Lecture #03 | 함수

SE213 프로그래밍 (2019)

Written by Mingyu Cho

Edited by Donghoon Shin

지난 시간에 다룬 내용

- 변수
  - 변수 대입문과 축약형
  - 변수의 저장공간
- 형변환 (type conversion)
- 조건문: if, else, elif
- 주석
- 모듈 사용: math, random

[Recap] 조건문: if, else, elif

```
if condition1:
    statement1_1
    statement1_2
elif condition2:
    statement2_1
    statement2_2
    statement2_3
elif condition3:
    statement3
else:
    else_statement
```

- if: 조건이 참일 때, 코드블럭(code block)을 실행
- elif
  - 추가적인 조건과 그 조건이 참인 경우 실행될 코드블럭을 지정
  - 0개 이상 사용 가능 (즉, 생략 가능하고, 개수의 제한이 없음)
- else
  - 만족하는 조건이 없을 때, 실행되는 코드블럭을 표시
  - 생략되거나 하나의 else절만 사용가능
- 의미
  - 위에서부터 조건을 검사하여 가장 먼처 참이되는 조건에 해당하는 코드블럭만을 실행
  - 만족하는 조건이 없을 때는 else 이후의 코드블럭을 실행

* ␣는 빈칸을 의미. python에서는 통상 4개의 빈칸을 사용하여 code block을 표시함

[Recap] 모듈/패키지 사용하기

- 모듈을 읽어들이는 방법
  - import module_name
  - module_name은 .py을 뺀 파일이름
- 모듈 사용방법: module_name.xxx 와 같이 사용하면 모듈에 속한 함수 혹은 변수 이름을 뜻한다. 사용 예:
  - module_name.function_name()
  - module_name.variable_name

- 네임스페이스(name space)[1]: 변수, 함수 등이 정의되어 있는 공간
  - 프로그램이 실행될 때, 전역 네임스페이스(global namespace)를 만듦
  - 새로운 변수가 정의되면, 네임스페이스에 변수의 값을 저장할 공간을 만듦
  - 참고: 함수, 모듈의 내부에서 독립적인 네임스페이스가 만들어짐

오늘 다룰 내용

- 함수
  - 함수 소개
  - 함수의 정의와 인자 전달 방법
  - 네임스페이스와 지역 변수
  - return문과 반환값

함수란 무엇이고, 왜 사용하는가?

- 함수(function or subroutine/procedure)란?
  - 입력과 출력의 관계
  - 특정한 기능을 하는 명령문들의 집합
- 함수를 사용하는 이유
  - 공통된 부분의 코드를 재사용할 수 있도록 하기 위해
  - 코드의 가독성을 높이기 위해
  - 반복과 오류를 줄이기 위해
  - 추상화를 제공하기 위해

함수의 정의와 호출

```
def say_hello():
    print('Hello, World!')



say_hello()
print('Hi, python!')
say_hello()
```

```
Hello, World!
Hi, python!
Hello, World!
```

함수의 정의와 호출: 매개변수/인자와 반환값이 있는 경우

```
def square(x):
    return x * x


two_squared = square(2)
print(two_squared)
five_squared = square(5)
print(five_squared)
```

```
4
25
```

수학의 함수와 python의 함수 비교

	수학 함수	python의 함수
정의 (definition)	$f(x) = x^2$	def f(x): return x * x
호출 (call, invocation)	$y = 2 \cdot f(5) + 3$	y = 2*f(5) + 3
매개변수 (parameter)	x	x
인자[1] (argument)	5	5
반환값[2] (return value)	명시할 필요없음 (수식 자체가 반환값을 의미)	명시해야 함

1 함수로 전달되는 값 2 함수에서 반환되는 값

python의 함수의 종류

- 내장 함수 (built-in functions)
  - 예: print(), input(), max(), min()
  - 68개 내장함수의 리스트: https://docs.python.org/3/library/functions.html
- 라이브러리 함수 (library functions)
  - 예: math 패키지의 exp(), log() 등
- 사용자 정의 함수 (user-defined functions)

함수 이름에 대한 규칙

- 변수명과 동일한 규칙: 알파벳, 숫자, _로 이루어지고, 알파벳 혹은 _로 시작
- python의 **예약어**들은 함수이름으로 사용할 수 없음, 예:

True	False	None	in	is
and	or	not	nonlocal	global
if	elif	else		
for	while	break	continue	
def	return	lambda	yield	class
import	from	as	with	finally
try	except	raise	assert	pass

- python 내장 함수의 이름을 사용할 수 있으나, 매우 권장되지 않음

함수를 정의하는 위치

- 함수를 부르기 전에 먼저 정의되어야 한다
  → 일반적으로 파일 앞부분에 함수를 정의
- 규모가 큰 프로그램을 짜게 될 경우에는,
  논리적 구조에 따라 여러 파일에 코드를 나눔
  (이 때 class 혹은 module 등을 사용)

