기초 컴퓨터 프로그래밍

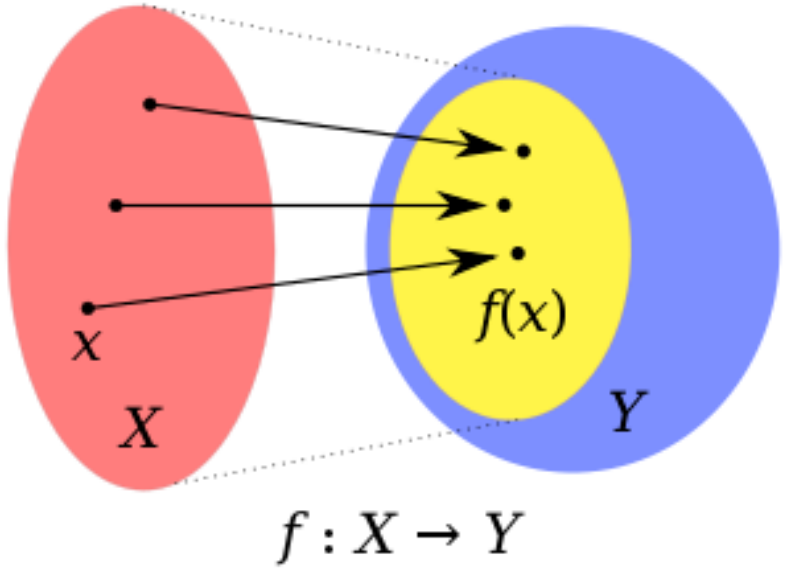
- Python 기본: 함수 -

충남대학교 의과대학 의공학교실

구 윤 서 교수

함수

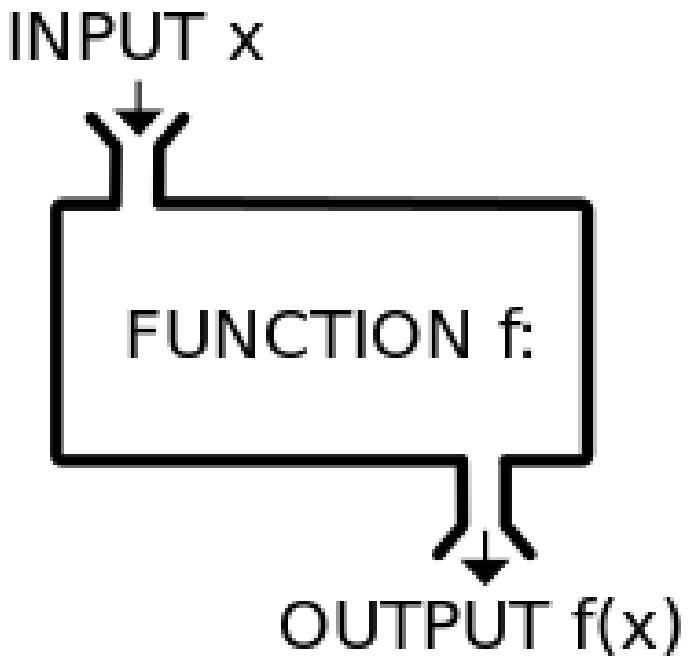
- 함수(수학): 첫 번째 집합의 임의의 한 원소를 두 번째 집합의 오직 한 원소에 대응시키는 대응 관계



https://ko.wikipedia.org/wiki/함수

함수

- 함수(프로그래밍): 특정한 기능을 수행하는 코드의 일정 부분



※ Input과 Output이 없는 함수도 존재

함수

- 어떤 식으로 프로그램이 동작할지 Flowchart 등을 이용하여 할 것인지 설계하게 되는데 그때 가장 중요한 부분 중 하나가 입출력의 선정임.

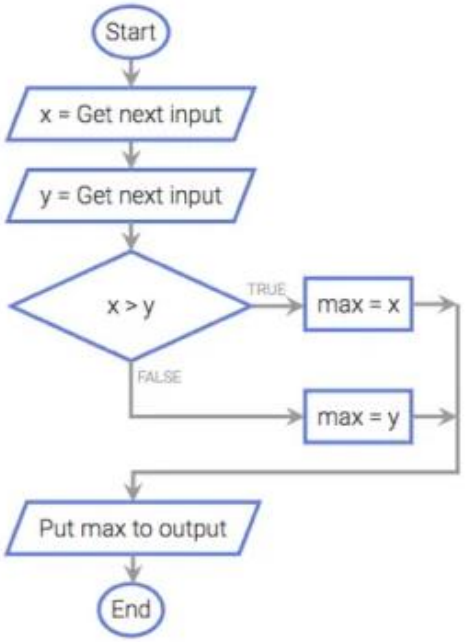
- 특정 프로그램만이 사용하는 함수를 만들 것인지 아니면 모든 프로그램이 공통으로 사용하는 함수를 만들 것인지는 입출력과 관련이 있음.

```
integer x
integer y
integer max

x = Get next input
y = Get next input

if x > y
   max = x
else
   max = y

Put max to output
```



