파이썬 프로그래밍

28차시

모듈의 이해와 활용 1

⚠ 학습개요

… 모듈(module)
… 모듈을 불러오는 다양한 import 구문
… 직접 모듈 생성과 사용

⚠ 학습목표

… 파이썬의 모듈을 이해할 수 있다.
… 표준 모듈과 써드 파티 모듈(third party modules)을 이해할 수 있다.
… 다양한 import 구문을 사용할 수 있다.
… 직접 모듈을 구현해서 활용할 수 있다.

Chapter 1.

모듈(module)

PYTHON PROGRAMMING

⚠ 함수나 변수의 파이썬 코드가 저장된 소스 모듈

+ 모듈(module)

- 함수나 변수, 클래스 정의 등의 파이썬 코드가 저장된 소스 파일
- 모듈은 파이썬 프로그램에서 불러와(import) 사용

+ 써드 파티 모듈(third party modules)

- 여러 회사나 전문가가 개발해 배포하는 모듈

+ 표준 모듈

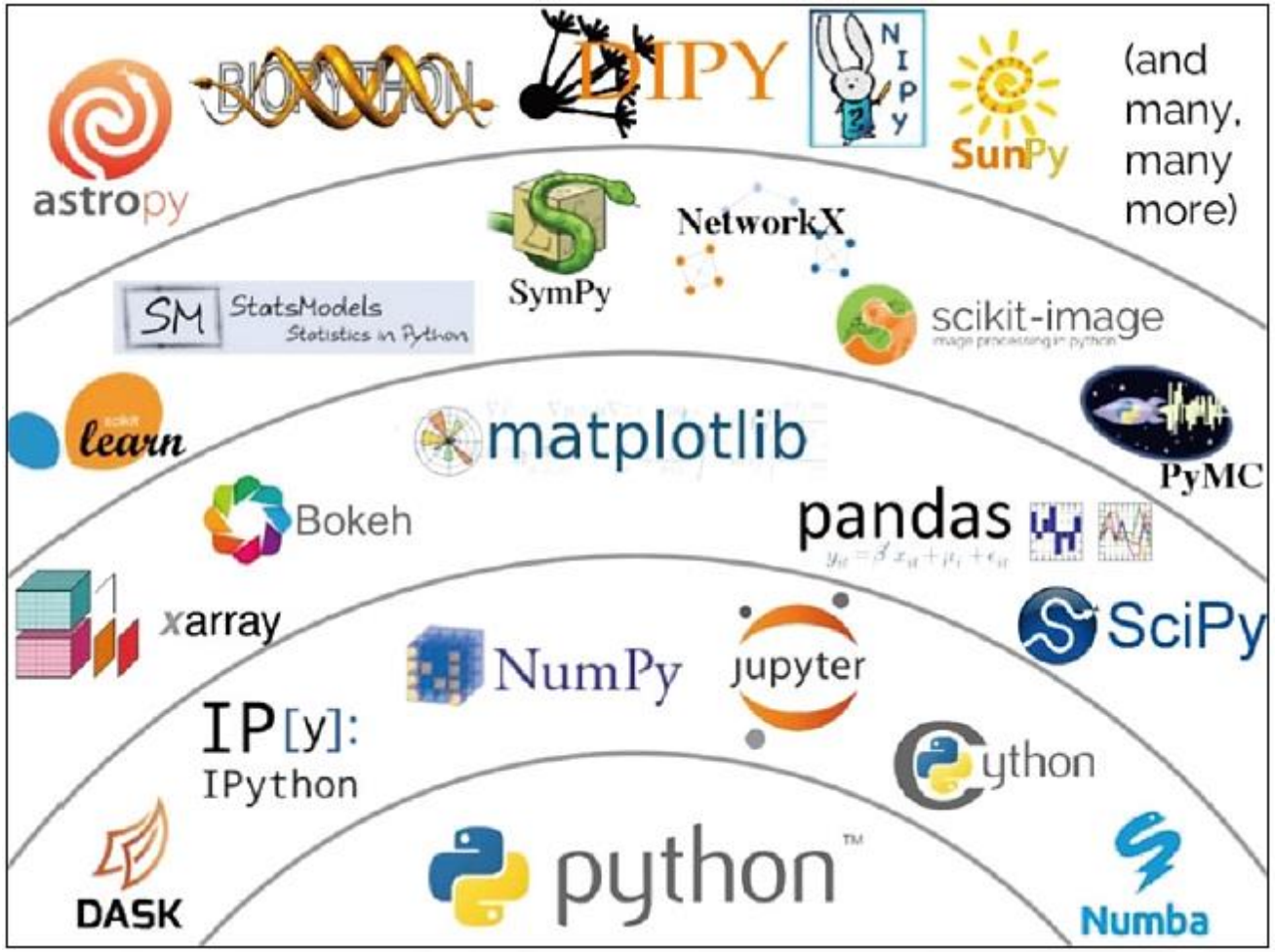
- 파이썬에 기본적으로 설치된 모듈
- random이 있으며 math와 statistics, turtle, datetime, sys, os 등 매우 다양
- 표준 모듈의 이름은 sys 모듈의 builtin_module_names 변수에 저장

