5장. 함수(메서드)

내장 함수(Built in Function)

❖ 내장 함수(Built in Function)

특정한 기능을 수행하는 프로그램의 일부분을 함수(Function)라 한다.

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

내장 함수(Built in Function)

함수	설명	사용 예
all(x)	반복가능한(iterable) 자료형 x가 모두 참이면 True, 거짓이 하나라도 있으면 Fasle	all([1,2,3]) True all([1,2,0]) False
any(x)	X가 하나라도 참이 있으면 True, x가 모두 거짓이면 Fasle	any([1,2,0]) True
eval(expression)	문자열 표현식을 숫자로 변환	eval('1+2') 3
list(s)	반복가능한 문자열을 입력받아 리스트로 반환	list("python") ['p', 'y', 't', 'h', 'o', 'n']
round(n, digit)	숫자를 입력받아 반올림하여 돌려줌	round(4.6) 5 round(4.4) 4
sum(iterable)	리스트나 튜플의 모든 요소의 합을 반환	sum([1, 2, 3]) 6 sum((1.2.3)) 6

사용자 정의 함수

❖ 함수의 정의와 사용(호출)

def 키워드를 사용한다.

def 함수 이름():
    함수의 내용

❖ 함수의 형태

1. return 반환값이 없고, 매개변수도 없는 경우

```
def hello():
    print("Hello Python!!")

hello()
hello()
```

함수의 정의

함수의 호출

함수 정의하고 호출하기

2. return이 없고 전달인자가 있는 함수

def 함수 이름(매개 변수):
    함수의 내용

```
def hello2(name):
    print('Hello', name)

hello2('obama')
hello2('황의조')
hello2('이순신')
```

인자는 변수이다.

함수 정의하고 호출하기

3. return값(반환값)이 있는 함수

def 함수 이름(매개 변수):
    함수의 내용
    return 반환값

```
def add(x, y):
    plus = x + y
    return plus

def sub(x, y):
    minus = x - y
    return minus

n1 = add(10, 20)
n2 = sub(10, 20)
print("두 수의 합 = {0}".format(n1))
print("두 수의 차 = {0}".format(n2))
```

구구단을 출력하는 함수

- 구구단을 출력하는 함수

```
def print_gugudan(dan):
    for i in range(1, 10):
        print(dan, 'x', i, '=', (dan * i))

print_gugudan(5)
```

도형의 면적 계산

- 도형의 면적을 계산하는 함수 정의와 사용

```
def square(w, h):
    area = w * h
    return area

def triangle(n, h):
    area = n * h / 2
    return area

print('사각형의 면적 : ', square(5, 4))
print('삼각형의 면적 : ', triangle(4, 7))
```

배송비 계산하기

주문 상품 가격이 20,000원 미만이면 배송비 (2,500원) 포함하고, 아니면 배송비를 포함하지 않는 프로그램을 작성하세요

☞ 실행 결과

```
상품1 가격은 30000원입니다.
상품2 가격은 17,500원입니다.
```

```
price1 = price(15000, 2)   #price 함수 호출
price2 = price(5000, 3)
print("상품1 가격은 " + str(price1) + "원입니다.")
print("상품2 가격은 " + format(price2, ',') + "원입니다.")
```

배송비 계산하기

주문 상품 가격이 20,000원 미만이면 배송비 (2,500원) 포함하고, 아니면 배송비를 포함하지 않는 프로그램을 작성하세요

```
def price(unit_price, quantity):
    delivery_fee = 2500              # 배송비
    price = unit_price * quantity  #상품가격 = 단위당 가격 * 수량
    if price < 20000:
        price += delivery_fee
        return price
    else:
        return price
```

함수 연습 문제

두 수를 매개변수 전달하여 서로 같으면 곱하고, 서로 다르면 더하는 함수를 정의하고 호출하는 프로그램을 작성하세요

```
def data(x, y):
    if x == y:
        return x * y
    else:
        return x + y

n1 = data(5, 5)
n2 = data(10, 5)

print("n1 = %d" % n1)
print("n2 = %d" % n2)
```

