[함수 (Function)]

함수 개요

- 프로그래밍에서 함수는 흔히 `블랙박스 black box` 라는 비유를 함
- 함수를 호출하는 사용자는 제대로 된 입력 값을 주고 그 결과인 출력(결과 값)을 사용만 하면 되기 때문
- **1.내장함수(built-in function), 2.외장함수(import), 3.사용자 정의 함수(user defined function)**

1. 내장함수(built-in function) : 파이썬이 미리 만들어서 제공 함수 (70개 정도)

- 예: input(), print(), len(), range()....등등

- 내장함수 목록

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

		내장 함수		
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()	

In [ ]:
```
# 내장함수(built-in function)
dir(__builtins__)
```

In [ ]:
```
# help(함수명)
help(abs)
```

In [ ]:
```
abs(-100) # 절대값을 반환하는 함수
```

In [ ]:
```
min(200, 100, 300, 400) # 여러 원소들 중 최솟값을 반환하는 함수
```

In [ ]:
```
int("100") + 1  # 문자형을 정수형으로 변환
```

In [ ]:
```
help(print)
```

In [ ]:
```
help(str.split)
```

[참고] 파이썬 구성

- 비슷한 종류의 일을 하는 함수들 끼리 따로 묶어서 `'모듈'` 이라는 묶음으로 제공
- 내장함수 (Built-in Fuction) : 바로 사용 가능
- 내장 자료형 관련 메소드 : 각각 자료형에 해당되는 메서드
- 모듈 : 외부함수/라이브러리 사용 전에 관련 모듈을 먼저 import 해야 함.

2. 외장함수 (import 함께) 예: import math

- **예) sys.path, random.random(), math.sin(x), datatime.time(x) ....**
- 파일과 디렉터리 접근 : sys, os
- 데이터 파일 저장 : pickle
- 수학 및 랜덤 : math, random
- 인터넷 액세스 : webbrowser, urllib
- 날씨와 시간 : time, datetime
- 등등....

In [ ]:
```
import math
dir(math) # math에 있는 함수와 데이터
#help(math)
```

In [ ]:
```
#import 모듈
import math
math.sin(10), math.sqrt(2.0), math.pi, math.log(3)
```

In [ ]:
```
from math import *
sin(10), sqrt(2.0), pi, log(3)
```

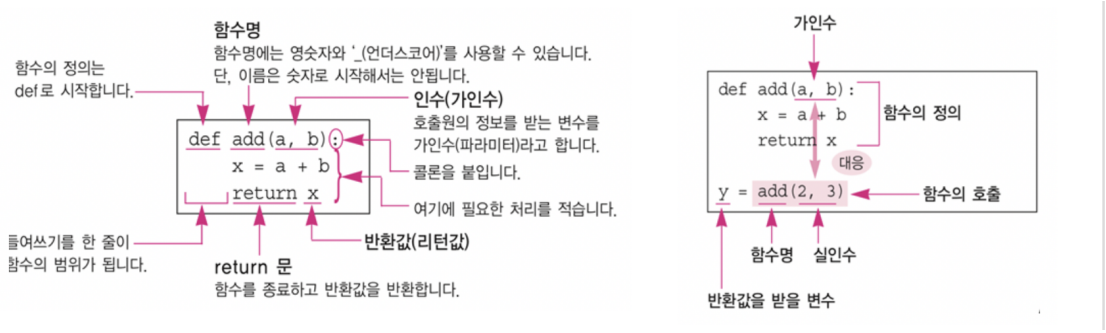
3. 사용자 정의 함수(user defined function) : 프로그래머가 직접 만드는 함수

- 함수(Function) : '무엇'을 넣으면, '어떤 것'을 돌려줌
- 메서드(Method)와 차이점 : 함수는 외부에 별도로 존재, 메서드는 클래스 안에 존재

[사용자 정의 함수]

함수 만들기 (함수정의)

- 매개변수(parameter): 함수의 입력 부분
- 인수(argument): 함수를 호출할 때 넘기는값
- 반환값(return): 결과값



- 일반 vs. 함수 (예 : 면적구하기)

