38.표준 라이브러리₩ -내장함수, 모듈

2018.12

일병 김재형

표준 라이브러리

표준 라이브러리(standard library)

- Python내부에 적재된 내장함수, 내장형, 모듈, 변수 등을 포함한다.
- 엄청나게 많다.
- 내장함수와 모듈 몇 개만 가져왔다.

내장함수(Built-in Functions)

내장함수(Built-in Functions)

- 여러 내장함수(hepl, print, super, id 등등)을 배웠다.
- 배우지 않았던 내장함수 중 사용할 만한 함수 일부를 가져왔다.
- 더 많은 내장함수는 https://docs.python.org/3/library/functions.html 를 참조한다.

내장함수(Built-in Functions)

abs(x)

- 숫자의 절대값을 돌려준다.

```
>>> abs(3)
3
>>> abs(-12)
12
```

내장함수(Built-in Functions)

all(반복 가능한(iterable) 자료형)

- 자료형 내부의 값이 전부 참이면 True, 거짓이 있으면 False를 반환

```
>>> a=["", "ab", "bc"]
>>> all(a)
False
>>> b=a[1:]
>>> b
['ab', 'bc']
>>> all(b)
True
```

내장함수(Built-in Functions)

any(반복 가능한(iterable) 자료형)

- 자료형 내부의 값 중 하나라도 참이면 True, 전부 거짓이면 False를 반환

```
>>> a=["", "ab", "bc"]
>>> any(a)
True
>>> b=[None, ""]
>>> any(b)
False
```

내장함수(Built-in Functions)

filter(함수, 반복 가능한(iterable) 자료형)

- 반복 가능한 자료형내의 요소를 함수에 입력해, 참(True)인 것만 추출한다.

```
>>> def negative(x):
...     return x<0
...
>>> a = filter(negative, [1, -2, -3, 4])
>>> a
<filter object at 0x7f42fd4ae630>
>>> a = list(a)
>>> a
[-2, -3]
```

내장함수(Built-in Functions)

map(함수, 반복 가능한(iterable) 자료형)

- 반복 가능한 자료형의 각 요소를 함수에 넣어 결과값을 묶어 반환한다.

```
>>> a = map(int, ['1', '2', '3'])
>>> a
<map object at 0x7f42fd4ae5c0>
>>> a = list(a)
>>> a
[1, 2, 3]
```

내장함수(Built-in Functions)

max, min(반복 가능한(iterable) 자료형)

max, min(매개변수1, 매개변수2, …)

- 반복 가능한 자료형이나 매개변수를 받아 최소와 최대값을 구한다.

```
>>> a=['a','b']
>>> max(a)
'b'
>>> min(1, 2, 3)
1
```

내장함수(Built-in Functions)

sorted(반복 가능한(iterable) 자료형)

- 반복 가능한 자료형을 받아 정렬한 후 리스트로 반환한다.
- list의 sort 메서드는 리스트를 반환하지 않고, 리스트 내부를 정렬한다.

```
>>> a = sorted((1, 3, 4, 0, -1))
>>> a
[-1, 0, 1, 3, 4]
>>> b = [1, 0, 9, 2]
>>> b.sort()
>>> b
[0, 1, 2, 9]
```

내장함수(Built-in Functions)

zip(반복 가능한(iterable) 자료형1, …)

- 동일한 개수의 반복 가능한 자료형을 받아 각 요소를 index별로 묶어 반환한다.

```
>>> a = list(zip([1, 2, 3, 4], [5, 6, 7, 8]))
>>> a
[(1, 5), (2, 6), (3, 7), (4, 8)]
>>> for one, two in a:
...     print(one, two)
...
1 5
2 6
3 7
4 8
```

```
>>> b = list(zip("abcd", "efgh"))
>>> b
[('a', 'e'), ('b', 'f'), ('c', 'g'), ('d', 'h')]
```

표준 모듈

os

- os자원인 환경 변수, 디렉터리, 파일등을 제어할 수 있게 하는 모듈
- os.environ
  - 환경변수값 확인하기

```
>>> os.environ
environ({'SHELL': '/bin/bash', 'TERM': 'xterm
eminar18', 'SSH_TTY': '/dev/pts/1', 'USER': '
```