[Flowchart]

함수의 기본 구조

```
def 함수명(매개변수): # def 다음에 콜론(:)을 잊지 말자!
        수행할 문장1
        수행할 문장2
        ...
```

- def는 함수를 만들 때 사용하는 예약어
- 함수명은 작성자가 임의로 작성 가능
- 괄호 안의 매개변수(parameter)는 이 함수에 입력으로 전달되는 값(argument)을 받는 변수
- 함수를 정의한 후 if문 등과 마찬가지로 함수에서 수행할 문장 입력(Indentation!)

함수 사용 시 장점

- 동일한 기능이 반복적으로 사용될 때 한 번만 작성하여 사용 가능하므로 간결한 코드 작성 가능
- 하나의 큰 프로그램을 여러 부분으로 분리함으로써 코드의 흐름 파악에 용이
- 함수의 기능과 내부 구현을 분리하는 캡슐화 (encapsulation) 수행
  - 입·출력 Interface와 기능만 알면 타인이 작성한 함수 사용 가능

함수 사용 시 장점

- 다수의 개발자가 프로그램을 기능 별로 나누어서 작성하고 통합하는 경우가 잦음.

- 프로그램을 기능 별로 나누는 방법
  - 함수 (Function)
  - 클래스 (Class)
  - 모듈 (Module)
  - 패키지 (Package)

함수 & 클래스

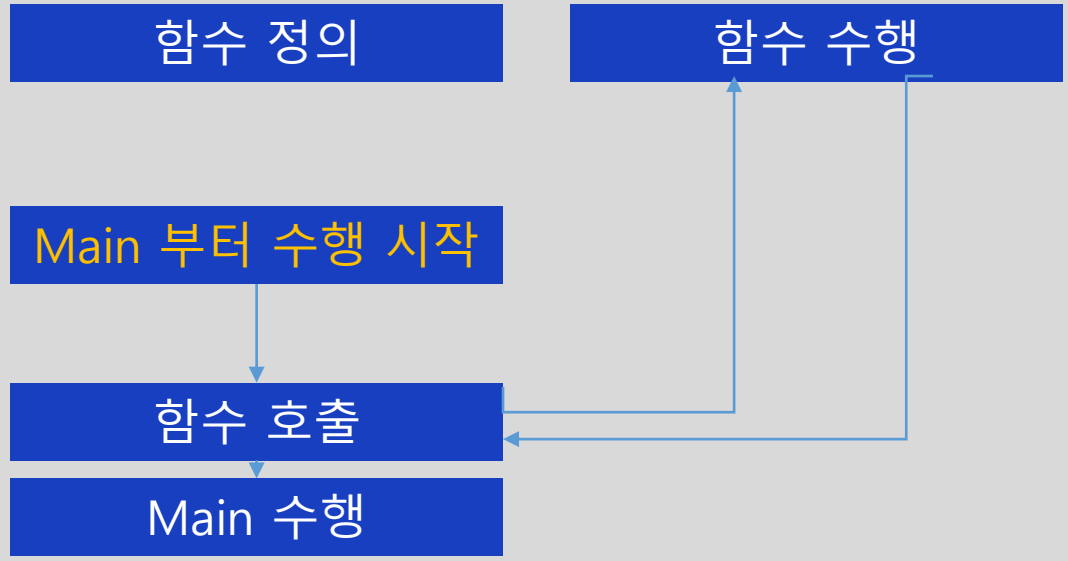
- 간단한 프로그램은 함수만으로 효율적으로 작성 가능 → 절차지향 프로그래밍 (procedure-oriented programming)
  - 가장 많이 사용하는 프로그래밍 언어 중 하나인 C 언어에는 클래스가 없음 → 굳이 클래스가 없어도 프로그램작성 가능

- 복잡한 프로그램 작성 시 객체지향 프로그래밍 (object-oriented programming)을 사용하면 구현이 용이 → 객체(object)를 만들어 내는 클래스 정의 필요
  - Python에서는 class라는 키워드로 클래스 정의

함수의 기본 구조

```
def sum_test(a, b):
        return a+b


a = 3
b = 4
c = sum_test(a, b)
print(c)
```




```
7
```

- 일반적으로 함수는 코드 가장 앞에 작성 (의무사항은 아님)
- 함수(def) 부분은 실행되지 않고 정의만 됨. → 실제 코드의 수행은 "a=3" 라인부터 시작됨.

