

# 06

## 람다 표현식

# 01

람다 표현식

## ■ 함수

- def로 함수를 정의

## ■ 람다 표현식

- 람다 표현식으로 익명 함수 생성 가능
  - 식 형태로 되어 있다고 해서 람다 표현식(Lambda expression) 이라고 부름
  - 함수를 간단하게 작성할 수 있어 다른 함수의 인수로 넣을 때 주로 사용
  - 단순히 람다식 만으로는 호출할 수 없음 : 익명함수(이름이 없기 때문)
    - 람다표현식으로 변수에 할당해 주면 됨

## ■ 람다 표현식으로 함수 만들기

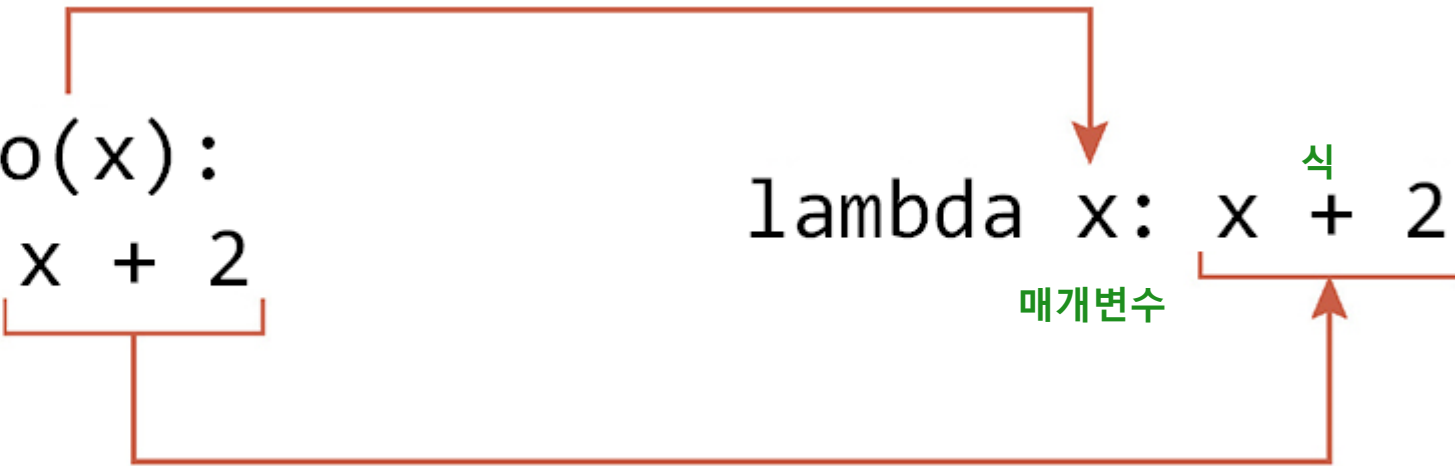
lambda 매개변수들 : 식

```
def plus_two(x):  
    return x + 2
```

```
lambda x: x + 2
```