⚠ 함수나 변수의 파이썬 코드가 저장된 소스 모듈

+ 표준 모듈

```
>>> import sys
>>> print(sys.builtin_module_names)
('_ast', '_bisect', '_blake2', '_codecs', '_codecs_hk',
'_codecs_iso2022','_codecs_jp', '_codecs_kr', '_codecs_tw',
'_collections', '_csv', '_datetime', '_functools', '_heapq',
'_imp', '_io', '_json', '_locale', '_lsprof', '_md5',
'_multibytecodec', '_opcode', '_operator', '_pickle', '_random',
'_sha1', '_sha256', '_sha3', '_sha512', '_signal', '_sre',
'_stat', 'string', '_struct', '_symtable', '_thread',
'_tracemalloc', '_warnings', '_wekref', '_winapi', 'array',
'atexit', 'audioop', 'binacii', 'builtins', 'cmath', 'errno',
'faulthandler', 'gc', 'itertools', 'marshal', 'math', 'mmap',
'msvcrt', 'nt', 'parser', 'sys', 'next', 'object', 'oct', 'open',
'open_in_spyder', 'ord', 'pow', 'time', 'winreg', 'xxsubtype',
'zipimport', 'zlip')
```

⚠ 다양한 써드 파티 모듈



[그림28-1] 다양한 써드 파티 모듈

끝

Chapter 2.

모듈을 불러오는 다양한 import 구문

PYTHON PROGRAMMING

⚠ 모듈을 불러오는 다양한 import 구문과 표준 모듈 math

+ 표준 모듈 math

```
>>> import math
>>> dir(math)
['__doc__', '__loder__', '__name__', '__package', '__spec__',
'acos','acosh', 'adinh', 'atan', 'atan2', 'atanh', 'ceil',
'cospysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp',
'expml',  'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gammma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isnif',
'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2',
'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt',
'tan', 'tanh', 'tau', 'trunc',]
```

⚠ 모듈을 불러오는 다양한 import 구문과 표준 모듈 math

```
>>> help(math.fsum)
Help on built-in function fsum in module math:
fsum(...)
    fsum(iterable)

    Return an accurate floating point sum of values in the
iterable.
    Assumes IEEE-754 floating point arithmetic

>>> import math as m
>>> help(m.radians)
Help on built-in function radians in module math:
radians(...)
   radians(x)

   Convert angle x from degrees to radians.
```

⚠ 모듈을 불러오는 다양한 import 구문과 표준 모듈 math

```
>>> from math import degrees
>>> help(degress)
Help on built-in function radians in module math:
degrees(...)
    degrees(x)

    Convert angle x from degrees to radians.
```

⚠ 모듈을 불러오는 다양한 import 구문과 표준 모듈 math

```
>>> from math import gcd, trunc
>>> print(gcd(12, 16))
4
>>> print(trunc(3.141592))
3


>>> import math, random
>>> print(math.fsum([1.2, 2.5]))
3.7
>>> print(random.randint(1, 10))
2



>>> from math import sqrt as sr
>>> print(sr(4))
2.0
```