- os.getcwd()
  - 현재 디렉터리 위치 반환
- os.chdir(인수)
  - 현재 디렉터리 위치 변경

```
>>> os.getcwd()
'/workspace/PythonSeminar18'
>>> os.chdir('./Exercise')
>>> os.getcwd()
'/workspace/PythonSeminar18/Exercise'
```

표준 모듈

os

- os.system(인수)
  - 인수에 사용할 명령어를 넣어 쉘에서 사용하는 명령어를 python내 에서 사용할 수 있다.
  - 인수를 사용자에게서 받아 사용할 경우 **보안상 취약**해짐으로 주의해야 한다.
  - ※ 만일 사용자가 rm –rf /를 입력하면 전체 데이터 삭제 가능
  - ※ 권한상승 등의 문제도 발생할 수 있다.

```
>>> os.getcwd()
'/workspace/PythonSeminar18/Exercise'
>>> os.system("ls -al")
합계 64
drwxrwxr-x 14 root root 4096  2월 22 10:07 .
drwxrwxr-x  7 root root 4096  2월 23 12:21 ..
drwxrwxr--  2 root root 4096 12월 22 09:17 BMI_calculator
drwxrwxr--  2 root root 4096  2월  4 03:15 LCD_display
-rw-rw-r--  1 root root  249 12월 25 05:37 README.md
```

표준 모듈

os

- 기타 유용한 os 관련 함수
- 고수준의 파일과 디렉터리 처리를 하기 위해서는 shutil모듈을 사용한다.

함수	설명
os.mkdir(디렉터리명)	디렉터리 생성
os.rmdir(디렉터리명)	디렉터리 삭제
os.unlink(파일명)	파일 삭제
Os.rename(src, dst)	원본 이름인 Src를 dst로 바꾼다.

표준 모듈

glob

- glob.glob(path)
  - 유닉스 셸이 사용하는 규칙에 따라 지정된 패턴과 일치하는 경로명을 찾는다.
  - 메타문자 *, ?를 통해 원하는 파일을 검색할 수 있다.

```
>>> glob.glob("/workspace/PythonSeminar18/*")
['/workspace/PythonSeminar18/jupyter_README_kor.md', '/w
goorm.manifest', '/workspace/PythonSeminar18/Users', '/w
/Diary', '/workspace/PythonSeminar18/TeachingMaterials']
>>> glob.glob("/workspace/PythonSeminar18/E*")
['/workspace/PythonSeminar18/Exercise']
```

표준 모듈

tempfile

- 파일을 임시로 만들어서 사용시 유용
- tempfile.mktemp()
  - 중복되지 않은 임시 파일의 이름을 무작위로 리턴
  - 이를 통해 임시 파일을 만든 후 종료해도 삭제되지 않는다.

```
>>> file = tempfile.mktemp()
>>> filepath = tempfile.mktemp()
>>> filepath
'/tmp/tmptrwfg1r4'
```

```
root@goorm:/tmp# ls -al
합계 12
drwxrwxrwt   3 root root 4096  2월 24 02:52 .
drwxr-xr-x 206 root root 4096 12월 29 00:14 ..
drwx------   2 root root 4096  5월 23  2018 tmplwt98u7t_kernels
```

표준 모듈

tempfile

- 파일을 임시로 만들어서 사용시 유용
- tempfile.mktemp()
  - 중복되지 않은 임시 파일의 이름을 무작위로 리턴
  - 이를 통해 임시 파일을 만든 후 종료해도 삭제되지 않는다.

```
>>> f = open(filepath, mode='w', encoding='utf-8')
>>> f
<_io.TextIOWrapper name='/tmp/tmptrwfglr4' mode='w' encoding='utf-8'>
>>> f.close()
```

```
root@goorm:/tmp# ls -al
합계 12
drwxrwxrwt   3 root root 4096  2월 24 02:59 .
drwxr-xr-x 206 root root 4096 12월 29 00:14 ..
drwx------   2 root root 4096  5월 23  2018 tmplwt98u7t_kernels
-rw-rw-r--   1 root root    0  2월 24 03:00 tmptrwfglr4
```

