

교수의 파이썬

03_2 반복자 객체 생성

창원대학교 정보통신공학과 교수 박동규

넌넌한 교수의 파이썬

03_2 반복자 객체 생성

창원대학교 정보통신공학과 교수 박동규

넌넌한 교수의 고급 파이썬

03_2 반복자 객체 생성

창원대학교 정보통신공학과 교수 박동규

넌넌한 교수의 고급 파이썬

03_2 반복자 객체 생성

창원대학교 정보통신공학과 교수 박동규

반복자 객체 생성

반복자 객체 생성

- 반복자를 만들기 위해서는 반드시 `__iter__()` 와 `__next__()` 메소드를 그 멤버로 가져야한다.
- 이 때, `__iter__()` 메소드는 반복자 객체 자신(`self` 라는 키워드로 정의 됨)을 반환하여야 하며, `__next__()` 메소드는 루프가 돌 때마다 지정된 값을 반환하는 일을 한다.

홀수 값을 반환하는 반복자 객체

홀수 값을 반환하는 반복자 객체

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 print(next(my_counter))
16 print(my_counter.__next__())
17 print(my_counter.__next__())
18 print(my_counter.__next__())
```


홀수 값을 반환하는 반복자 객체

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t           # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 print(next(my_counter))
16 print(my_counter.__next__())
17 print(my_counter.__next__())
18 print(my_counter.__next__())
```

홀수 값을 반환하는 반복자 객체

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 print(next(my_counter))
16 print(my_counter.__next__())
17 print(my_counter.__next__())
18 print(my_counter.__next__())
```

홀수 값을 반환하는 반복자 객체

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시변수 t에 저장해 두고
11        self.n += 2       # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 print(next(my_counter))
16 print(my_counter.__next__())
17 print(my_counter.__next__())
18 print(my_counter.__next__())
```

실행 결과

1
3
5
7

Lab

1 - 20 사이의 홀수를 출력하는 경우

for 문을 이용해서 1에서 20 사이의 홀수를 출력

1 - 20 사이의 홀수를 출력하는 경우

for 문을 이용해서 1에서 20 사이의 홀수를 출력

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n      # self.n을 임시 변수 t에 저장해 두고
11        self.n += 2     # self.n을 2증가시킨다
12        return t        # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 for x in my_counter:
16     if x > 20: # 반복을 종료하는 조건 : x > 20
17         break
18     print(x, end = ' ')
```

1 - 20 사이의 홀수를 출력하는 경우

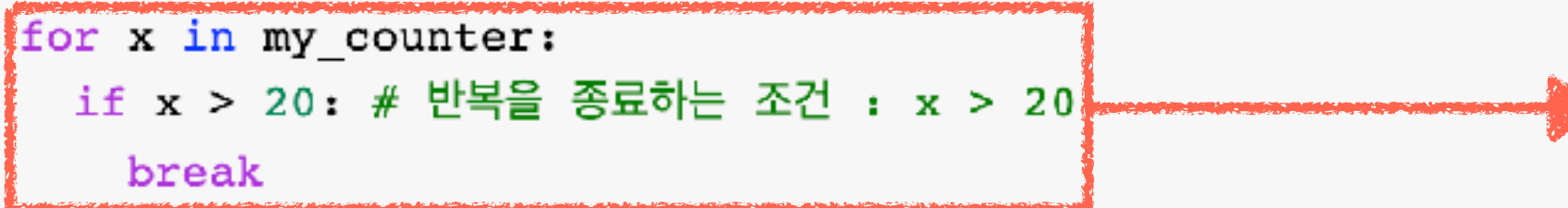
for 문을 이용해서 1에서 20 사이의 홀수를 출력

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시 변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 for x in my_counter:
16     if x > 20: # 반복을 종료하는 조건 : x > 20
17         break
18     print(x, end = ' ')
```

1 - 20 사이의 홀수를 출력하는 경우

for 문을 이용해서 1에서 20 사이의 홀수를 출력

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시 변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 for x in my_counter:
16     if x > 20: # 반복을 종료하는 조건 : x > 20
17         break
18     print(x, end = ' ')
```



1 - 20 사이의 홀수를 출력하는 경우

for 문을 이용해서 1에서 20 사이의 홀수를 출력

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시 변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 for x in my_counter:
16     if x > 20: # 반복을 종료하는 조건 : x > 20
17         break
18     print(x, end = ' ')
```

x > 20을 만족하는지 검사하는
조건 검사문을
for 문 내부에 넣어주어야 하며,
매번 루프를 돌 때마다
조건검사를 하는 번거로움 존재

1 - 20 사이의 홀수를 출력하는 경우

for 문을 이용해서 1에서 20 사이의 홀수를 출력

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시 변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 for x in my_counter:
16     if x > 20: # 반복을 종료하는 조건 : x > 20
17         break
18     print(x, end = ' ')
```

실행 결과

x > 20을 만족하는지 검사하는
조건 검사문을
for 문 내부에 넣어주어야 하며,
매번 루프를 돌 때마다
조건검사를 하는 번거로움 존재

1 - 20 사이의 홀수를 출력하는 경우

for 문을 이용해서 1에서 20 사이의 홀수를 출력

