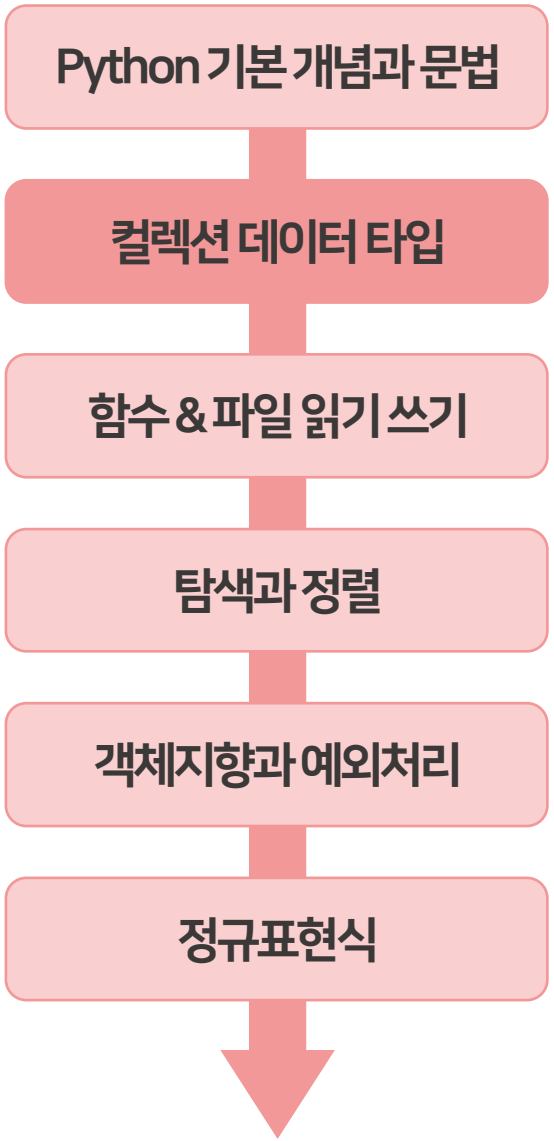
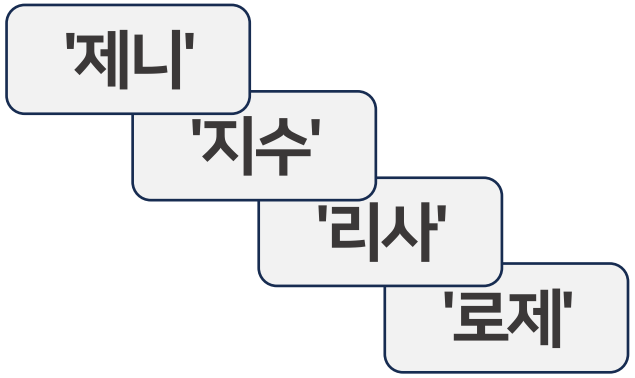

python™



1. 리스트
2. 리스트의 수정, 삭제
3. 리스트 관련 함수
4. 구조 분해 할당
5. List Comprehesion
6. 리스트 형태로 입력받기
7. 튜플
8. 딕셔너리, 딕셔너리의 추가, 수정, 삭제
9. 딕셔너리 관련 함수
10. 세트
11. 세트의 활용
12. 세트 관련 함수

리스트




```
black_pink = ['제니', '지수', '리사', '로제']
```

0	'제니'
1	'지수'
2	'리사'
3	'로제'

```
empty_list = []
black_pink = ['제니', '지수', '리사', '로제']
odd_number = [1, 3, 5, 7, 9]
mixed_list = ['손흥민', 7, 183.6, True]
```

```
alpabet = ['A', 'B', 'C', 'D', 'E']

# vow = alpabet[0:5:4]
vow = [alpabet[0], alpabet[4]]
consonant = alpabet[1:4]

new_alpabet = [vow, consonant]
print(new_alpabet)
```

```
[['A', 'E'], ['B', 'C', 'D']]
```

