5 장. 람다 표현식 사용하기

파이썬에는 한 줄로 함수를 표현하는 람다(lambda) 함수가 있습니다.

람다 함수는 구성이 단순해 간단한 연산을 하는데 종종 사용합니다.

람다 표현식은 식 형태로 되어 있다고 해서 람다 표현식(lambda expression)이라고 부릅니다.

람다 표현식은 함수를 간편하게 작성할 수 있어서 다른 함수의 인수로 넣을 때 주로 사용합니다.

이번에는 람다 표현식으로 익명 함수를 만드는 방법을 알아보겠습니다.

lambda <인자> : <인자 활용 수행 코드>

람다 함수는 <인자>를 전달하면 <인자 활용 수행 코드>를 수행한 후 결과를 바로 반환합니다.

<인자>는 콤마 (,)로 구분해 여러 개를 사용할 수 있습니다.

람다를 사용할 때는 다음처럼 람다 함수 전체를 소괄호로 감싸고 그 다음에 별도의 소괄호 안에 인자를 사용합니다.

(lambda <인자> : <인자 활용 수행 코드>) (<인자>)

이처럼 사용하는 것이 기본적인 방법이지만 보통은 사용의 편리성을 위해 다음과 같이 람다 함수를 다른 변수에 할당하고 이 변수를 함수명 처럼 이용해 람다 함수를 호출합니다.

이때 정의한 <인자>도 함께 입력합니다.

람다 함수를 변수에 할당할 때는 람다 함수 전체를 소괄호로 감싸지 않아도 됩니다.

lambda_function = lambda <인자> : <인자 활용 수행 코드>

lambda_function(<인자>)

5.1 람다 표현식으로 함수 만들기

람다 표현식을 사용하기 전에 먼저 숫자를 받은 뒤 10 을 더해서 반환하는 함수 plus_ten 을 만들어보겠습니다.

```
>>> def plus_ten(x):
...     return x + 10
...
>>> plus_ten(1)
11
```

return x + 10 으로 매개변수 x 에 10 을 더한 값을 반환하는 간단한 함수입니다.

그럼 이 plus_ten 함수를 람다 표현식으로 작성해보겠습니다.

다음과 같이 lambda 에 매개변수를 지정하고 :(콜론) 뒤에 반환값으로 사용할 식을 지정합니다.

- **lambda 매개변수들: 식**

```
>>> lambda x: x + 10
<function <lambda> at 0x02C27270>
```

실행을 해보면 함수 객체가 나오는데, 이 상태로는 함수를 호출할 수 없습니다. 왜냐하면 람다 표현식은 이름이 없는 함수를 만들기 때문입니다. 그래서 람다 표현식을 익명 함수(anonymous function)로 부르기도 합니다.

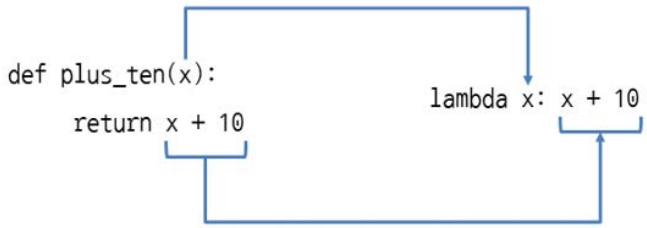
lambda 로 만든 익명 함수를 호출하려면 다음과 같이 람다 표현식을 변수에 할당해주면 됩니다.

```
>>> plus_ten = lambda x: x + 10
>>> plus_ten(1)
11
```

이제 람다 표현식을 살펴보면 lambda x: x + 10 은 매개변수 x 하나를 받고, x 에 10 을 더해서 반환한다는 뜻입니다. 즉, 매개변수, 연산자, 값 등을 조합한 식으로 반환값을 만드는 방식입니다.

다음 그림과 같이 def 로 만든 함수와 비교해보면 쉽게 알 수 있습니다.

▼ **그림 5-1** def로 만든 함수와 람다 표현식



5.1.1 람다 표현식 자체를 호출하기

람다 표현식은 변수에 할당하지 않고 람다 표현식 자체를 바로 호출할 수 있습니다. 다음과 같이 람다 표현식을 ( )(괄호)로 묶은 뒤에 다시 ( )를 붙이고 인수를 넣어서 호출하면 됩니다.