표준 모듈

tempfile

- 파일을 임시로 만들어서 사용시 유용
- tempfile.TemporaryFile(mode, encoding)
  - file위치를 적지 않는 것 말고는 거의 유사
  - 임시파일(파일과 유사한 객체)을 만든 후, 파일이 종료되면 삭제
  - 기본적으로 w+b(쓰기, 바이너리 모드)로 열린다.
  - 파일이 닫히면 삭제된다.

```
>>> f = tempfile.TemporaryFile(mode='w', encoding='utf-8')
>>> f
<_io.TextIOWrapper name=4 mode='w' encoding='utf-8'>
```

표준 모듈

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.e
2.718281828459045
>>> math.factorial(6)
720
```

math
- pi(π)와 자연상수 e
- 팩토리얼(factorial)
- 삼각함수

함수	설명
cos()	라디안에 대한 코사인 값 계산
sin()	라디안에 대한 사인 값 계산
tan()	라디안에 대한 탄젠트 값 계산
acos()	cos()의 역함수
asin()	sin()의 역함수
atan()	tan()의 역함수

표준 모듈

math

- 제곱근(sqrt(인수))과 제곱(pow(인수1, 인수2))
  - Sqrt(인수)는 인수의 제곱근을 구한다.
  - pow(인수1, 인수2)는 인수1을 인수2로 제곱한 값을 구한다.
  - 제곱은 **를 사용해도 된다.

```
>>> math.sqrt(2)
1.4142135623730951
>>> math.pow(4, 3)
64.0
```

표준 모듈

math

- 로그
  - Log(인수1, 인수2)
    - 첫 번째 인수의 로그를 구한다. 두 번째 인수는 밑수이고, 생략 시 자연상수e로 간주한다.
  - log10()
    - 밑수가 10인 로그 계산

```
>>> math.log(math.e)
1.0
>>> math.log10(10)
1.0
```

표준 모듈

datetime

- 모듈 내부에 여러 클래스가 존재
- 몇 가지를 대표적으로 본다.
- datetime.date 클래스
- datetime.time 클래스
- datetime.datetime 클래스
- datetime.timedelta 클래스
- dateutil 패키지

표준 모듈

datetime.date 클래스

- datetime.date(year, month, day)
  - 숫자로 년, 월, 일을 입력받아 date객체 생성
- date.today()
  - 오늘 날짜가 포함된 date인스턴스 반환
- 인스턴스 속성
  - date.year(년)
  - date.month(월)
  - date.day(일)

```
>>> date = datetime.date.today()
>>> print(date)
2019-02-22
>>> date.year
2019
>>> date.month
2
>>> date.day
22
```

표준 모듈

datetime.date 클래스

- 인스턴스 메서드
  - date.weekday()
    - 요일을 정수로 변환하여 반환
    - 월요일=0, 일요일=6
  - date.isoweekday()
    - 월요일=1, 일요일=7
  - Date.strftime(format)
    - 지정된 포맷에 맞춰 문자열로 반환

```
   February 2019
Mo Tu We Th Fr Sa Su
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28
```

```
>>> date = datetime.date.today()
>>> print(date)
2019-02-22
>>> date.weekday()
4
>>> date.isoweekday()
5
>>> date.strftime("%y.%m.%d")
'19.02.22'
```

표준 모듈

datetime.time 클래스

- datetime.time(hour[, minute[, second[, microsecond[, tzinfo]]]])
  - 숫자로 시, 분, 초, 마이크로 초, 시간대 (time zone)을 입력받아 time객체를 생성
- 인스턴스 속성
  - time.hour(시)
  - time.minute(분)
  - time.second(분)
  - time.microsecond(마이크로초)

```
>>> a = datetime.time(12, 5, 6)
>>> print(a)
12:05:06
>>> a.hour
12
>>> a.microsecond
0
```

표준 모듈

- datetime.time 클래스
  - 인스턴스 메서드
  - time.isoformat()
    - HH:MM:SS.mmmmmm형식 HH:MM:SS 형식의 문자열 반환
  - time.strftime(format)
    - 지정된 포맷에 맞춰 문자열로 반환