```
1 class OddCounter:
2     ''' 1부터 증가하는 홀수를 반환하는 클래스 '''
3     def __init__(self, n = 1): # 초기화 메소드 n을 1로 둔다
4         self.n = n
5
6     def __iter__(self): # 반복자는 __iter__() 함수를 가져야 함
7         return self
8
9     def __next__(self): # 반복자는 __next__() 함수를 가져야 함
10        t = self.n        # self.n을 임시 변수 t에 저장해 두고
11        self.n += 2        # self.n을 2증가시킨다
12        return t          # t부터 출력해야 1이 가장 먼저 출력된다
13
14 my_counter = OddCounter()
15 for x in my_counter:
16     if x > 20: # 반복을 종료하는 조건 : x > 20
17         break
18     print(x, end = ' ')
```

실행 결과

1 3 5 7 9 11 13 15 17 19

x > 20을 만족하는지 검사하는
조건 검사문을
for 문 내부에 넣어주어야 하며,
매번 루프를 돌 때마다
조건검사를 하는 번거로움 존재

1-20 사이의 홀수를 출력하는 경우

특정 조건을 만족하면 StopIteration을 raise

1-20 사이의 홀수를 출력하는 경우

특정 조건을 만족하면 StopIteration을 raise

```
1 class OddCounter:
2     def __init__(self, n = 1):
3         self.n = n
4
5     def __iter__(self):
6         return self
7
8     def __next__(self):
9         if self.n < 20: # 반복자가 수행되는 조건
10            t = self.n
11            self.n += 2
12            return t
13            raise StopIteration # 조건을 만족하지 않으면 StopIteration을 raise함
14
15 my_counter = OddCounter()
16 for x in my_counter:
17     print(x, end = ' ')
```

1-20 사이의 홀수를 출력하는 경우

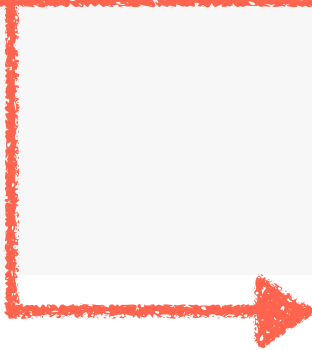
특정 조건을 만족하면 StopIteration을 raise

```
1 class OddCounter:
2     def __init__(self, n = 1):
3         self.n = n
4
5     def __iter__(self):
6         return self
7
8     def __next__(self):
9         if self.n < 20: # 반복자가 수행되는 조건
10            t = self.n
11            self.n += 2
12            return t
13        raise StopIteration # 조건을 만족하지 않으면 StopIteration을 raise함
14
15 my_counter = OddCounter()
16 for x in my_counter:
17     print(x, end = ' ')
```

1-20 사이의 홀수를 출력하는 경우

특정 조건을 만족하면 StopIteration을 raise

```
1 class OddCounter:
2     def __init__(self, n = 1):
3         self.n = n
4
5     def __iter__(self):
6         return self
7
8     def __next__(self):
9         if self.n < 20: # 반복자가 수행되는 조건
10            t = self.n
11            self.n += 2
12            return t
13            raise StopIteration # 조건을 만족하지 않으면 StopIteration을 raise함
14
15 my_counter = OddCounter()
16 for x in my_counter:
17     print(x, end = ' ')
```



1-20 사이의 홀수를 출력하는 경우

특정 조건을 만족하면 StopIteration을 raise

```
1 class OddCounter:
2     def __init__(self, n = 1):
3         self.n = n
4
5     def __iter__(self):
6         return self
7
8     def __next__(self):
9         if self.n < 20: # 반복자가 수행되는 조건
10            t = self.n
11            self.n += 2
12            return t
13            raise StopIteration # 조건을 만족하지 않으면 StopIteration을 raise함
14
15 my_counter = OddCounter()
16 for x in my_counter:
17     print(x, end = ' ')
```

클래스 외부에서 조건을 검사하지 않아도
클래스 내부의 `__next__()` 메소드에서
`self.n` 값이 20이 되는 순간
StopIteration 예외를 생성하여
for 루프가 중지된다

1-20 사이의 홀수를 출력하는 경우

특정 조건을 만족하면 StopIteration을 raise

```
1 class OddCounter:
2     def __init__(self, n = 1):
3         self.n = n
4
5     def __iter__(self):
6         return self
7
8     def __next__(self):
9         if self.n < 20: # 반복자가 수행되는 조건
10            t = self.n
11            self.n += 2
12            return t
13            raise StopIteration # 조건을 만족하지 않으면 StopIteration을 raise함
14
15 my_counter = OddCounter()
16 for x in my_counter:
17     print(x, end = ' ')
```

실행 결과

클래스 외부에서 조건을 검사하지 않아도
클래스 내부의 `__next__()` 메소드에서
`self.n` 값이 20이 되는 순간
StopIteration 예외를 생성하여
for 루프가 중지된다

1-20 사이의 홀수를 출력하는 경우

특정 조건을 만족하면 StopIteration을 raise

```
1 class OddCounter:
2     def __init__(self, n = 1):
3         self.n = n
4
5     def __iter__(self):
6         return self
7
8     def __next__(self):
9         if self.n < 20: # 반복자가 수행되는 조건
10            t = self.n
11            self.n += 2
12            return t
13            raise StopIteration # 조건을 만족하지 않으면 StopIteration을 raise함
14
15 my_counter = OddCounter()
16 for x in my_counter:
17     print(x, end = ' ')
```

실행 결과

1 3 5 7 9 11 13 15 17 19

클래스 외부에서 조건을 검사하지 않아도
클래스 내부의 __next__() 메소드에서
self.n 값이 20이 되는 순간
StopIteration 예외를 생성하여
for 루프가 중지된다

Lab

정리

- 반복자를 만들기 위해서는 반드시 `__iter__()` 와 `__next__()` 메소드를 그 멤버로 가져야하므로
- 이 클래스를 정의해 보았음
- 이 때, `__iter__()` 메소드는 반복자 객체 자신(`self` 라는 키워드로 정의 됨)을 반환함.
- `__next__()` 메소드는 루프가 돌 때마다 주어진 값을 반환

감사합니다.