

## [Effective Python..]

Ch18 가변 위치 인수로 깔끔하게 보이게 하자

Ch19 키워드 인수로 선택적인 동작을 제공하자

Ch20 동적 기본 인수를 지정하려면 None과 docstring 을 사용하자

Ch21 키워드 전용 인수로 명료성을 강요하자

## [번외]

PIP 로 Python Package import

1. 이 인수들을 이용하여 코딩가독성을 높이자
2. Python open package 를 import 시켜보자

# 인수들이 참 많다

위치 인수 ( positional argument )  
동적 기본 인수 ( dynamic default argument )  
가변 위치 인수 ( variable argument )  
키워드 인수 ( keyword argument )  
키워드 전용 인수 ( keyword only argument )

Ch18 가변 위치 인수로 깔끔하게 보이게 하자

## #가변인수 란?

```
def log_variable_args(message, *values):
```

#선언

함수 인수명 앞에 ‘\*’ 을 붙인다

#사용

가변인수는 ‘튜플’ 타입이다

# 1. Only Positional Argument

```
def log(message, values):  
    if not values:  
        print(message)  
    else:  
        values_str = ', '.join(str(x) for x in values)  
        print('%s: %s' % (message, values_str))
```

```
log('My numbers are', [1, 2])  
log('My numbers are', [])
```

# 2. Using Variable Argument

```
def log_variable_args(message, *values):  
    if not values:  
        print(message)  
    else:  
        values_str = ', '.join(str(x) for x in values)  
        print('%s: %s' % (message, values_str))
```

```
log_variable_args('My numbers are', 1, 2)  
log_variable_args('Hi there')
```

## 가변인수를 쓰면?

- 빈 리스트를 넘겨 줘야하는 귀찮음과 구림이 없다
- 코드가 이빠진다

```
def my_generator():  
    for i in range(3):  
        yield i
```

```
def my_func(*args):  
    print(args)
```

```
it = my_generator()
```

```
my_func(*it)  # *연산자로 제너레이터의 모든 아이템을 함수의 위치 인수로 건넴  
             # print(*it) 이렇게 쓴다는 것
```

### 가변인수와 시퀀스 \*연산이 만나면?

- 시퀀스의 모든 아이템이 튜플 메모리를 잡기 때문에 메모리가 터질 위험이 있다  
( 사실은 예측하기가 어렵다 )

```
#4.
def log_danger(sequence, message, *values):
    if not values:
        print('%s: %s' % (sequence, message))
    else:
        values_str = ', '.join(str(x) for x in values)
        print('%s: %s: %s' % (sequence, message, values_str))

log_danger(1, 'Favorites', 7, 33)    # 새로운 용법은 OK
log_danger('Favorites', 7, 33)      # 오래된 용법은 제대로 동작하지 않음
```

```
> > 1: Favorites: 7, 33
      Favorites: 7: 33
```

**기존 함수에 가변인수를 추가하면?**

**-기존 코드들이 오동작할 확률이 높다**

## #핵심정리

- Def 문에서 \*args 를 사용하면 가변 개수의 위치 인수를 받을 수 있다
- \* 연산자를 쓰면 시퀀스에 들어 있는 아이템을 함수의 위치 인수로 사용할 수 있다
- 제너레이터와 \* 연산자를 함께 사용하면 프로그램이 메모리 부족으로 망가질 수 있다
- Args 를 받는 함수에 새 위치 파라미터를 추가하면 정말 찾기 어려운 버그가 생길 수 있다



Ch19 키워드 인수로 선택적인 동작을 제공하자

## #키워드 인수 란?

- 함수 호출 / 정의에 인수명을 명시적으로 쓰고, '=' 으로 값이 정해지는 인수

## #키워드 인수 사용 문법?

#위치 인수를 모두 키워드로 전달 할 수 있다

#위치 인수는 키워드 인수 앞에 지정해야한다

#각 인수는 한 번만 지정할 수 있다

```
remainder(20, 7) # OK
```

```
remainder(20, divisor=7) # OK
```

```
remainder(number=20, divisor=7) # OK
```

```
remainder(divisor=7, number=20) # OK
```

```
remainder(number=20, 50) # Syntax Error. Positional argument after keyword argument
```

```
remainder(divisor=7, divisor=5) # Syntax Error. Keyword argument repeated
```