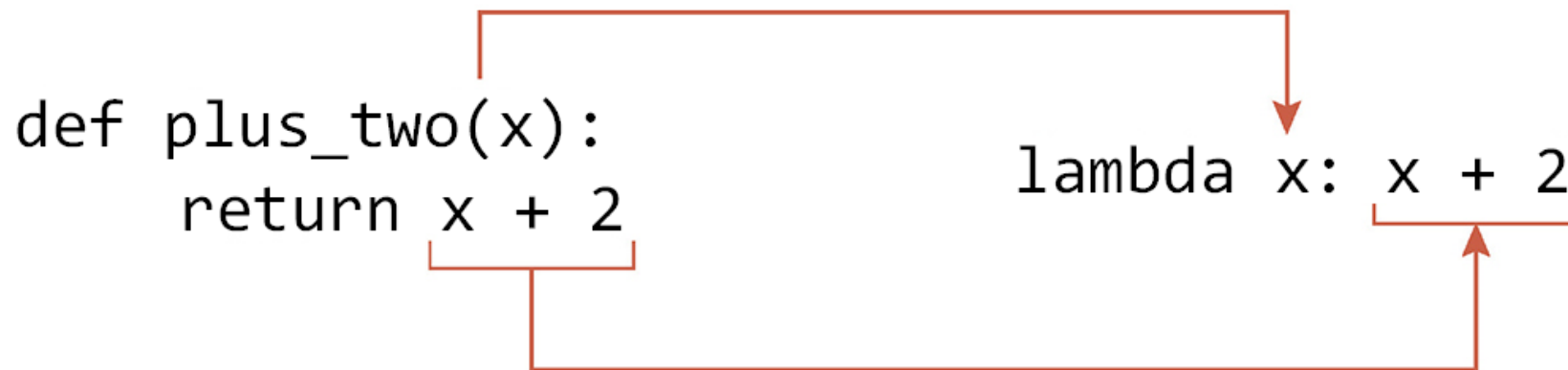
매개변수

식



(lambda 매개변수들 : 식)(인수들)

```
def plus_two(x):  
    return x + 2  
  
num = 3  
print(plus_two(num))
```



```
num = 3  
print((lambda x: x + 2)(num))
```

## ■ 람다표현식을 인수로 사용하기

- 람다 표현식을 사용하는 이유는 간단하게 함수를 만들기 위함

```
def plus_two(x):  
    return x + 2
```

리스트 직접입력

```
print(list(map(plus_two, [1,2,3])))
```

[3,4,5]

```
def plus_two(x):  
    return x + 2
```

lambda x: x + 2



```
print(list(map(lambda x: x + 2, [1,2,3])))
```

[3,4,5]

## ■ 람다표현식을 인수로 사용하기

- 람다 표현식을 사용하는 이유는 간단하게 함수를 만들기 위함

```
def plus_two(x):  
    return x + 2
```

변수로 리스트입력

```
mylist = [1,2,3]  
print(list(map(plus_two, mylist)))
```

[3,4,5]

```
def plus_two(x):  
    return x + 2
```

lambda x: x + 2



```
mylist = [1,2,3]  
print(list(map(lambda x: x + 2, mylist)))
```

[3,4,5]

## ■ 람다표현식을 인수로 사용하기

- 람다 표현식을 사용하는 이유는 간단하게 함수를 만들기 위함

```
def plus_two(x):  
    return x + 2
```

변수로 리스트입력

```
mylist = [1,2,3]  
print(list(map(plus_two, mylist)))
```

[3,4,5]

```
def plus_two(x):  
    return x + 2
```

lambda x: x + 2



```
mylist = [1,2,3]  
print(list(map(lambda x: x + 2, mylist)))
```

[3,4,5]



## ■ 람다표현식을 사용하여 매개변수 없는 함수 만들기

```
def plus_two():  
    return 2  
  
print(plus_two())
```

2

```
def plus_two(x):  
    return x + 2
```

lambda x: x + 2

```
print((lambda : 2)())
```

2

## ■ 람다표현식을 사용하여 매개변수 없는 함수 만들기

```
def plus_two():  
    return x  
  
x = 2  
print(plus_two())
```

2

```
def plus_two(x):  
    return x + 2
```

```
lambda x: x + 2
```

```
x = 2  
print((lambda : x)())
```

