Lab3. Lists II and Functions I

CSED101 LAB

들어가기 전

- add, sub, mul, div 함수 정의해 보기
  - add: 더하기, sub: 빼기, mul: 곱하기, div: 나누기

```
# 함수 정의
...


a = 1
b = 2

c = add(a, b)
d = sub(a, b)
e = mul(a, b)
f = div(a, b)

print(c, d, e, f)    # 3 -1 2 0.5
```

리스트 2

리스트 더하기(+) 및 반복(*)

- + 연산자
  - 자료형이 동일한 두 시퀀스 자료를 연결하여 새로운 시퀀스 자료를 생성
  - 리스트 사이에 사용하여 합치는 기능을 함

※ 시퀀스(Sequence) 자료형
- 내부 원소들이 순서를 가지고 구성된 자료형
- 리스트, 문자열, 튜플

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

- * 연산자
  - 동일한 시퀀스 자료를 여러 번 반복하여 새로운 시퀀스 자료를 생성

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

멤버확인 - 리스트

- 'in'키워드를 사용하여 특정 값이 시퀀스 자료의 요소로 속해 있는지 확인

<값> in <자료>

```
>>> color =  ['red', 'green', 'blue']
>>> 'red' in color
True

>>> 'orange' in color
False
```

리스트 관련 함수

- 리스트 생성

```
>>> color = list()   # 또는 color = []
```

- 리스트이름.append(x)

: 기존 리스트 끝에 x 삽입

```
>>> color
[]
>>> color.append("red")
>>> color.append("green")
>>> color.append("blue")
>>> print(color)
['red', 'green', 'blue']
```

- 리스트이름.remove(x)

: 리스트에서 첫 번째로 나오는 x를 제거

```
>>> color.append("red") # 항목 추가
>>> print(color)
['red', 'green', 'blue', 'red']

>>> color.remove("red")
>>> print(color)
???
```

리스트 관련 함수

- 리스트이름.sort()

  : 리스트의 항목을 오름차순으로 정렬

```
>>> a = [1, 7, 9, 2, 5]
>>> a.sort()
>>> a
[1, 2, 5, 7, 9]

>>> f = ['banana', 'apple', 'orange']
>>> f.sort()
>>> f
['apple', 'banana', 'orange']

>>> a.sort(reverse=True) # 내림차순
```

- sorted(리스트 이름)
  - sorted()는 리턴 값이 있고,
    sort()는 리턴 값이 없다.

```
>>> a = [1, 7, 9, 2, 5]
>>> a1 = a.sort()
>>> print(a1)
None

>>> a = [1, 7, 9, 2, 5]
>>> a2 = sorted(a)
>>> print(a2)
[1, 2, 5, 7, 9]

>>> print(a)
[1, 7, 9, 2, 5]
```

리스트 관련 함수

- 리스트의 길이, 최대값, 최소값, 총합

```
>>> a = [45, 87, 99, 21, 55]
>>> len(a)
5
>>> max(a)
99
>>> min(a)
21
>>> sum(a)
307
```

함수 1

함수

- 사용 이유
  - 코드 재사용

    중복되는 부분을 한 번만 작성. 코드를 간결하게 만들어 줌

  - 절차적 분해

    한 번에 큰 처리 절차를 구현하는 것보다 작은 작업을 구현하는 것이 더 쉬움

- 이미 Python에는 많은 함수들이 존재
  - print(), input(), int(), sorted(), map(), …

함수

일반적 구조

```
def 함수이름(매개변수) :
    수행할 문장
    …
    return 반환값
```

좋은 함수는 "매개변수를 가지고 함수이름의 작업을 해서 반환값을 주는" 함수로 설명이 가능함

- 매개변수: 없을 수 있음
- 반환값: 없을 수 있음 (None)

함수 정의 및 호출

```
#함수 정의
def add(a, b) :
    result = a + b
    return result

#함수 호출
add_result = add(100, 200)
print(add_result)
```

※ a,b가 주어질 때, a와 b를 더한 결과를 반환하는 함수

```
def add3add5() :
    return 8
```

```
def print_this(s) :
    print(s)
```

