

IT 개론

7장. 파이썬 모듈

목차

1. 모듈 소개하기
2. 모듈 사용하기
3. random 모듈
4. time 모듈
5. sys 모듈

<https://docs.python.org/3/library/>

1. 모듈 소개하기

◆ 파이썬 구성 요소

modules + **classes** + **built-in functions** (내장함수)

random module

random()
shuffle()
randint()

math module

pow ()
sin()
cos()

..... module들

내장 함수들

abs() input() sum()
type() print() pow()

class들

class list

append ()
insert()
sort()

class str

upper ()
lower()
count()

1. 모듈 소개하기

◆ 모듈 (module)

- 코드들을 한 단위로 묶어 사용할 수 있게 하는 하나의 단위.
- 서로 연관된 작업을 하는 코드들의 모임으로 구성됨.
- 모듈의 종류
 - ① 표준 모듈 – 파이썬 패키지 안에 포함된 모듈
 - ② 사용자 모듈 – 사용자가 만드는 모듈
 - ③ 써드 파티 (third party) 모듈 – 개인이 만들어서 제공하는 모듈

1. 모듈 소개하기

◆ 모듈 사용의 장점

- 코드의 재사용
- 코드를 이름으로 구분하고 관리할 수 있음.
 - 모듈은 각각의 이름과 동일한 이름 공간을 가짐.
 - `math`라는 모듈을 `import`하면 `math`라는 이름 공간이 생성되고 그 `module` 내에 있는 속성과 메소드들을 사용할 수 있음.
 - 다른 모듈에 같은 이름의 메소드가 있어도 소속 모듈이 다르기 때문에 충돌이 생기지 않음.

2. 모듈 사용하기

◆ 모듈은 반드시 import 후에 사용할 수 있다

```
>>> import math
>>> math.pow(2,3)      # 모듈 math 내의 pow 함수
8.0
>>> math.pi           # 모듈 math 내의 pi 속성 (상수)
3.141592653589793
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__',
'__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan',
'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs',
'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp',
'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf',
'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan',
'tanh', 'trunc']
```

2. 모듈 사용하기

◆ 모듈 import 방법 - 4 가지

① import <모듈>

```
>>> import math
>>> math.pi      # math 모듈에 있는 pi 속성
3.141592653589793
>>> math.pow(2,5) # math 모듈에 있는 pow 메소드
32.0
```

② from <모듈> import <함수>

```
>>> from math import pow
>>> pow(2,5)      # 이 경우 메소드명만 사용 가능하다
32.0
```

2. 모듈 사용하기

◆ 모듈 import 방법 - 4 가지

- ③ `from <모듈> import *` - <모듈>에 있는 모든 메소드를 이름으로 사용 가능함.

```
>>> from math import *
>>> log2(1024)
10.0
>>> pow(3,5)
243.0
```

- ④ `import <모듈> as <alias>` - <모듈> 대신 <alias>를 이름으로 사용함.

```
>>> import math as mt
>>> mt.pow(3,5)
243.0
>>> mt.log2(1024)
10.0
```


3. random 모듈

◆ random 모듈 사용 예

- 임의의 수 하나를 선택하기 (`random()`)
- 주사위 던지기 (`randrange()`)
- 아이팟의 셔플처럼 데이터 섞기 (`shuffle()`)
- 주어진 데이터 중에서 임의로 하나의 데이터 뽑기 (`choice()`)

3. random 모듈

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

3. random 모듈

메소드	설명
<code>choice(seq)</code>	입력받은 seq 객체의 임의의 아이템을 반환함.
<code>gauss(m,sb)</code>	가우스 random number를 반환함.
<code>randint(a,b)</code>	$a \leq N \leq b$ 사이의 임의의 정수 N을 반환함.
<code>random()</code>	$0.0 \leq F < 1.0$ 사이의 임의의 float 숫자를 반환함.
<code>randrange([start], stop[,step])</code>	<code>range()</code> 결과 중에서 임의로 선택해서 반환함.
<code>sample(seq,k)</code>	seq에서 k개의 원소를 임의로 중복없이 반환함.
<code>shuffle(x[,random])</code>	입력받은 시퀀스 객체를 섞는다.
<code>uniform(a,b)</code>	a와 b 사이의 임의의 float 숫자를 반환함.

