파이썬 프로그래밍

8. 자료구조와 알고리즘

00 Contents

❖ 수업 목표

- 리스트 데이터 SWAP 프로그램을 작성할 수 있다.
- 최대값 찾기 알고리즘을 이해할 수 있다.
- 선택 정렬 알고리즘을 이해할 수 있다.
- 딕셔너리를 매개변수로 활용할 수 있다.

❖ 세부 목표

00

1. 리스트 데이터 SWAP

❖ 변수 데이터 값 서로 바꾸기

- na=10, nb=20을 nb=10, na=20으로 스왑
- 함수 사용 하지 않기

```
na=10
nb=11
print("na=", na, "nb=", nb)
temp=na
na=nb
nb=temp
print("na=", na, "nb=", nb)
```

〈화면 출력〉

```
na=10 nb=11
na=11 nb=10
```

00

1. 리스트 데이터 SWAP

❖ 리스트 데이터 값 서로 바꾸기

- 리스트 내부 할당 데이터 서로 바꾸기

```
ca=[10, 11]
print("ca[0]의 값은", ca[0], "ca[1]의 값은", ca[1])
temp=ca[0]
ca[0]=ca[1]
ca[1]=temp
print("ca[0]의 값은", ca[0], "ca[1]의 값은", ca[1])
```

00

1. 리스트 데이터 SWAP

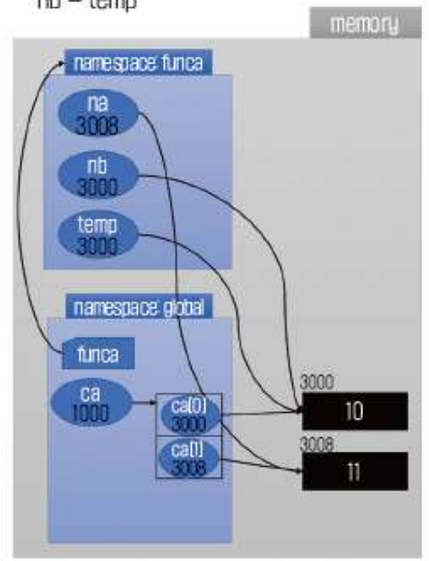
- swap을 함수로 정의하고 호출
  - ca 리스트의 요소를 서로 바꾸도록 funca 함수 정의
  - 리스트 요소 각각을 인수로 전달
  - 결과: 스왑 안됨
    (함수 영역 내부 처리 후 외부 전달 안됨)

```
                    ④  ⑤
def funca(na, nb): # ---> ⑪
    temp=na # ---> ⑦
    na=nb # ---> ⑧
    nb=temp # ---> ⑨
ca=[10, 11] # ---> ①
⑥ na=ca[0] ②
  nb=ca[1] ③
print("ca[0] 값은", ca[0], "ca[1] 값은", ca[1])
funca(ca[0], ca[1]) # ---> ⑩
print("ca[0] 값은", ca[0], "ca[1] 값은", ca[1])
                                    # ---> ⑪
```

〈화면 출력〉

```
ca[0] 값은 10 ca[1] 값은 11
ca[0] 값은 10 ca[1] 값은 11
```

```
def funca(na, nb)
    temp = na
    na = nb
    nb = temp
```




```
ca = [10, 11]
funca(ca[0], ca[1])
```

00

1. 리스트 데이터 SWAP

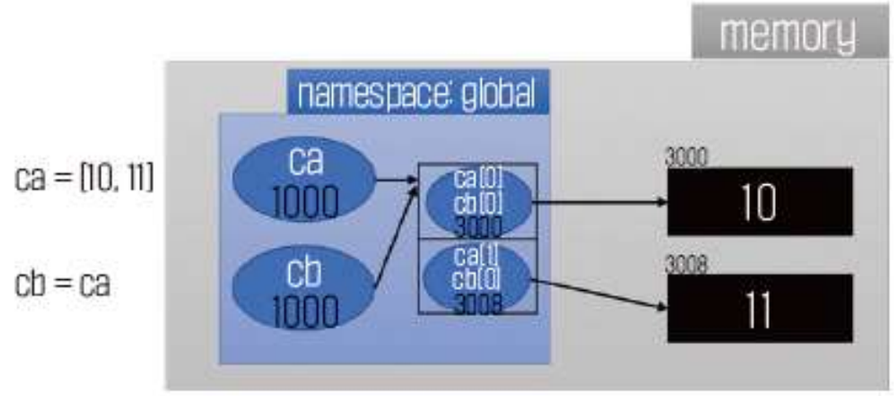
❖ 리스트에 새 이름(별명) 추가

- 새 변수에 기 생성된 리스트를 할당 => 얕은 복사
- 메인 메모리에 변수 공간을 생성하고 기존의 리스트 주소를 할당

```
ca=[10, 11]
cb=ca
print("리스트ca 값은", ca)
print("리스트cb 값은", cb)
```

