

파이썬 프로그래밍

강의 노트 #08

자료구조 1
리스트(list), 튜플(tuple)

리스트 (List)

- 리스트는 순서가 있는 요소들의(Sequential Items) 모음
 - 변경 가능한 자료구조
 - 크기를 동적으로(run-time) 조절 가능
 - 요소를 치환해서 변환 가능
 - 인덱싱을 이용해서 요소를 참조 가능
 - 문자열과 마찬가지로 연결하기, 슬라이싱(slicing), 반복하기, 멤버 검사, 길이 정보 등과 같은 다양한 연산 지원
 - 리스트는 [] 형태로 표현

```
[ ]      # 빈 리스트(empty list)
[1, 2, 3, "abc"] # 요소가 4개인 리스트
[1, 2, 3, [1, 2]] # 요소에 리스트가 포함됨
```

리스트 생성

- 빈 리스트 생성 후 요소 추가
- 빈 리스트 생성

```
변수_이름 = []  
변수_이름 = list()
```

- 리스트 요소 추가

- `append()` 함수는 리스트의 마지막에 새로운 요소들을 추가

```
a = []  
a.append(4)          # [4]  
a.append('5')        # [4, '5']  
a.append([2, 3])     # [4, '5', [2, 3]]
```

리스트 (List)

▣ 리스트의 요소 개수 확인하기

■ len() 함수 사용

```
e = []      # empty list
l1 = [1, 2, 3, "abc"] # 요소가 4개인 리스트
l2 = [1, 2, 3, [1, 2]] # 요소에 리스트가 포함됨
len([])     # 0
len(e)
len(l1)
len(l2)
len([1, 2, 3, "abc"])
len([1, 2, 3, [1, 2]])
```

리스트 (List)

□ 인덱싱

- 특정(단일) 요소에 접근(access)해서 값을 읽거나 변경하는 방법
- [인덱스]를 사용해서 요소에 접근

[0] [1] ... [3]

1	2	3	abc
---	---	---	-----

[-4] [-3] ... [-1]

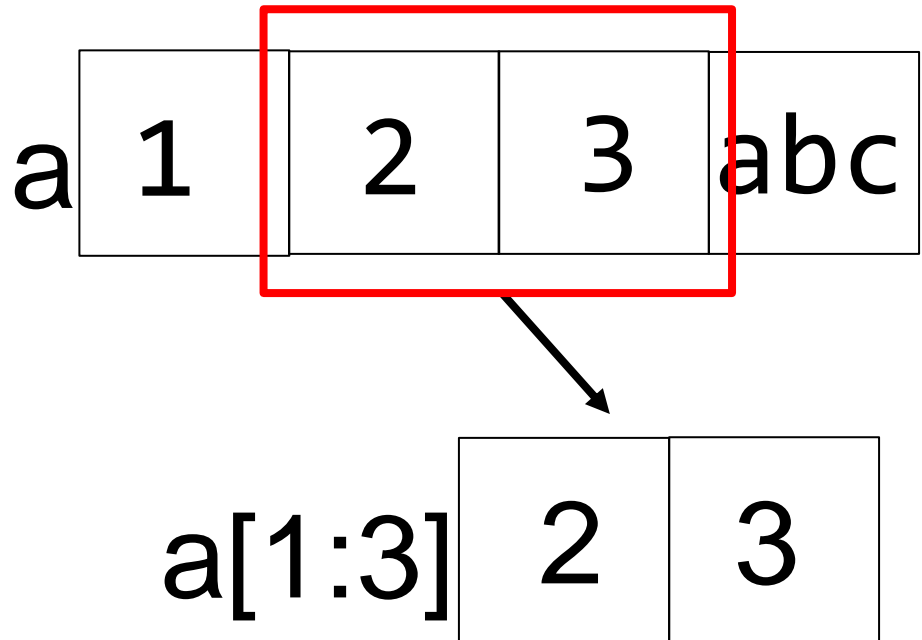
```
l = [1, 2, 3, "abc"] # 리스트 생성
print(l[0], l[1], l[2], l[3]) # 순서대로 출력
print(l[-1], l[-2], l[-3], l[-4]) # 거꾸로 출력
```

리스트 (List)

□ 슬라이싱(slicing)

- 여러 개 요소에 접근하는 방법
- **[시작인덱스:끝인덱스]**를 사용해서 요소들에 접근
- 시작인덱스에 해당되는 요소부터 끝인덱스 - 1 까지의 요소들을 포함하는 리스트 참조

```
a = [1, 2, 3, "abc"]  
a[1:3]
```