리스트의 수정

```
practice_list = [1, 2, 3, '파이', 'Apple', ['가', '나', '다', '라']]

# 리스트에서 '파이'를 3.14로 수정하기
practice_list[3] = 3.14
print(practice_list)

# 리스트에서 3을 4, 8로 수정하기
~~practice_list[2] = [4, 8]~~
practice_list[2:3] = [4, 8]
print(practice_list)
```

리스트의 삭제

```
practice_list = [1, 2, 4, 8, 3.14, 'Apple', ['가', '나', '다', '라']]

# 리스트에서 'Apple'을 삭제하기
~~practice_list[5] = []~~
practice_list[5:6] = []
print(practice_list)

# 리스트에서 '라'를 삭제하기
del practice_list[-1][-1]
print(practice_list)

# 리스트에서 '나','다'를 삭제하기
del practice_list[-1][1:]
print(practice_list)
```

리스트 관련 함수

함수	설명	함수	설명
append(element)	리스트 끝에 요소를 추가	sort()	리스트 정렬
insert(index, element)	리스트 중 특정 위치에 요소 삽입	reverse()	리스트 순서 뒤집기
pop()	리스트에서 마지막 요소 꺼내기	extend(list)	리스트 확장
pop(index)	리스트에서 해당 위치의 요소 꺼내기	index(element)	리스트 내 같은 값 위치 반환 (중복 시 첫번째)
element in list	리스트 내 해당 요소 존재 여부 반환	count(element)	리스트 내 같은 값 개수 세기

구조 분해 할당

```
my_list = ['홍길동', 20]
name, age = my_list
```

```
my_list = [1, 2, 4, 8]

odd, *even = my_list
```

```
my_list = ['홍길동', 'TMI', 20, 'TMI', '서울 성북구']
name, _, age, _, address = my_list
```

```
my_list = ['홍길동', 'TMI', 'TMI', 'TMI', '20']
name, *_, age = my_list
```

List Comprehension

```
num_list = [1, 2, 3, 4]
even_list = []
for element in num_list:
    even_list.append(element * 2)

print(even_list)
```

```
num_list = [1, 2, 3, 4]
even_list = [element * 2 for element in num_list]

print(even_list)
```

List Comprehension

```
my_list = [ 변수를활용한값 for 사용한변수명 in 반복객체 ]
```

List Comprehension의 활용

```
name_list = ['손흥민', '조규성', '김민재', '이강인', '이승우', '황희찬']
lee_list = [ name for name in name_list if name[0]=='이' ]
son_lee_list = [ name for name in name_list if name[0]=='이' or name[0]=='손' ]

print(lee_list)
print(son_lee_list)
```

리스트 형태로 입력받기

```
my_input = input("이름과 나이를 입력하세요. (예: 홍길동 20)\n")
my_input_list = my_input.split(" ")
name, age = my_input_list
age = int(age)
```

리스트 형태로 입력받기

```
my_input = input("공백을 기준으로 숫자를 입력하세요. (예: 1 2 3)\n")
my_inputs = my_input.split()

my_numbers = []
for i in my_inputs:
    my_numbers.append(int(i))
```

```
number_list = [int(x) for x in input("숫자를 입력하세요. (예: 1 2 3)").split()]
```

```
number_list = list(map(int, input("숫자를 입력하세요. (예: 1 2 3)").split()))
```

시작과 끝

입력으로 주어지는 리스트 x 의 첫 번째 요소와 마지막 요소의 합을 반환하는 프로그램을 완성하세요.

입력	출력
1356	7

입력	출력
7193	10

최대값 https://www.acmicpc.net/problem/2562

9개의 서로 다른 자연수가 주어질 때, 이들 중 최댓값을 찾고 그 최댓값이 몇 번째 수인지를 구하는 프로그램을 작성하시오.

예를 들어, 서로 다른 9개의 자연수 [3, 29, 38, 12, 57, 74, 40, 85, 61]이 주어지면, 이들 중 최댓값은 85이고, 이 값은 8번째 수이다.

