python

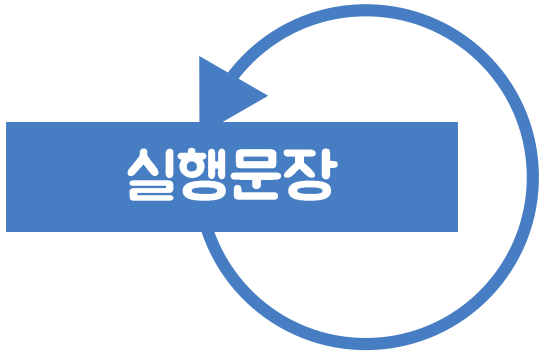
대한상공회의소
서울기술교육센터

나 예 호 교수

python 반복문 이란?

- 프로그램 내에서 똑같은 명령을 일정 횟수만큼 반복하여 수행하도록 제어하는 명령문

- 반복문 종류는 while문, for문이 있다.



while : 반복 횟수가 명확하지 않을 때

for : 반복 횟수가 명확할 때

1부터 1000까지 출력하시오.

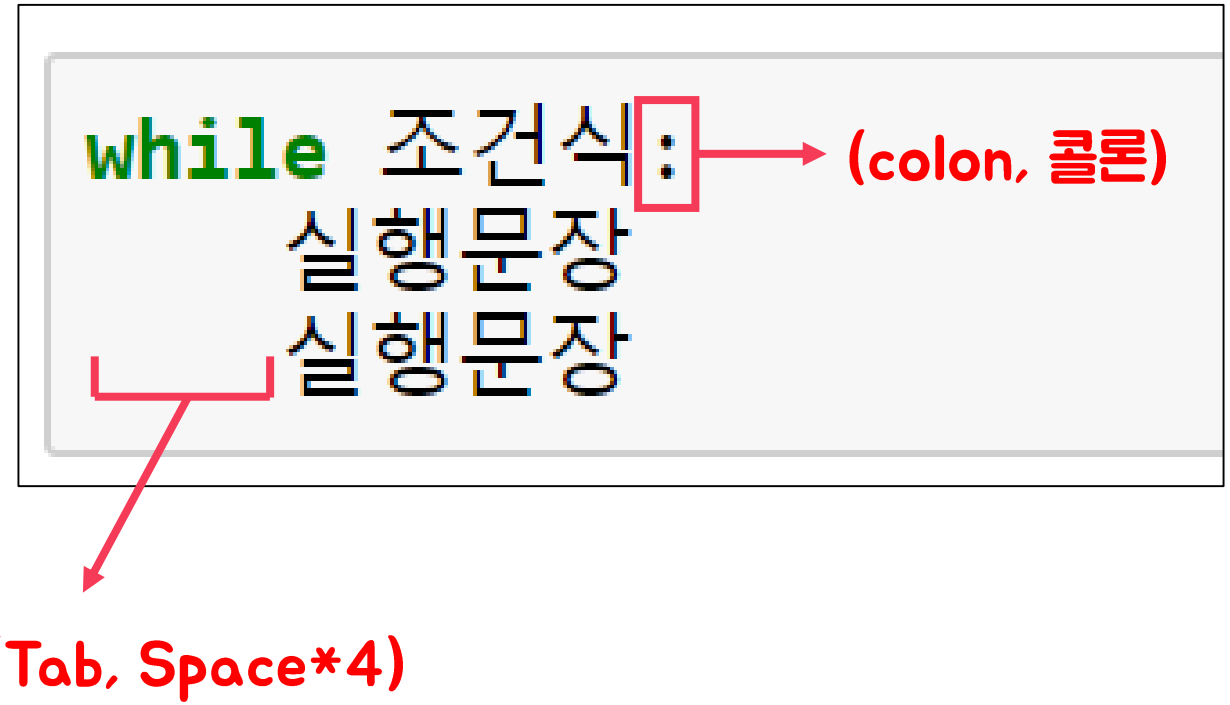
```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
...
```

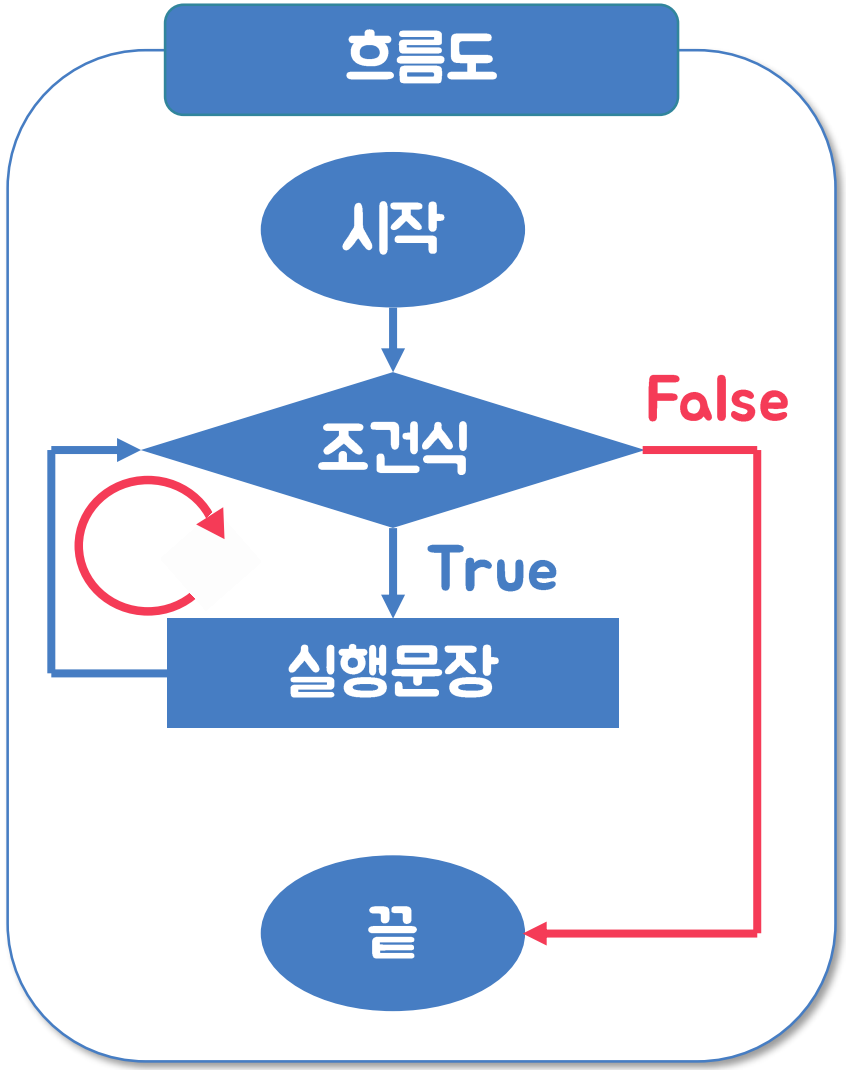
```
1
2
3
4
5
6
```

```
for i in range(1,1001):
    print(i)
```

```
1
2
3
4
996
997
998
999
1000
```

조건식이 True일 경우 실행문장 반복





1부터 3까지 출력하시오.

```
number = 1
while number <= 3:
    print(number)
    number += 1
```

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪

while문을 사용해서 "파이썬 최고!!"를 13번 출력하시오.

```
while 조건식:
    실행문장
```

```
num = 0
while num < 10:
    print("파이썬 최고!!")
    num += 1
```

```
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
파이썬 최고!!
```

break

- 반복문을 나가는 기능

```
while True:
    print("무한루프")
```

```
무한루프
무한루프
무한루프
무한루프
무한루프
무한루프
무한루프
무한루프
무한루프
무한루프
```

```
while True:
    print("무한루프")
    break
```

```
무한루프
```

```
number = 1
while True:
    print(number)
    number += 1
    if number > 3:
        break
```

```
1
2
3
```

두개의 정수를 입력 받아서 더하는 코드를 작성하시오.
(단, 두개의 정수가 0이 들어올 때 까지 반복한다.)

```
첫 번째 정수 입력 >> 1
두 번째 정수 입력 >> 2
두 정수의 합 : 3
첫 번째 정수 입력 >> 7
두 번째 정수 입력 >> 3
두 정수의 합 : 10
첫 번째 정수 입력 >> 13
두 번째 정수 입력 >> 77
두 정수의 합 : 90
첫 번째 정수 입력 >> 0
두 번째 정수 입력 >> 0
프로그램이 종료되었습니다.
```

다이어트 관리 프로그램

1. 현재 몸무게와 목표몸무게를 입력 받고 주차 별 감량 몸무게를 입력 받으세요.
2. 목표몸무게를 달성하면 축하한다는 문구를 출력하고 입력을 멈추세요!

```
현재 몸무게 : 80
목표 몸무게 : 70
1주차 감량 몸무게 : 2
2주차 감량 몸무게 : 3
3주차 감량 몸무게 : 4
4주차 감량 몸무게 : 5
66 kg 달성!! 축하합니다!
```