리스트 (List)

□ 슬라이싱(slicing)

```
lst1 = [1, 2, 3, "abc", 4, 5]
lst1[2:4]      # [3, "abc"]
lst1[:3]       # [1, 2, 3]
lst1[2:]       # [3, "abc", 4, 5]
lst1[:]        # [1, 2, 3, "abc", 4, 5]
```

□ 확장 슬라이싱

- [시작인덱스:끝인덱스:스텝]
- 시작인덱스부터 끝인덱스 - 1까지 스텝만큼 건너뛰면서 리스트 참조

```
lst1 = [1, 2, 3, "abc", 4, 5]
lst1[2:5:2]     # [3, 4]
lst1[2:6:2]     # [3, 4]
lst1[:3:2]      # [1, 3]
lst1[2::2]      # [3, 4]
```

리스트 (List)

□ 연결하기

- + 연산자를 이용해서 두 개 이상의 리스트를 연결해서 새로운 리스트를 생성할 수 있음

```
a = [1, 2, 3]
b = ["abc", 4, 5]
c = a + b # 두 개의 리스트를 붙여서 c에 저장
print(c)  # [1, 2, 3, "abc", 4, 5]
```

□ 반복하기

- * 연산자를 이용해서 요소들이 반복되는 리스트 생성

```
a = [1, 2]
b = a * 4
print(b)          # [1, 2, 1, 2, 1, 2, 1, 2]
```


리스트 (List)

□ 리스트 요소 수정

■ 특정 요소의 값을 변경

□ 리스트이름[인덱스] = 새로운 값

```
a = [1, 2, 3, "abc"]  
a[2] = 4          # [1, 2, 4, "abc"]  
a[3] = ["ab", 'a', "def"] # [1, 2, 4, ["ab", 'a', "def"]]
```

■ 여러 개 요소를 새로운 값으로 치환하고자 할 때에는 아래와 같이 슬라이싱 방법을 사용해서 수정

■ 리스트이름[시작인덱스:끝인덱스] = 새로운 값

```
a = [1, 2, 3, "abc"]  
a[1:2] = ['g', 'h'] # [1, 'g', 'h', 3, "abc"]  
a[2:4] = ["ab", 'a', "def"]  
# [1, 'g', "ab", 'a', "def", "abc"]
```

리스트 (List)

□ 리스트 요소 삭제

- del 명령을 이용해서 특정 요소 삭제
- del 삭제하고 싶은 요소 또는 del(삭제하고 싶은 요소)

```
a = [1, 2, 3, "abc", 'g', 'h']  
del a[1] # a[1:2] = [], [1, 3, "abc", 'g', 'h']
```

- 범위 삭제
- **리스트이름[시작인덱스:끝인덱스] = []**

```
a = [1, 2, 3, "abc", 'g', 'h']  
a[1:2] = [] # [1, 3, "abc", 'g', 'h']  
a[2:4] = [] # [1, 3, 'h']  
del(a[0:2]) # ['h']
```

리스트 (List)

□ 리스트 요소 제거

- 리스트의 요소를 제거하려면 remove(x) 활용
- 리스트에서 첫 번째로 나오는 x 제거

```
a = [1, 3, 5, 7, 9, 7, 5, 3, 1]
a.remove(5)      # [1, 3, 7, 9, 7, 5, 3, 1]
a.remove(3)      # [1, 7, 9, 7, 5, 3, 1]
a.remove(3)      # [1, 7, 9, 7, 5, 1]
```

□ 리스트에 포함된 요소 x의 개수 세기

- count(x) 함수 활용
- 리스트에서 x의
개수 반환

```
a = [1, 5, 7, 5, 7, 5, 3, 1]
a.count(5)       # 3
a.count(3)       # 1
a.count(7)       # 2
```

리스트 (List)

□ 요소의 존재 여부 확인하기

■ in 사용

```
a = [4, 8, 7, 2, 1]
8 in a           # True
9 in a           # False
```

□ 인덱스 확인하기

```
a = [4, 8, 7, 2, 1]
a.index(8)        # 1
a.index(1)        # 4
```

리스트 (List)

□ 리스트에 요소 삽입

- 요소 삽입은 `insert(x, y)` 함수 활용. `x` index 위치에 `y` 삽입

```
a = [4, 8, 7, 2, 1]
a.insert(2, 5)  # [4, 8, 5, 7, 2, 1]
a.insert(1, [1, 6])
# [4, [1, 6], 8, 5, 7, 2, 1]
```