- **(lambda 매개변수들: 식)(인수들)**

```
>>> (lambda x: x + 10)(1)
11
```

5.1.2 람다 표현식 안에서는 변수를 만들 수 없다

람다 표현식에서 주의할 점은 람다 표현식 안에서는 새 변수를 만들 수 없다는 점입니다. 따라서 반환값 부분은 변수 없이 식 한 줄로 표현할 수 있어야 합니다. 변수가 필요한 코드일 경우에는 def로 함수를 작성하는 것이 좋습니다.

```
>>> (lambda x: y = 10; x + y)(1)
SyntaxError: invalid syntax
```

단, 람다 표현식 바깥에 있는 변수는 사용할 수 있습니다. 다음은 매개변수 x와 람다 표현식 바깥에 있는 변수 y를 더해서 반환합니다.

```
>>> y = 10
>>> (lambda x: x + y)(1)
11
```

5.1.3 람다 표현식을 인수로 사용하기

람다 표현식을 사용하는 이유는 함수의 인수 부분에서 간단하게 함수를 만들기 위해서 입니다. 이런 방식으로 사용하는 대표적인 예가 map 입니다.

람다 표현식을 사용하기 전에 먼저 def 로 함수를 만들어서 map 을 사용해보겠습니다. 다음과 같이 숫자를 받은 뒤 10 을 더해서 반환하는 함수 plus_ten 을 작성합니다. 그리고 map 에 plus_ten 함수와 리스트 [1, 2, 3]을 넣습니다. 물론 map 의 결과는 map 객체이므로 눈으로 확인할 수 있도록 list 를 사용해서 리스트로 변환해줍니다.

```
>>> def plus_ten(x):
...     return x + 10
>>> list(map(plus_ten, [1, 2, 3]))
[11, 12, 13]
```

plus_ten 함수는 매개변수 x 에 10 을 더해서 반환하므로 리스트 [1, 2, 3]이 [11, 12, 13]으로 바뀌었습니다. 지금까지 map 을 사용할 때 map(str, [1, 2, 3])와 같이 자료형 int, float, str 등을 넣었죠? 사실 plus_ten 처럼 함수를 직접 만들어서 넣어도 됩니다.

이제 람다 표현식으로 함수를 만들어서 map 에 넣어보겠습니다.

```
>>> list(map(lambda x: x + 10, [1, 2, 3]))
[11, 12, 13]
```

plus_ten 함수 대신 람다 표현식 lambda x: x + 10 을 넣었습니다. 전체적으로 보면 코드가 세 줄에서 한 줄로 줄었죠? 이처럼 람다 표현식은 함수를 다른 함수의 인수로 넣을 때 매우 편리합니다.

참고 | 람다 표현식으로 매개변수가 없는 함수 만들기
람다 표현식으로 매개변수가 없는 함수를 만들 때는 lambda 뒤에 아무것도 지정하지 않고 : (콜론)을 붙입니다. 단, 콜론 뒤에는 반드시 반환할 값이 있어야 합니다. 왜냐하면 표현식(expression)은 반드시 값으로 평가되어야 하기 때문입니다.

```
>>> (lambda : 1)()
1
>>> x = 10
>>> (lambda : x)()
10
```

5.2 람다 표현식과 map, filter, reduce 함수 활용하기

람다 표현식 작성 방법을 알아보았으니 이번에는 람다 표현식과 map, filter, reduce 함수를 함께 사용해보겠습니다.

5.2.1 람다 표현식에 조건부 표현식 사용하기

먼저 람다 표현식에서 조건부 표현식을 사용하는 방법을 알아보겠습니다.