랜덤으로 1부터 50사이의 숫자를 뽑으면 뽑은 숫자를 맞추는 Up, Down게임 예제

```
숫자를 입력하세요 >> 25
25보다 작은 수 입니다.
숫자를 입력하세요 >> 13
13보다 작은 수 입니다.
숫자를 입력하세요 >> 5
5보다 큰 수 입니다.
숫자를 입력하세요 >> 8
8보다 작은 수 입니다.
숫자를 입력하세요 >> 6
6보다 큰 수 입니다.
숫자를 입력하세요 >> 7
정답을 맞추셨습니다.
```

random 라이브러리 사용

라이브러리 import

```
import random
```

random.randint() 사용

- 1~10 사이의 숫자 랜덤 추출

```
random.randint(1, 10)
```

```
5
```

랜덤으로 1부터 50사이의 숫자를 뽑으면 뽑은 숫자를 맞추는 Up, Down게임 예제

```
숫자를 입력하세요 >> 25
25보다 작은 수 입니다.
숫자를 입력하세요 >> 13
13보다 작은 수 입니다.
숫자를 입력하세요 >> 5
5보다 큰 수 입니다.
숫자를 입력하세요 >> 8
8보다 작은 수 입니다.
숫자를 입력하세요 >> 6
6보다 큰 수 입니다.
숫자를 입력하세요 >> 7
정답을 맞추셨습니다.
```

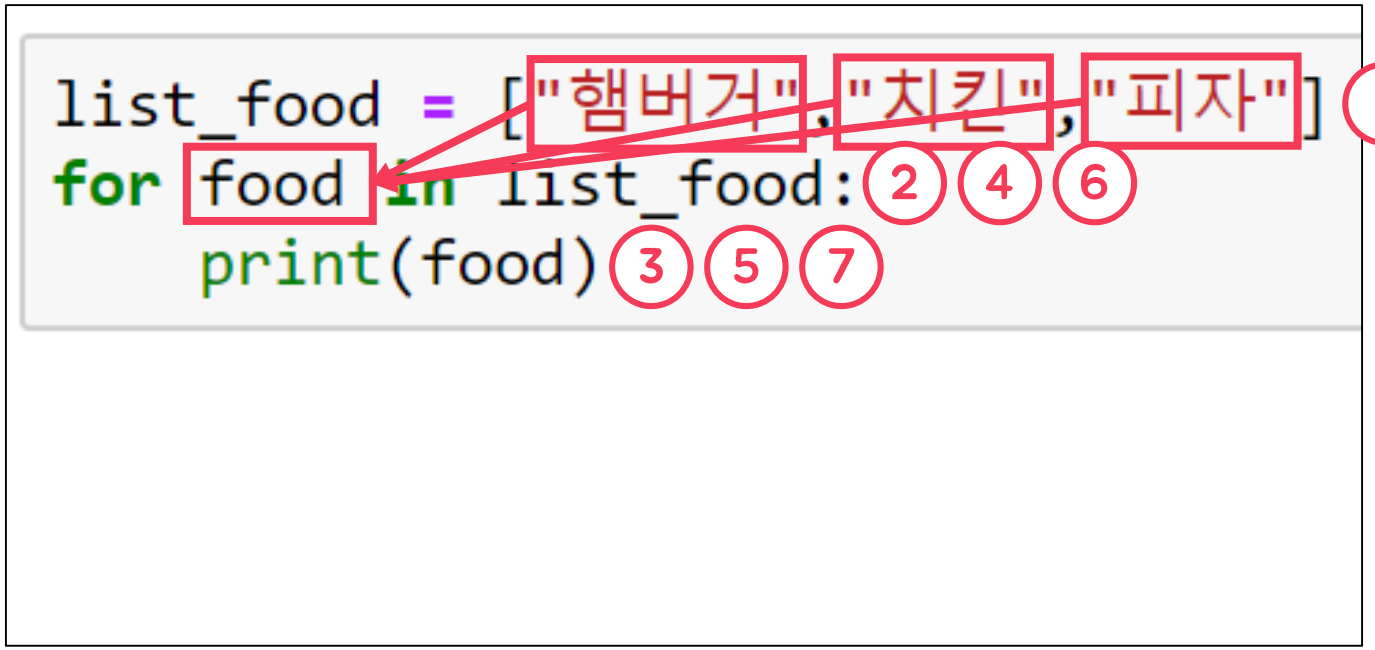
문자열 또는 리스트 또는 튜플이 들어갔을 때
안에 있는 요소를 하나씩 반복

```
for 변수 in 문자열(or 리스트 or 튜플):
    print(변수)
```

들여쓰기 (Tab, Space*4)

(colon, 콜론)

for문 예시



for문 예시

for문 예시

```
hi = "안녕하세요"
for s in hi:
    print(s)
```

```
안
녕
하
세
요
```

```
tuple_food = ("햄버거","치킨","피자")
for food in tuple_food:
    print(food)
```

```
햄버거
치킨
피자
```

1. 1부터 100사이의 숫자 중 3의 배수인 값들의 합을 출력하세요.

정답 : 1683

2. for문을 이용하여 구구단 2단을 출력하시오.

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
```

For문을 이용하여 다음 list에 들어있는 요소 중
가장 큰 수를 찾아 출력하세요.

```
list2 = [4,5,2,1,99,15,2,7,27]
?
```

```
99
```

For문을 이용하여 다음 list에 들어있는 요소 중
가장 작은 수를 찾아 출력하세요.

```
list2 = [4,5,2,1,99,15,2,7,27]
?
```

```
1
```

5명에 대한 정보처리기사 자격증 시험 점수가 리스트에 담겨있습니다. 이때 각 점수가 합격 점수인지 불합격 점수인지 판별하여 출력하시오.

(60점 이상 합격)

```
score_list = [90, 45, 70, 60, 55]
?
```

```
1번 학생은 합격입니다.
2번 학생은 불합격입니다.
3번 학생은 합격입니다.
4번 학생은 합격입니다.
5번 학생은 불합격입니다.
```

range() 함수 사용

- 필요한 만큼의 숫자를 만들어내는 유용한 기능
- range(시작할 숫자, 종료할 숫자, 증가량)
- range(1, 10, 1) →1부터 9까지 1씩 증가
- range(1, 100, 3) → 1부터 99까지 3씩 증가
- range(10, 1, -1) → 10부터 2까지 1씩 감소(-1씩 증가)

range() 함수 사용

```
for i in range(1, 10, 1):
    print(i)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
1
2
3
4
5
6
7
8
9
```

print() 함수

- end 속성

```
for i in range(1, 10, 1):
    print(i, end=" ")
```

```
1 2 3 4 5 6 7 8 9
```

```
for i in range(1, 10, 1):
    print(i, end="\n")
```

```
1
2
3
4
5
6
7
8
9
```

range() 함수 사용

- 필요한 만큼의 숫자를 만들어내는 유용한 기능
- range(기본값 0, 종료할 숫자, 기본값 1)

- range(3, 10) →3부터 9까지 1씩 증가

- range(10) → 0부터 9까지 1씩 증가

```
for i in range(3, 10):
    print(i, end=" ")
```

```
3 4 5 6 7 8 9
```

```
for i in range(10):
    print(i, end=" ")
```

```
0 1 2 3 4 5 6 7 8 9
```

1. for문을 이용하여 97부터 77까지 출력하시오.

```
?
```

```
97 96 95 94 93 92 91 90 89 88 87 86 85 84 83 82 81 80 79 78 77
```

2. for문을 이용하여 23부터 40까지 출력하시오

```
?
```

```
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
```

```
list1 = [[1,2],[3,4],[5,6]]
for i,j in list1:
    print(i, j)
```

```
1 2
3 4
5 6
```

```
a, b = 1, 7
print(a)
print(b)
```

```
1
7
```

두개의 정수를 키보드로 입력 받아 첫 번째 정수부터 두 번째 정수까지 출력되는 소스코드를 작성하시오.

```
start = ?
end = ?
?
```

```
첫 번째 정수 입력 >> 10
두 번째 정수 입력 >> 30
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

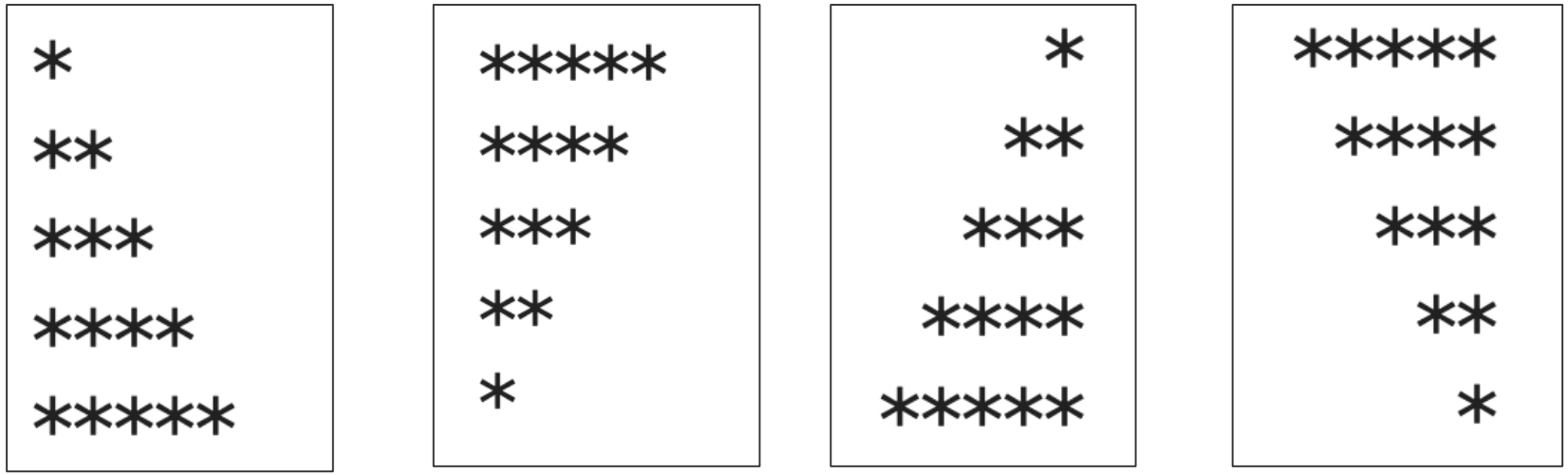
숫자를 입력 받고 입력 받은 숫자의 약수를 구하시오.