리스트 (List)

□ 리스트 요소 정렬(sort)

■ sort()함수 활용

```
a = [4, 8, 7, 2, 1]
a.sort()           # [1, 2, 4, 7, 8]
a.append('a')      # [1, 2, 4, 7, 8, 'a']
a.sort()           # Error 발생: 문자열과 정수를 비교함
```

```
b = ['g', 'o', 'o', 'd']
b.sort()           # ['d', 'g', 'o', 'o']
```

□ 리스트 요소 뒤집기

■ reverse()함수 활용

```
a = [4, 8, 7, 2, 1]
a.reverse()         # [1, 2, 7, 8, 4]
```

for 반복문과 리스트 사용하기

- 리스트는 순서가 있는 객체
- for 반복문을 이용해서 리스트의 각 요소에 접근 가능

```
a = [4, 8, 7, 2, 1]
for n in a:
    print(n)

l = [1, 'g', 'h', 3, "abc"]
for n in l:
    print(n)

# 리스트에서 4보다 큰 정수만 출력
for n in a:
    if n > 4:
        print(n)
```

실습문제 1

□ 문제

- 사용자로부터 세 개의 숫자를 입력받아 리스트에 저장한 후,
- 리스트의 모든 요소의 합을 계산하는 프로그램을 작성하세요.

□ 요구사항

- 함수구성
- 사용자로부터 입력받는 값은 양의 정수로 가정
- 함수에서 정수를 입력받고 리스트 구성후 합계 출력

실습문제 1

▣ 최종 코드

```
def sum_list_elements():  
    # 사용자로부터 세 개의 숫자를 입력받습니다.  
    numbers = [] # 숫자를 저장할 빈 리스트 생성  
    numbers.append(int(input("첫 번째 숫자를 입력하세요: ")))  
    numbers.append(int(input("두 번째 숫자를 입력하세요: ")))  
    numbers.append(int(input("세 번째 숫자를 입력하세요: ")))  
  
    # 리스트의 요소를 더합니다.  
    total = sum(numbers)  
    print(f"합계: {total}")  
  
# 함수 호출  
sum_list_elements()
```

실습문제 2

□ 문제

- 사용자로부터 세 개의 숫자를 입력받아 리스트에 저장하고, 그 중에서 최대값을 찾아 출력하는 프로그램을 작성하세요.

□ 요구사항

- `max()`, `min()` 명령 활용하여 결과 출력

실습문제 2

□ 최종 코드

```
def find_min_max_value():
    # 사용자로부터 세 개의 숫자를 입력받습니다.
    numbers = [] # 숫자를 저장할 빈 리스트 생성
    numbers.append(int(input("첫 번째 숫자를 입력하세요: ")))
    numbers.append(int(input("두 번째 숫자를 입력하세요: ")))
    numbers.append(int(input("세 번째 숫자를 입력하세요: ")))

    # 최대값을 찾습니다.
    min_value = min(numbers)
    max_value = max(numbers)
    print(f"최대값: {max_value} 이고 , 최소값:{min_value} ")

# 함수 호출
find_min_max_value()
```

튜플 (Tuple)

- 튜플은 리스트와 비슷하게 순서가 있는 요소들의 (Sequential Items) 목록
 - 변경할 수 없는 자료구조 (리스트와의 큰 차이)
 - 리스트와 마찬가지로 연결하기, 슬라이싱(slicing), 반복하기, 멤버 검사, 길이 정보 등과 같은 다양한 연산 지원 (새로운 튜플 생성 가능)
 - () 형태로 표현 (괄호를 생략하는 경우도 있음)

```
()      # 빈 튜플(empty tuple)
(1, 2, 3, "abc") # 요소가 4개인 튜플
(1, 2, 3, (1, 2)) # 요소에 튜플 포함 가능
(1, 2, 3, [1, 2]) # 요소에 리스트 포함 가능
tuple1 = (2, 3)
tuple2 = 2, 3      # 괄호 생략 가능
tuple3 = 2, 3, [1, 2], (1, 2)
```

튜플 (Tuple)

