

- 함수란?
  - 매개변수를 전달받아 필요한 처리를 하고 그 결과를 반환하는 프로그램 코드
- 함수는 반복 사용 가능
- 전체 기능을 작은 기능으로 분할하고, 작은 기능을 함수로 구현하면 전체 프로그램을 쉽고 편리하게 작성할 수 있다
  - 프로그램을 작성할 때 가급적 함수를 이용하는 것이 좋다
- 함수의 종류
  - 내장 함수(built-in function)
  - 사용자 정의함수(user defined function)

```
def 함수이름(매개변수):  
    # 들여쓰기  
    함수 몸체  
    (return 반환값)
```

# 1 부터 n 까지 합을 구하는 함수

```
def sum(n): # n = 최종값
    sum = 0 # 합을 초기값
    # 1부터 n까지 차례로 더한다
    for i in range(1, n+1): # 1~n까지 반복
        sum = sum + i      # 합에 새로운 수를 더한다
    return sum            # 합을 반환

# 함수를 호출하고 반환값을 변수에 저장한다
a = sum(100)
b = sum(1000)
print(a)
print(b)
```

# 함수, 리스트, 튜플, 디셔너리 등을 반환할 수 있다

```
# 정사각형의 면적과 둘레를 반환하는 함수
def square(s) : # s = 정사각형 한 변의 길이
    area = s*s
    circumference = 4*s
    return (area, circumference) # 튜플 반환

s = float(input("정사각형 한변의 길이 : "))

# 함수를 호출하고 반환값을 언패킹하여 받는다
a, c = square(s)
print("정사각형 넓이="+str(a)+", 둘레="+str(c))
```

# 함수의 변수

- 함수의 변수는 변수가 정의된 위치에 따라 유효범위가 구분된다
- **지역 변수(local variable)**
  - 함수 내부에서 정의된 변수
  - 함수 내부에서만 유효하다
- **전역 변수(global variable)**
  - 함수 밖에서 정의된 변수
  - 프로그램 전체에서 유효하다
  - global을 이용하여 명시적 선언 가능
- 전역 변수와 동일한 이름을 가진 지역 변수가 함수 내에서 정의된 경우, 해당 함수 내에서 지역 변수가 전역 변수보다 우선한다.

# 함수의 변수

```
def func1() :  
    v = 10 # v는 함수 func1()안의 지역 변수  
    print("func1()의 v = %f" % v)  
  
def func2() :  
    global v # v를 전역변수로 선언  
    v = 30 #전역변수 변경  
    print("func2()의 v = %d" % v) |  
  
v = 20 # 전역변수 선언부, 전역 변수이므로 프로그램 전체에서 유효  
  
func1() # 지역 변수 값인 10을 출력. 지역 변수가 없을 때 전역 변수 사용  
func2() # 전역변수 값이 출력된다  
print(v) # func2()에서 변경된 전역변수
```

```
func1()의 v = 10.000000  
func2()의 v = 30  
30
```

# 함수의 매개변수(parameter)

- 함수를 호출될 때 매개변수를 사용하면 함수에 정보를 전달
  - 함수는 이 정보를 사용하여 작업을 수행
- 매개변수의 종류
  - 위치 매개변수
    - 매개변수 이름만 지정되어 있고 초기값이 없는 변수
    - 위치 매개변수는 항상 기본값 매개변수 보다 먼저 지정되어야 한다
  - 기본(키워드) 매개변수
    - 매개변수 이름과 함께 초기값이 할당된 변수
  - 가변 길이 매개 변수(\*args, \*\*kwargs)
    - \*args (임의의 위치 인자): 위치 인자들을 튜플(tuple) 형태로 묶어서 받는 기능
    - \*\*kwargs (임의의 키워드 인자): 키워드 인자들을 딕셔너리(dictionary) 형태로 묶어서 받는 기능

# 매개변수의 값 전달

```
def hello(msg):  
    print(msg)  
  
def greeting(person, msg="Hello"):  
    print(person, ", ", msg)  
  
hello("안녕하세요")  
  
# 기본값 매개변수는 인수를 생략할 수 있다  
greeting("Kim")  
greeting("Dr. Kim", msg="반갑습니다")  
  
# 위치 매개변수를 생략하면 오류 발생  
greeting(msg="반갑습니다")
```

# 키워드 매개변수를 이용한 값 전달

```
def func(a, b=2, c=3): #a: 위치 매개변수, b, c: 키워드 매개변수
    print('a=', a, 'b=', b, 'c=', c)
    return a + b + c
```

```
print(func(4, 5)) #인자 위치에 의해 a=4, b=5로 전달
```

```
print(func(5, c=7)) #키워드 c를 이용한 전달. a=5, c=7, b는 기본값
```

```
print(func(b=5, c=6)) #위치 매개변수를 생략하면 에러
```

# 가변 길이 매개변수의 전달: 튜플

- 매개변수를 \*args로 표시하면 함수 내에서 매개변수를 Tuple로 처리한다.

```
def total(*numbers):  
    print(type(numbers))  
    sum = 0 # 초기값  
    for n in numbers: # 전달 받은 값을 튜플로 처리  
        sum += n  
    return sum  
  
print (total(1,2,3)) #3개의 인자  
print (total(1,2,3,4,5)) #5개의 인자
```

# 가변 길이 매개변수의 전달: 딕셔너리

- 매개변수를 **\*\*kwargs**로 표시하면 매개변수는 함수 내에서 딕셔너리 형으로 처리된다.
- 인수는 (key1=value1, key2=value2)형으로 전달한다

```
def dicPresident(**keywords):  
    print(type(keywords))  
    for key in keywords.keys():  
        print("%s : %d-th president" %(key, keywords[key]))
```

```
dicPresident(Kennedy=35, Obama=44, Trump=45) # (key=value)로 매개변수 전달
```

# 내장 함수: enumerate

- index와 함께 반복 객체의 요소를 하나씩 호출

```
names = ['a', 'b', 'c', 'd', 'e']  
print(enumerate(names, start=1))  
  
# index와 names 요소를 차례로 출력, index는 0부터 시작  
for elem in enumerate(names, start=0):  
    #print(i, name)  
    print(elem)
```

```
<enumerate object at 0x000001E444169D00>  
(0, 'a')  
(1, 'b')  
(2, 'c')  
(3, 'd')  
(4, 'e')
```

# 내장 함수: filter

- 반복 객체의 요소를 인수로 사용하여 함수를 호출하고 **조건을 만족하는 요소만을 걸러내어** 반복 객체로 반환한다

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

def is_even(n):
    if (n % 2) == 0:
        return True

result = filter(is_even, target)
print(list(result))
```

lambda와 filter 함수를 같이 사용한 예제

```
nums = [1, 2, 3, 11, 12, 13, 21, 22, 23]
# 짝수만 찾아낸다
list(filter(lambda x: (x % 2) == 0, nums))
```

# 내장 함수: map

- 반복 객체의 요소를 인수로 사용하여 함수를 호출하고 **각 요소를 새로운 값으로 변환한 뒤** 반복 객체로 반환한다.

```
target = [1, 2, 3, 4, 5]
result = map(lambda x : x+1, target)
print(list(result))
```

```
[2, 3, 4, 5, 6]
```

# 내장 함수: zip

- 반복객체1과 반복객체2, ...의 요소를 하나씩 튜플로 결합하여 반복 객체로 반환한다.

```
Number = [1,2,3]
Name = ['hong','gil','dong','nim']
Number_Name = zip(Number,Name)
print(list(Number_Name))
```

```
[(1, 'hong'), (2, 'gil'), (3, 'dong')]
```