매개변수로 배열 전달 1

- 배열을 매개변수로 전달하여 평균 계산하기

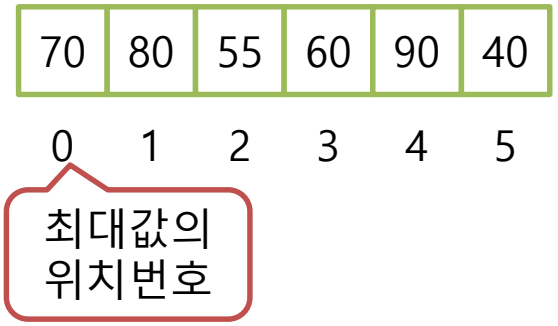
```
# 리스트가 매개변수인 함수 정의
def calc_age(a):
    sum_v = 0
    for i in a:
        sum_v += i
    avg = sum_v / len(a)
    return avg

v = [70, 90, 50, 80, 100]
average = calc_age(v)

print("평균 :", average)
```

매개변수로 배열 전달 2

- 최대값과 최대값의 위치 구하기



1. 첫번째 숫자 70을 최대값으로 기억한다.
2. 두 번째 숫자 80을 최대값 70과 비교하여 최대값은 80이 된다.
3. 계속 다음 숫자와 비교과정을 반복하여 최대값을 결정한다.

매개변수로 배열 전달 2

- 최대값과 최대값의 위치 구하기

```
# 최대값 구하기
def find_max(a):
    max = a[0]
    for i in a:
        if max < i:
            max = i
    return max

# 최대값의 위치 구하기
def find_max_idx(a):
    max_idx = 0
    n = len(a)
    for i in range(1, n):
        if a[max_idx] < a[i]:
            max_idx = i
    return max_idx

v = [70, 80, 55, 60, 90, 40]
print(find_max(v))
print(find_max_idx(v))
```

매개변수로 배열 전달 3

- 리스트를 매개변수로 새로운 리스트 만들기

```
def times(a):
    li = []
    for i in a:
        li.append(i * 4)
    return li

v = [ 1, 2, 3, 4]
print(times(v))
```

변수의 유효 범위 – 지역변수

- **지역 변수(local variable)의 유효 범위**

  지역변수는 함수나 명령문(조건, 반복)의 블록 안에서 생성되며 블록{ }을 벗어나면 메모리에서 소멸한다.

```
def one_up():
    x = 1
    x += 1
    return x

print(one_up())
print(one_up())
print(one_up())
# x값 출력
print(x)
```

```
2
2
2
Traceback (most recent call last):
  File "C:\pyworks\ch05\one_up.py",
    print(x)
NameError: name 'x' is not defined
```

변수의 유효 범위 – 전역변수

- **전역 변수의 유효 범위**

  전역 변수는 메인 함수의 위쪽에서 선언하여 사용하고 영향 범위가 전체로 미친다.

  프로그램이 종료되면 메모리에서 소멸한다.

```
# 전역변수의 범위
def price():
    price = 250 * quantity
    print("{}원입니다.".format(price))

quantity = 2;
print("{0}개에".format(quantity))
price()
```

변수의 유효 범위 – 정적변수

- 전역 변수의 유효 범위 – global 키워드 사용

```
def one_up():
    global x
    x += 1
    return x

x = 1
print(one_up())
print(one_up())
print(one_up())
# x값 출력
print(x)
```