## #키워드 인수의 이점

1. 함수 호출문만 보아도, 함수의 인수가 무엇인지 알 수 있다  
: 선언/정의문을 안가도 된다는 것
2. 함수 정의 시, 디폴트값을 설정할 수 있다  
: 함수 호출 할 때, 타이핑을 덜 칠 수 있다는 것
3. 기존의 함수 호출코드와 호환성을 유지하며,  
함수의 파라미터를 확장시킬 수 있다

## #핵심정리

- 함수의 인수를 위치 또는 키워드로 지정할 수 있다
- 위치 인수만으로는 이해하기 어려울 때 키워드 인수를 쓰면 각 인수를 사용하는 목적이 명확해진다
- 키워드 인수에 기본값을 지정하면 함수에 새 동작을 쉽게 할 수 있다. 특히, 함수를 호출하는 기존 코드가 있을 때 사용하면 좋다
- 선택적인 키워드 인수는 항상 오버워치가 아닌 키워드로 알고 있는게 맞다

Ch20 동적 기본 인수를 지정하려면 None과 docstring을 사용하자

```
#1.  
def log(message, when=datetime.datetime.now()):  
    print('%s: %s' % (when, message))
```

```
log('Hi there!')  
sleep(0.3)  
log('Hi again!')
```

```
> >
```

```
2016-10-28 00:10:02.477348: Hi there!
```

```
2016-10-28 00:10:02.477348: Hi again!
```

<< Log 함수 호출할 때마다, when 에 현재 시간이 대입되지 않는다

키워드 인수의 기본값은 모듈이 로드 될 때 딱 한 번만 평가 된다  
== 키워드 인수의 기본값은 static 하다

```
#2.
def log_none(message, when=None):
    """Log a message with a timestamp.

    Args
        :param message: Message to print
        :param when:     datetime of when the message occurred.
                        Defaults to the present time.

    :return:
    """
    when = datetime.datetime.now() if when is None else when
    print('%s: %s' % (when, message))

log_none('Hi there!')
sleep(0.3)
log_none('Hi again!')
```

-키워드 인수의 값이 기본값일 때 (None 일 때)  
함수 내부에서 키워드 인수 값을 동적으로 세팅

-Docstring 으로 실제 함수 로직을 문서화

```
> > 2016-10-28 00:16:44.041810: Hi there!
      2016-10-28 00:16:44.341985: Hi again!
```

## #핵심정리

- 기본 인수는 모듈 로드 시점에 함수 정의 과정에서 딱 한 번만 평가된다. 그래서 ({} , [] 와 같은) 동적 값에는 오동작의 원인이 되기도 한다
- 값이 동적인 키워드 인수에는 기본값으로 None 을 사용하자. 그리고 나서 함수의 docstring 에 실제 기본 동작을 문서화하자



Ch21 키워드 전용 인수로 명료성을 강요하자



## #키워드 전용 인수를 이용

```
# 3. Only Python3 - keyword-only argument (키워드 전용 인수 - 키워드로만 넘길 수 있고, 위치 인수로 절대 넘길 수 없다)
def safe_division_c(number, divisor, *,
                    ignore_overflow=False,
                    ignore_zero_division=False):
    try:
        return number / divisor
    except OverflowError:
        if ignore_overflow:
            return 0
        else:
            raise
    except ZeroDivisionError:
        if ignore_zero_division:
            return float('inf') # infinity
        else:
            raise

safe_division_c(1, 10**500, True, False) # >>TypeError: safe_division_c() takes 2 positional arguments but 4 were given
safe_division_c(1, 0, ignore_zero_division=True) # Ok

try:
    safe_division_c(1, 0)
except ZeroDivisionError:
    pass
```

\* 기호로, 위치 인수의 끝, 키워드 전용 인수의 시작을 구분 짓는다

## #Python2 에서는

```
# 4. Python2
def print_args(*args, **kwargs):
    print('Positional:', args)
    print('Keyword: ', kwargs)

print_args(1, 2, foo='bar', stuff='meep')

# error 도 hierarchy 가 있네 https://docs.python.org/3/library/exceptions.html
def safe_division_d(number, divisor, **kwargs):
    ignore_overflow = kwargs.pop('ignore_overflow', False)
    ignore_zero_div = kwargs.pop('ignore_zero_div', False)
    if kwargs:
        raise TypeError('Unexpected **kwargs: %r' % kwargs)
    try:
        return number / divisor
    except OverflowError:
        if ignore_overflow:
            return 0
        else:
            raise
    except ZeroDivisionError:
        if ignore_zero_div:
            return float('inf') # infinity
        else:
            raise

safe_division_d(1, 10) # Ok
safe_division_d(1, 0, ignore_zero_div=True) # Ok
safe_division_d(1, 10**500, ignore_overflow=True) # Ok

safe_division_d(1, 0, False, True) # TypeError: safe_division_d() takes 2 positional arguments but 4 were given
safe_division_d(0, 0, unexpected=True) # TypeError: Unexpected **kwargs: {'unexpected': True}
```