(약수란 어떤 수를 나누어 떨어지게 하는 수)

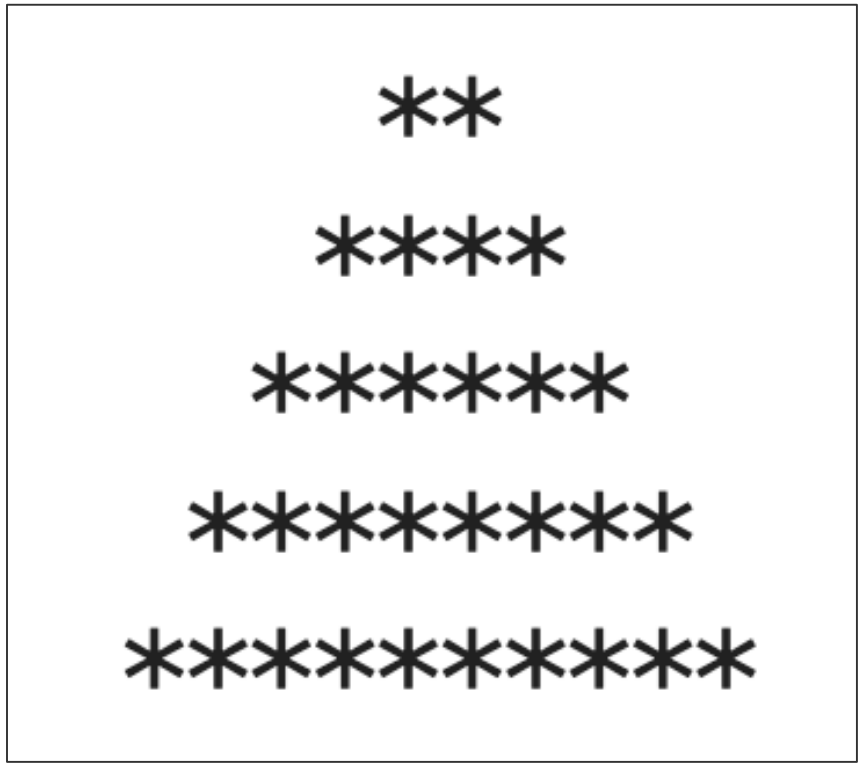
```
num = int(input("정수 입력 >> "))
?
```

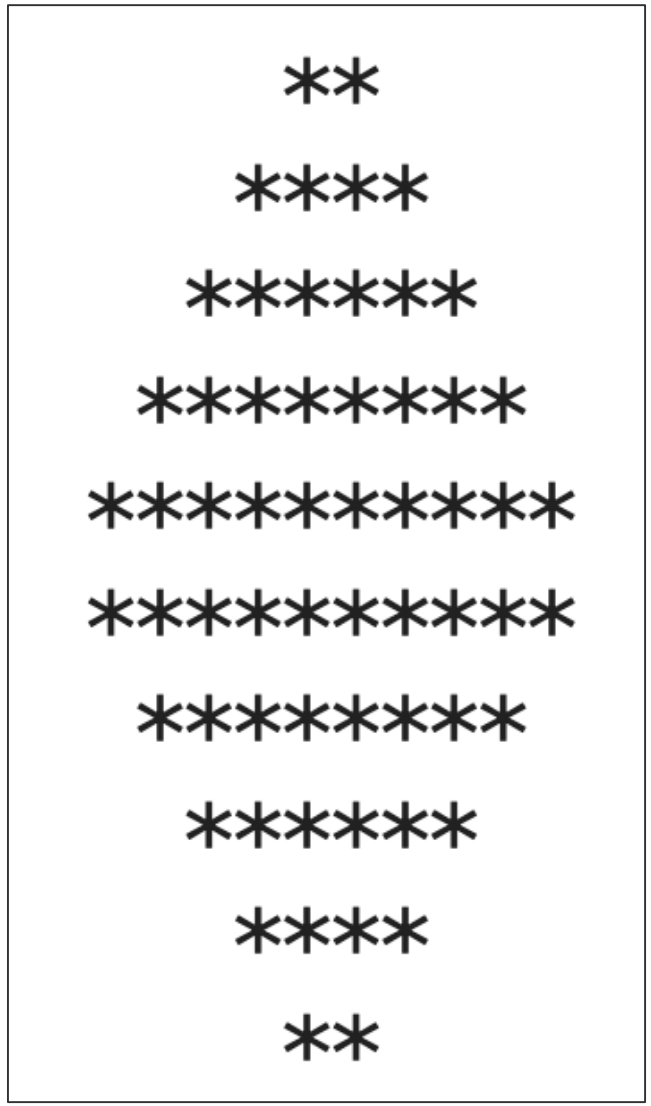
```
정수 입력 >> 32
32의 약수 : 1 2 4 8 16 32
```