- **lambda 매개변수들: 식 1 if 조건식 else 식 2**

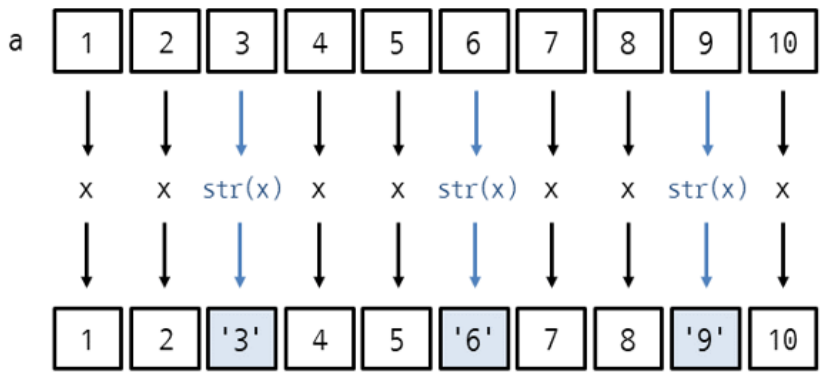
다음은 map 을 사용하여 리스트 a 에서 3 의 배수를 문자열로 변환합니다.

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(map(lambda x: str(x) if x % 3 == 0 else x, a))
[1, 2, '3', 4, 5, '6', 7, 8, '9', 10]
```

map 은 리스트의 요소를 각각 처리하므로 lambda 의 반환값도 요소라야 합니다. 여기서는 요소가 3 의 배수일 때는 str(x)로 요소를 문자열로 만들어서 반환했고, 3 의 배수가 아닐 때는 x 로 요소를 그대로 반환했습니다.

▼ 그림 5-2 map에 람다 표현식 사용하기

```
list(map(lambda x: str(x) if x % 3 == 0 else x, a))
```



람다 표현식 안에서 조건부 표현식 if, else 를 사용할 때는 :(콜론)을 붙이지 않습니다. 일반적인 if, else 와 문법이 다르므로 주의해야 합니다. 조건부 표현식은 **식 1 if 조건식 else 식 2** 형식으로 사용하며 **식 1** 은 조건식이 참일 때, **식 2** 는 조건식이 거짓일 때 사용할 식입니다.

특히 람다 표현식에서 if 를 사용했다면 반드시 else 를 사용해야 합니다. 다음과 같이 if 만 사용하면 문법 에러가 발생하므로 주의해야 합니다.

```
>>> list(map(lambda x: str(x) if x % 3 == 0, a))
SyntaxError: invalid syntax
```

그리고 람다 표현식 안에서는 elif 를 사용할 수 없습니다. 따라서 조건부 표현식은 **식 1 if 조건식 1 else 식 2 if 조건식 2 else 식 3** 형식처럼 if 를 연속으로 사용해야 합니다. 예를 들어 리스트에서 1 은 문자열로 변환하고, 2 는 실수로 변환, 3 이상은 10 을 더하는 식은 다음과 같이 만듭니다.

- **lambda 매개변수들: 식 1 if 조건식 1 else 식 2 if 조건식 2 else 식 3**

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(map(lambda x: str(x) if x == 1 else float(x) if x == 2 else x + 10, a))
['1', 2.0, 13, 14, 15, 16, 17, 18, 19, 20]
```

별로 복잡하지 않은 조건인데도 알아보기가 힘듭니다. 이런 경우에는 억지로 람다 표현식을 사용하기 보다는 그냥 def 로 함수를 만들고 if, elif, else 를 사용하는 것을 권장합니다.

```
>>> def f(x):
...     if x == 1:
...         return str(x)
...     elif x == 2:
...         return float(x)
...     else:
...         return x + 10
...
>>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(map(f, a))
['1', 2.0, 13, 14, 15, 16, 17, 18, 19, 20]
```

복잡하고 어렵게 코드를 작성하면 나중에 시간이 지나서 자기가 만든 코드인데도 못 알아보는 경우가 생깁니다. 코드는 길이가 조금 길어지더라도 알아보기 쉽게 작성하는 것이 좋습니다.

5.2.2 map 에 객체를 여러 개 넣기

map 은 리스트 등의 반복 가능한 객체를 여러 개 넣을 수도 있습니다. 다음은 두 리스트의 요소를 곱해서 새 리스트를 만듭니다.

```
>>> a = [1, 2, 3, 4, 5]

>>> b = [2, 4, 6, 8, 10]

>>> list(map(lambda x, y: x * y, a, b))

[2, 8, 18, 5, 50]
```

이렇게 리스트 두 개를 처리할 때는 람다 표현식에서 lambda x, y: x * y 처럼 매개변수를 두 개로 지정하면 됩니다. 그리고 map 에 람다 표현식을 넣고 그다음에 리스트 두 개를 콤마로 구분해서 넣어줍니다. 즉, 람다 표현식의 매개변수 개수에 맞게 반복 가능한 객체도 콤마로 구분해서 넣어주면 됩니다.