〈화면 출력〉

```
리스트ca 값은 [10, 11]
리스트cb 값은 [10, 11]
```



00

1. 리스트 데이터 SWAP

❖ 리스트에 새 이름 (별명) 추가

- 리스트에 별명 붙이고 별명 변수명으로 데이터 맞교환
  - 리스트 이름을 할당(얕은 복사)하여 데이터를 맞교환

```
ca=[10, 11]
cb=ca # ----------------------> ①
print("리스트ca 값은", ca)
print("리스트cb 값은", cb)
temp=cb[0]
cb[0]=cb[1]  ②
cb[1]=temp
print("리스트ca 값은", ca) # --------> ③
print("리스트cb 값은", cb)
print("ca[0]=", ca[0], "ca[1]=", ca[1])
print("cb[0]=", cb[0], "cb[1]=", cb[1])
```

〈화면 출력〉

```
리스트ca 값은 [10, 11]
리스트cb 값은 [10, 11]
리스트ca 값은 [11, 10]
리스트cb 값은 [11, 10]
ca[0]=11 ca[1]=10
cb[0]=11 cb[1]=10
```

00

1. 리스트 데이터 SWAP

▪ 리스트 이름을 매개변수로 사용한 SWAP

• ca=[10,20]을 서로 맞바꾸는 funca 함수 정의

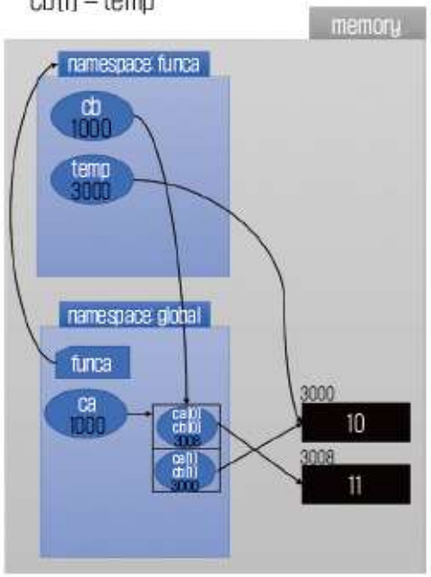
- 리스트명에 별명을 붙이는 코드와 swap 기능 부분을 funca 함수로 정의
  - » 별명을 붙이는 과정 cb=ca를 함수 정의 시 매개변수와 함수 호출 시 인수의 관계로 표현

```
def funca(cb):
    temp=cb[0]
    cb[0]=cb[1]
    cb[1]=temp
ca=[10, 11]
print("ca[0] 값은", ca[0], "ca[1] 값은 ", ca[1])
funca(ca) # ---------------> ①
print("ca[0] 값은", ca[0], "ca[1] 값은 ", ca[1])
```

〈화면 출력〉

```
ca[0] 값은 10 ca[1] 값은  11
ca[0] 값은 11 ca[1] 값은  10
```

```
def funca(cb)
    temp = cb[0]
    ca[0] = cb[1]
    cb[1] = temp
```




```
ca = [10, 11]
funca(ca)
```

2. id 함수와 리스트

❖ 리스트 메모리 활용

- 정수 4개로 구성된 리스트 a 정의

```
a=[10, 11, 12, 13]
print("리스트 a 값", a)
b=a
print("리스트 b 값",b)
```