# \*\* 연산자로 키워드 인수만 받게 한다

# \*\* 연산자로 받은 키워드 인수들은 Dictionary 타입 이다

## #핵심정리

- 키워드 인수는 함수 호출의 의도를 더 명확하게 해준다
- 특히 불 플래그를 여러 개 받는 함수를 호출할 때 키워드 인수를 넘기게 하려면 키워드 전용 인수를 사용하자
- 파이썬 3은 함수의 키워드 전용 인수 문법을 명시적으로 찾아보려고해
- 파이썬 2에서는 kwargs 를 사용하고 TypeError 예외를 직접 일으키는 방법으로 함수의 키워드 전용 인수를 흉내낼 수 있다

PIP 와 Python Open Package Import 시키기

# PIP ?

파이썬으로 작성된 패키지 소프트웨어를설치/관리하는 패키지 관리 시스템

# PIP ?

파이썬 2.79 이후 버전, 파이썬 3.4 이후 버전은 pip 가 기본 포함되어 있다



# PIP ?

cmd>> pip install -U python\_package\_index

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\WatMERRY PARK>pip install -U memory_profiler
```

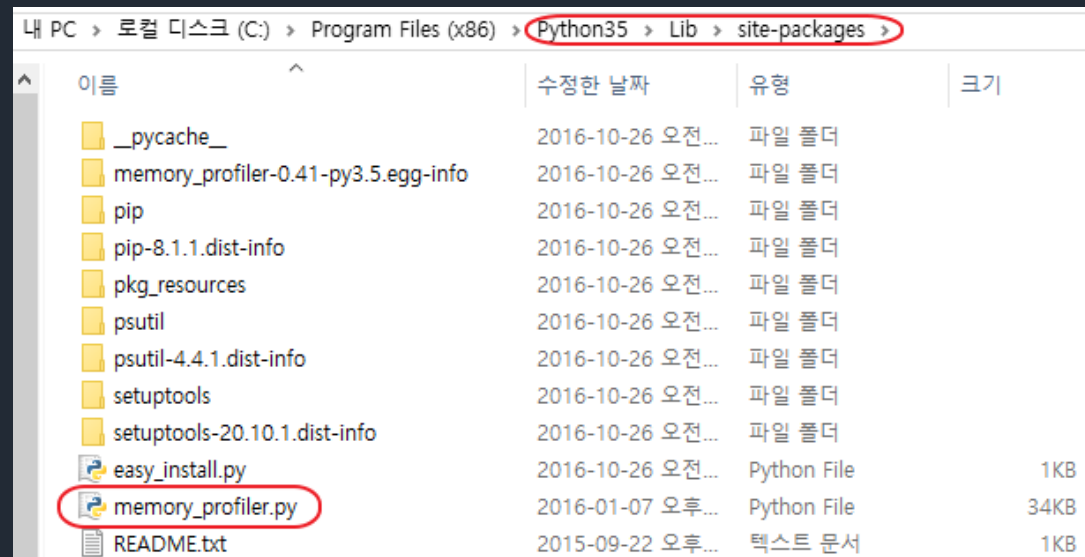
# pip install ex)

```
C:\> 선택 관리자: 명령 프롬프트

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>pip install -U memory_profiler
Collecting memory-profiler
  Using cached memory_profiler-0.41.tar.gz
Installing collected packages: memory-profiler
  Running setup.py install for memory-profiler
Successfully installed memory-profiler-0.41
You are using pip version 7.1.2, however version 8.1.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