2

## lambda 매개변수들 : 식1 **if** 조건식 **else** 식2

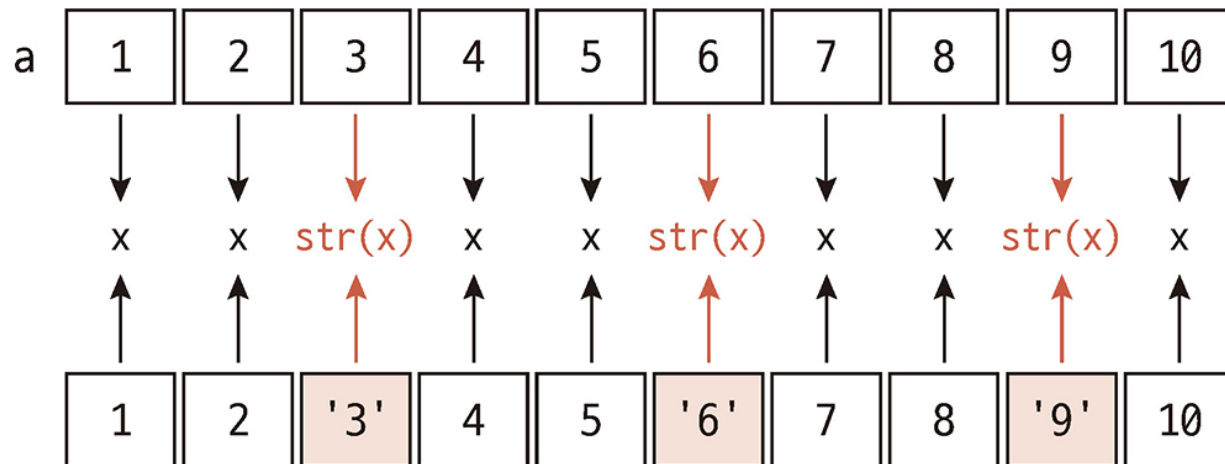
- 1~10까지의 리스트를 생성하고 리스트 가운데 3으로 나눈 나머지가 0 인 값을 문자로 만들기

```
a = list(range(1,11))
```

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

```
[1, 2, '3', 4, 5, '6', 7, 8, '9', 10]
```

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



```
def f(x):  
    if x % 3 == 0:  
        return str(x)  
    else:  
        return x  
  
a = list(range(1,11))  
print(list(map(f, a)))
```

```
[1, 2, '3', 4, 5, '6', 7, 8, '9', 10]
```

# 람다 표현식과 map, filter, reduce 함수 활용

## ■ 람다식에서 if문 사용시 주의 사항

- If를 사용시 반드시 else를 사용할 것
- If ~ elif ~ else문의 경우 함수를 만들어 사용 하는것이 더 나음

lambda 매개변수들 : 식1 if 조건식1 else 식2 if 조건식2 else 식3

```
a = list(range(1,11))
```

```
print(list(map(lambda x: str(x) if x == 1 else float(x) if x == 2 else x + 1, a)))
```

```
['1', 2.0, 4, 5, 6, 7, 8, 9, 10, 11]
```

lambda 매개변수들 : 식1 if 조건식1 else 식2 if 조건식2 else 식3

```
def f(x):  
    if x == 1:  
        return str(x)  
    elif x==2:  
        return float(x)  
    else:  
        return x + 1  
  
a = list(range(1, 11))  
print(list(map(f, a)))
```

```
['1', 2.0, 4, 5, 6, 7, 8, 9, 10, 11]
```

## ■ Filter는 반복 가능한 객체에서 특정 조건에 맞는 요소만 가져옴

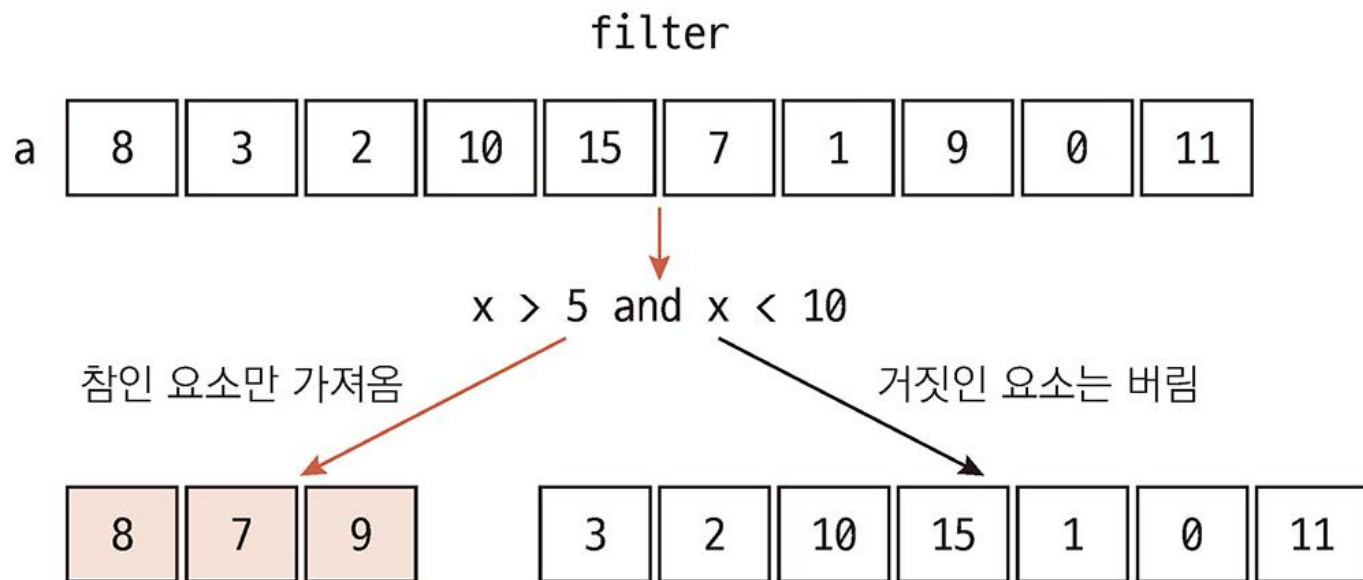
- Filter에 지정한 함수의 반환값이 True일때만 해당요소를 가져옴