- 요소가 한 개인 튜플을 생성할 때에는 반드시 콤마(',')로 끝내야 함

```
tuple1 = (2,)      # 튜플 생성
tuple2 = 2,        # 튜플 생성
tuple3 = ("hello",) # 튜플
tuple4 = "hello",  # 튜플
tuple5 = (2)        # 정수
tuple6 = ("hello")  # 문자열
```

튜플 (Tuple)

▣ 튜플의 요소 개수 확인하기

■ len() 함수 사용

```
e = ()  
t1 = (1, 2, 3, "abc")  
t2 = (1, 2, 3, (1, 2))  
t3 = 1, 2, 3, [1, 2], (1, 2)  
len(())      # 0  
len(e)  
len(t1)  
len(t2)  
len()  
len((1, 2, 3, "abc"))  
len((1, 2, 3, [1, 2]))
```

튜플 (Tuple)

□ 인덱싱

- 특정(단일) 요소에 접근(access)해서 값을 읽거나 변경하는 방법
- [인덱스]를 사용해서 요소에 접근

[0] [1] ... [3]

1	2	3	abc
---	---	---	-----

[-4] [-3] ... [-1]

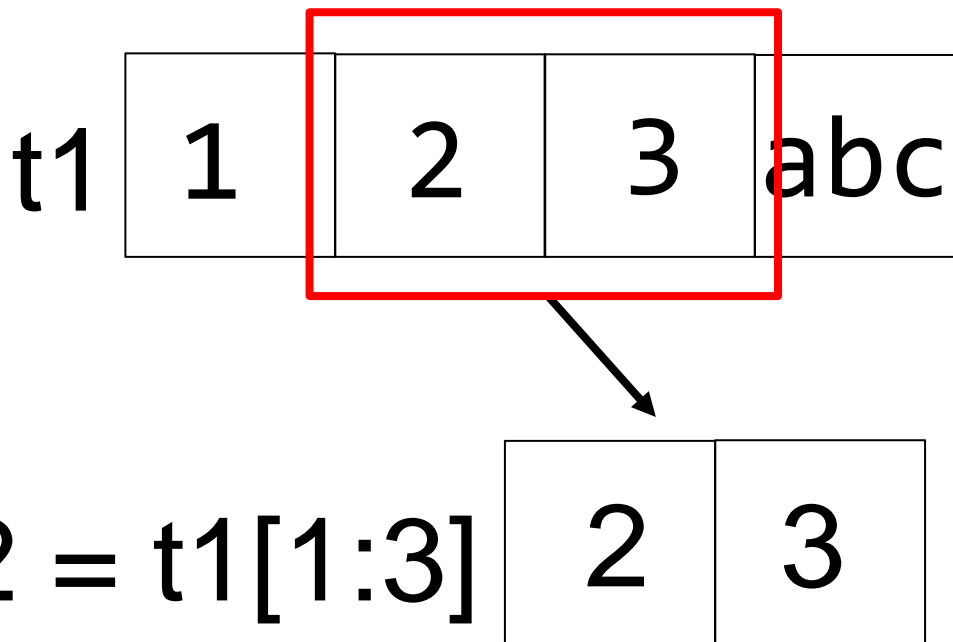
```
a = (1, 2, 3, "abc") # 튜플 생성
print(a[0], a[1], a[2], a[3]) # 순서대로 출력
print(a[-1], a[-2], a[-3], a[-4]) # 거꾸로 출력
```

튜플 (Tuple)

□ 슬라이싱(slicing)

- 여러 개 요소에 접근하는 방법
- **[시작인덱스:끝인덱스]**를 사용해서 요소들에 접근
- 시작인덱스에 해당되는 요소부터 끝인덱스 - 1 까지의 요소들을 포함하는 튜플 생성

```
t1 = (1, 2, 3, "abc")  
t2 = t1[1:3]
```



튜플 (Tuple)

▣ 슬라이싱(slicing)

```
a = (1, 2, 3, "abc", 4, 5)
a[2:4]      # (3, "abc")
a[:3]       # (1, 2, 3)
a[2:]       # (3, "abc", 4, 5)
a[:]        # (1, 2, 3, "abc", 4, 5)
```

▣ 확장 슬라이싱

- [시작인덱스:끝인덱스:스텝]

- 시작인덱스부터 끝인덱스 - 1까지 스텝만큼 건너뛰면서 튜플 참조

```
a = (1, 2, 3, "abc", 4, 5)
a[2:5:2]     # (3, 4)
a[2:6:2]     # (3, 4)
a[:3:2]      # (1, 3)
a[2::2]      # (3, 4)
```