5.2.3 filter 사용하기

이번에는 filter 를 사용해보겠습니다. filter 는 반복 가능한 객체에서 특정 조건에 맞는 요소만 가져오는데, filter 에 지정한 함수의 반환값이 True 일 때만 해당 요소를 가져옵니다.

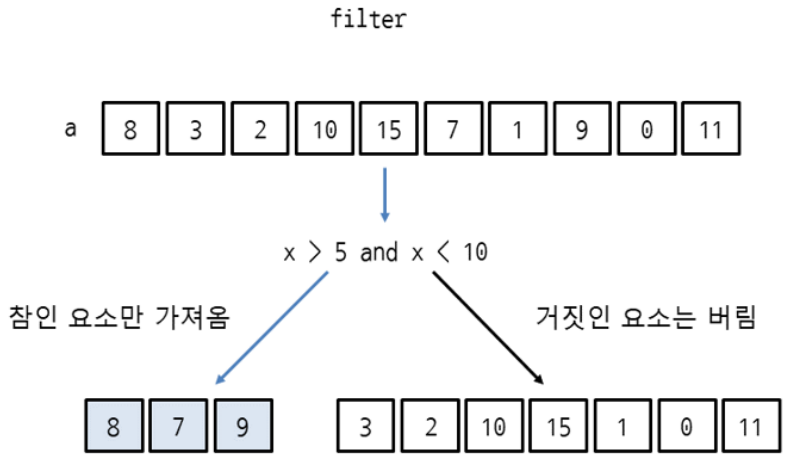
- **filter(함수, 반복가능한객체)**

먼저 def 로 함수를 만들어서 filter 를 사용해보겠습니다. 다음은 리스트에서 5 보다 크면서 10 보다 작은 숫자를 가져옵니다.

```
>>> def f(x):

...     return x > 5 and x < 10

...

>>> a = [8, 3, 2, 10, 15, 7, 1, 9, 0, 11]

>>> list(filter(f, a))

[8, 7, 9]
```

리스트 a 에서 8, 7, 9 를 가져왔습니다. 즉, filter 는 x > 5 and x < 10 의 결과가 참인 요소만 가져오고 거짓인 요소는 버립니다.

▼ 그림 5-3 filter 함수



그럼 함수 f 를 람다 표현식으로 만들어서 filter 에 넣어보겠습니다.

```
>>> a = [8, 3, 2, 10, 15, 7, 1, 9, 0, 11]
>>> list(filter(lambda x: x > 5 and x < 10, a))
[8, 7, 9]
```

람다 표현식 lambda x: x > 5 and x < 10 을 filter 에 넣어서 5 보다 크면서 10 보다 작은 수를 가져오도록 만들었습니다.

5.2.4 reduce 사용하기

마지막으로 reduce 를 사용해보겠습니다. reduce 는 반복 가능한 객체의 각 요소를 지정된 함수로 처리한 뒤 이전 결과와 누적해서 반환하는 함수입니다.(reduce 는 파이썬 3 부터 내장 함수가 아닙니다. 따라서 functools 모듈에서 reduce 함수를 가져와야 합니다)

- **from functools import reduce**
- **reduce(함수, 반복가능한객체)**

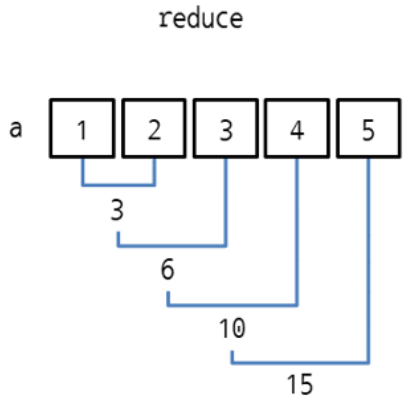
다음은 리스트에 저장된 요소를 순서대로 더한 뒤 누적된 결과를 반환합니다.

```
>>> def f(x, y):
...     return x + y
...
```

```
>>> a = [1, 2, 3, 4, 5]
>>> from functools import reduce
>>> reduce(f, a)
15
```

reduce 의 반환값이 15 가 나왔습니다. 함수 f 에서 x + y 를 반환하도록 만들었으므로 reduce 는 그림과 같이 요소 두 개를 계속 더하면서 결과를 누적합니다.