```
>>> a = datetime.time(12, 0, 52, 120)
>>> print(a)
12:00:52.000120
>>> a.isoformat()
'12:00:52.000120'
>>> a.strftime("%H:%M-%S.%f")
'12:00-52.000120'
```

표준 모듈

datetime.datetime 클래스

- datetime.datetime(year, month, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]])
  - datetime.date와 datetime.time을 합친 클래스
- 인스턴스 속성과 메서드는 date와 time클래스의 인스턴스 속성과 클래스를 사용할 수 있다.
- datetime.date()
  - datetime의 날짜 정보로 date객체 생성 후 반환
- datetime.time()
  - datetime의 시간 정보로 time 객체 생성 후 반환

표준 모듈

datetime.timedelta 클래스

- datetime.timedelta(days[, seconds[, microseconds[, milliseconds[, minutes[, hours[, weeks]]]]]])
  - 두 날짜나 시간 사이의 차이를 표현
  - 동일한 기간을 다양하게 표현(1주일 == 7일)할 수 있기 때문에 정규화를 통해 유일한 표현방식으로 저장한다.

```
>>> datetime.timedelta(days=-3)
datetime.timedelta(-3)
>>> datetime.timedelta(weeks=2, days=3, hours=-3)
datetime.timedelta(16, 75600)
```

표준 모듈

datetime.timedelta 클래스

- timedelta끼리의 계산
  - timedelta가 반환
  - 절대값(abs)를 통해 차이를 양수로 만들 수 있다.

표준 모듈

datetime.timedelta 클래스

- timedelta와 date와 datetime의 계산
  - date와 datetime이 반환되며 차이만큼 이동한 값이 생성
  - date나 datetime사이에서 뺄셈연산을 하면 timedelta가 반환된다.

```
>>> a=datetime.timedelta(days=1)
>>> b=datetime.date(2018, 12, 5)
>>> c=datetime.date(2018, 11, 1)
>>> b-c
datetime.timedelta(34)
>>> b+a
datetime.date(2018, 12, 6)
>>> b-a
datetime.date(2018, 12, 4)
>>> b+c
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'datetime.date' and 'datetime.date'
```

표준 모듈

datetime.datetime.strftime()

- 날짜와 시간정보를 문자열로 바꿔주는 메서드

날짜 및 시간 지정 문자열	의미
%Y	앞의 빈자리를 0으로 채우는 4자리 연
%m	앞의 빈자리를 0으로 채우는 2자리 달
%d	앞의 빈자리를 0으로 채우는 2자리 일
%H	앞의 빈자리를 0으로 채우는 24시간 형식 2자리 시
%M	앞의 빈자리를 0으로 채우는 2자리 분
%S	앞의 빈자리를 0으로 채우는 2자리 초
%A	영어로 된 요일
%B	영어로 된 월

표준 모듈

datetime.datetime.strptime()

- strptime(날짜와 시간 정보 포함 문자열, 문자열 해독 형식 문자열)
- 문자열로부터 날짜와 시간정보를 읽어 datetime클래스 반환

```
>>> datetime.datetime.strptime("2020/04/02", "%Y/%m/%d")
datetime.datetime(2020, 4, 2, 0, 0)
```

표준 모듈

dateutil.parser.parse()

- 자동으로 형식 문자열을 찾아 datetime.datetime 객체 생성

```
>>> import dateutil.parser as dtparser
>>> dtparser.parse('2019.02.05')
datetime.datetime(2019, 2, 5, 0, 0)
```

표준 모듈

이 외에도 많은 모듈이 존재한다.

- 필요한 것이 있을 때 이미 존재하는지 검색하여 사용한다.

기본과제-전역일 계산기

전역일 계산기(discharged_date.py)

- PythonSeminar18/Exercise/discharged_date/data.txt를 활용(복사하여 자신의 디렉터리에 넣어)하여 기수별 입대일과 전역일을 받아서 사용한다
- 기수별 계산기
  - 기수를 입력하면
  - 입대일부터 오늘까지 복무일수를 출력합니다.
  - 오늘부터 남은 복무기간 출력합니다.

기본과제-전역일 계산기

전역일 계산기(discharged_date.py)

- 비교 계산기
  - 기수를 두 개 입력하면
    각 기수의 입대일과 복무일수, 남은 복무기간을 출력합니다.