함수의 정의와 사용

- 함수의 정의

```
def function_name(parameters):
    statement1
    statement2
    ...
    return return_value
```

함수명과 매개변수*(parameter, 함수 내부에서 사용되는 입력 변수의 이름)

함수가 호출될 때마다 실행되는 코드블럭 (들여쓰기로 구분됨)

반환값(return value)를 돌려주기 위해 return문을 사용

- 함수의 사용(실행)

```
function_name(argements)
```

전달인자*(arguments): 함수를 호출할 때 전달하는 값

* 매개변수와 인자가 2개 이상인 경우에 수학 함수에서와 같이 ,를 이용하여 구분함

함수의 정의와 사용

- [참고] 함수 설명문

```
def function_name(parameters):
    ''' this is an example function '''
    statement1
    ...
    return return_value
```

```
>>> help(print)
Help on built-in function print in module builtins:
print(...)
    print(value, ..., sep=' ', end='₩n', file=sys.stdout, flush=False)
    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
>>>
```

예제 1: 매개변수/인자와 반환값이 없는 경우

```
def say_hello():
    print('Hello, World!')



say_hello()
print('Hi, python!')
say_hello()
```

```
Hello, World!
Hi, python!
Hello, World!
```

예제 2: 매개변수/인자와 반환값이 있는 경우

```
def square(x):
    return x * x


two_squared = square(2)
print(two_squared)
five_squared = square(5)
print(five_squared)
```

```
4
25
```

위치를 이용한 인자 전달

- 함수에 정의되어 있는 매개변수 위치(순서)에 따라서 값 전달
- 참고. 전달 인자의 개수를 사전에 지정하지 않고 tuple을 이용하여 동적 전달 가능

```
def power(base, exponent):
    return base ** exponent

def power2(*args):
    base, exponent = args
    return base ** exponent

result = power(2, 2)
print(power(2, 10))
print(power2(2, 10))
```

```
1024
1024
```

키워드를 이용한 인자 전달

- 매개변수 이름을 키워드로 지정할 수 있음 → 순서가 중요하지 않음
- 디폴트 매개변수가 여러 개 있을 때, 그 중 일부 매개변수에만 인자를 넘길 때 주로 사용함

```
def power(base, exponent):
    return base ** exponent

print(power(2, 10))
print(power(base=2, exponent=10))
print(power(exponent=10, base=2))
```

```
1024
1024
1024
```

매개변수의 기본값 지정

- 함수를 정의할 때, 매개변수의 기본값(default value)을 지정할 수 있음
  - 매개변수 이름 뒤에 =*default_value*로 기본값을 명시할 수 있음
  - 함수를 호출할 때, 인자를 넘겨주지 않은 경우에는 기본값을 사용함
- 기본값이 있는 매개변수들은 기본값이 없는 매개변수들 뒤에 나타나야 함
- 기본값이 있는 매개변수들이 여러 개가 있을 때, 앞쪽에 있는 매개변수는 기본값을 사용하고 뒤쪽에 있는 매개변수에 인자를 넘겨줄 수 없음

```
def add1(value1, value2=1, value3=2):
    return value1 + value2 + value3

print(add1(42, 1024, 23))
print(add1(42, 1024))
print(add1(42))
```

```
1089
1068
45
```

print() 함수

- print() 함수: 인자들을 출력할 때 인자들 사이에 sep, 가장 마지막에 end를 출력
  print(*objects, sep=' ', end='₩n', file=sys.stdout, flush=False)

```
print(1, 2, 3)
print(4, 5)
print(1, 2, 3, end='')
print(4, 5)
print(1, 2, 3, sep='')
print(1, 2, 3, sep='hey')
print() # blank line
print('After one blank line')
print('₩n') # two blank lines
print('After two blank lines')
```

```
1 2 3
4 5
1 2 34 5
123
1hey2hey3

After one blank line



After two blank lines
```

* https://docs.python.org/3/library/functions.html#print

네임스페이스와 지역 변수

- 네임스페이스 (namespace): 변수, 함수 등이 정의되어 있는 공간
- (단순화시킨) python에서 네임 스페이스의 종류와 그 생성/소멸에 관한 설명
  - 전역 네임스페이스 (global name space): 프로그램이 실행될 때 생성됨
    - 내장 함수와 전역 변수(함수 외부에서 정의된 변수)를 저장
  - 지역 네임스페이스 (local name space): 함수가 호출될 때 생성되고, 함수의 실행이 끝나면 소멸됨
    - 매개 변수와 지역 변수(함수 안에서 정의된 변수)를 저장

변수의 범위 (scope)