▼ **그림 5-4** reduce 함수



이제 함수 f 를 람다 표현식으로 만들어서 reduce 에 넣어보겠습니다.

```
>>> a = [1, 2, 3, 4, 5]
>>> from functools import reduce
>>> reduce(lambda x, y: x + y, a)
15
```

lambda x, y: x + y 와 같이 매개변수 x, y 를 지정한 뒤 x 와 y 를 더한 결과를 반환하도록 만들었습니다.

지금까지 람다 표현식으로 익명 함수를 만들고, map, filter, reduce 와 응용하는 방법을 알아보았습니다. 람다 표현식은 간단하게 함수를 만들 때 자주 사용합니다. 그러므로 람다 표현식은 꼭 익혀 두는 것이 좋습니다.

참고 | map, filter, reduce와 리스트 표현식

리스트(딕셔너리, 세트) 표현식으로 처리할 수 있는 경우에는 map, filter 와 람다 표현식 대신 리스트 표현식을 사용하는 것이 좋습니다. list(filter(lambda x: x > 5 and x < 10, a))는 다음과 같이 리스트 표현식으로도 만들 수 있습니다.

```
>>> a = [8, 3, 2, 10, 15, 7, 1, 9, 0, 11]
>>> [i for i in a if i > 5 and i < 10]
[8, 7, 9]
```

리스트 표현식이 좀 더 알아보기 쉽고 속도도 더 빠릅니다.

또한, for, while 반복문으로 처리할 수 있는 경우에도 reduce 대신 for, while 을 사용하는 것이 좋습니다. 왜냐하면 reduce 는 코드가 조금만 복잡해져도 의미하는 바를 한 눈에 알아보기가 힘들기 때문입니다. 이러한 이유로 파이썬 3 부터는 reduce 가 내장 함수에서 제외되었습니다.

reduce(lambda x, y: x + y, a)는 다음과 같이 for 반복문으로 표현할 수 있습니다.

```
>>> a = [1, 2, 3, 4, 5]
>>> x = a[0]
>>> for i in range(len(a) - 1):
...     x = x + a[i + 1]
...
>>> x
15
```

5.4 연습문제: 이미지 파일만 가져오기

다음 소스 코드를 완성하여 확장자가 .jpg, .png 인 이미지 파일만 출력되게 만드세요. 여기서는 람다 표현식을 사용해야 하며 출력 결과는 리스트 형태라야 합니다. 람다 표현식에서 확장자를 검사할 때는 문자열 메서드를 활용하세요.

practice_lambda.py

```
files = ['font', '1.png', '10.jpg', '11.gif', '2.jpg', '3.png', 'table.xslx', 'spec.docx']


print(                                                    )
```

실행 결과

```
['1.png', '10.jpg', '2.jpg', '3.png']
```

정답

```
list(filter(lambda x: x.find('.jpg') != -1 or x.find('.png') != -1, files))
```

해설

리스트 files 에는 여러 종류의 파일 이름이 들어있는데 .jpg, .png 파일만 가져오려면 filter 를 사용해야 합니다. 그리고 filter 는 lambda 또는 함수의 반환값이 True 일 때 해당 요소를 가져옵니다. 따라서 람다 표현식을 작성할 때 매개변수가 '.jpg', '.png'이면 True 를 반환하도록 만듭니다.

먼저 문자열에서 find 메서드를 사용하면 찾을 문자열이 있을 때 인덱스를 반환하고 없을 때는 -1 을 반환합니다. 그래서 조건식은 x.find('.jpg') != -1 or x.find('.png') != -1 과 같이 만들고 .jpg, .png 둘 중 하나라도 참이면 되므로 find 를 or 연산자로 연결해주면 됩니다.