입력	출력
3 29 38 12 57 74 40 85 61	85 8

공 바꾸기 https://www.acmicpc.net/problem/10813

바구니를 총 N개 가지고 있고, 각각의 바구니에는 1번부터 N번까지 번호가 매겨져 있다.
바구니에는 공이 1개씩 들어있고, 처음에는 바구니에 적혀있는 번호와 같은 번호가 적힌 공이 들어있다.

앞으로 M번 공을 바꾸려고 한다. 공을 바꿀 바구니 2개를 선택하고, 두 바구니에 들어있는 공을 서로 교환한다.
공을 어떻게 바꿀지가 주어졌을 때, M번 공을 바꾼 이후에 각 바구니에 어떤 공이 들어있는지 구하는 프로그램을 작성하시오.

첫째 줄에 N (1 ≤ N ≤ 100)과 M (1 ≤ M ≤ 100)이 주어진다.
둘째 줄부터 M개의 줄에 걸쳐서 공을 교환할 방법이 주어진다.
각 방법은 두 정수 i j로 이루어져 있으며, i번 바구니와 j번 바구니에 들어있는 공을 교환한다는 뜻이다.

입력	출력
5 4 1 2 3 4 1 4 2 2	3 1 4 2 5

애너그램 https://www.acmicpc.net/problem/6996

두 단어 A와 B가 주어졌을 때, A에 속하는 알파벳의 순서를 바꾸어서 B를 만들 수 있다면,
A와 B를 애너그램이라고 한다.

두 단어가 애너그램인지 아닌지 구하는 프로그램을 작성하시오.

첫째 줄에 테스트 케이스의 개수(<100)가 주어진다.
둘째 줄부터는 테스트 케이스(공백을 기준으로 두 개의 단어)가 주어진다.
단어는 알파벳 소문자로만 이루어져 있다.

입력	출력
3 blather reblath maryland landam bizarre brazier	blather & reblath are anagrams. maryland & landam are NOT anagrams. bizarre & brazier are anagrams.

튜플

1. 리스트는 대괄호 `[]`를 이용하지만, 튜플은 소괄호 `()`를 이용합니다.

```
my_tuple = ()

my_int = (1) # 튜플이 아닌 정수형
my_tuple = (1,) # 튜플로 인식

my_tuple1 = (1, 2, 3)
my_tuple2 = 1, 2, 3
print(my_tuple1 == my_tuple2)
```

2. 리스트는 요소의 값을 바꾸는 것이 가능하지만, 튜플은 값을 바꿀 수가 없습니다.

```
my_tuple = ('a', 'b', ('ab', 'cd'))
del my_tuple1[0] # [오류 발생!] 튜플의 값은 한 번 정하면 삭제 불가능
my_tuple[0] = 'A' # [오류 발생!] 튜플의 값은 한 번 정하면 수정 불가능
```

X보다 작은 수

https://www.acmicpc.net/problem/10871

정수 N개로 이루어진 수열 A와 정수 X가 주어진다.

이때, A에서 X보다 작은 수를 모두 출력하는 프로그램을 작성하시오

입력	출력
10 5 1 10 4 9 2 3 8 5 7 6	1 4 2 3

딕셔너리

1. 딕셔너리는 Key와 Value의 대응으로 자료를 저장하는 데이터 타입으로, 리스트나 튜플처럼 순서가 있지 않습니다.
2. 딕셔너리는 중괄호 `{}`를 이용하고, Key와 Value는 콜론 `:`으로 구분합니다.
3. 딕셔너리에서 Key는 고유한 값이므로, 중복된 Key 값을 사용해서는 안됩니다.

```
my_dictionary = {}
my_dictionary = dict()

my_dictionary = {
    'name': '손흥민',
    'age': 29,
    'address': ['대한민국', '영국', '독일']
}
```

딕셔너리의 추가, 수정, 삭제

```
son_dictionary = {
    'name': '손흥민',
    'age': 29,
    'address': ['대한민국', '영국', '독일']
}

son_dictionary['job'] = 'football player'    # 추가
son_dictionary['name'] = 'son'               # 수정
del son_dictionary['address']                # 삭제

print(son_dictionary)
print(son_dictionary['name'])
```

