- 함수가 여러값을 반환하는 경우 절대로 네 개 이상은 언패킹하지 말라

이유 1: 반환값이 많으면 실수하기도 쉬워 귀찮다

이유 2: 가독성이 나빠진다.

- None을 반환하기보다는 예외를 발생시켜라

- 변수 영역과 클로저의 상호작용 방식을 이해하라.

  - 파이썬 closure 지원: closure는 자신을 둘러싼 scope의 상태값을 기억하는 함수.

  - 파이썬에서 함수가 first-class citizen(일등시민) 객체 : first-class citizen 객체는 다음 3가지 조건을 만족한다. 1. 변수나 데이터에 할당될 수 있어야 한다. 2. 객체의 인자로 넘길 수 있어야 한다.

  3. 객체의 리턴값으로 리턴 될 수 있어야 한다.

```
def sort_priority(values, group):
    def helper(x):
        if x in group:
            return (0, x)
        return (1, x)
    values.sort(key=helper)


numbers = [8, 3, 1, 2, 5, 4, 7, 6]
group = {2, 3, 5, 7}
sort_priority(numbers, group)
print(numbers)
```

```
[2, 3, 5, 7, 1, 4, 6, 8]
```

- 문제점

```
def sort_priority2(numbers, group):
    found = False
    def helper(x):
        if x in group:
            found = True # 문제를 쉽게 해결할 수 있을 것 같다
            return (0, x)
        return (1, x)
    numbers.sort(key=helper)
    return found

found = sort_priority2(numbers, group)
print('발견:', found)
print(numbers)
```

```
발견: False
[2, 3, 5, 7, 1, 4, 6, 8]
```

→ True가 반환되길 의도했지만, False가 반환됨 ⇒ helper 함수의 클로저 안에서 이 대입문은 helper 영역안에 새로운 변수를 정의하는 것으로 취급되며, sort_priority2 안에서 같은 변수에 값을 대입하는 것으로 취급되지 않.

⇒ 해결책: nonlocal 문. 클로저 밖으로 데이터를 꺼내는 구문. nonlocal은 전역 영역을 더럽혀서 오염되지 않도록 주변 영역까지 데이터를 끌어내지는 않는다. 이는 변수 대입 시 직접 모듈 영역(전역 영역)을 사용해야 한다고 지정하는 global 문을 보완한다.

```
def sort_priority2(numbers, group):
    found = False
    def helper(x):
        nonlocal found       # 추가함
        if x in group:
            found = True
            return (0, x)
        return (1, x)
    numbers.sort(key=helper)
    return found

found = sort_priority2(numbers, group)
print('발견:', found)
print(numbers)
```

```
발견: True
[2, 3, 5, 7, 1, 4, 6, 8]
```

→ nonlocal을 사용하는 방식이 복잡해지면, 도우미 함수로 상태를 감싸는 것이 가독성이 더 좋다.

```
class Sorter:
    def __init__(self, group):
        self.group = group
        self.found = False

    def __call__(self, x):
        if x in self.group:
            self.found = True
            return (0, x)
        return (1, x)

sorter = Sorter(group)
numbers.sort(key=sorter)
print('발견', sorter.found)
print(numbers)
```

```
발견 True
[2, 3, 5, 7, 1, 4, 6, 8]
```

- 변수 위치 인자를 사용해 시각적인 잡음을 줄여라.

  ↳ 변수 개수를 가변적으로 받을 수 있음. 가변위치(*args) 나 스타 인자 (star args) 라고 부르기도 함. 하지만 가변인자에 많은 값들이 들어가게 되면 이로 인해 메모리가 아주 많이 소비되어 프로그램이 중단될 수 있다.

```
def log(message, *values):
    print(f"들어온 values 값은 : {values}")
    if not values:
        print(message)
    else:
        values_str = ', '.join(str(x) for x in values)
        print(f'{message}: {values_str}')

numbers = [1, 2]
log('내 숫자는 ', numbers)
log('안녕')
```

```
들어온 values 값은 : ([1, 2],)
내 숫자는 : [1, 2]
들어온 values 값은 : ()
안녕
```

```
def log(message, *values):
    print(f"들어온 values 값은 : {values}")
    if not values:
        print(message)
    else:
        values_str = ', '.join(str(x) for x in values)
        print(f'{message}: {values_str}')

numbers = [1, 2]
log('내 숫자는 ', *numbers)
log('안녕')
```

```
들어온 values 값은 : (1, 2)
내 숫자는 : 1, 2
들어온 values 값은 : ()
안녕
```

*변수로 시퀀스의 원소를 함수의 가변 인자로 넘겨준 것과 동일하다.