함수

- 튜플이나 리스트를 이용해 여러 개의 값을 반환할 수 있음

```
# 함수 정의
def add_and_mul(a, b):
    return (a+b, a*b)


# 함수 호출
add_res, mul_res = add_and_mul(3, 4)
print(add_res)
print(mul_res)
```

실습 1

- 초를 전달받아 시간, 분, 초를 반환하는 함수를 작성하시오.

```
def sec_to_hms(seconds):
    …
    …


hr, min_, sec = sec_to_hms(57894)
print("%d시간 %d분 %d초" % (hr, min_, sec))
```

```
16시간 4분 54초
```

실습 2

- 숫자로 구성된 리스트 2개를 받아서 이 두 리스트를 합친 후에 정렬한 새로운 리스트를 반환하는 함수를 작성하시오.

```
def merge_list(l1, l2):
    …
    …
```

```
l = [3, 5, 9, 1, 2]
ml1 = merge_list(l, [2, 1])
ml2 = merge_list([6, 9, 4], l)

print(ml1) # [1, 1, 2, 2, 3, 5, 9]
print(ml2) # [1, 2, 3, 4, 5, 6, 9, 9]
```

지역변수, 전역변수

- 지역변수 (Local Variable)
  : 함수 안에서 만들어져
  함수 안에서만 사용하는 변수

```
def add_value(a, b) :
    add_res = a + b
    return add_res

x = 10
y = 20
res = add_value(x, y)
print(add_res)  # 에러 발생
```

- 전역변수 (Global Variable)
  : 함수 밖에서 만들어져
  아무데서나 사용할 수 있는 변수
  단, 함수 안에서 전역변수 수정 불가

```
def addX(_x) :
    return x + _x

x = 10
y = 20

res = addX(y)
print(res)  # ??
```

지역변수, 전역변수

- 함수 안에서 전역변수 수정 불가, 수정하려면 global 키워드 사용

```
def addX(_x) :
    x = x + _x  # 에러 발생


x = 10
y = 20

addX(y)
print(x)
```

```
def addX(_x) :
    global x  # 전역변수임을 명시
    x = x + _x

x = 10
y = 20

addX(y)
print(x)
```

실습 3

- 숫자로 구성된 리스트를 전달받아 최댓값, 최솟값을 반환하고, 두 수를 리스트에서 제거하는 함수를 작성하시오.

```
def get_min_max(l):


```

```
nlist = [3, 5, 9, 1, 2]
min_val, max_val = get_min_max(nlist)
print(min_val)
print(max_val)
print(nlist)
```

```
1
9
[3, 5, 2]
```

- (가정)
  - 리스트의 원소는 2개 이상
  - 중복된 원소는 없음
- (힌트)
  - 리스트 관련 함수(6~8쪽) 사용 가능

랜덤(random) 모듈

Ex] 난수(Random Number) 생성

- 난수는 random 모듈을 이용하여 생성

```
>>> import random
```

- 리스트에서 랜덤하게 선택

```
>>> random.choice([1,2,3,4,5])
```

- a이상 b이하인 임의의 정수 뽑기 (a, b: int 타입)

```
>>> random.randint(a, b)          # a~b 정수 값 중 하나
>>> random.randrange(a, b+1) # randint(a, b)와 동일한 기능
```

Ex] 난수(Random Number) 생성

- 리스트 내용 랜덤으로 섞기

```
>>> Ls = [1,2,3,4,5]
>>> random.shuffle(Ls)
>>> print(Ls)
```

- 리스트 중 랜덤으로 여러 개 뽑기

```
>>> random.sample([1, 2, 3, 4, 5, 6], 3)
```

실습 4

- 점심 메뉴 리스트에서 임의로 하나의 메뉴를 골라 정해주는 프로그램을 완성하시오.

실행 예제1)

```
오늘 점심은 부대찌개입니다.
```

실행 예제2)

```
오늘 점심은 파스타입니다.
```

```
# 채우기

menu = ["된장찌개", "부대찌개", "김치찌개", "삼계탕", "파스타", "돈까스", "쌀국수"]

# 채우기
```