함 수

용어 정리

- 정의(Definition) : 어떤 이름을 가진 코드가 구체적으로 어떻게 동작하는지를 '구체적으로 기술하는 것'.
- 호출(Call) : 함수를 부르는 행위.
- 호출자(Caller) : 함수를 부르는 코드.
- 반환(Return) : 함수가 호출자에게 결과를 돌려주는 것.

함수 정의하기

- 정의 문법

```
def 함수이름(인자1, 인자2 ...) :
    # 코드 블록
    return 반환값
```

함수 정의하기

- 기본값 매개변수(Default Argument Value)

```
def print_string(text, count=1) :
    for i in range(count) :
        print(text)


print_string('안녕하세요')
print_string('안녕하세요', 5)
```

함수 정의하기

- 키워드 매개 변수(Keyword Argument)

```
def print_personnel(name, position='staff', nationality='Korea') :
    print('name = {0}'.format(name))
    print('position = {0}'.format(position))
    print('nationality = {0}'.format(nationality))
```

함수 정의하기

- 가변 매개 변수(Arbitrary Argument List)

```
def 함수이름( *매개변수 ) :
    코드블록
    …
```

예1)

```
def merge_string(*text_list) :
    result = ''
    for s in text_list :
        result += s

    return result

print(merge_string('아버지가 ', '방에 ', '들어가신다.'))
```

함수 정의하기

- 가변 매개 변수(Arbitrary Argument List)

```
def 함수이름( **매개변수 ) :
    코드블록
    …
```

예2)

```
def print_team(**players) :
    for k in players.keys() :
        print('{0} = {1}'.format(k, players[k]))

print_team(선동렬='야구', 안정환='축구', 신진식='배구', 서장훈='농구')
```

함수 정의하기

- 가변 매개 변수(Arbitrary Argument List)

※ 주 의

```
def print_args1(argc, *argv) :
    for i in range(argc) :
        print(argv[i])
print_args1()                    # 호출 시 주의
```

함수 정의하기

- 가변 매개 변수(Arbitrary Argument List)

※ 주 의

```
def print_args2(*argv , argc) :
    for i in range(argc) :
        print(argv[i])
print_args2()                    # 호출 시 주의
```

호출자에게 반환하기

- *함수 즉시 종료/호출자에게 결과 전달*

```
def multiply(a, b) :
    return a * b
result = multiply(3, 5)
print(result)


# 한 함수 안에 여러 개의 return 배치 가능.
def my_abs1(arg) :
    if(arg < 0) :
        return arg * -1
    else:
        return arg

result = my_abs1(-1)
print(result)
```

호출자에게 반환하기

- *함수 즉시 종료/호출자에게 결과 전달*

※ 주의 필요

```
def my_abs2(arg) :
    if(arg < 0) :
        return arg * -1
    elif(arg > 0):
        return arg

result = my_abs2(-1)
print(result)
result = my_abs2(1)
print(result)
result = my_abs2(0)    # return을 실행하지 못하고 함수가 종료되면 함수는 호출자에게 None을 반환.
print(result)
```

호출자에게 반환하기

- *반환 데이터없이 '함수 종료'의 의미로 사용.*

```
def ogamdo(num) :
    for i in range(1, num+1):
        print('제 {0}의 아해'.format(i))
        if i == 5:
            return

ogamdo(3)
ogamdo(5)
ogamdo(8)
```

호출자에게 반환하기

- *반환 결과 없고, 함수 중간에 종료 시킬 일도 없을 때, return문 생략 가능.*

```
def print_something(*args) :
    for s in args:
        print(s)

print_something(1, 2, 3, 4, 5)
```

변수의 유효 범위(Scope)

- **def** scope_test() :
    a = 1 *# 함수 호출시 메모리 생성*
    print('**a:{0}**'.format(a))

  a = 0 *# 함수 밖 변수 선언*
  scope_test() *# 함수 호출시 가까운 내부 변수 참조 - 1*
  print('**a:{0}**'.format(a)) *# 함수 밖 변수 a 값 출력 – 0*

- **def** scope_test() :
    global a *# global 키워드 : 전역 변수*
    a = 1
    print('**a:{0}**'.format(a))

  a = 0
  scope_test()
  print('**a:{0}**'.format(a))

함수를 변수에 담아 사용

- ```
  def print_something(a) :
      print(a)

  p = print_something
  p(123)
  p('abc')
  ```

- ```
  def plus(a, b) :
      return a + b

  def minus(a, b) :
      return a - b

  flist = [plus, minus]
  flist[0](1, 2)          # plus(1, 2)와 동일
  flist[1](1, 2)          # minus(1, 2)와 동일
  ```

중첩 함수

- **import** math

```
def stddev(*args) :
    def mean() :                                # 중첩함수
        return sum(args) / len(args)

    def variance(m) :                           # 중첩함수
        total = 0
        for arg in args:
            total += (arg - m) ** 2
        return total / (len(args) - 1)
    v = variance(mean())
    return math.sqrt(v)

print(stddev(2.3, 1.7, 1.4, 0.7, 1.9))
```