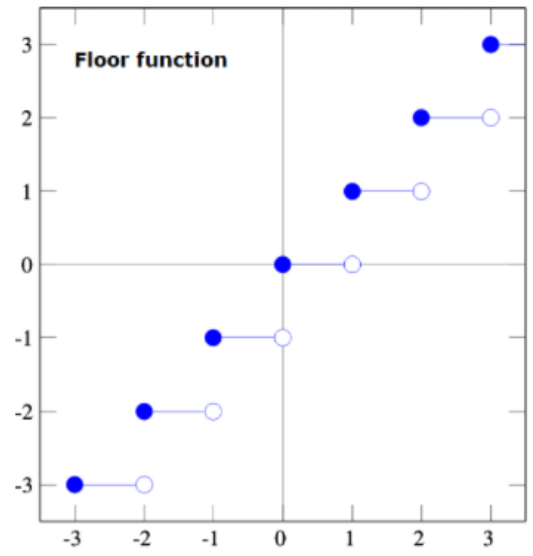
⚠ import 구문으로 모듈 math를 불러오기

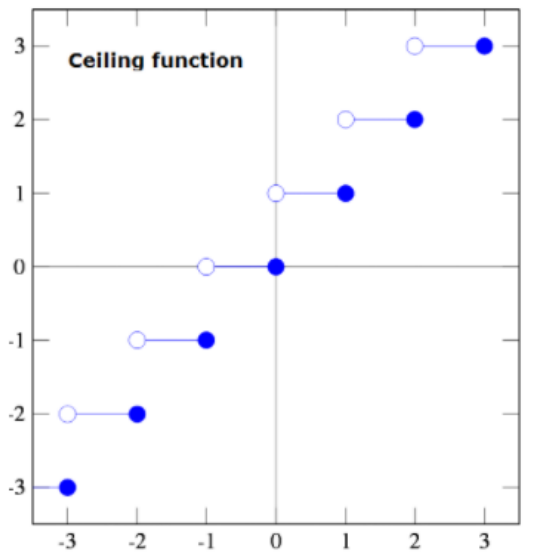
+ 함수 floor(실수)

실수보다 작거나 같은 가장 큰 정수, 즉 바닥 정숫값

+ 함수 ceil(실수)

실수보다 크거나 같은 가장 작은 정수, 즉 천정 정숫값





⚠ import 구문으로 모듈 math를 불러오기

[코딩실습] import 구문으로 모듈 math를 불러오기 난이도 기본

```
1. import math
2. print(math.pi) # 원주율 pi
3.
4. from math import e # 자연수 e
5. print(e)
6. from math import degrees, radians
7. print(degrees(math.pi), radians(90))
8. from math import floor, ceil # 바닥 값, 천정 값 함수
9. print(floor(3.8), ceil(3.1))
10. from math import factorial as fact # n! 함수
11. print(fact(5))
```

결과

```
3.141592653589793
2.718281828459045
180.0 1.5707963267948966
3 4
120
```

⚠ import 구문으로 모듈 math를 불러오기

✚ 모듈 이용의 다양한 구문

문장	설명	비고
import math	모듈 사용 구문으로, 콤마로 나열해 여러 모듈을 한 번에 사용 가능	import math, random
import math as m	모듈을 별칭인 다른 이름으로 지정 가능	
from math import degrees	• 모듈에서 지정된 이름을 모듈 이름 없이 사용가능, import *는 모듈의 모든 이름 사용 가능 • 모듈에서 지정된 이름도 사용 가능	from math import * from math import gcd, trunc
from math import sqrt as sr	모듈에서 지정된 이름을 다른 이름으로 사용	

⚠ 표준 모듈 random의 사용

+ 난수를 위한 표준 모듈 random

```
>>> import random as rd
>>> dir(rd)
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random',
'SG_MAGICCONST','Systemrandom', 'TWOPI', '_BuiltinMethodType',
'_MethodType', '_Sequence', '_Set', '__all__', '__builtians__',
'__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__',  '_acos', '_bisect', '_ceil', '_cos',
'_e', '_exp', '_inst', '_itertools', '_log', '_pi', '_random',
'_sha512', '_sin', '_sqrt', '_test', '_test_generator',
'_urandom', '_warn', 'betavariate', 'choice', 'choices',
'expovariate', 'gammmavariate', 'gauss', 'getrandbits',
'getstate', 'lognormvariate', 'normalvariate', 'paretovariate',
'randint', 'random', 'randrange', 'sample', 'seed', 'setstate',
'shuffle', 'triangular', 'uniform', 'vonmisesvariate',
'weibullvariate']
```

⚠ 표준 모듈 random의 사용

+ 난수를 위한 표준 모듈 random

- 함수 random()
  - 0에서 1보다 작은 실수 난수를 반환

```
>>> help(rd.random)
Help on built-in function random!
random (...) method of random. Random instance
    random() -> x in the interval[0, 1).
```

⚠ 표준 모듈 random의 사용

+ 난수를 위한 표준 모듈 random

- 함수 shuffle(list)
  - 인자인 list의 순서를 임의로 섞어(shuffling) 바꿈

```
>>> help(rd.shuffle)
Help on method shuffle in module random!
Shuffle(x, random =None) method of random.Random instance
    Shuffle list x in place, and return None.

    Optional argument random is a 0-argument function returning a
    random float in [0.0 , 1.0); if it is the default None, the
    standard random. Random will be used.
```