- 변수의 범위 (scope): python에서 네임 스페이스, 즉 변수 혹은 함수가 직접 접근가능한 영역

- 변수 혹은 함수가 사용될 때,
  1. 지역 네임스페이스에서 검색
  2. 전역 네임스페이스에서 검색
  3. NameError를 발생

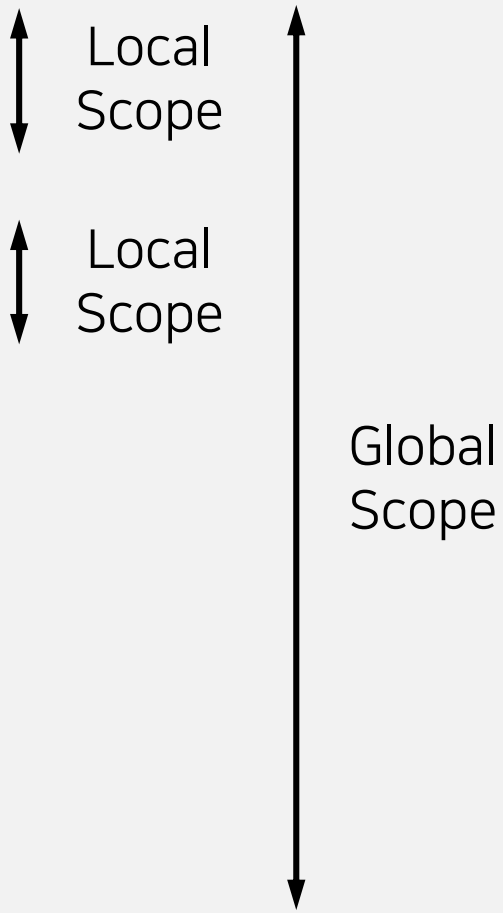
예제: 매개변수의 이름

- 매개 변수의 이름은 함수 내부에서만 의미가 있고,
  함수 외부(함수를 호출하는 쪽)에는 차이를 만들지 않음
- 참고: 아래 두 가지 형태의 함수는 호출하는 쪽에서는 동일한 함수

```
def square(x):
    return x * x

def square(y):
    return y * y

value = square(3)
```

예제: 지역 변수과 변수의 범위(scope)

```
def foo(x):
    x = 28

def bar(y):
    y = 496

x = 6
y = 42
foo(x)
bar(x)
bar(y)
print(x, y)
```




```
6 42
```

리턴문(return statement)와 반환값

- 함수 내부에서 명령문 중 리턴문을 수행하면, 반환값을 돌려주고 함수의 실행을 종료함
  - python이 지원하는 모든 객체가 반환값이 될 수 있음
  - 참고. 1개 이상의 값이 (tuple) 형태로 반환 가능

- 반환값이 없는 것을 나타내기 위해 None[1]을 사용함
  - return문이 없는 경우
  - return문 뒤에 값이 주어지지 않은 경우

1 문자열이 아닌, python 내부의 특수한 값

예시: return문

```
def age_group(age):
    if age <= 12:
        return 'childhood'
    elif age <= 18:
        return 'adolescent'
    elif age <= 65:
        return 'adult'
    else:
        return 'aged'

    # the following line will not be executed
    print('The last line of age_group()')


print(age_group(42))
```

```
adult
```

예시: return문 with tuple

```
def divider(a, b):
    return a//b, a%b

print(divider(10, 3))

q, r = divider(10, 3)
print(q, r)
```

```
(3, 1)
3 1
```

예시: 반환값이 없는 경우

```
def foo():
    pass

def bar():
    return

print(foo())
print(bar())
```

```
None
None
```

* pass 명령문은 아무 일도 하지 않고, 보통 코드블록이 필요한 경우에 placeholder도 사용됨

lambda 함수의 정의와 사용

- lambda 함수
  - 익명 함수로 lambda 키워드를 이용하여 정의
  - 한 줄로 표시되는 함수
  - 비교적 간단한 기능의 함수를 정의할 때 유용하며, 다른 함수의 인자로 함수를 넘겨줄 때 유용
  - filter, reduce 등 함수와 자주 함께 사용됨
- lambda 함수 표현식
  - lambda parameters: expression

```
def square(x):
    x_squared = x * x
    return x_squared

print(square(10))
```

```
square = lambda x: x * x
print(square(10))

print((lambda x: x*x)(10))
```

요약: 함수가 호출될 때, 수행되는 작업

- 인자가 있는 경우, 인자의 값을 먼저 계산함
- 함수를 위한 독립적인 지역 네임스페이스를 생성함
- 매개변수가 있는 경우,
  - 새로 만들어진 네임스페이스에 매개변수를 생성
  - 호출될 때 제공된 인자 혹은 디폴트 인자를 매개변수에 대입
- 함수 내부의 명령문을 실행함
  - 참고: 함수 내부에서는 임의의 명령문이 실행될 수 있음
- 함수의 코드블럭의 끝 혹은 return문이 있는 경우 함수의 실행을 종료함
- 반환값 혹은 None을 함수를 호출한 곳으로 반환함
  - 반환된 값은 호출한 곳, 즉 함수가 표현식에 사용된 곳에서 함수값으로 사용됨
- 지역 네임스페이스를 소멸시킴

ANY QUESTIONS?