별을 찍어보자

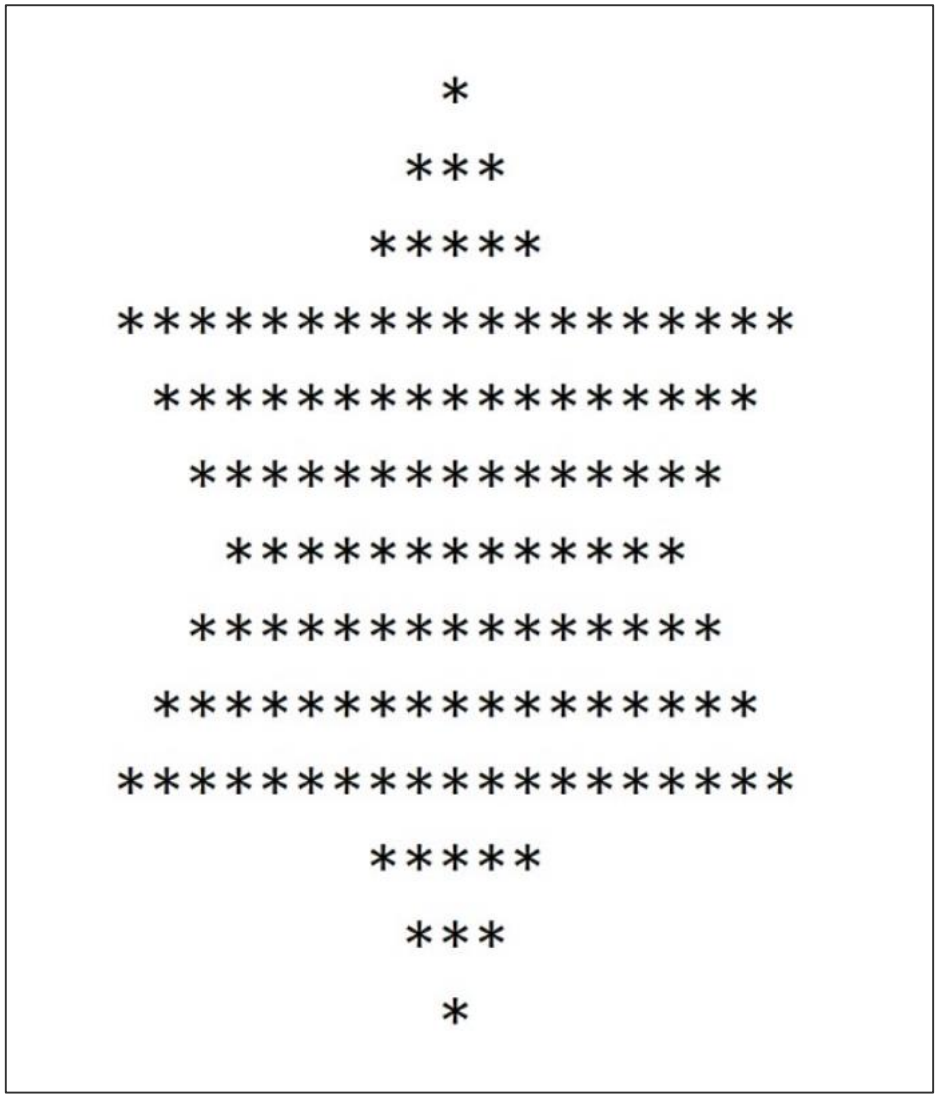


별을 찍어보자





별을 찍어보자

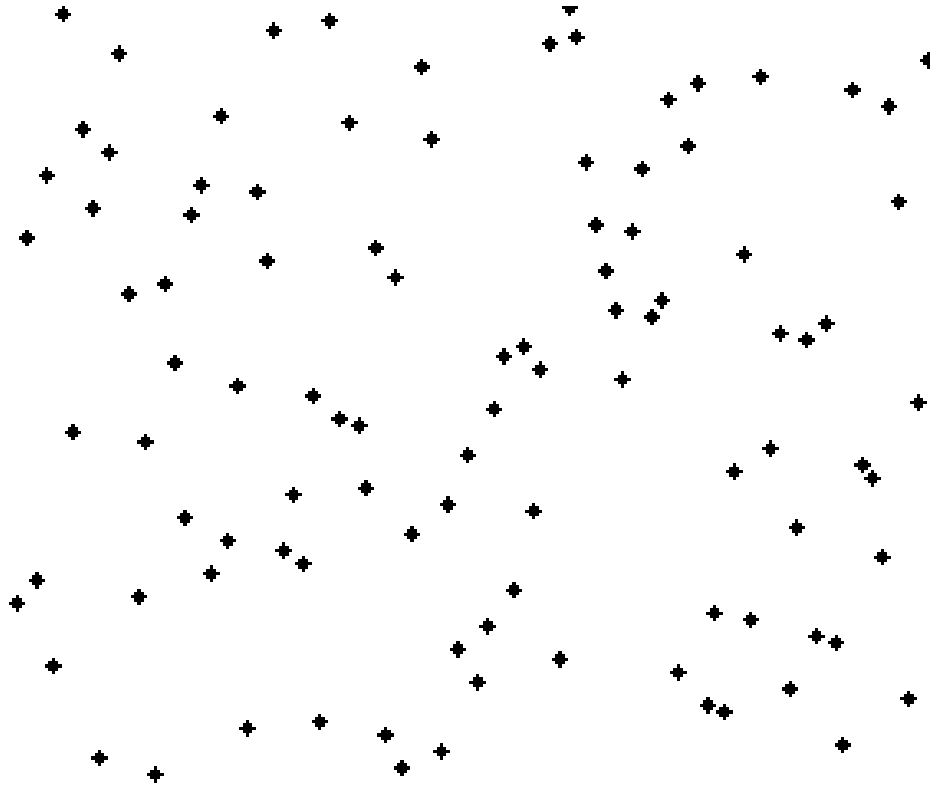


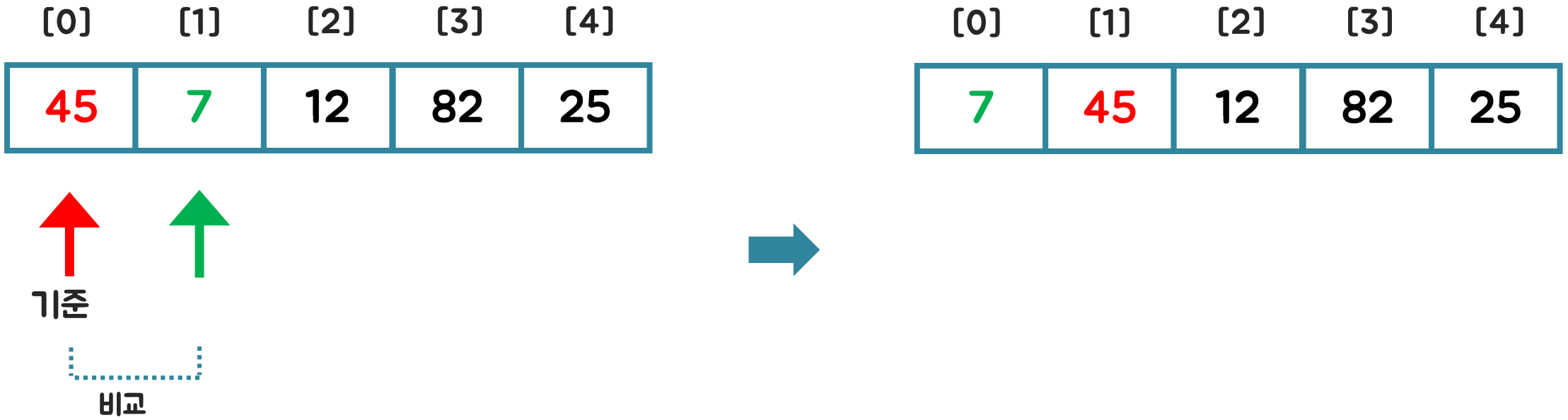
정렬 알고리즘