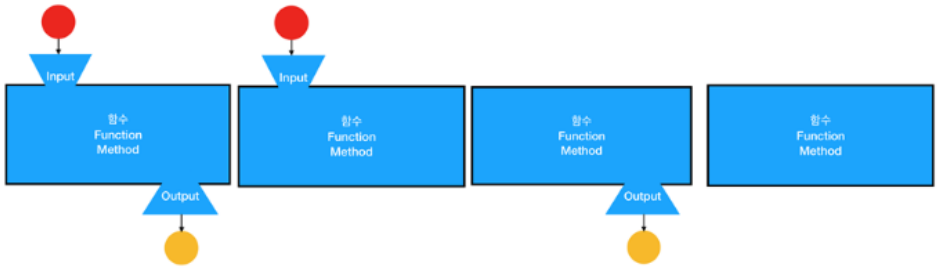
In [ ]:
```
# 1.일반(면적구하기)
h = 10
v = 20

area = h * v
print(area)
```

In [ ]:
```
# 2. 함수사용
def area(h, v):
    return h * v
```

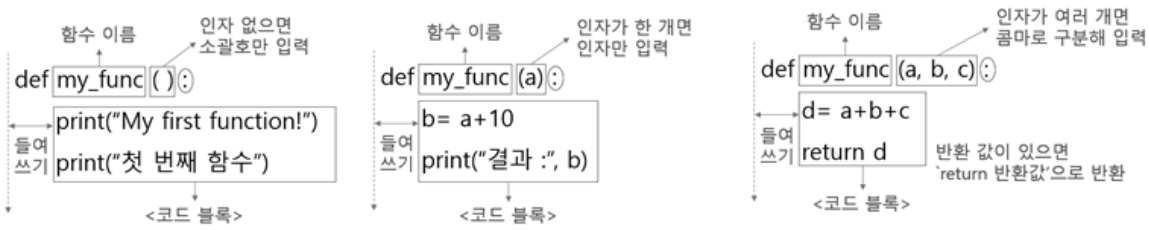
In [ ]:
```
a = area(10, 20)
print(a)
```

입력값과 반환값에 따른 함수 종류

- **입력값 --> 함수 --> 반환**
- 함수의 형태는 입력값과 결과값의 존재 유무에 따라 4가지 유형



- 인자(인수)도 반환 값도 없는 함수
- 인자는 있으나 반환 값이 없는 함수
- 인자도 있고 반환 값도 있는 함수



1. 인자값X - 출력값X 함수

In [ ]:
```
def say():
    print('Hi')

# main
say()
```

2. 인자값X - 출력값O 함수

In [ ]:
```
def say():
    return 'Hi'

# main
a = say()
print(a)
```

3. 인자값O - 출력값X 함수

In [ ]:
```
def add(a, b):
    print("{},{}의 합은 {}입니다.".format(a, b, a+b))

# main
add(3, 4)
```

4. 인자값O-출력값O 함수(일반적인 함수)

- 입력값이 있고 결과값이 있는 함수가 일반적인 함수이다

In [ ]:
```
def add(a, b):
    c = a + b
    return c

## main
x = add(3, 4)
print(x)
```

함수 매개 변수

1. 인수 전달 시 매개변수 지정

In [ ]:
```
def sub(a, b):
    print(a - b)

sub(1, 2)
```

```
sub(a=1, b=2)
sub(b=1, a=2)
```

In [ ]:
```
# 함수 선언 부분
def add2(v1, v2) :
    x2 = v1 + v2
    return x2

def add3(v1, v2, v3) :
    x3 = v1 + v2 + v3
    return x3

# 전역 변수 선언 부분
tot = 0

# 메인 코드 부분
tot = add2(10, 20)
print("매개변수2개 함수 =>", tot)

tot = add3(10, 20, 30)
print("매개변수3개 함수 =>", tot)
```

2. 매개변수의 기본값 설정(default parameters)

- 매개변수 2개, 3개가 있으면 함수를 2개 만드는데 불편 --> 3개중 1개를 기본값설정

In [ ]:
```
def total(a, b=0, c=0):
    print(a + b + c)

total(1)
total(1, 2)
total(1, 2, 3)
```