### filter(함수, 반복가능 객체)

```
def f(x):  
    return 5 < x < 10  
  
a = [8, 3, 2, 10, 15, 7, 1, 9, 0, 11]  
  
print(list(filter(f, a)))
```

```
[8, 7, 9]
```

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



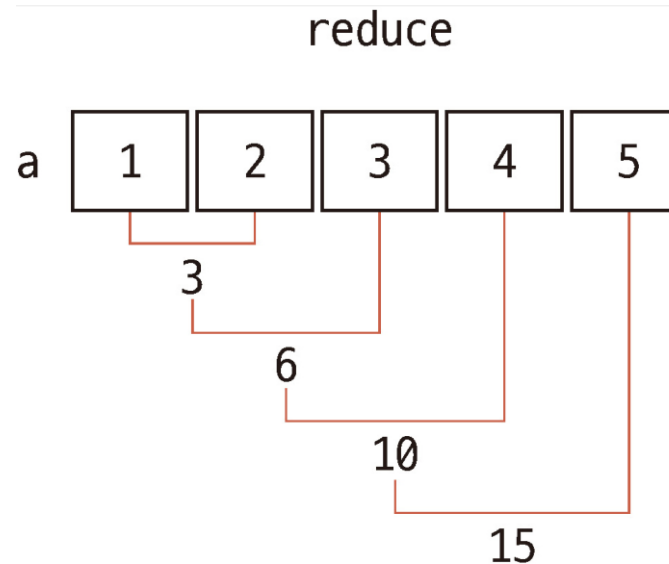
## ■ reduce

- 반복가능한 객체의 각 요소를 지정된 함수로 처리한 뒤 이전결과와 누적해서 반환

### reduce(함수, 반복가능 객체)

```
def f(x,y):  
    return x+y  
  
from functools import reduce  
  
a = list(range(1,6))  
print(reduce(f, a))
```

15



```
from functools import reduce  
  
a = list(range(1,6))  
print(reduce(lambda x,y: x+y, a))
```



다음의 files 리스트에서 확장자가 .jpg, .png인 이미지 파일만 출력하라

- 람다 표현식을 사용하시오

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

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

## ANSWER

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

## 표준입력으로 숫자.확장자 형식으로 된 파일 이름을 여러 개 입력받고 리스트 형태로 출력하는 프로그램 작성

- 다음을 만족하여야 함
- 1. 파일 이름을 입력 받고 숫자는 3자리(100보다 작을 때는 앞에 0을 붙임)형태로 출력
- 2. 결과는 리스트 형태로 출력

1.jpg 2.jpg 15.jpg 99.jpg 101.jpg

['001.jpg', '002.jpg', '015.jpg', '099.jpg', '101.jpg']

```
def f(x):  
    if len(x) == 5:  
        return '00' + x  
    elif len(x) == 6:  
        return '0' + x  
    else:  
        return x  
  
files = input().split(" ")  
print(list(map(f, files)))
```