딕셔너리 관련 함수

함수	설명
keys()	딕셔너리 내 Key만을 모아서 dict_keys라는 객체를 반환
values()	딕셔너리 내 Value만을 모아서 dict_values라는 객체를 반환
items()	딕셔너리 내 Key와 Value의 쌍을 튜플로 묶어서 dict_items라는 객체를 반환
clear()	딕셔너리 안의 모든 요소 삭제
get(key)	Key에 대응되는 Value를 반환, 없으면 None을 반환
get(key, default)	Key에 대응되는 Value를 반환, 없으면 default 값을 반환
key in dictionary	딕셔너리 안에 해당 Key가 있는지 여부를 반환

애너그램 https://www.acmicpc.net/problem/6996

두 단어 A와 B가 주어졌을 때, A에 속하는 알파벳의 순서를 바꾸어서 B를 만들 수 있다면, A와 B를 애너그램이라고 한다.

두 단어가 애너그램인지 아닌지 구하는 프로그램을 작성하시오.

첫째 줄에 테스트 케이스의 개수(<100)가 주어진다.
둘째 줄부터는 테스트 케이스(공백을 기준으로 두 개의 단어)가 주어진다.
단어는 알파벳 소문자로만 이루어져 있다.

입력	출력
3 blather reblath maryland landam bizarre brazier	blather & reblath are anagrams. maryland & landam are NOT anagrams. bizarre & brazier are anagrams.

세트

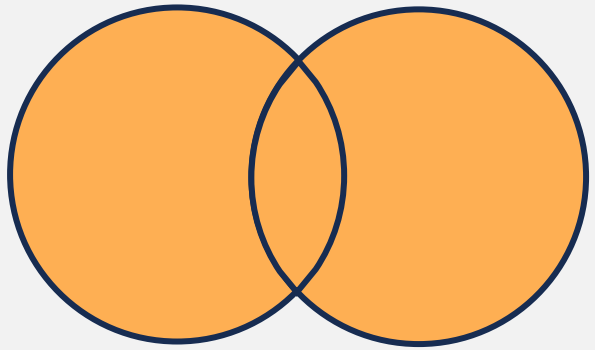
1. 세트는 집합 자료형으로, 순서가 없고 중복을 허용하지 않습니다.
    - 따라서 딕셔너리와 마찬가지로 인덱싱과 슬라이싱이 불가능합니다.

```
my_set = set()

my_set1 = set([1, 2, 3, 4, 5])
my_set2 = set([3, 1, 2, 5, 4])
print(my_set1 == my_set2)

my_set3 = set("Python is Interesting")
print(my_set3)
```

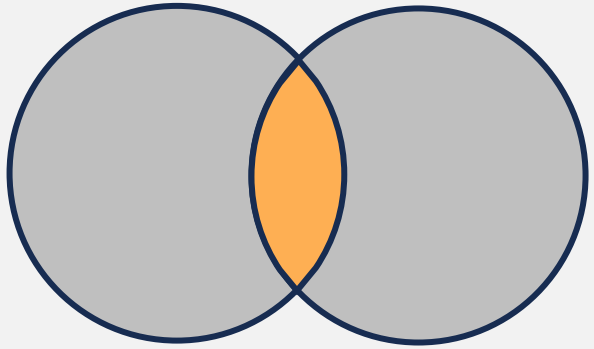
세트의 활용

```
my_set1 = set([1,2,3,4,5,6])
my_set2 = set([4,5,6,7,8,9])
```