In [ ]:
```
# 함수 선언 부분
def add23(v1, v2, v3=0) :
    x23 = 0
    x23 = v1 + v2 + v3
    return x23

# 전역 변수 선언 부분
tot = 0

# 메인 코드 부분
tot = add23(10, 20)
print("매개변수 2개 함수 =>",tot)

tot = add23(10, 20, 30)
print("매개변수 3개 함수 =>",tot)  # 기본값 무시 됨
```

3. 가변 매개변수(variable parameters)

- 매개변수에 개수를 지정하지 않고 전달
- 입력값이 몇 개가 될지 모를 때는 어떻게 해야 할까?
- 인자의 수가 정해지지 않은 가변 인자 arbitrary argument
- def 함수이름(*매개변수):

In [ ]:
```
def add(*paras):
    print(paras)
    total = 0
```

```
    for para in paras:
        total += para
    return total

print(add(10))
print(add(10, 100))
print(add(10, 100, 1000))
```

In [ ]:
```
# 함수 선언 부분
def greet(*names):
    for name in names:
        print('안녕하세요 ! ', name, '씨')

# 메인 코드 부분
greet('홍길동', '박수원', '김미래') # 인자가 3개

greet('James', 'Thomas') # 인자가 2개
```

In [ ]:
```
# 함수 선언 부분
def add_mul(choice, *args):
    if choice == "add":
        result = 0
        for i in args:
            result = result + i
    elif choice == "mul":
        result = 1
        for i in args:
            result = result * i
    return result


result = add_mul('add', 1,2,3,4,5)
print("함수 덧셈 결과 =>", result)

result = add_mul('mul', 1,2,3,4,5,6,7)
print("함수 곱셈 결과 =>", result)
```

In [ ]:
```
# 가변 매개변수로 딕셔너리를 사용 위해, (**)기호 사용
def print_map(**dicts):
    for item in dicts.items():
        print(item)

# 메인 코드 부분
print_map(하나=1); print("--------------")
print_map(one=1, two=2); print("--------------")
print_map(하나=1, 둘=2, 셋=3); print("--------------")
```

변수의 유효 범위: 지역변수(Local), 전역(global)변수

1. 전역 변수(Global Variable)
   - 함수 바깥에서 선언하는 변수
   - 어디에서나 참조 할수 있다
2. 지역 변수(Local Variable)
   - Variable with local scope
     + 함수 내부에서 선언하는 변수
     + 함수 내부에서만 사용
   - Variable with global scope
     + 변수를 함수나 블록 내에서 값을 지정하지만, 어디서나 사용 가능하게 정의
     + global count

In [ ]:
```
# 함수 정의
def f():
    x = 10 # 지역변수
    print(x)

# 함수 호출
f()
```

In [ ]:
```
# 전역 변수
b = 20

# 함수정의
def f():
    print(b)

def g():
    print(b + 3)

# 함수호출(main)
f()
g()
```

In [ ]:
```
# 전역 변수
a = 35

# 함수 정의
def f1() :
    a = 10 # 지역 변수
    print("f1()에서 a값(지역변수)", a) # 지역변수 a

def f2() :
    print("f2()에서 a값(전역변수)", a) # 전역변수 a

# 함수 호출
f1()
f2()
```

In [ ]:
```
# 같은 이름의 변수를 지역 변수와 전역 변수로 모두 사용한 예
# global 키워드를 사용한 전역변수의 참조 방법

# 전역변수
a = 5

# 함수 정의
def func1():
    a = 1 # 지역 변수. func1()에서만 사용
    print("[func1] 지역 변수 a =", a)

def func2():
    a = 2 # 지역 변수. func2()에서만 사용
    print("[func2] 지역 변수 a =", a)

def func3():
    print("[func3] 전역 변수 a =", a)

# global 키워드를 사용한 전역변수의 참조 방법
def func4():
    global a    # global 변수: 함수 내에서 전역 변수를 변경하기 위해 선언
    a = 4       # 전역 변수의 값 변경
    print("[func4] global a =",a)

# 함수 호출
```

```
func1() #함수 func1() 호출
func2() #함수 func2() 호출
print("## 전역 변수 a =", a) # 전역 변수 출력

func3() # 함수 func3() 호출 - 비교 확인
func4() # 함수 func4() 호출

func3() # 함수 func3() 호출 - 비교 확인

print(a)
```

---------- END ----------