함수의 기본 구조

- **매개변수 (parameter)**: 함수에 입력으로 전달된 값을 받는 변수
- **인수 (argument)**: 함수를 호출할 때 전달하는 입력 값

```
def sum_test(a, b):  # a와 b는 매개변수 (parameter)
        return a+b


c = sum_test(3, 4)  # 3과 4는 인수 (argument)
print(c)
```

함수의 사용

```
입력 #1
수행할 문장1
수행할 문장2
출력 #1
…
입력 #2
수행할 문장1
수행할 문장2
출력 #2
…
입력 #3
수행할 문장1
수행할 문장2
출력 #3
```



```
def func(입력):
        수행할 문장1
        수행할 문장2
        return 결과값

…
출력 #1=func(입력 #1)
…
출력 #2=func(입력 #2)
…
출력 #3=func(입력 #3)
```

간결한 코드 작성 가능

입력값과 결과값이 모두 있는 함수

- 수학에서의 함수와 유사

```
def sum_test(a, b):
        result_test = a+b
        return result_test


y = sum_test(3, 4)
print(y)
```

Return 값이 존재

```
7
```

입력값과 결과값이 모두 있는 함수

f(x) = 2x+7, g(x) = x^2 이고 x = 2일 때, f(x)+g(x)+f(g(x))+g(f(x))의 값은?

>>> 11+4+15+121 = 151

```
def f(x):
        return 2*x+7
def g(x):
        return x**2
x=2
print(f(x)+g(x)+f(g(x))+g(f(x)))
```

```
151
```

결과값이 없는 함수

```
def sum_test(a, b):
        print("%d, %d의 합은 %d입니다." % (a, b, a+b))

sum_test(3,4)
```

Return 값이 없음

```
3, 4의 합은 7입니다.
```

```
def sum_test(a, b):
        print("%d, %d의 합은 %d입니다." % (a, b, a+b))

y = sum_test(3,4)
print(y)
```

Return 값이 없음

```
None
```

입력값도 결과값도 없는 함수

```
def say():
    print("Hi")



say()
```

Hi

Variable-Length (가변 인자: *args)

```
def sum_many(*args):
    sum_value = 0
    for i in args:
        sum_value += i
    return sum_value


result = sum_many(1, 2, 3)
print(result)

result = sum_many(1, 2, 3, 4, 5)
print(result)
```

args는 arguments(인수)의 약자이며 관례적으로 사용

매개변수 이름 앞에 *을 붙이면 입력 값을 전부 모아서 tuple로 만듦

```
6

15
```

Variable-length (가변 인자)

실습 #1

```
def sum_mul(choice, *args):
    if choice == "sum":
        result = 0
        for i in args:
            result += i
    elif choice == "mul":
        result = 1
        for i in args:
            result *= i
    return result


result = sum_mul("sum", 1, 2, 3, 4, 5)
print(result)

result = sum_mul("mul", 1, 2, 3, 4, 5)
print(result)
```

*args 매개변수 앞에 choice 매개변수 추가
(*args는 가장 마지막에 한 번만 사용 가능)

15

120

Keyword Variable-Length

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


print_kwargs(a = 1)

print_kwargs(name = 'foo', age = 3)
```

**을 붙이면 매개변수 kwargs는 dictionary가 되고, key=value 형태로 dictionary에 저장

```
{'a': 1}

{'name': 'foo', 'age': 3}
```

함수 안에서 선언된 변수의 효력 범위 (scope)

- 변수의 범위(Scoping Rule)
  - 지역 변수(local variable): 함수 내에서만 사용
  - 전역 변수(global variable): 프로그램 전체에서 사용

```
def vartest(a):
    a += 1
    print("In function, variable 'a' is :", a)


a = 1
vartest(a)
print("In main, variable 'a' is :", a)
```

함수 안에서 선언된 변수의 효력 범위

```
def vartest(a):
    a += 1
    print("In function, variable 'a' is :", a)


a = 1
vartest(a)
print("In main, variable 'a' is :", a)
```

Return 값(=출력값)이 없음!