3. random 모듈

◆ 임의의 정수 생성

```
>>> import random
>>> random.randrange(10)
3
>>> random.randrange(5,10)
9
>>> random.randrange(5,10)
5
>>> random.randrange(5,15,3)
5
>>> random.randrange(5,15,3)
11
>>> random.randrange(5,15,3)
8
```

3. random 모듈

◆ 임의의 정수 생성

```
>>> import random
```

```
>>> [random.randrange(20) for i in range(10)] # 중복 허용  
[8, 6, 12, 8, 8, 0, 12, 11, 6, 8]
```

```
>>> random.sample(range(20),10) # 중복 허용하지 않음  
[12, 8, 17, 1, 3, 6, 9, 0, 11, 10]
```

```
>>> [random.randrange(0,20,3) for i in range(5)]  
[3, 9, 0, 0, 18]
```

3. random 모듈

◆ 임의의 실수 생성

- 0.0에서 1.0 사이의 float 값을 임의로 생성하기

```
>>> import random  
>>> random.random()  
0.9438646187963933  
>>> random.random()  
0.2983023008638803
```

- 입력받은 두 값 사이의 float 값을 임의로 생성하기

```
>>> import random  
>>> random.uniform(3,4)  
3.9694488711194635
```

3. random 모듈

◆ 시퀀스 객체 관련 연산

```
>>> L = list(range(10))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> random.choice(L)
6
>>> [random.choice(L) for i in range(3)]
[1, 6, 4]
>>> random.sample(L,3)
[5, 0, 4]
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> random.shuffle(L)
>>> L
[1, 4, 0, 6, 3, 7, 5, 2, 8, 9]
```

```
>>> L = list(range(10))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> s = random.sample(L, len(L))
>>> s
[9, 4, 5, 3, 2, 6, 8, 1, 0, 7]
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```

3. random 모듈

◆ 시퀀스 객체 관련 연산

```
>>> L = ['red', 'yellow', 'blue', 'green']
>>> random.shuffle(L) # L의 원소들을 섞는다. (카드 섞기 등에 이용)
>>> print(L)
['green', 'red', 'yellow', 'blue']
>>> random.choice(L) # L에서 임의의 원소를 하나 알려 준다.
'green'
>>> name = "Alice Wonderland" # 문자열에서도 임의의 문자를 뽑는다
>>> random.choice(name)
'd'
```


4. time 모듈

◆ time 모듈

- 시간을 표현하고 처리하는 데 이용하는 모듈
- 타임스탬프(timestamp) – 1970년 1월 1일 자정 이후로 초 단위로 측정한 절대 시간
- 타임스탬프를 이용하여 사람이 이해할 수 있는 시간으로 변환한다. 이 때 년, 월, 일, 시, 분, 초와 같은 정보가 필요하다. 파이썬에서는 `struct_time` 시퀀스 객체를 이용하여 이를 표현한다.

4. time 모듈

◆ struct_time 객체

속성	내용
tm_year	년도
tm_mon	월 (1~12)
tm_mday	일 (1~31)
tm_hour	시 (0~23)
tm_min	분 (0~59)
tm_sec	초 (0~61)
tm_wday	요일 (월요일이 '0'임)
tm_yday	1월 1일부터 오늘까지 누적된 날짜를 반환함. (1~366)
tm_isdst	서머타임 (0, 1, -1)

4. time 모듈

```
>>> import time
>>> dir(time)
['_STRUCT_TM_ITEMS', '__doc__', '__loader__', '__name__',
 '__package__', '__spec__', 'altzone', 'asctime', 'clock',
 'ctime', 'daylight', 'get_clock_info', 'gmtime', 'localtime',
 'mktime', 'monotonic', 'perf_counter', 'process_time',
 'sleep', 'strftime', 'strptime', 'struct_time', 'time',
 'timezone', 'tzname']
```

4. time 모듈

메소드	설명
<code>time()</code>	1970년 1월 1일 자정 이후로 누적된 초를 float로 변환. >>> <code>time.time()</code> # 1970년 1월 1일부터 누적 초 1459079499.528705
<code>gmtime([secs])</code>	입력된 초를 변환해, UTC 기준의 <code>struct_time</code> 시퀀스 객체로 반환함. 인자없이 호출되면, <code>time()</code> 을 이용해 현재 시간을 변환함. >>> <code>t = time.gmtime(time.time())</code> >>> <code>t</code> <code>time.struct_time(tm_year=2016, tm_mon=3, tm_mday=27, tm_hour=11, tm_min=53, tm_sec=56, tm_wday=6, tm_yday=87, tm_isdst=0)</code> >>> <code>t.tm_mon</code> 3 >>> <code>t.tm_hour</code> 11

4. time 모듈

메소드	설명
<code>asctime([t])</code>	<p><code>struct_time</code> 시퀀스 객체를 인자로 받음.</p> <pre>>>> t = time.gmtime(time.time()) >>> time.asctime(t) 'Sun Mar 27 11:53:56 2016'</pre>
<code>localtime([secs])</code>	<p>입력된 초를 지방표준시 기준의 <code>struct_time</code> 시퀀스 객체를 반환함. 인자없이 호출되면, <code>time()</code>을 이용해 현재 시간을 변환함.</p> <pre>>>> time.localtime() time.struct_time(tm_year=2016, tm_mon=3, tm_mday=27, tm_hour=21, tm_min=13, tm_sec=2, tm_wday=6, tm_yday=87, tm_isdst=0)</pre>
<code>mktime(t)</code>	<p>지방표준시인 <code>struct_time</code> 시퀀스 객체를 인자로 받아 타임스탬프를 반환함.</p> <pre>>>> time.mktime(time.localtime()) 1459080873.0</pre>

4. time 모듈

메소드	설명
sleep(secs)	secs 초만큼 정지시킴.