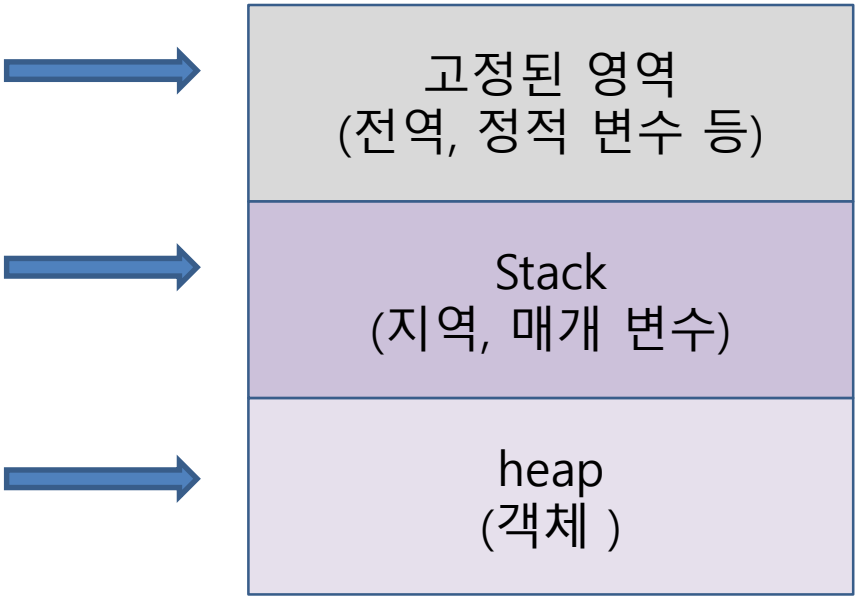
```
2
3
4
4
```

변수의 메모리 영역

데이터 영역 : **전역 변수**가 저장되는 영역

스택 영역 : **매개 변수 및 중괄호(블록) 내부에 정의된 변수**들이 저장되는 영역

힙 영역 : **동적으로 메모리를 할당하는 변수**들이 저장되는 영역 (객체 - 인스턴스 변수)



변수의 유효범위 연습 문제

1~100까지의 자연수 중 배수와 배수의 개수를 계산하는 함수를 정의하시오.

```
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51
54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
배수의 개수 : 33
```

```python
def times(x):
    global count
    for i in range(1, 101):
        if i % x == 0:
            count += 1
            print(i, end=' ')

count = 0
times(3)
print("\n배수의 개수 : %d" % count)
```

함수의 기본 매개변수

- **기본 매개변수**

  매개변수를 초기화하여 선언하고 함수 호출시 매개변수를 생략하면 기본 값으로 출력된다.

def 함수 이름**(변수, 변수=0):**
    코드블럭

```
# 기본매개 변수
def print_string(text, count=1):
    for i in range(count):
        print(text)

print_string("Hello")
print_string("Hello", 3)
```

함수의 가변 매개변수

- **가변 매개변수**

  매개변수의 입력값이 정해지지 않고 변경해야 할때 사용하는 변수이다.

  변수이름 앞에 *를 붙인다.

def 함수 이름**(*변수):**

    코드블럭

```
def calc_avg(*numbers):
    sum_v = 0
    avg = 0.0
    for i in numbers:
        sum_v += i
    avg = sum_v / len(numbers)
    return avg

avg1 = calc_avg(1, 2)
avg2 = calc_avg(1, 2, 3, 4)
print(avg1)
print(avg2)
```

함수의 가변 매개변수

- 가변 매개변수

```
def merge_text(*text):
    result = ""
    for s in text:
        result += s
    return result

str1 = merge_text("코스모스", "민들레")
str2 = merge_text("코스모스", "민들레", "국화")
print(str1)
print(str2)
```

함수의 매개변수

- **키워드 매개변수**

  매개변수 앞에 별 2개(**)를 붙인다.

  딕셔너리(dictionary)형의 자료로 만들어짐

```
def print_kw(**kwargs):
    print(kwargs)

print_kw(name="지수")
print_kw(name="지수", age=9)
```

재귀 함수(recursive function)

- **재귀 함수(recursive function)**

  어떤 함수 안에서 자기 자신을 부르는 것을 말한다.

  재귀호출은 무한 반복하므로 **종료 조건**이 필요함

```
def func(입력 값):
    if 입력값이 충분히 작으면: #종료 조건
        return 결과값
    else:      # 더 작은 값으로 호출
        return 결과값
```

재귀 함수(recursive function)

- 재귀 함수(recursive function)

```
# 재귀 호출
def sos(i):
    print("help me!")
    if i <= 1:  # 가장 작은 횟수 처리
        return ""
    else:
        return sos(i-1)

sos(5)  #help me!
#sos(4) #help me!
#sos(3) #help me!
#sos(2) #help me!
#sos(1) #help me!
#공백을 출력하고 종료
```

팩토리얼(factorial)

- 팩토리얼을 구하는 재귀 함수

```python
# 1부터 n까지의 곱 구하기(1x2x3x...xn)
def facto(n):
    gob = 1
    for i in range(1, n+1):
        gob *= i
        #print(i, gob)
    return gob

print(facto(1))
print(facto(4))
print(facto(5))
```