모듈이 설치되는 경로 : Python 설치경로\Lib\site-packages



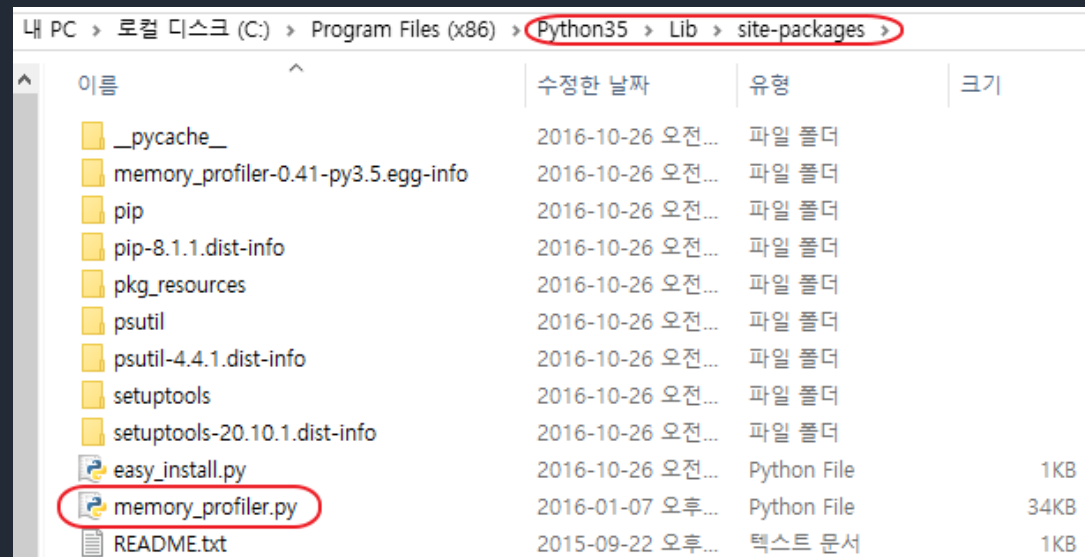
# pip install ex)

```
C:\> 선택 관리자: 명령 프롬프트

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.


C:\WINDOWS\system32>pip install -U memory_profiler
Collecting memory-profiler
  Using cached memory_profiler-0.41.tar.gz
Installing collected packages: memory-profiler
  Running setup.py install for memory-profiler
Successfully installed memory-profiler-0.41
You are using pip version 7.1.2, however version 8.1.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

모듈이 설치되는 경로 : Python 설치경로\Lib\site-packages



# pip install ex)

사용 중인 파이썬 인터프리터에서 사용가능 한 모듈을 확인해보면,  
memory-profiler 가 있는 것을 확인할 수 있다

Project Interpreter:  3.5.2 (C:\Program Files (x86)\Python35\python.exe)		
Package	Version	Latest
memory-profiler	0.41	
pip	8.1.1	➡ 8.1.2
psutil	4.4.1	➡ 4.4.2
setuptools	20.10.1	➡ 28.6.1

# pip install ex)

```
from memory_profiler import profile

@profile
def my_func():
    a = [1] * (10 ** 6)
    b = [2] * (2 * 10 ** 7)
    del b
    return a

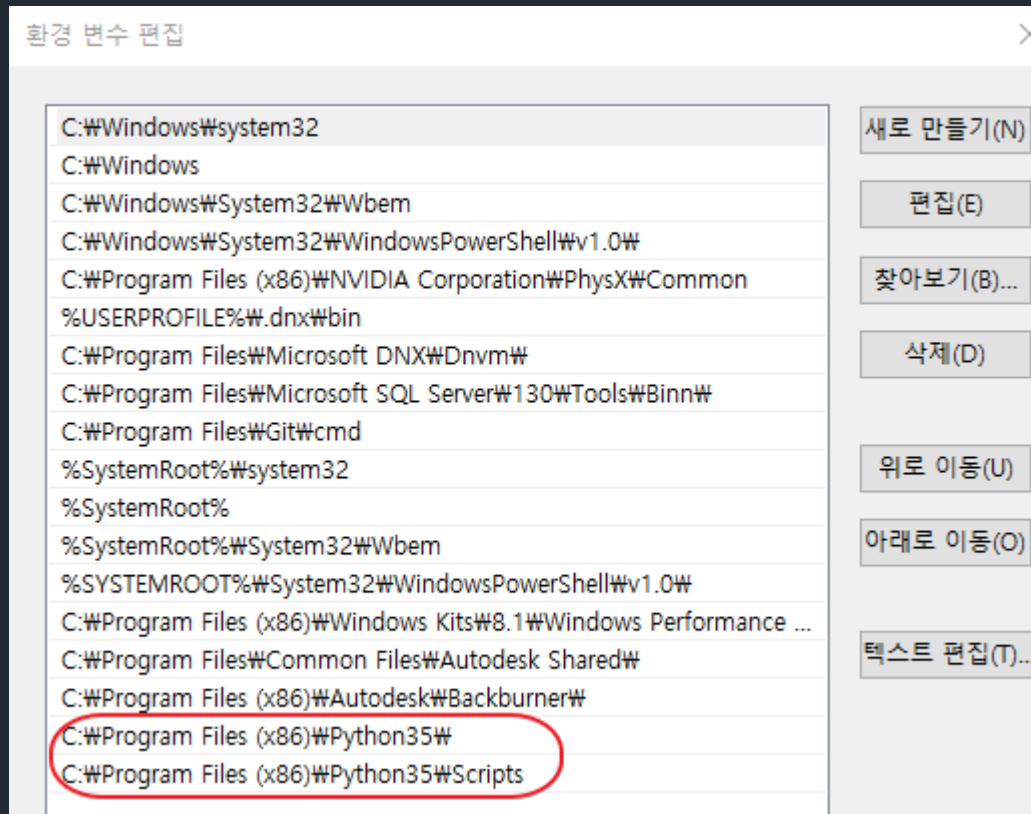
if __name__ == '__main__':
    my_func()
```