튜플 (Tuple)

□ 연결하기

- + 연산자를 이용해서 두 개 이상의 튜플을 연결해서 새로운 튜플을 생성할 수 있음

```
t1 = (1, 2, 3)
t2 = ("abc", 4, 5)
t = t1 + t2 # 두 개 튜플을 붙여서 t에 저장
print(t)    # (1, 2, 3, "abc", 4, 5)
```

□ 반복하기

- * 연산자를 이용해서 요소들이 반복되는 튜플 생성

```
a = (1, 2)
b = a * 4
print(b)          # (1, 2, 1, 2, 1, 2, 1, 2)
```

튜플 (Tuple)

▣ 튜플 요소 변경해보기

```
t = (1, 2, [1, 2])  
t[1] = 3 # 오류 발생  
t[2] = [1, 3]  
del(t[1])
```

```
list1 = t[2]  
list1[0] = 3  
t[2][0] = 4
```

▣ 튜플의 구조를 바꾸는 것은 불가능

실습문제 3

□ 문제

- 이름, 나이를 함수 인자로 전달하고 전달받은 값을, 변수에 담고, 튜플로 리턴하는 함수를 만들자
- 출력은 리턴 받은 튜플 데이터를 출력한다.

□ 요구사항

- Name, age를 입력 받아서 문자열로 함수로 전달한다.

실습문제 3

▣ 최종 코드

```
def get_user_data(str1, str2):  
    # 사용자의 이름과 나이를 반환  
    name = str1  
    age = str2  
    return (name, age) # 튜플로 반환  
  
# 함수 호출  
  
str1 = input("이름을 입력해주시요:")  
str2 = str(input("나이를 입력해주시요:"))  
user_data = get_user_data(str1, str2)  
print(user_data) # ('Alice', 30)
```

실습문제 4

□ 문제

- 사용자로부터 세 개의 숫자를 입력받아 튜플에 저장한 후, 저장된 튜플을 출력하고 튜플의 첫 번째 요소와 마지막 요소를 출력하는 프로그램을 작성하세요.

□ 요구사항

- 함수 활용
- Input() 활용
- 튜플 요소 접근

실습문제 4

▣ 최종 코드

```
def print_tuple_elements():  
    # 사용자 입력 받기  
    a = int(input("첫 번째 숫자를 입력하세요: "))  
    b = int(input("두 번째 숫자를 입력하세요: "))  
    c = int(input("세 번째 숫자를 입력하세요: "))  
  
    # 튜플 생성  
    my_tuple = (a, b, c)  
  
    # 튜플과 요소 출력  
    print("생성된 튜플:", my_tuple)  
    print("튜플의 첫 번째 요소:", my_tuple[0])  
    print("튜플의 마지막 요소:", my_tuple[-1])  
  
# 함수 호출  
print_tuple_elements()
```

함수에서 여러 개 값을 반환하기

□ 함수에서 여러 값을 함께 반환

- 튜플이나 리스트를 사용해서 반환할 수 있음
- 함수 결과를 받는 변수 개수는 1 또는 리스트/튜플의 길이

```
def funcReturnsTuple():  
    return 1, 2
```

```
def funcReturnsList():  
    return [3, 4, 5]
```

```
tuple1 = funcReturnsTuple()  
list1 = funcReturnsList()  
x, y = funcReturnsTuple()  
a, b, c = funcReturnsList()
```


함수에서 여러 개 값을 반환하기

- 두 개의 숫자 값을 입력으로 전달 받고, 비교해서 작은 것, 큰 것 순서대로 값들을 다시 반환하는 함수를 작성
 - 숫자 비교
 - 작은 값, 큰 값 순서대로 튜플 또는 리스트로 구성
 - 튜플 또는 리스트 반환

함수에서 여러 개 값을 반환하기

□ 튜플 버전

```
def orderNumbers(num1, num2):  
    if num1 <= num2:  
        return num1, num2  
    else:  
        return num2, num1
```

□ 리스트 버전

```
def orderNumbers(num1, num2):  
    if num1 <= num2:  
        return [num1, num2]  
    else:  
        return [num2, num1]
```

리스트와 튜플 간 상호 변환

□ 리스트를 튜플로 변환

■ tuple()함수 사용

```
list1 = [1, 2]  
tuple1 = tuple(list1)  
tuple2 = tuple([1, 2, 3])
```

□ 튜플을 리스트로 변환

■ list()함수 사용

```
tuple1 = (1, 2)  
list1 = list(tuple1)  
list2 = list((1, 2, 3))
```

실습문제 5

□ 문제

- 사용자로부터 세 개의 숫자를 입력받아 튜플에 저장한 후, 이 튜플의 요소들의 합, 최대값, 최소값을 계산하여 출력하는 프로그램을 작성하세요.