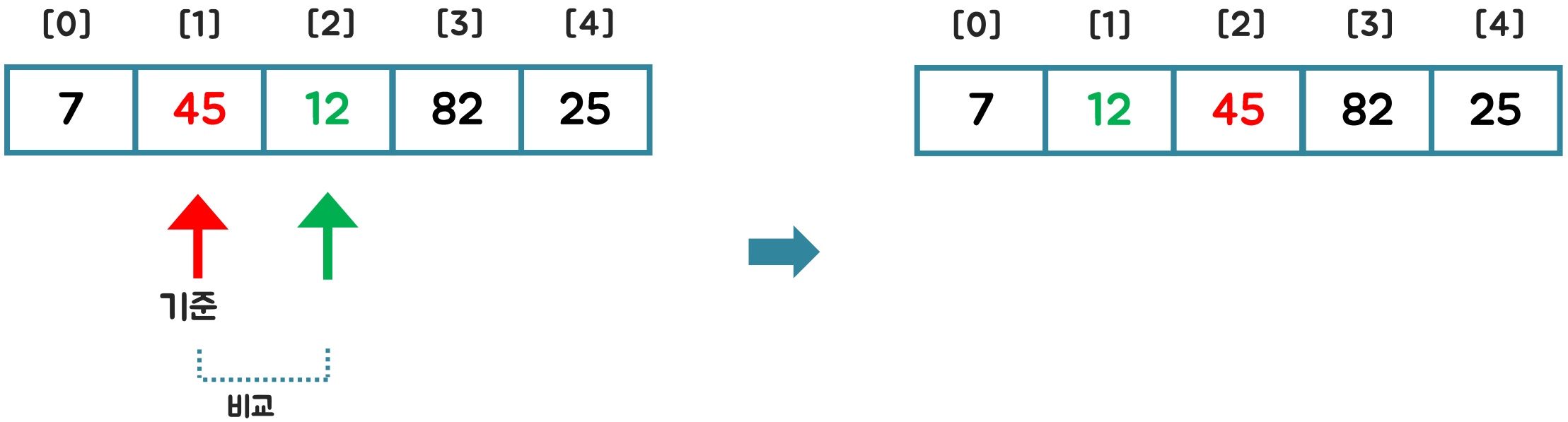
원소들을 일정한 순서대로 열거하는 알고리즘

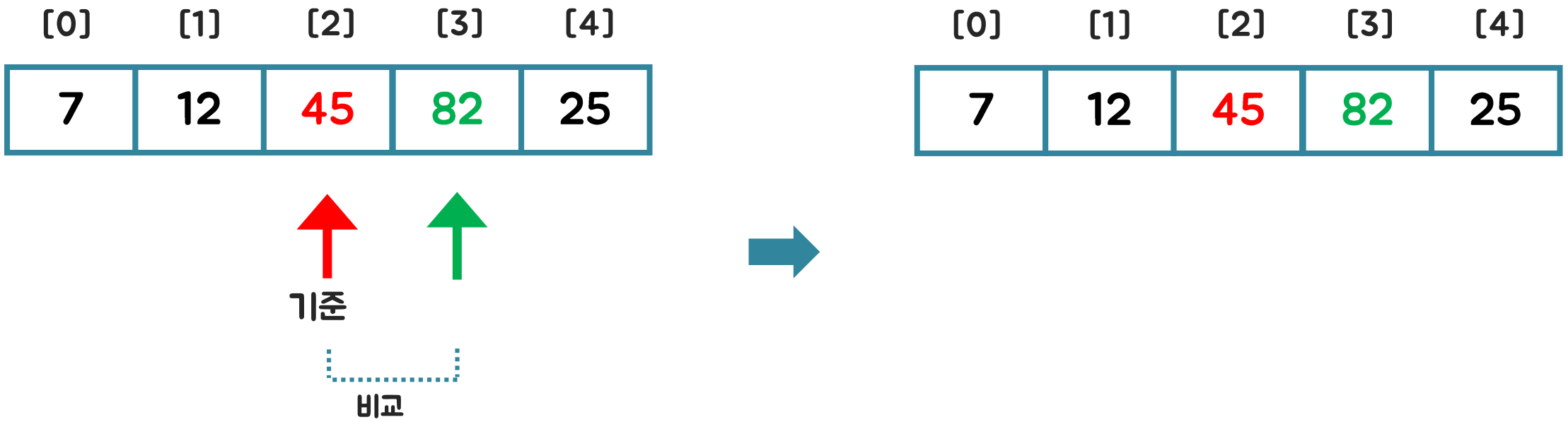
Bubble sort

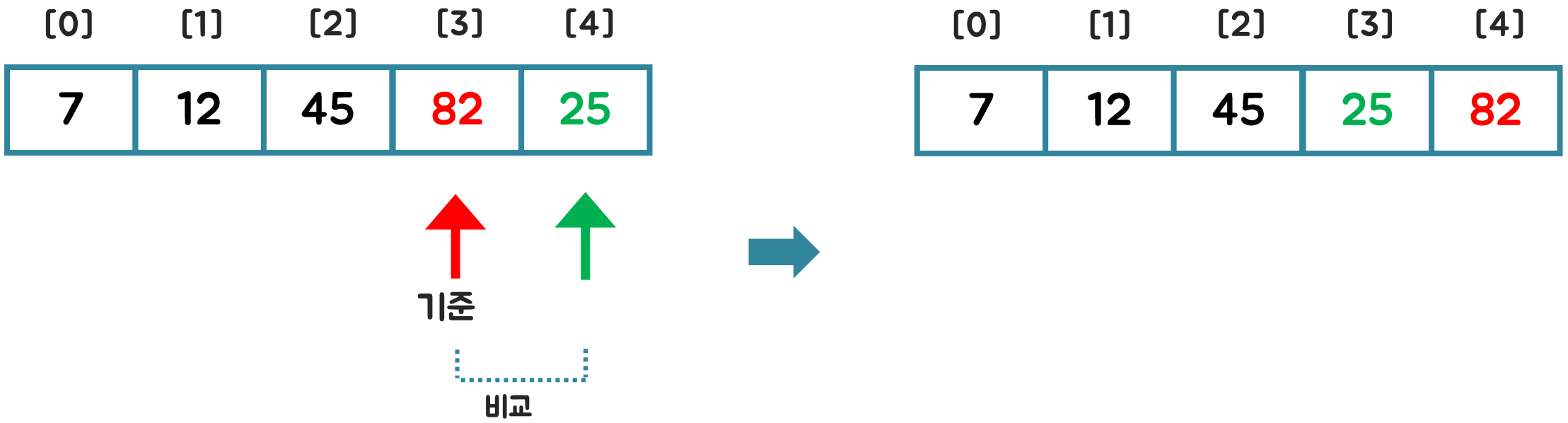
두 인접한 원소를 **비교**하여 정렬하는 방법
속도는 느리지만 코드가 단순하다.