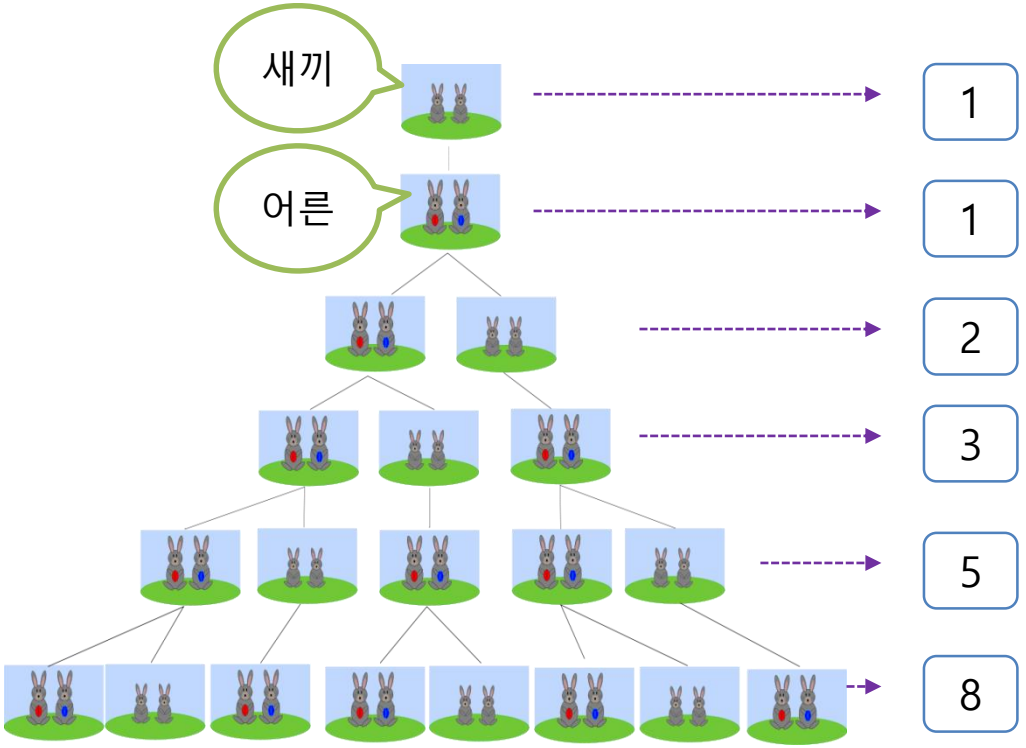
```python
# 팩토리얼(factorial) - 5! = 5 x 4 x 3 x 2 x 1
def facto(n):
    if n <= 1:
        return 1
    else:
        return n * facto(n-1)

"""
  5!
  = 5 x (5-1)!
  = 5 x (5-1) x (5-2)!
  = 5 x (5-1) x (5-2) x (5-3)!
  = 5 x (5-1) x (5-2) x (5-3) x (5-4)!
"""
print(facto(1))
print(facto(5))
```

피보나치 수열

- **피보나치(Fibonacci) 수열**

  수학에서 피보나치 수는 첫째 및 둘째 항이 1이며, 그 뒤의 모든 항은 바로 앞 두 항의 합인 수열이다. 처음 여섯 항은 각각 1, 1, 2, 3, 5, 8이다.



첫번째 달에 새로 태어난 토끼 한쌍이 있고, 둘째달에 토끼가 커서 그대로 어른토끼 한쌍, 세째달에는 새끼를 한쌍 낳아 어른, 새끼 두쌍, 네째달에는 어른이 새끼를 낳고, 새끼는 어른이 되어 총 세쌍, 이렇게 계속 새끼를 낳고, 죽지 않는다는 가정을 세우면 피보나치의 수가 된다.

