# 고급통계프로그래밍 Solution 6

## 컴퓨터과학부 2017920054 이호준

과제 index는 번역본 기준입니다.

#### # Ex 12.1

#### Solution

```
def sumall(*args):
    result = 0
    for elem in args:
        result += elem
    return result
```

- \*args 를 이용해서 가변 인자를 입력받는다. (입력받은 인자들은 하나의 튜플의 원소로 각각 들어가게 된다.)
- for문을 돌면서 부분합 result에 값을 갱신하는 방법을 이용해 튜플의 전체합을 구한다.
- 반복이 종료되면 최종 갱신된 result를 반환한다.

#### Result

```
# Test
print(sumall(1, 2, 3))
print(sumall(1, 4, 7, 10))
print(sumall(25, 75, 100, 200))
# in Stdout:
6
22
400
```

## #Ex 12.2

## Solution

```
import random

def sort_by_random(words):
    t = []
    for word in words:
        t.append((len(word), word))

    t.sort(key=lambda x: (-x[0], random.random()))

res = []
    for _, word in t:
        res.append(word)
    return res
```

- random.random을 사용하기 위해 random 모듈을 호출한다.
- 기존의 로직과 유사하게 리스트 t를 만든 후, 길이와 해당 문자열을 넣어준다.
- t.sort()의 key 인자로 lambda 함수를 넣어주었다.
  - 첫 번째 기준은 x[0], 즉 len(word)를 기준으로 하되, 앞에 -를 붙여 음수의 오름차순, 즉 길이의 내림차순으로 우선적으로 정렬을 진행하도록 했다.
  - 두 번째 기준은 random.random()으로, [0, 1) 범위의 float 값을 반환한다. 이를 기준으로 정렬한다.
  - 정리하면 길이의 내림차순으로 정리하되, 이것이 같으면 임의의 순서대로 정렬이 이루어진다.
- res 리스트를 만들고, 정렬된 t에서 word만을 추출하고, 이를 반환한다.

#### Result

```
# Test - 각 함수를 3번씩 호출
print(sort_by_random(("time", "series", "data")))
print(sort_by_random(("time", "series", "data")))
print(sort_by_random(("time", "series", "data")))
print(sort_by_random(("apple", "banana", "carrot")))
print(sort_by_random(("apple", "banana", "carrot")))
print(sort_by_random(("apple", "banana", "carrot")))

# in Stdout:
['series', 'data', 'time']
['series', 'time', 'data']
['series', 'time', 'data']
['banana', 'carrot', 'apple']
['banana', 'carrot', 'apple']
['carrot', 'banana', 'apple']
```

## # Ex 12.3

#### Solution

```
def most_frequent(string):
    my_dict = {}

for elem in string:
    my_dict[elem] = my_dict.get(elem, 0) + 1

freq_set = [(i, my_dict[i]) for i in my_dict]
    freq_set.sort(key=lambda x : -x[i])

print(freq_set)
```

- 빈 딕셔너리 my\_dict를 선언한다.
- 이전 과제에 있었던 로직을 바탕으로 string에 있는 elem의 개수를 세어준다.
- freq\_set 리스트를 선언하고, List Comprehension의 방식으로 my\_dict를 순회하면서 (key, value)가 원소인 리스트를 만들어 준다.
- value(== key의 빈도)를 기준으로 내림차순 정렬을 진행하고, 이를 출력한다.

## Result

```
# Test
most_frequent("Lorem ipsum dolor sit amet, consectetur adipiscing elit.")
most_frequent("life is short, you need python")

# in Stdout:
[(' ', 7), ('i', 6), ('e', 5), ('t', 5), ('o', 4), ('s', 4), ('r', 3), ('m', 3), ('c', 3), ('p', 2), ('u', 2), ('d', 2), ('l', [(' ', 5), ('e', 3), ('o', 3), ('i', 2), ('s', 2), ('h', 2), ('t', 2), ('y', 2), ('n', 2), ('l', 1), ('f', 1), ('r', 1), (',', 1), ('r', 1), (',', 1), ('r', 1),
```

#### # Ex 14.2

### Solution

```
def sed(src, dst, fin_dir, fout_dir):
    try:
        fin = open(fin_dir)
        fout = open(fout_dir, 'w')
        for line in fin:
            fout.write(line.replace(src, dst))
        fin.close()
        fout.close()
    except:
        print("An Exception Occured")
```

- try 를 통해 다음 명령을 시도한다:
  - input을 담당하는 파일 객체 fin과 output을 담당하는 파일 객체 fout을 선언한다. fout의 경우에는 파일에 값을 써야하므로 (write), 인자로 'w'를 전달한다.

- fin의 line을 순회하면서, 각 line의 src를 dst로 대체하는 .replace() 함수를 적용한 문자열을 fout에 write한다.
- 반복이 끝나면 fin을 닫는다
- except 가 일어나는 경우 "An Exception Occured"를 출력한다.

## Result

```
# ./in file
I love python
python is great!
Do you know python?

# in ex14-2.py
sed("python", "C++", "in", "out")

# ./out file
I love C++
C++ is great!
Do you know C++?
```

## # Exercise 05

#### Solution

```
def get_abs():
    try:
        ipt = int(input())
        print(abs(ipt))
    except ValueError:
        print("Your input is not integer")
```

- try-except문을 이용해서 구현하였다.
- 숫자 입력을 받고, 이를 절댓값 함수를 이용해서 출력을 진행한다
- 만약 ValueError가 발생한 경우, "Your input is not integer" 를 출력한다.

#### Result

```
# Test
10
-15
statistics
# in Stdout:
10
15
Your input is not integer
```