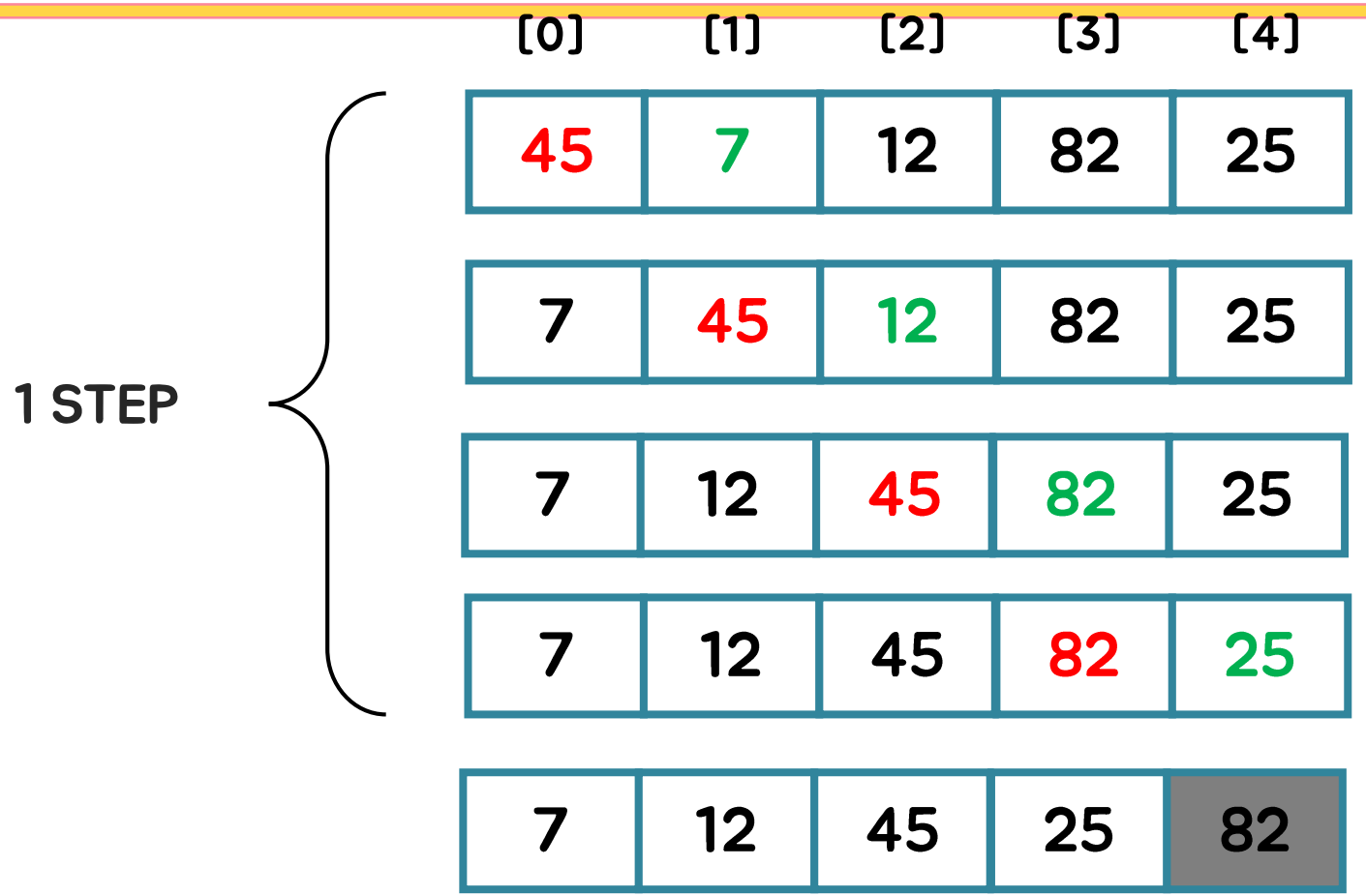


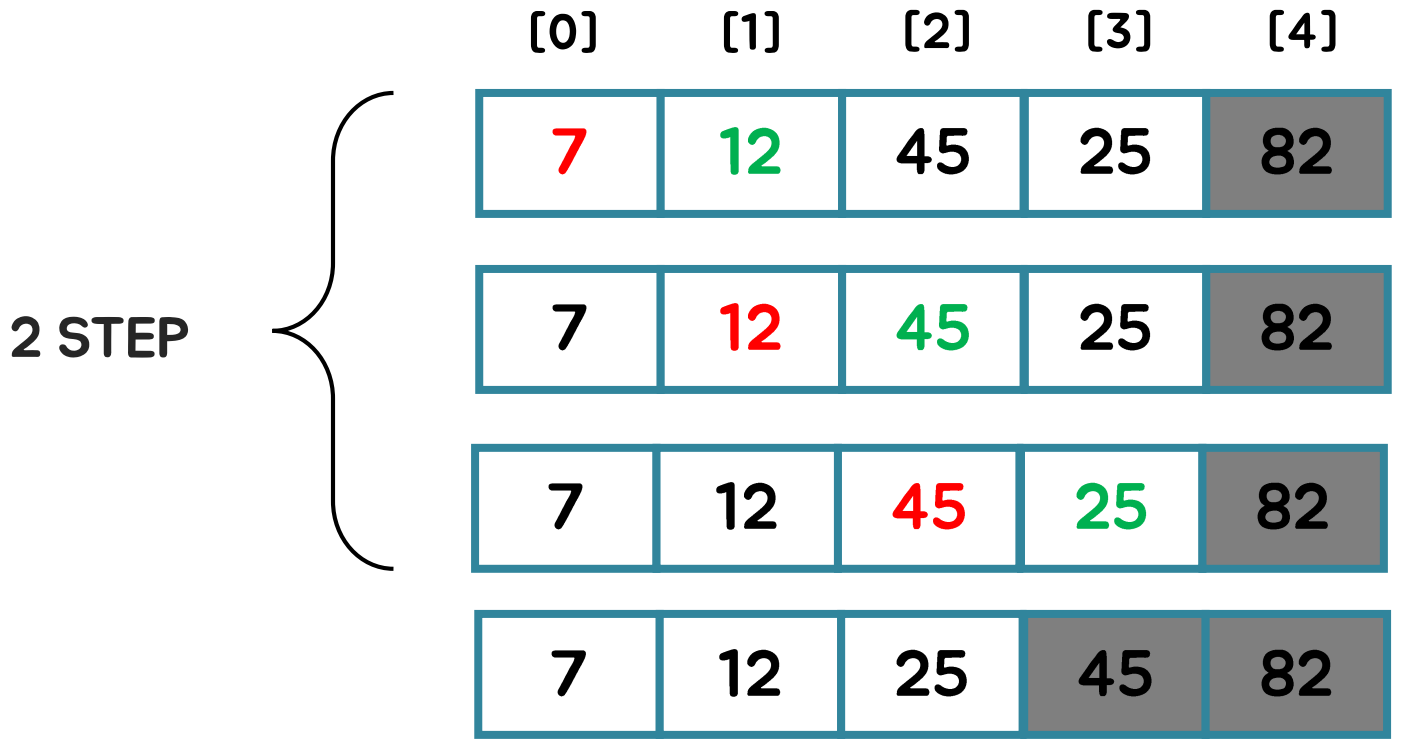

[0]
[1]
[2]
[3]
[4]
45
7
12
82
25
기준
비교
[0]
[1]
[2]
[3]
[4]
7
45
12
82
25


[0]
[1]
[2]
[3]
[4]
7
45
12
82
25
기준
비교
[0]
[1]
[2]
[3]
[4]
7
12
45
82
25


[0]
[1]
[2]
[3]
[4]
7
12
45
82
25
기준
비교
[0]
[1]
[2]
[3]
[4]
7
12
45
82
25

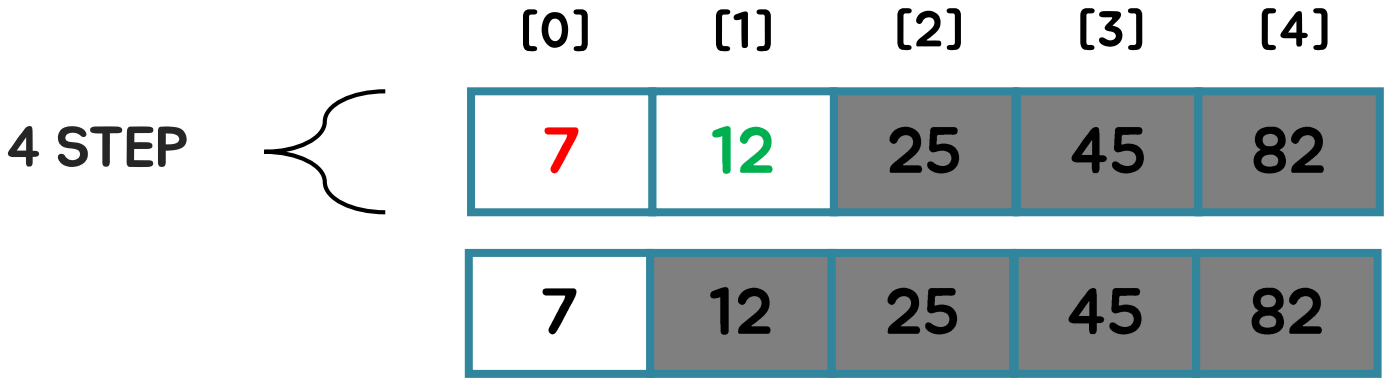

[0]
[1]
[2]
[3]
[4]
7
12
45
82
25
기준
비교
[0]
[1]
[2]
[3]
[4]
7
12
45
25
82


[0] [1] [2] [3] [4]
45 7 12 82 25
7 45 12 82 25
7 12 45 82 25
7 12 45 82 25
7 12 45 25 82
1 STEP


[0]
[1]
[2]
[3]
[4]
2 STEP
7 12 45 25 82
7 12 45 25 82
7 12 45 25 82
7 12 25 45 82

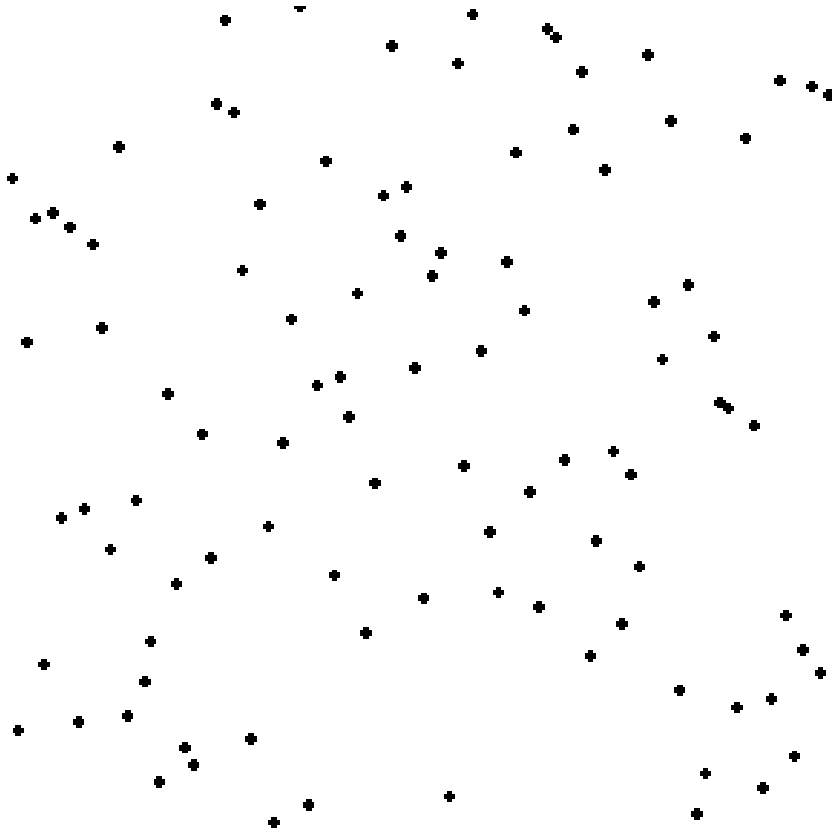
3 STEP

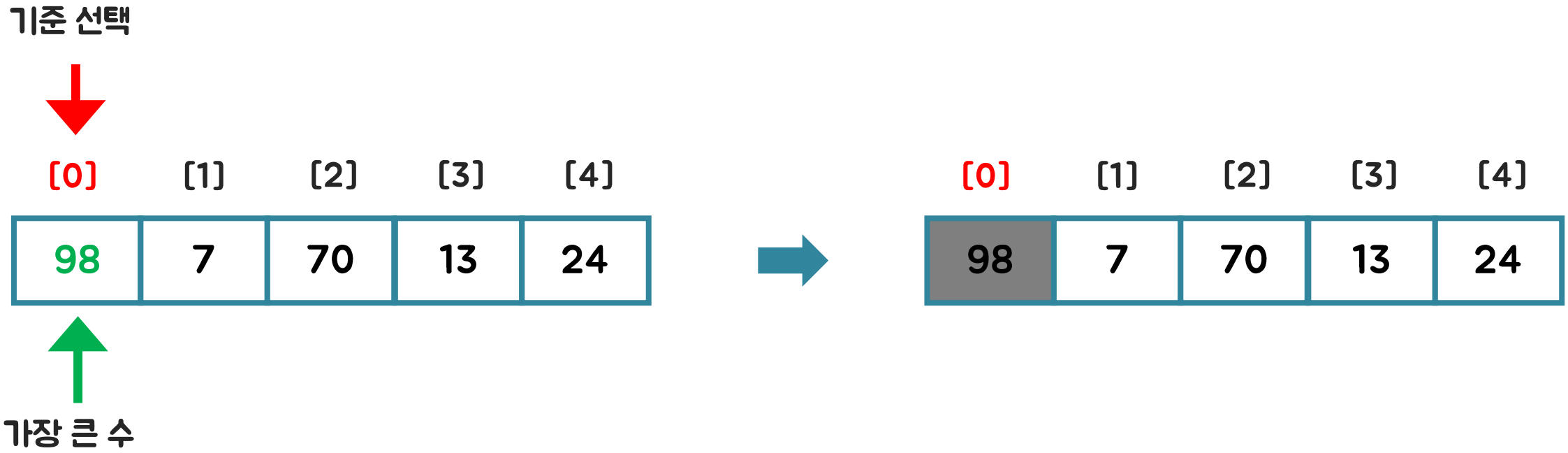
[0]	[1]	[2]	[3]	[4]
7	12	25	45	82
7	12	25	45	82
7	12	25	45	82