```
> >
=====
Line #   Mem usage   Increment   Line Contents
=====
5        13.4 MiB      0.0 MiB      @profile
6                                     def my_func():
7        17.2 MiB      3.8 MiB          a = [1] * (10 ** 6)
8        93.5 MiB     76.3 MiB          b = [2] * (2 * 10 ** 7)
9        17.2 MiB    -76.3 MiB          del b
10       17.2 MiB      0.0 MiB          return a
```

## 사전 세팅

1. 환경 변수 설정
2. Python OpenPackage 물색
3. Case-By-Case 로 설치

## 1. 환경 변수 설정



## 2. Python Open Package 물색

# 알잘. 찾자

### Installation

To install through easy\_install or pip:

```
$ easy_install -U memory_profiler # pip install -U memory_profiler
```

To install from source, download the package, extract and type:

```
$ python setup.py install
```



### 3. Case-By-Case

```
C:\Users\WATMERRY\>pip install -U memory_profiler
Collecting memory_profiler
  Downloading memory_profiler-0.41.tar.gz
Installing collected packages: memory_profiler
  Running setup.py install for memory_profiler
    Complete output from command "c:\program files (x86)\python34\python.exe" -c "import setuptools, tokenize;__file__='
C:\Users\WATMERRY\AppData\Local\Temp\pip-build-j4ngu4mo\memory_profiler\setup.py';exec(compile(getattr(tokenize,
'open', open)(__file__).read().replace('\r\n', '\n'), __file__, 'exec'))" install --record C:\Users\WATMERRY\AppData\Local\Temp\pip-9ms6h020-record\install-record.txt --single-version-externally-managed --compile:
    running install
    running build
    running build_py
    creating build
    creating build\lib
    copying memory_profiler.py -> build\lib
    running build_scripts
    creating build\scripts-3.4
    copying and adjusting mprof -> build\scripts-3.4
    running install_lib
    copying build\lib\memory_profiler.py -> c:\program files (x86)\python34\lib\site-packages
    error: could not create 'c:\program files (x86)\python34\lib\site-packages\memory_profiler.py': Permission denied
```

```
-----
Command "'c:\program files (x86)\python34\python.exe" -c "import setuptools, tokenize;__file__='C:\Users\WATMERRY\AppData\Local\Temp\pip-build-j4ngu4mo\memory_profiler\setup.py';exec(compile(getattr(tokenize, 'open', open)(__file__).read().replace('\r\n', '\n'), __file__, 'exec'))" install --record C:\Users\WATMERRY\AppData\Local\Temp\pip-9ms6h020-record\install-record.txt --single-version-externally-managed --compile" failed with error code 1 in C:\Users\WATMERRY\AppData\Local\Temp\pip-build-j4ngu4mo\memory_profiler
You are using pip version 7.1.2, however version 8.1.2 is available.
You should consider upgrading via the 'python -m pip --upgrade' command
```

```
C:\Users\WATMERRY\>
```

```
File "C:\Program Files (x86)\Python34\lib\site-packages\memory_profiler.py", line 110, in _get_memory
    raise NotImplementedError('The psutil module is required for non-unix '
NotImplementedError: The psutil module is required for non-unix platforms
```