```
In function, variable 'a' is : 2
In main, variable 'a' is : 1
```

- 함수 안에서 새로 만들어진 매개변수는 함수 안에서만 사용되는 변수 → 지역 변수(local variable)
- 전역 변수와 같은 이름의 변수로 선언하면 새로운 지역 변수가 생김

함수 안에서 함수 밖의 변수를 변경하는 방법

- 지역 변수를 반환하여 함수 밖의 변수 변경

```
def vartest(a):
        a += 1
        return a


a = 1
a = vartest(a)
print(a)
```

```
2
```

함수 안에서 함수 밖의 변수를 변경하는 방법

- global 명령어를 사용하여 전역 변수로 변경

```
def vartest(): # 입력받는 argument가 없음을 주의할 것!
        global a
        a += 1


a = 1
vartest()
print(a)
```

```
2
```

※ 복잡한 프로그램 작성 시 여러 함수 내에서 일일이 어떤 변수가 global 변수 확인하는 것이 어려워 global 사용은 추천하지 않음.

변수의 범위 예제 #1

```
def func_test(t):
    t = 20
    print("In function :",t)


x = 10
print("Before :", x)
func_test(x)
print("After :", x)
```

변수의 범위 예제 #1

```
def func_test(t):
    t = 20
    print("In function :",t)


x = 10
print("Before :", x)
func_test(x)
print("After :", x)
```

```
Before : 10
In function : 20
After : 10
```

변수의 범위 예제 #2

```
def func_test(t):
    t = 20
    print("In function :",t)


x = 10
print("Before :", x)
func_test(x)
print("After :", x)
print(t)
```

변수의 범위 예제 #2

```
def func_test(t):
    t = 20
    print("In function :",t)

x = 10
print("Before :", x)
func_test(x)
print("After :", x)
print(t)
```

```
Before : 10
In function : 20
After : 10
```

```
NameError: name 't' is not defined
```

함수 밖에는 't'가 선언되지 않았으므로 에러 발생

Python 내장함수 (Built-in functions)

Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

		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()	

https://docs.python.org/3/library/functions.html

Enumerate 함수

- **List의 element를 번호를 붙여서 추출**

```
for i, j in enumerate(['zero', 'one', 'two']): # list index와 값 unpacking
    print(i, j)

enum_example1 = ['zero','one','two','three']
enum_example1_list = list(enumerate(enum_example1)) # unpacking하여 list로 저장
print(enum_example1_list)

enum_example2_dict = {i:j for i,j in enumerate('word1 word2 word3 word4'.split())}
print(enum_example2_dict)
```

```
0 zero
1 one
2 two
[(0, 'zero'), (1, 'one'), (2, 'two'), (3, 'three')]
{0: 'word1', 1: 'word2', 2: 'word3', 3: 'word4'}
```

Zip 함수

- **두 개의 list의 값을 병렬적으로 추출 (Zipper로 묶어서 보관)**

```
zip_example_data = ['data1', 'data2', 'data3']
zip_example_label = ['class2', 'class1', 'class2']

for x, y in zip(zip_example_data, zip_example_label): # 병렬적으로 값 추출
    print (x,y)

id0,id1,id2 =zip((1,2,3),(10,20,30),(100,200,300)) # 같은 tuple index 끼리 grouping
print(id0, id1, id2)

zip_example2_list = [sum(x) for x in zip((1,2,3), (10,20,30), (100,200,300))]
print(zip_example2_list)
```

data1 class2
data2 class1
data3 class2

(1, 10, 100) (2, 20, 200) (3, 30, 300)

[111, 222, 333]

Enumerate & Zip

```
data = ['data1', 'data2', 'data3']
label = ['class2', 'class1', 'class2']

for i, (x, y) in enumerate(zip(data, label)):
    print (i, x, y) # index data[index] label[index] 표시
```

0 data1 class2
1 data2 class1
2 data3 class2

Split 함수

- **String Type의 값을 나눠서 List 형태로 변환**
- **문자와 숫자가 섞여 있는 data 처리 시 자주 활용**

```
split_example1 ='Seoul Gwacheon Daejeon Sejong'.split() # space 기준으로 분해하여 list 생성
print(split_example1)

split_example2 ='Seoul,Gwacheon,Daejeon,Sejong'

split_example2_comma = split_example2.split(",") # comma 기준으로 분해하여 list 생성
print(split_example2_comma)

a,b,c,d =split_example2.split(",")
print(a, b, c, d)
```

['Seoul', 'Gwacheon', 'Daejeon', 'Sejong']

['Seoul', 'Gwacheon', 'Daejeon', 'Sejong']

Seoul Gwacheon Daejeon Sejong

Join 함수

- **String List를 합쳐 하나의 String으로 반환**

```
example1 =['Seoul','Gwacheon','Daejeon','Sejong']
join_example1 ='' .join(example1)
print(join_example1)

join_example2 =' '.join(example1) #space
print(join_example2)

join_example3 =', '.join(example1) #comma
print(join_example3)
```

SeoulGwacheonDaejeonSejong

Seoul Gwacheon Daejeon Sejong

Seoul, Gwacheon, Daejeon, Sejong

Next class

- 클래스 (Class)
- 모듈 (Module)
- 패키지 (Package)