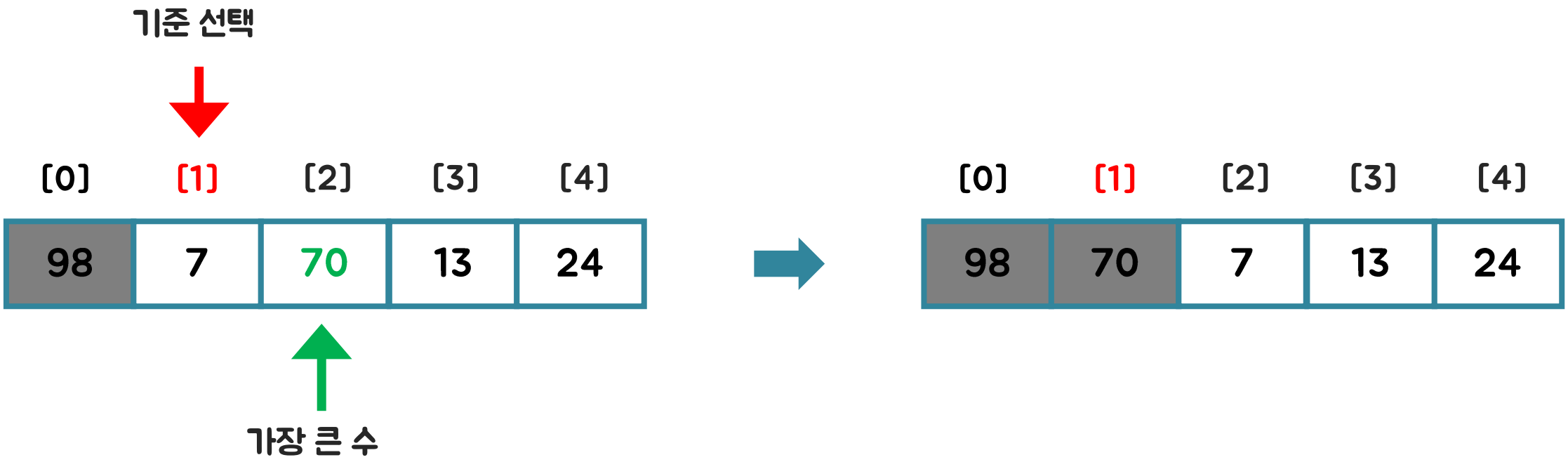

[0]
[1]
[2]
[3]
[4]
4 STEP
7
12
25
45
82
7
12
25
45
82

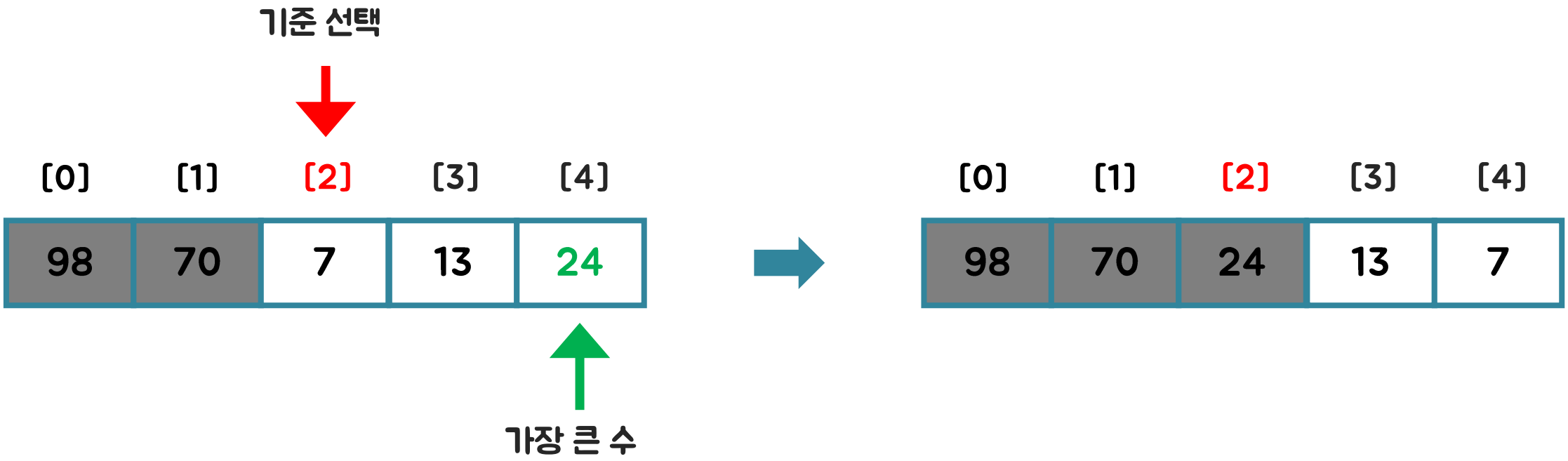
Selection sort

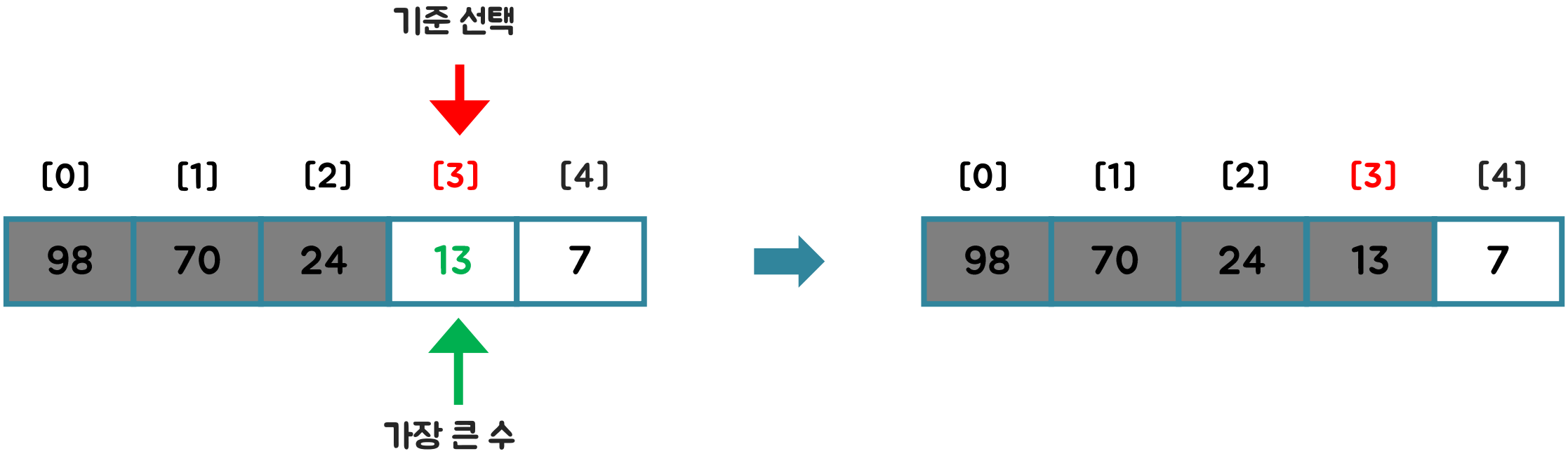
가장 큰 원소 또는 작은 원소를 찾아
주어진 위치(리스트 처음~끝)를 교체해 나가는 정렬 방법




기준 선택
[0]
[1]
[2]
[3]
[4]
98
7
70
13
24
가장 큰 수
[0]
[1]
[2]
[3]
[4]
98
7
70
13
24


기준 선택
[0]
[1]
[2]
[3]
[4]
98
7
70
13
24
가장 큰 수
[0]
[1]
[2]
[3]
[4]
98
70
7
13
24


기준 선택
[0]
[1]
[2]
[3]
[4]
98
70
7
13
24
가장 큰 수
[0]
[1]
[2]
[3]
[4]
98
70
24
13
7


기준 선택
[0]
[1]
[2]
[3]
[4]
98
70
24
13
7
가장 큰 수
[0]
[1]
[2]
[3]
[4]
98
70
24
13
7

검색 알고리즘

특정 원소를 검색하는 알고리즘

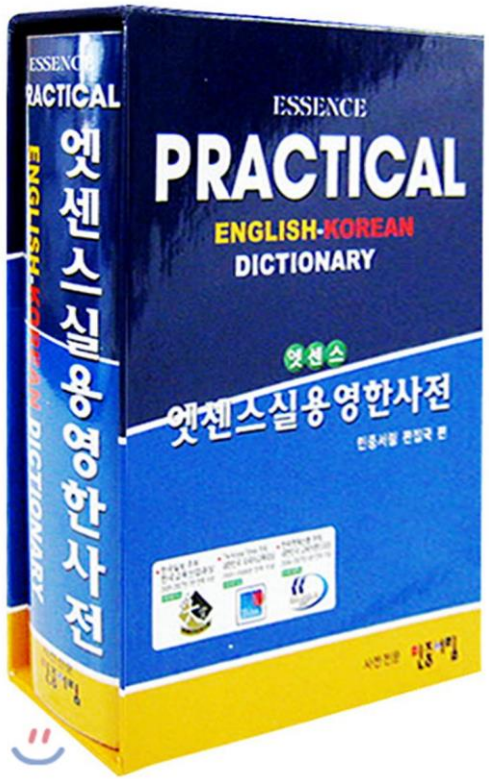
Sequential search

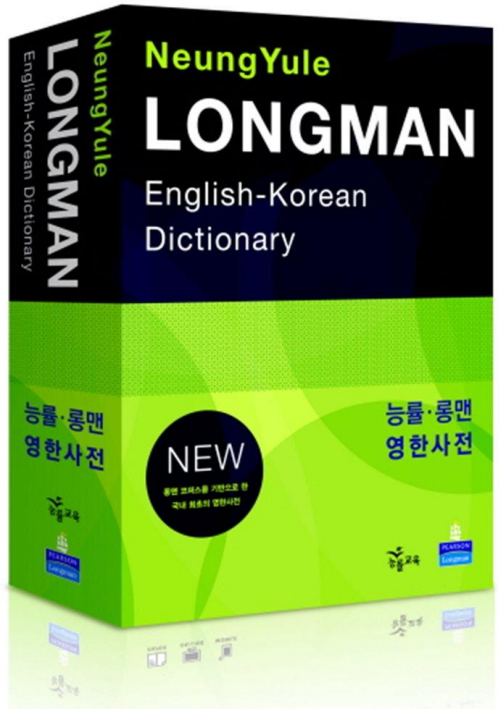
가장 단순한 검색 방법으로 원소의 정렬이 필요 없다.
하지만 리스트 길이가 길면 비효율적

처음부터 끝까지 순차적으로 비교

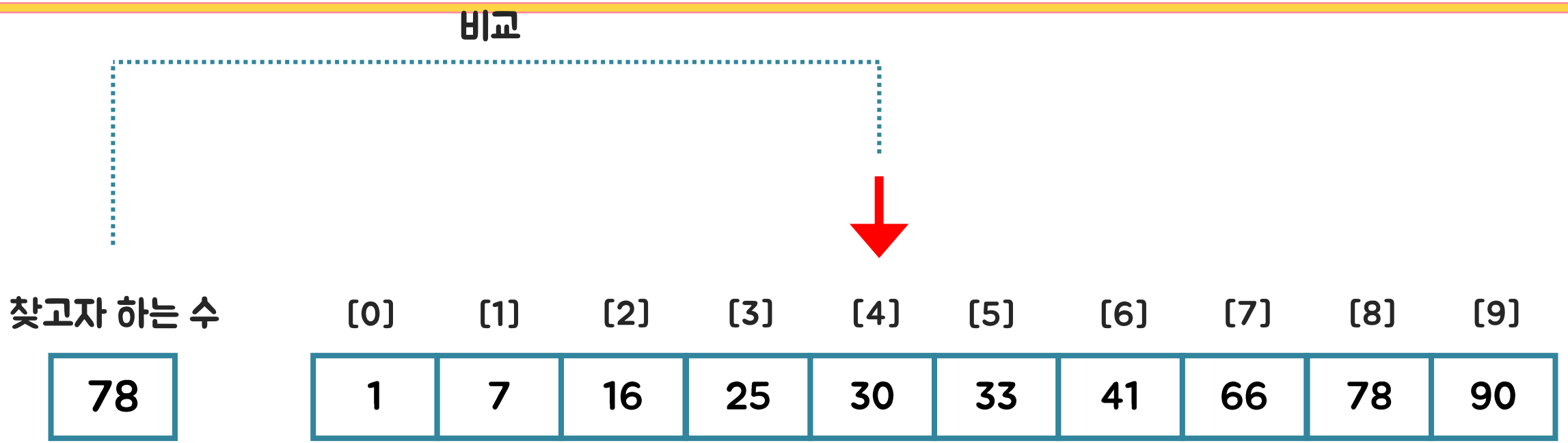
Binary search

리스트의 중간 값을 정해 크고 작음을 비교해
검색하는 알고리즘 정렬된 리스트에 사용 할 수 있다.