재귀 호출

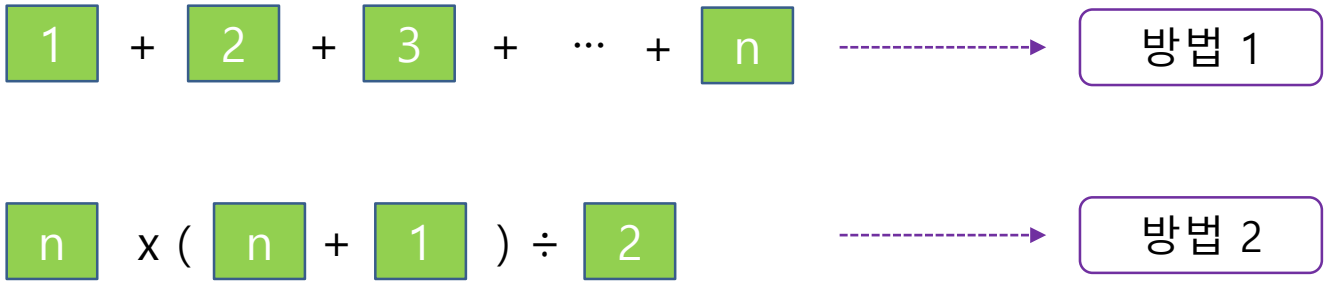
```python
# fibonacci - 1, 1, 2, 3, 5, 8...
# 앞항 + 뒤항 = 항 반복
def fibo(n):
    if n <= 2:
        return 1
    else:
        return fibo(n-2) + fibo(n-1)

"""
    n = 1    fibo(1) = 1
    n = 2    fibo(2) = 1
    n = 3    fibo(1) + fibo(2) = 2
    n = 4    fibo(2) + fibo(3) = 3
    n = 5    fibo(3) + fibo(4) = 5
"""
print(fibo(1))
print(fibo(2))
print(fibo(3))
print(fibo(4))
```

```
1
1
2
3
```

알고리즘 계산 복잡도

➢ 1부터 n까지의 합을 구하기




```python
def sum_n(n):
    sum_v = 0
    for i in range(1, n+1):
        sum_v += i
    return sum_v

print(sum_n(10))
```

```python
def sum_n2(n):
    sum_v = (n * (n + 1)) // 2
    return sum_v

print(sum_n2(10))
```