```
import time
t = time.time()
time.sleep(10)
t2 = time.time()

spendtime = t2 - t
print('Before timestamp :', t)
print('After timestamp  :', t2)
print('Wait {0} seconds'.format(spendtime))
```

```
Before timestamp : 1459082125.995574
After timestamp  : 1459082135.996146
Wait 10.000571966171265 seconds
```

4. time 모듈

메소드	설명
<code>strftime(format[, t])</code>	직접 포맷을 지정해서 출력함.
<code>strptime(string[, format])</code>	

지시자	내용	지시자	내용
<code>%y</code>	연도를 축약하여 표시	<code>%p</code>	오전(AM)/오후(PM) 표시
<code>%Y</code>	연도를 축약하지 않고 표시	<code>%a</code>	축약된 요일 이름
<code>%b</code>	축약된 월 이름	<code>%A</code>	축약되지 않은 요일 이름
<code>%B</code>	축약되지 않은 월 이름	<code>%w</code>	요일을 숫자로 표시 (일요일 0)
<code>%m</code>	숫자로 표현한 월 (01~12)	<code>%j</code>	1월 1일부터 누적 날짜 (001~336)
<code>%d</code>	일 (01~31)		
<code>%H</code>	24시를 기준으로 한 시 (00~23)		
<code>%I</code>	12시를 기준으로 한 시 (01~12)		
<code>%M</code>	분 (00~59)		
<code>%S</code>	초 (00~61)		

4. time 모듈

```
>>> from time import localtime, strftime
>>> strftime("%B %dth %A %I:%M", localtime())
'March 27th Sunday 11:03'

>>> strftime("%Y-%m-%d %I:%M", localtime())
'2016-03-27 11:03'

>>> strftime("%y/%m/%d %H:%M:%S", localtime())
'16/03/27 23:04:11'

>>> strftime("%y/%m/%d %H:%M:%S")
'16/03/27 23:04:36'

>>> strftime("%x %X", localtime())
'03/27/16 23:05:07'
```


4. time 모듈

```
>>> import time
>>> t = time.ctime(time.time())
>>> t
'Sun Mar 27 23:06:01 2016'
```

```
>>> time.strptime(t)
time.struct_time(tm_year=2016, tm_mon=3, tm_mday=27,
tm_hour=23, tm_min=6, tm_sec=1, tm_wday=6, tm_yday=87,
tm_isdst=-1)
```

```
>>> time.strptime(t, "%a %b %d %H:%M:%S %Y")
time.struct_time(tm_year=2016, tm_mon=3, tm_mday=27,
tm_hour=23, tm_min=6, tm_sec=1, tm_wday=6, tm_yday=87,
tm_isdst=-1)
```

5. sys 모듈

◆ sys 모듈

- 파이썬 인터프리터와 관련된 정보와 기능을 제공

```
>>> import sys
>>> name = sys.stdin.readline()
Alice Wonderland
>>> name
'Alice Wonderland\n'
```

```
import sys
r = lambda:sys.stdin.readline()
n = int(r())
a = []
for k in range(n):
    x, y = map(int, r().split())
    a.append((x,y))
print(a)
```

```
5
4 9
3 0
8 3
10 20
5 6
[(4, 9), (3, 0), (8, 3), (10, 20), (5, 6)]
```