ESSENCE
PRACTICAL
ENGLISH-KOREAN
DICTIONARY
엣센스
엣센스실용영한사전

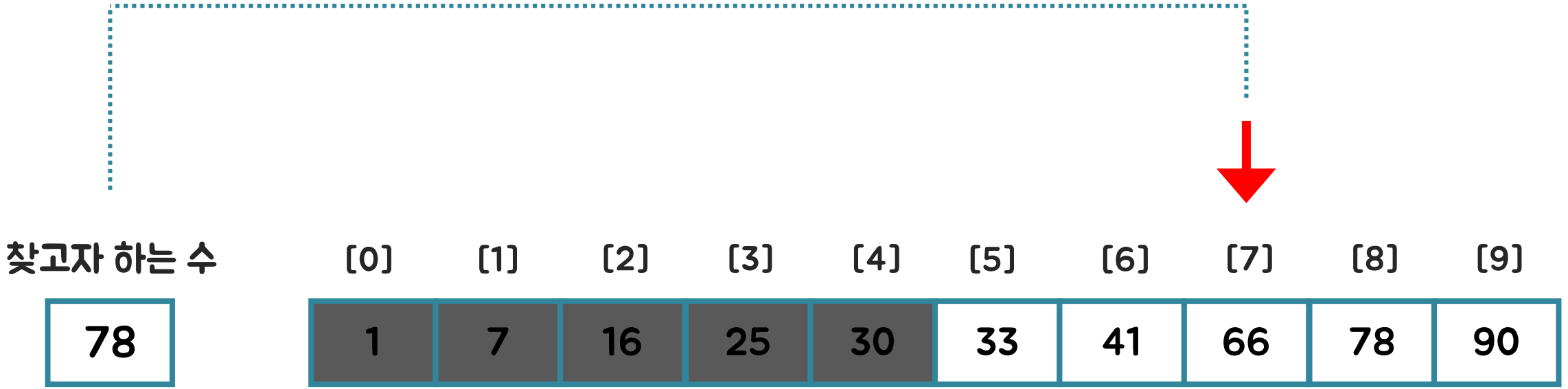

NeungYule
LONGMAN
English-Korean
Dictionary
능률·롱맨
영한사전
NEW


24!
30?
24! 업!
30? 다운!
2/14



lowIndex = 0, highIndex = 9

비교



lowIndex = 5, highIndex = 9

비교

찾고자 하는 수

78

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
1	7	16	25	30	33	41	66	78	90

lowIndex = 8, highIndex = 9

python 딕셔너리(dictionary) 란?

- dictionary는 단어 그대로 해석하면 사전이라는 뜻

- “people”이라는 단어는 “사람”, “baseball”이라는 단어는 “야구”에 부합되듯이 dictionary는 Key와 Value를 한쌍으로 갖는 자료형

- 딕셔너리 타입은 immutable한 key와 mutable한 value로 맵핑되어 있는 순서가 없는 집합

딕셔너리명 = {Key : Value, Key : Value, ... }

※ Key에는 변하지 않는 값을 사용하고,
Value에는 변하는 값과 변하지 않는 값 모두 사용할 수 있다.

```
a = {}

b = { "name" : "YH" }

c = { 1 : 5, 2 : 3}
```

python 딕셔너리(dictionary) 선언 및 타입 확인


```
dic1 = {"name":"YH", "age":20, "mbti":"ENFJ"}
print(dic1)
```

```
{'name': 'YH', 'age': 20, 'mbti': 'ENFJ'}
```

key	value
name	YH
age	20
mbti	ENFJ

딕셔너리 정보

python 딕셔너리(dictionary) 값 추가

딕셔너리명[key] = value

```
dic1 = {"name":"YH", "age":20, "mbti":"ENFJ"}
print(dic1)
```

```
{'name': 'YH', 'age': 20, 'mbti': 'ENFJ'}
```

```
dic1["birth"] = "08/01"
```

key	value
name	YH
age	20
mbti	ENFJ
birth	08/01

딕셔너리 정보

```
print(dic1)
```

```
{'name': 'YH', 'age': 20, 'mbti': 'ENFJ', 'birth': '08/01'}
```

python 딕셔너리(dictionary) 값 추가 실습

1. 변수 dic_song를 다음과 같이 만드시오.

key	value
노래제목	HOME SWEET HOME

2. 딕셔너리 추가를 통해 다음과 같이 정보를 저장 시키시오.

key	value
노래제목	HOME SWEET HOME
가수	G-DRAGON
날짜	2025.02.06

```
print(dic_song)
```

```
{'노래제목': 'HOME SWEET HOME', '가수': 'G-DRAGON', '날짜': '2025.02.06'}
```

del 딕셔너리명[key]

```
dic2 = {"name":"YH", "age":20, "mbti":"ENFJ"}
del dic2["age"]
print(dic2)
```

```
{'name': 'YH', 'mbti': 'ENFJ'}
```

key	value
name	YH
mbti	ENFJ

딕셔너리 정보

python 딕셔너리(dictionary) 값 가져오기

딕셔너리명[Key]

```
print(dic2)
```

```
{'name': 'YH', 'mbti': 'ENFJ'}
```

```
print(dic2["name"])
```

```
YH
```

```
print(dic2["mbti"])
```

```
ENFJ
```

key	value
name	YH
mbti	ENFJ

딕셔너리 정보

python 딕셔너리(dictionary) 값 가져오기

딕셔너리명.get(Key)

```
print(dic2)
```

```
{'name': 'YH', 'mbti': 'ENFJ'}
```

```
print(dic2.get("name"))
```

```
YH
```

```
print(dic2.get("mbti"))
```

```
ENFJ
```

key	value
name	YH
mbti	ENFJ

딕셔너리 정보

딕셔너리명[key] VS 딕셔너리명.get(Key)

```
dic2["성별"]
```

```
----------------------------
--
KeyError
t)
<ipython-input-31-e10b9ca4
----> 1 dic2["성별"]

KeyError: '성별'
```

```
temp = dic2.get("성별")
print(temp)
```

```
None
```

False

python 딕셔너리(dictionary) Key만 가져오기

딕셔너리명.keys()

```
dic3 = {"name":"YH", "age":20, "mbti":"ENFJ"}
print(dic3.keys())
```

```
dict_keys(['name', 'age', 'mbti'])
```

```
print(type(dic3.keys()))
list3 = list(dic3.keys())
print(list3)
print(type(list3))
```

```
<class 'dict_keys'>
['name', 'age', 'mbti']
<class 'list'>
```

key	value
name	YH
age	20
mbti	ENFJ

딕셔너리 정보

딕셔너리명.values()

```
dic3 = {"name":"YH", "age":20, "mbti":"ENFJ"}
print(dic3.values())
```

```
dict_values(['YH', 20, 'ENFJ'])
```

key	value
name	YH
age	20
mbti	ENFJ

딕셔너리 정보

딕셔너리 for문 활용

```
for key in dic3.keys():
    print(key)
```

```
name
age
mbti
```

```
for value in dic3.values():
    print(value)
```

```
YH
20
ENFJ
```

```
for key, value in dic3.items():
    print(key, value)
```

```
name YH
age 20
mbti ENFJ
```

key in 딕셔너리명

- in은 딕셔너리의 키에 한에서 동작한다.

```
print("name" in dic3)
```

```
True
```

```
print("성별" in dic3)
```

```
False
```

```
print("YH" in dic3)
```

```
False
```

딕셔너리명.clear()

```
print(dic3)
```

```
{'name': 'YH', 'age': 20, 'mbti': 'ENFJ'}
```

```
dic3.clear()
print(dic3)
```

```
{}
```

여러 학생의 성적점수가 Dictionary로 아래와 같이 구성되어있다.
과목 별 합을 구하여 새로운 Dictionary로 구성하시오.

```
dic_score = {"나예호" : {"수학" : 55, "영어" : 23, "국어" : 41},
             "공유" : {"영어" : 67, "국어" : 87, "수학" : 67},
             "수지" : {"수학" : 99, "국어" : 75, "영어" : 80}}
```

```
{'수학': 221, '국어': 203, '영어': 170}
```