알고리즘 계산 복잡도

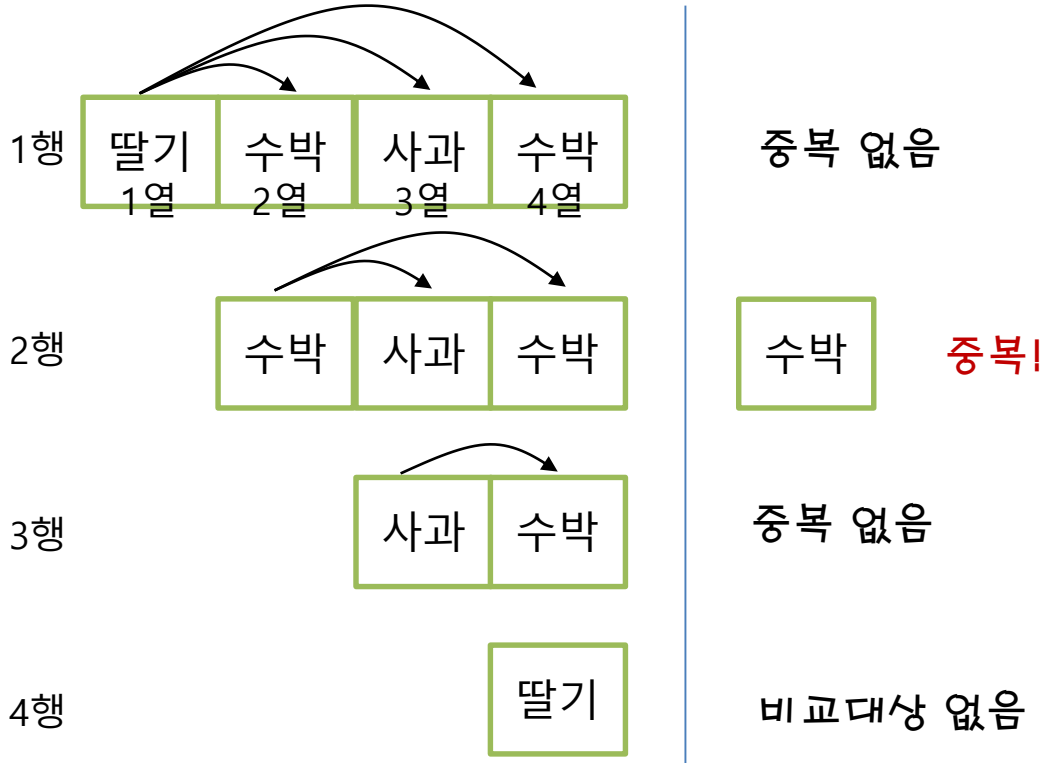
➢ 계산 복잡도

- 입력 크기와 계산 횟수
  - 첫 번째 알고리즘 : 덧셈 n번
  - 두 번째 알고리즘 : 덧셈, 곱셈, 나눗셈(총 3번)

- 대문자 O표기법 : 계산 복잡도 표현
  - O(n) : 필요한 계산횟수가 입력 크기 n과 비례할 때
  - O(1) : 필요한 계산횟수가 입력 크기 n과 무관할 때
- **판단**
  - 두번째 방법이 계산 속도가 더 빠름

동명이인 찾기 – 중복 검사

♥ 같은 이름 찾기



동명이인 찾기 – 중복 검사

```python
# 동명 이인 찾기
def find_same_name(li):
    same_name = []
    n = len(li)
    for i in range(0, n-1):
        for j in range(i+1, n):
            if li[i] == li[j]:
                same_name.append(li[i])
    return same_name
'''
1행 li[0] == li[1], li[0] == li[2], li[0] == li[3], li[0] == li[4]
2행 li[1] == li[2], li[1] == li[3], li[1] == li[4]
3행 li[2] == li[3], li[2] == li[4](중복!)
4행 li[3] == li[4]
5행 li[4] 비교 대상 없음
'''
name = ['콩쥐', '팥쥐', '흥부', '놀부', '흥부']
result = find_same_name(name)
print(result)
```

lambda Expressions(람다식)

- **Lambda Expressions(람다식)**

  lambda 키워드로 익명 함수를 만들수 있다.

lambda 매개변수 : 표현식

```
#일반 함수
def add(x, y):
    return  x + y

print(add(1, 2))
```

```
#lambda 함수
add = lambda x, y : x + y
print(add(1, 2))
```

lambda Expressions(람다식)

- 매개변수가 1개인 람다 함수

```
# 1 더하기
oneup = lambda x : x + 1
print(oneup(1))
print((lambda x : x + 1)(1))

# 3의 배수
times = lambda x : x * 3
print(times(2))
print((lambda x : x * 3)(2))

# 제곱수
square = lambda x : x * x
print(square(4))
print((lambda x : x * x)(4))
```

lambda Expressions(람다식)

- 매개변수가 2개인 람다 함수

```
# 사칙연산
add = lambda x, y : x + y
print(add(3, 4))
print((lambda x, y : x + y)(3, 4))

sub = lambda x, y : x - y
print(sub(3, 4))
print((lambda x, y : x - y)(3, 4))

mul = lambda x, y : x * y
print(mul(3, 4))
print((lambda x, y : x * y)(3, 4))

div = lambda x, y : x / y
print(div(3, 4))
print((lambda x, y : x / y)(3, 4))
```

lambda Expressions(람다식)

- 매개 변수가 없는 람다 함수

```
# 함수를 매개 변수로 전달
print("콜백(callback) 함수")
def call_10(func):
    for i in range(10):
        func()

def hello():
    print("안녕하세요")

# 호출
call_10(hello)

# lamba 함수로 정의 - 매개변수가 없는 경우
hello2 = lambda : print("안녕하세요")
call_10(hello2)
```

lambda(람다) 프로그래밍

- **lambda - map() 함수와 함께 사용**

```
def times(a):
    li = []
    for i in a:
        li.append(i * 3)
    return li

v = [1, 2, 3, 4]
print(times(v))

#lambda - map() 함수
times = lambda x : x * 3
result = map(times, v)
print(list(result))
```

lambda(람다) 프로그래밍

- **lambda - filter() 함수와 함께 사용**

```
def negative(a):
    li = []
    for i in a:
        if i < 0:
            li.append(i)
    return li

v = [-5, 1, 2, -11, 76]
print(negative(v))

#lambda - filter() 함수
negative = lambda x : x < 0
result = filter(negative, v)
print(list(result))
```