, 한 번 실행 된다
- 각 모듈은 각자의 symbol table을 갖는다
- 모듈의 변수들과 기존에 사용하던 변수와는 충돌 하지 않는다

- 만약 모듈을 잘 알고 있다면, 그들의 변수들을 수정 할 수 있다.

```
modname.itemname
```

6.1 More on Module

Variant of the import

- import시 모듈의 이름을 알면 직접적으로 import 할 수 있다.

```
>>> from fibo import fib, fib2
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

- 모든 모듈을 직접 가져 오는 방법도 있다.

```
>>> from fibo import *
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

'_'로 시작하는 모듈들은 제외된다

6.1 More on Module

6.1.1 Executing moduels as scrpit

- 모듈을 스크립트와 같이 사용 하는 방법

```
python fibo.py <arguments>
```

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

```
$ python fibo.py 50
1 1 2 3 5 8 13 21 34
```

```
>>> import fibo
>>>
```

import를 해도 아무런 변화를 주지 않는다!

6.1 More on Module

6.1.2 The Module Search Path

- import spam.py 가 interpret 되면
- First, built-in module 에서 찾는다. 없다면
- Second, sys.path에서 찾는다
  - input script가 포함된 폴더
  - PYTHONPATH (환경변수)
  - 설치 할 때 추가한 dependent

6.1 More on Module

6.1.3 “Compiled” Python files

- 모듈을 로드하는 속도를 빠르게 하는 방법이 있다!
- “__pycache__” 폴더에 module.version.pyc
  - version은 python의 버전 정보가 들어간다.
  - 예를 들면, Cpython 3.3에서 spam.py 가 있다면
    __pycache__/spam.cpython-33.pyc
  - 오래되거나 recompile 되어야된다면 알아서 함.

- Cahce를 살펴보지 않는 두 가지 환경

  항상 recompile되며 결과를 저장하지 않고 cmd line 에서 호출

  source module이 아닐 경우(compiled module)

Tips for Expert는 나중에 전문가가 되면 다뤄보자

6.2 Standard Modules

What is Standard?

- Python Library Reference

- 몇 개의 모듈은 인터프리터에 포함 되었다.
- 어떤 모듈은 인터프리터 작동에 접근하지만, 언어의 핵심에는 포함 되지 않아도 빌트인 효율을 위해서나 system call 과 같은 OS 우선순위에 접근하기 위해 사용됨

- Winreg 와 같은 모듈은 Windows OS 에서만 사용 가능

- 재밌는 예로 sys 모듈이 있는데, interactive mode 수정 가능

- 모듈 검색 디렉토리 추가도 가능

```
>>> import sys
>>> sys.path.append('/ufs/guido/lib/python')
```

```
>>> import sys
>>> sys.ps1
'>>> '
>>> sys.ps2
'... '
>>> sys.ps1 = 'C> '
C> print('Yuck!')
Yuck!
C>
```

6.3 The dir( ) Function

What did you define?

- dir 함수는 선언된 모듈들의 리스트를 반환해준다 (sorted list of string)

- Argument를 입력하지 않으면 현재 모듈에서 사용하는 모듈들의 알려준다

```
>>> import fibo, sys
>>> dir(fibo)
['__name__', 'fib', 'fib2']
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '_
 '__package__', '__stderr__', '__stdin__', '__stdou
```

```
>>> a = [1, 2, 3, 4, 5]
>>> import fibo
>>> fib = fibo.fib
>>> dir()
['__builtins__', '__name__', 'a', 'fib', 'fibo', 'sys']
```

6.4 Packages

- 패키지는 모듈의 모임
- A.B 와 같은 형식으로 사용

  패키지 A 안의 B 모듈을 지칭

```
import sound.effects.echo
sound.effects.echo.echofilter(input, output, delay=0.7, atten=4)
from sound.effects import echo
echo.echofilter(input, output, delay=0.7, atten=4)
from sound.effects.echo import echofilter
echofilter(input, output, delay=0.7, atten=4)
```

6.4 Packages

6.4.1 Importing * From a Package

```
from sound.effects import *
```

- 위와 같은 양식으로 모든 모듈을 가져올 수 있다

  (package 내의 __init__.py 에서 __all__ 이 설정 되어야 됨)

  (__init__.py 가 존재 않지 않는다면, sound.effects 모듈만)

6.4 Packages

6.4.2 Intra-package References

- 패키지 내부에서 하나의 모듈만 콕 찝어서 가져 올 수 있다

- Relative-import

현재 내가 A.B.C 모듈에서 작업중이라면

```
from . import D     # import A.B.D
from .. import E    # import A.E
from ..F import G # import A.F.G
```

6.4 Packages

6.4.3 Packages in Multiple Directories

- __path__ 라는 attribute 이 있는데
- __init__.py 의 경로를 저장하고 있다
- 이를 수정하는 경우는 매우 드물지만, 사용 하는 패키지의 일부 모듈을 확장하는 경우에 이용된다.

감사합니다