파이썬 프로그래밍

제어문과 함수 기초

한국기술교육대학교
온라인평생교육원

▣ 파이썬 함수 기초

1. 함수의 장점 및 함수 사용법

• 함수의 장점
- 함수는 반복적인 코드를 없에 주어 코드의 길이를 짧게 만들어 준다.
- 코드의 유지보수를 쉽게 만들어 준다.

- 연관된 코드들을 여러 번 수행 가능하도록 함수에 엮음

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

print add(3, 4)

print add([1,2,3], [4,5,6])
```

```
7
[1, 2, 3, 4, 5, 6]
```

- 코드의 양이 많을 경우 함수를 사용하여 한번에 호출 가능
- 코드의 길이가 짧아지고 코드의 유지보수가 쉬움
- 함수를 정의하는 방식: def 사용 (define의 약자)
- 함수의 인자도 타입형을 선언해 주지 않음
- a와 b의 타입은 함수를 호출할 때 결정됨
- a 타입 → int, b의 타입 → int
- 인자를 받을 때 타입이 결정되기 때문에 동적인 특징을 가짐
- 7을 리턴 받아서 리턴받은 7의 값을 프린트
- [1, 2, 3] + [4, 5, 6] = [1, 2, 3, 4, 5, 6]

```
c = add(10, 30)
print c
```

```
40
```

- c = add(10, 30) = 40

▣ 파이썬 함수 기초

1. 함수의 장점 및 함수 사용법

•함수 이름에 저장된 레퍼런스를 다른 변수에 할당하여 그 변수를 이용한 함수 호출 가능

```
f = add
print f(4, 5)

print f

print f is add
```

```
9
<function add at 0x10d8916e0>
True
```

- add → 하나의 식별자 (함수 식별자)
- 함수 = 객체 → 모든 변수 = 객체
- 1은 객체, a는 1이라는 객체를 가리키는 포인터(레퍼런스)
- add 변수 → 함수 객체를 가리키고 있는 레퍼런스
- f하고 add가 동일한 함수를 가리키고 있는 식별자
- <f 식별자의 형, 실제 식별자 이름, 객체가 놓여져 있는 메모리 위치>
- is 연산자: 일종의 예약어
- f와 add가 같은 객체인지, 같은 식별자인지 알아보는 키워드
- f is add = true

▣ 파이썬 함수 기초

1. 함수의 장점 및 함수 사용법

•함수의 몸체에는 최소한 한개 이상의 statement가 존재해야 함
그러므로, 아무런 내용이 없는 몸체를 지닌 함수를 만들 때에는 pass 라는 키워드를 몸체에 적어주어야 함

```
def simple():
    pass

print simple()
```

```
None
```

- simple 함수: 인자를 받는 것이 하나도 없음
- 내용을 안 적고 싶어도 반드시 적어야 함
- 비어 있는 경우 pass 예약어 사용
- pass 함수가 리턴하는 값이 없음
- simple 을 호출해서 받아내는 것이 없다. → none 객체 리턴

▣ 파이썬 함수 기초

1. 함수의 장점 및 함수 사용법

•함수에서 다른 함수 호출 가능

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

def myabs(x):
    if x < 0:
        x = -x
    return x

def addabs(a, b):
    c = add(a, b)
    return myabs(c)

print addabs(-5, -7)
```

```
12
```

- myabs는 인자를 x로 받아, x가 0보다 작으면 부호를 바꿔줌
- c = add(-5, -7) = -5 + -7 = -12
- myabs(-12) = 12

▣ 파이썬 함수 기초

1. 함수의 장점 및 함수 사용법

•인자의 이름과 함께 인자 값을 넘겨줄 수 있다.

```
def minus(a, b):
    return a - b
print minus(a=12, b=20)
print minus(b=20, a=12)
```

```
-8
-8
```

- def 함수이름(식별자) 인자 :
- def minus(12, 20)
- 인자의 이름을 적어주면 반드시 a값 먼저 안 적어도 괜찮음

•인자의 디폴트 값을 지정할 수 있다.

```
def incr(x, y=1):
    return x + y

print incr(5)

print incr(5, 10)
```

```
6
15
```

- y=1 → y의 디폴트 값이 1이다.
- 디폴트 값이 정해져 있으면 인자를 하나만 줘도 괜찮음
- 5 + 1 = 6

▣ 파이썬 함수 기초

1. 함수의 장점 및 함수 사용법

•두 개 이상의 값을 동시에 반환할 수 있다.

```
def calc(x, y):
    return x + y, x - y, x * y, x / y

print calc(10, 2)
```

```
(12, 8, 20, 5)
```

- 인자는 2개인데, 총 4개(더하기, 빼기, 곱하기, 나누기한 값)을 리턴
- 여러 개의 값이 콤마의 형태로 묶여있는 자료: tuple(튜플)

▣ 파이썬 함수 기초

2. 함수 호출시 동적인 자료형 결정

•파이썬에서는 모든 객체는 동적으로 (실행시간에) 그 타입이 결정된다.
- 함수 인자는 함수가 호출되는 순간 해당 인자에 전달되는 객체에 따라 그 타입이 결정된다.
- 함수 몸체 내에서 사용되는 여러 가지 연산자들은 함수 호출시에 결정된 객체 타입에 맞게 실행된다.

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

c = add(1, 3.4)
d = add('dynamic', 'typing')
e = add(['list'], ['and', 'list'])
print c
print d
print e
```

```
4.4
dynamictyping
['list', 'and', 'list']
```

- add(a, b) → a는 정수, b는 실수
- 정수 + 실수 = 실수
- 문자열 + 문자열 = 문자열
- 리스트 + 리스트 = 리스트

▣ 파이썬 함수 기초

3. 재귀적 함수 호출

•재귀(Recursive) 함수: 함수 몸체에서 자기 자신을 호출하는 함수
 - 수학에서 점화식과 유사한 코드
 - 반드시 종결 조건 및 종결 조건이 만족할 때의 반환 값이 있어야 한다.
•1부터 N까지 더하는 재귀 함수

```
def sum(N):
    if N == 1:            # 종결 조건
        return 1          # 종결 조건이 만족할 때의 반환 값
    return N + sum(N-1)  # 재귀 호출

print sum(10)
```

```
55
```

- return 10 + sum(10-1)
- return 10 + 9 + 8+ 7+ 6+ 5+ 4+ 3+ 2+ 1 = 55
- if절이 종결되었을 때 (만족되었을 때) return문이 함께 보여야 함