⚠ 모듈 random의 주요 함수

[코딩실습] 모듈 random의 주요 함수

난이도 기본

```
1. import random as rd
2.
3. for i in range(3):
4.     print(rd.random()) # 0에서 1보다 작은 실수 난수 생성
5.
6. cards = [[1, '송학'], [2, '메조'], [3, '벚꽃'], [4, '흑싸리'],
   [5, '초'], [6, '모란']]
7. print(cards)
8. for _ in range(3):
9.     rd.shuffle(cards) #카드를 섞음
10.    print(cards)
11.
12.print(rd.sample(cards, 2)) #카드에서 2개를 선택
```

⚠ 모듈 random의 주요 함수

결과

```
0.36302499440976255
0.19348727139204103
0.07595135197379321
[[1, '송학'], [2, '메조'], [3, '벚꽃'], [4,'흑싸리'], [5, '초'],
[6, '모란']]
[[2, '메조'], [3, '벚꽃'], [6, '모란'], [4,'흑싸리'], [5, '초'],
[1, '송학']]
[[5, '초'], [1, '송학'], [4, '흑싸리'], [2,'메조'], [3, '벚꽃'],
[6, '모란']]
[[5, '초'], [1, '송학'], [3, '벚꽃'], [4, '흑싸리'], [6, '모란'],
[2, '메조']]
[3, '벚꽃'], [6, '모란']]
```

끝

Chapter 3.

직접 모듈 생성과 사용

PYTHON PROGRAMMING

⚠ 직접 모듈을 작성해 프로그램에서 실행

+ 모듈 작성

- 일반 소스와 동일
  - 다른 파이썬 프로그램에서 활용될 수 있는 함수, 클래스 및 변수의 정의 등이 포함
- 폴더 'D:\Python Code\ch09'에 저장
- 모듈 이름: hello, hello.py로 저장

hello.py

```
def hi():
    print('Hi, Python!')

msg = '파이썬, 재미있네요!'
```

mypg.py

```
import hello
hello.hi()

from hello import msg
print(msg)
```

[그림28-2] 모듈 hello.py와 모듈을 불러 사용하는 프로그램 mypg.py

⚠ 직접 모듈을 작성해 프로그램에서 실행

✚ 오른쪽 mypg.py를 파이썬 IDLE 쉘에서 실행한 결과

```
>>>
========== RESTART: D:/Paython Code/ch09/mypg.py ==========
Hi, Python!
파이썬, 재미있네요!
```

⚠ 쉘에서 자신이 만든 모듈을 실행

+ 쉘에서 모듈 hello를 사용하는 두 가지 방법

+ 첫 번째 방법

- 써드 파티 모듈 설치 폴더
- site-packages에 자신이 만든 모듈을 복사 - 폴더 'lib\site-packages'

```
>>> import sys
>>> print(sys.path)
['', 'c:\\Python\\Python37-32\\lib\\idlelib', 'c:\\Python\\Pyhon37-32\\python37.zip',
'c:\\Python\\Python37-32\\DLLs', 'c:\\Python\\Python37-32\\lib', 'c:\\ Python\\Python37-32',
'c:\\Python\\Python37-32\\lib\\site-package']

>>> import hello
>>> hello.hi()
Hi, Python!
>>> hello.msg
'파이썬, 재미있네요!'
```

⚠ 쉘에서 자신이 만든 모듈을 실행

+ 두 번째 방법 2

- sys.path에 append( ) 메소드로 추가
  - 원래 hello.py가 저장된 폴더인 'D:\Python Code\ch09'

```
>>> import sys
>>> print (sys.path)
['', 'c:\\Python\\Python37-32\\Lib\\idlelib', 'c:\\Python\\Pyhon37-
32\\python37.zip', 'c:\\Python\\Python37-32\\DLLs',
'C:\\Python\\Python37-32\\lib', 'c:\\ Python\\Python37-
32', 'C:\\Python\\Python37-32\\lib\\site-package', 'D:\\Python Code
\\ch09']
>>> import hello
>>>
```

끝

SUMMARY

⚠모듈(module)

… 함수나 변수, 클래스 정의 등의 파이썬 코드가 저장된 소스 파일

… 표준 모듈과 써드 파티 모듈(third party modules)

⚠모듈을 불러오는 다양한 import 구문

… 표준 모듈 random 활용

⚠직접 모듈 생성과 사용

… 쉘에서 사용 방법