Ask Django

빌트인 함수, 정렬

Agenda

- 빌트인 함수 활용
  - sorted, filter, map, max, min
- **list**의 sort 멤버함수
- 임의 기준으로 정렬하기

builtin 함수 활용 #doc

sorted

정렬된 리스트를 반환

```
sorted_list = sorted(iterable, key=None, reverse=False)
```

- key : 정렬기준값을 생성할 함수를 지정. iterable 객체의 각 원소마다 **key함수**가 호출되고 그 리턴값으로 정렬을 수행

```
>>> def sort_fn(value):
        return value % 10  # 1의 자리수로서 정렬을 수행할려면

>>> sorted_list = sorted([19, 25, 32, 45], key=sort_fn, reverse=True)
>>> sorted_list
[19, 25, 45, 32]

>>> sorted([19, 25, 32, 45], key=sort_fn, reverse=False)
[32, 25, 45, 19]
```

filter

지정함수로 필터링된 결과를 생산할 Iterator를 반환

각 원소마다 지정함수가 호출되어, 리턴값이 **True** 판정될 경우 **통과**

```
iterator = filter(필터링여부를결정할함수, iterable)
```

```
>>> def judge_fn(value):
        return value % 2 == 0  # 짝수만 통과 (PASS)

>>> iterator = filter(judge_fn, [1, 2, 3, 4, 5, 6])

>>> iterator
<filter at 0x104878208>

>>> list(iterator)
[2, 4, 6]
```

map

지정함수의 리턴값을 반환할 Iterator 반환

```
iterator = map(값을변환할함수, iterable)

def power_fn(value):
    return value ** 2

>>> iterator = map(power_fn, [1, 2, 3, 4, 5])

>>> iterator
<map at 0x104877c18>

>>> list(iterator)
[1, 4, 9, 16, 25]
```

max / min

iterable에서 key함수를 거친 결과값 중에 가장 큰 결과값의 원래값을 반환

```
최대값 = max(iterable [, default=obj][, key=값을변환할함수])
최소값 = min(iterable [, default=obj][, key=값을변환할함수])
```

- iterable이 비었을 경우, **default** 값을 반환
- 디폴트값을 지정하지 않고 Iterable 객체가 비었을 경우, **ValueError: max() arg is an empty sequence** 예외가 발생

- 프로그램 수행 시에 iterable가 비었을 수도 있으니, default 값은 필히 지정해주세요.

예시

```
>>> max([], default=0)
0
>>> max([1, 2, 3], default=0)
3
>>> max([1, 2, 3, 11], key=lambda i: i%10, default=0)
3

>>> max([1, 2, 3, -11], key=abs, default=0)
-11
>>> min([1, 2, 3, -11], key=abs, default=0)
1
```

list 의 sort 멤버함수

- **sorted** : 다양한 **iterable**객체를 정렬한 새로운 리스트를 리턴
  - 원본 <iterable 객체>의 순서는 변경하지 않음.
- **list**는 자체적으로 **sort**함수를 지원
  - list 자체의 순서를 변경
  - sorted와 다르게 리턴값이 없습니다. 즉 None을 리턴합니다.

```
>>> mylist = [10, 9, 1, 2, -1]
>>> mylist.sort(key=None, reverse=False)  # 리턴값이 없습니다. (None)
>>> mylist
[-1, 1, 2, 9, 10]
```

임의 기준으로 정렬하기

- sorted, mylist.sort 함수에 **key**함수를 제공하여, 임의 기준으로 정렬

```
mylist = [5, -6, 6, -4, 9, -3, -2, -6, 4, 3]

# 절대값을 기준으로 정렬
>>> sorted(mylist, key=lambda i: abs(i))
[-2, -3, 3, -4, 4, 5, -6, 6, -6, 9]

>>> sorted(mylist, key=lambda i: abs(i), reverse=True)
[9, -6, 6, -6, 5, -4, 4, -3, 3, -2]

# 3으로 나눈 나머지 값을 기준으로 내림차순 정렬
>>> sorted(mylist, key=lambda i: i%3, reverse=True)
[5, -4, -2, 4, -6, 6, 9, -3, -6, 3]

# list의 sort함수를 호출하여, list 자체의 순서를 변경
>>> mylist.sort(key=lambda i: i%3, reverse=True)
>>> mylist
[5, -4, -2, 4, -6, 6, 9, -3, -6, 3]
```

대소비교

- 정렬을 위해서는 각 값들 간에 **대소비교**가 가능해야합니다.
  - 대소비교가 지원되지 않으면, 정렬이 불가

```
>>> 0 < 'a'
TypeError: unorderable types: int() < str()
```

- 한 문자끼리 비교는 각 문자에 매핑된 문자코드를 따라 비교

```
>>> 'a' < 'b'  # 'a'의 ascii코드 : 97, 'b'의 ascii코드 : 98
True
```

- 문자열끼리의 비교는 첫번째 인덱스부터 대소비교가 판가름이 날때까지 같은 인덱스의 문자끼리 비교.

```
>>> 'abcdef' < 'axab'  # 0번 인덱스 (무승부), 1번 인덱스 (우항이 크다) -> 비교 끝
True
```

- list/tuple 끼리의 비교도 문자열 비교와 동일

```
>>> [9] < [0]
False
>>> [0, 9] < [9]
True
```

Quiz) 다수 기준으로 정렬하기

- 다음 기준으로 정렬해보세요.
  - 1차 기준 : 자릿수 (ex - 11 은 2, 123 은 3, 1111 은 4)
  - 2차 기준 : 1의 자리 숫자 (ex - 11 은 1, 123 은 3, 1111 은 1)

```
def sort_fn(value):
    pass  # FIXME: 구현해주세요.


mylist = [10, 11, 9, 20, 12, 313, 211, 121]
mylist.sort(key=sort_fn)
print(mylist)  # [9, 10, 20, 11, 12, 211, 121, 313]
```

힌트

```
str(11)     # "11"
len("11")   # 2
```

문제를 꼭 고민해보시고, 코드를 확인해주세요. :-)

코드 확인하기