□ 요구사항

- `sum()`, `max()`, `min()` 활용

실습문제 6

□ 문제

- 사용자로부터 문자열을 입력받아 리스트에 추가하고, 튜플로 변환하여 출력하는 프로그램을 작성하세요.

□ 요구사항

실습문제 7

□ 문제

- 두 개의 리스트와 두 개의 튜플을 생성하고, 이들을 각각 결합하여 새로운 리스트와 튜플을 만드는 프로그램을 작성하세요.

□ 요구사항

- 함수안에서 각각 두개씩 리스트, 튜플 생성, 병합, 출력 진행

실습문제 8

□ 문제

- 사용자로부터 숫자를 입력받아, 주어진 리스트와 튜플 내에 그 숫자가 있는지 확인하는 프로그램을 작성하세요.

□ 요구사항

- `if 변수 in 객체`

실습문제 9

□ 문제

- 사용자로부터 여러 개의 숫자를 입력받아, 리스트를 생성하고, 이 리스트를 역순으로 출력하는 프로그램을 작성하세요.

□ 요구사항

- 리스트에서 역순 처리를 위해 '공백 구분'
- `reverse()` 활용

실습문제 10

□ 문제

- 튜플 (1, 2, 3, 4, 5, 6)에서 인덱스 2부터 4까지의 요소를 슬라이싱하여 출력하는 프로그램을 작성하세요.

□ 요구사항

실습문제 11

□ 문제

- 중첩된 리스트 `[[1, 2], [3, 4], [5, 6]]`를 평탄화(1차원 리스트)하여 모든 요소를 하나의 리스트로 만드는 프로그램을 작성하세요.

□ 요구사항

- `[표현식 for 항목 in 리스트 for 항목 in 리스트]`

실습문제 12

□ 문제

- 사용자로부터 여러 개의 숫자를 입력받아 리스트로 저장한 후, 이 리스트를 튜플로 변환하여 출력하는 프로그램을 작성하세요.

□ 요구사항

- `input("숫자들을 입력하세요 (공백으로 구분): ")`

실습문제 13

□ 문제

- 사용자로부터 여러 개의 숫자를 입력받아 리스트로 저장한 후, 중복된 요소를 제거하고 결과를 출력하는 프로그램을 작성하세요.

□ 요구사항

- 중복 제거를 위해 `set()` : 집합 활용해 보자
- `list(set(numbers))`

실습문제 14

□ 문제

- 사용자로부터 정수 N 을 입력받아,
- 리스트 $[1, 2, 3, 4, 5]$ 를 N 번만큼 순환 시킨 결과를 출력하는 프로그램을 작성하세요.

□ 요구사항

실습문제 15

□ 문제

- 변수 a, b, c 에 각각 1, 2, 3을 패킹하여 튜플로 만든 후, 이를 다시 언패킹하여 출력하는 프로그램을 작성하세요.

□ 요구사항

- 패킹 : 튜플화
- 언패킹 : 변수화
- $a, b, c = 1, 2, 3$ 변수 할당하여 진행

실습문제 16

□ 문제

- 사용자로부터 여러 개의 숫자를 입력받아 리스트로 저장한 후, 이 리스트를 오름차순으로 정렬하여 출력하는 프로그램을 작성하세요.

□ 요구사항

- `input("숫자들을 입력하세요 (공백으로 구분): ")`
- 입력 받은 값을 리스트로 변환
- `sorted()` 활용 정렬 (`sorted` 함수는 파이썬 내장 정렬함수)

실습문제 17

□ 문제

- 튜플 (10, 20, 30, 40, 50)에서 30보다 큰 수만을 출력하는 프로그램을 작성하세요.

□ 요구사항

- for문과 if 조건문이 한개일 때 작성 형식 :
[식 for 변수 in 리스트 if 조건식]

실습문제 17

▣ 최종 코드

```
def filter_tuple():  
    my_tuple = (10, 20, 30, 40, 50)  
    filtered = [x for x in my_tuple if x > 30]  
    print("30보다 큰 수:", filtered)  
  
# 함수 호출  
filter_tuple()
```

