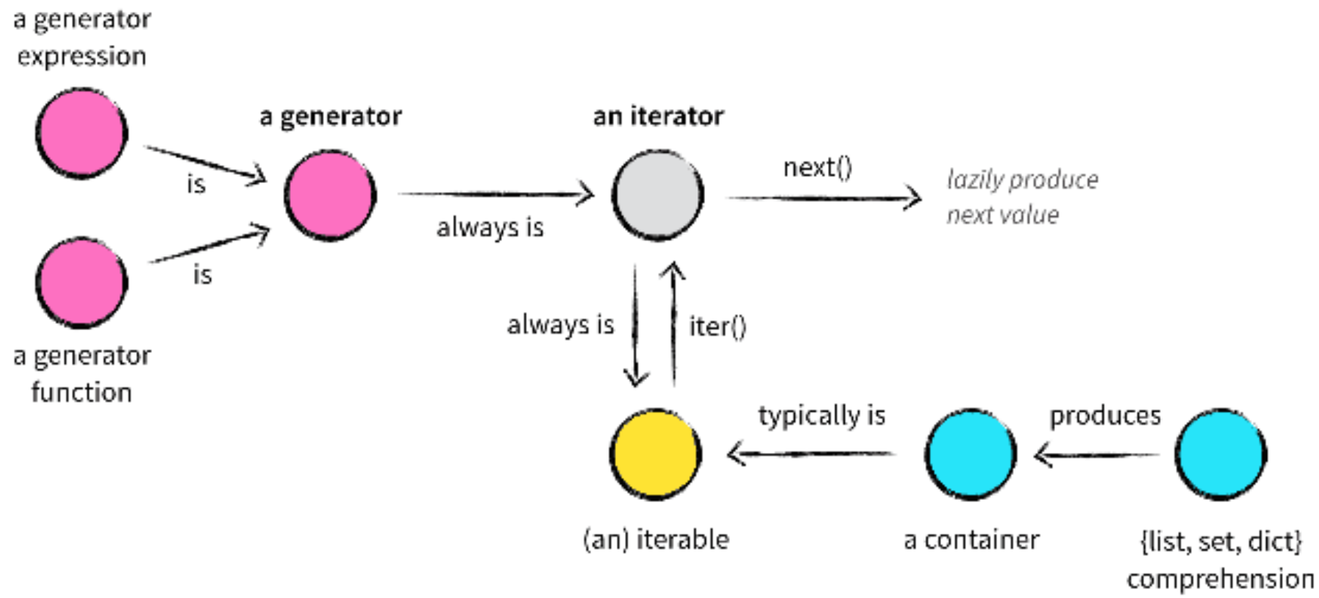
Ask Django

순회 가능한 (Iterable) 객체

참고) Iterables vs. Iterators vs. Generators #doc #번역



- iterable : 순회 가능한 객체
- Generator : 값을 생산해내는 객체
  - generator expression : 제너레이터 표현식
  - generator function : 제너레이터 함수

Generator 맛보기

```
# a generator expression
>>> (i ** 2 for i in range(10))
<generator object <genexpr> at 0x104c8d4c0>

# generator expression 으로 list 생성
>>> list(i**2 for i in range(10))
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

# a generator function
def power():
    for i in range(10):
        yield i ** 2      # 함수 내에서 return 대신에 yield. yield시마다 값을 생산

>>> power()
<generator object power at 0x10667dbf8>

>>> list(power())
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

순회 가능한 (Iterable) 객체

- set, list, dict, tuple, string, generator 는 모두 순회 가능한 객체
- Custom 클래스에 대해서도 순회가능토록 만들 수 있습니다.
  - `__iter__` 멤버함수 구현 : self가 iterator로서 동작을 하기 위해 self를 반환
  - `__next__` 멤버함수 구현 : iterator로서 동작
- `for in` 구문에서 활용 가능

```
for ch in "hello world":
    print(ch)  # 글자 별로 순회

for i in [1, 2, 3]:
    print(i)   # 숫자 별로 순회
```

사전(dict)의 경우

```
mydict = {'a': 1, 'b': 2}

for key in mydict:                      # key 별로 순회
    print(key, mydict[key])

for key in mydict.keys():               # key 별로 순회
    print(key, mydict[key])

for value in mydict.values():           # values 별로 순회
    print(value)

for (key, value) in mydict.items():     # (key, value) 쌍으로 순회
    print(key, value)
```

클래스를 통한 Iterable 객체 만들기

```
class MyRange:
    def __init__(self, start, end):
        self.start = start
        self.end = end

    def __iter__(self):
        return self             # iterator를 요구받고, 현 instance에서 next처리

    def __next__(self):
        if self.start >= self.end:
            raise StopIteration  # 남은 요소가 없을 때, StopIteration 강제 발생
        value = self.start
        self.start += 1
        return value             # 다음 요소를 return

>>> iterable = MyRange(0, 3)

>>> # object가 iterator가 아닐 경우, iterator(반복자)를 요구받음 - 인스턴스일 경우 __iter__ 호출
>>> # for 루프를 돌며, iterator에 다음 요소를 요구 - 인스턴스일 경우 __next__ 호출
>>> for i in iterable:
        print(i)

1
2
3
```