세트의 활용

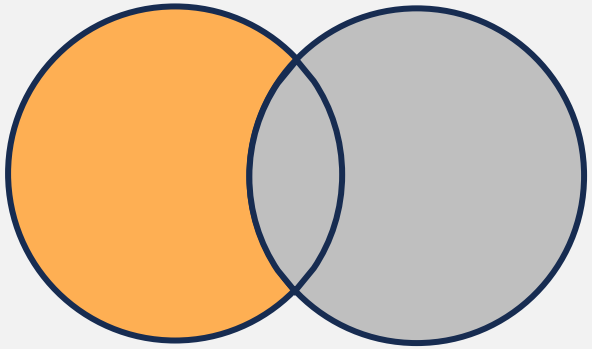


```
result1 = my_set1 | my_set2
result2 = my_set1.union(my_set2)

my_set1.update(my_set2)
```

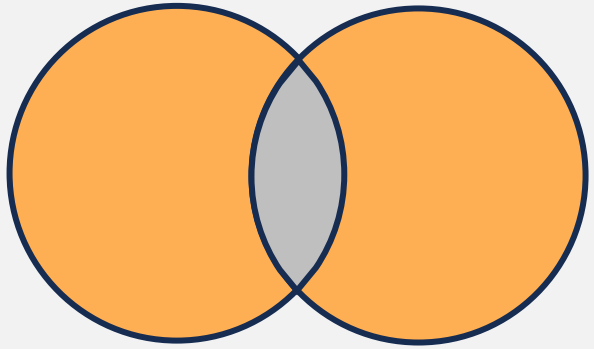


```
result1 = my_set1 & my_set2
result2 = my_set1.intersection(my_set2)

my_set1.intersection_update(my_set2)
```

세트의 활용



```
result1 = my_set1 - my_set2
result2 = my_set1.difference(my_set2)

my_set1.difference_update(my_set2)
```



```
result1 = my_set1 ^ my_set2
result2 = my_set1.symmetric_difference(my_set2)

my_set1.symmetric_difference_update(my_set2)
```

세트 관련 함수

함수	설명
add(element)	세트에 새로운 요소를 추가합니다.
update(set)	세트에 다른 세트를 추가
pop()	세트 내 임의의 요소를 제거하고 반환 (set가 비어있으면, KeyError 발생)
remove(element)	세트 내 요소를 제거 (존재하지 않는 요소인 경우, KeyError 발생)
discard(element)	세트 내 요소를 제거 (존재하지 않는 요소인 경우, 세트 변경되지 않음)
clear()	세트 내 모든 요소를 제거

함수	설명
copy()	세트를 복사
isdisjoint(set)	다른 세트와 교집합 요소 존재 여부 반환
issubset(set)	다른 세트와 부분 집합 관계 여부 반환
issuperset(set)	다른 세트의 상위 집합 여부 반환
len(set)	세트의 요소 개수 반환
element in set	세트 안에 해당 요소 존재 여부 반환

집합 https://www.acmicpc.net/problem/11723

비어있는 공집합 S가 주어졌을 때, 아래 연산을 수행하는 프로그램을 작성하시오.

- add x: S에 x를 추가한다. S에 x가 이미 있는 경우에는 연산을 무시한다.
- remove x: S에서 x를 제거한다. S에 x가 없는 경우에는 연산을 무시한다.
- check x: S에 x가 있으면 1을, 없으면 0을 출력한다.
- toggle x: S에 x가 있으면 x를 제거하고, 없으면 x를 추가한다.
- all: S를 {1, 2, ..., 20} 으로 바꾼다.
- empty: S를 공집합으로 바꾼다.

첫째 줄에 수행해야 하는 연산의 수 M이 주어진다.
둘째 줄부터 M개의 줄에 수행해야 하는 연산이 한 줄에 하나씩 주어진다.
check 연산이 주어질 때마다, 결과를 출력한다.

입력	출력
16 add 2 check 1 check 2 remove 2 check 1 check 2 toggle 3 check 3 all check 20 toggle 10 remove 20 check 10 check 20 empty check 1	 0 1 0 0 1 1 0 0 0

동명이인

공백을 기준으로 여러 명의 이름을 입력받은 뒤,
동명이인을 찾아 집합으로 반환하는 프로그램을 만들어보세요.

입력	출력
Tom Jerry Mike Tom Mike	{'Tom', 'Mike'}