〈화면 출력〉

```
리스트 a 값 [10, 11, 12, 13]
리스트 b 값 [10, 11, 12, 13]
```

2. id 함수와 리스트

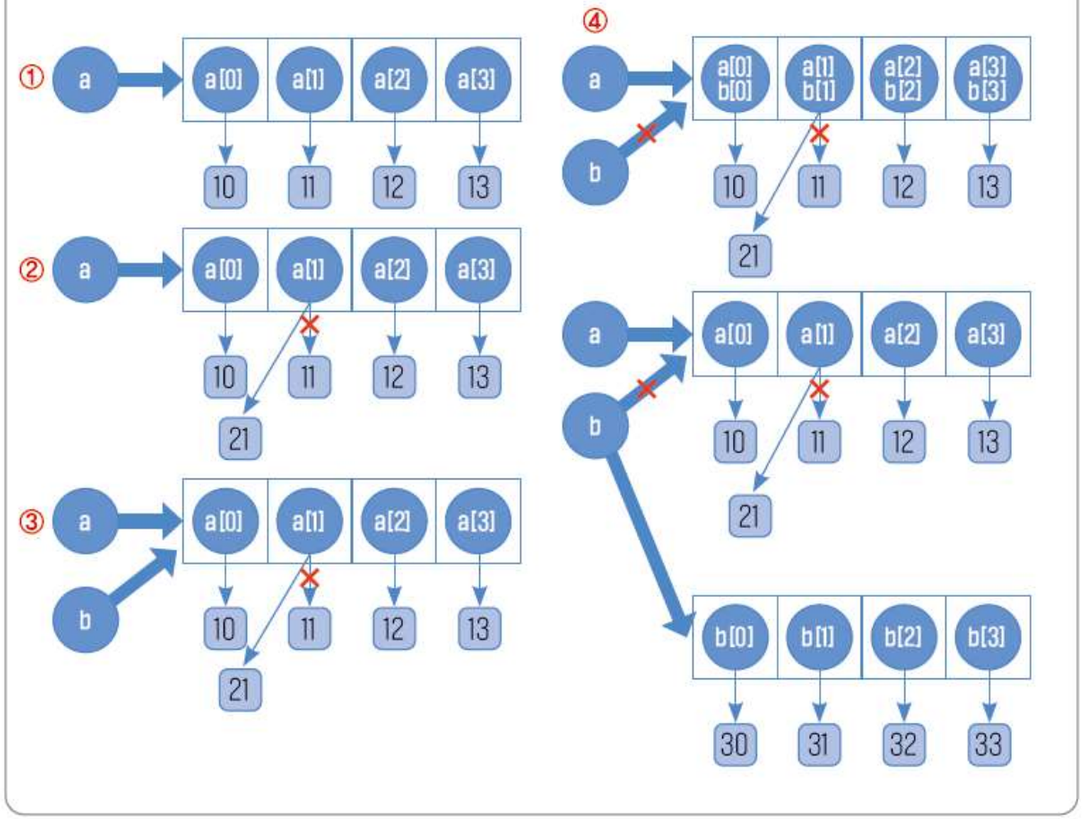
❖ 리스트 메모리 활용

- 리스트 변화에 따른 메모리 활용

```
a=[10,11,12,13] # --> ①
print("리스트 a 값", a)

a[1]=21 # --> ②
print("리스트 a 값",a)

b=a # --> ③
print("리스트 b 값",b)

b=[30, 31, 32, 33] # --> ④
print("리스트 b 값",b)
```

〈화면 출력〉

```
리스트 a 값 [10, 11, 12, 13]
리스트 a 값 [10, 21, 12, 13]
리스트 b 값 [10, 21, 12, 13]
리스트 b 값 [30, 31, 32, 33]
```



2. id 함수와 리스트

❖ id 함수와 리스트

- **id() 함수: 메모리 주소 값 반환**

```
a=[10, 11, 12, 13]
print(id(a))
print(id(a[0]))
print(id(a[1]))
print(id(a[2]))
a[1]=21
print(id(a))
print(id(a[0]))
print(id(a[1]))
print(id(a[2]))
b=a
print(id(b))
print(id(b[0]))
print(id(b[1]))
print(id(b[2]))
b=[30, 31, 32, 33]
print(id(b))
print(id(b[0]))
print(id(b[1]))
print(id(b[2]))
print(id(b[3]))
```

〈화면 출력〉

```
2483374094024
1721550800
1721550832
1721550864
2483374094024
1721550800
1721551152
1721550864
2483374094024
1721550800
1721551152
1721550864
2483374160520
1721551440
1721551472
1721551504
1721551536
```

메모리 주소이므로 코딩 결과는 다르게 나온다.

2. id 함수와 리스트

❖ id 함수와 리스트

- 리스트를 return 하는 함수 예제 프로그램

```
def fk(cb):
    total=0
    for sb in range(0, 3, 1):
        total=total+cb[sb]
    cb[2]=total
    return cb # ---> ①
ca=[10, 20, 30]
print(ca)
cd=fk(ca) # ---> ②
print(ca) # ---> ③
print(cd) # ---> ④
```

〈화면 출력〉

```
[10, 20, 30]
[10, 20, 60]
[10, 20, 60]
```

3. 최대값 찾기(1)

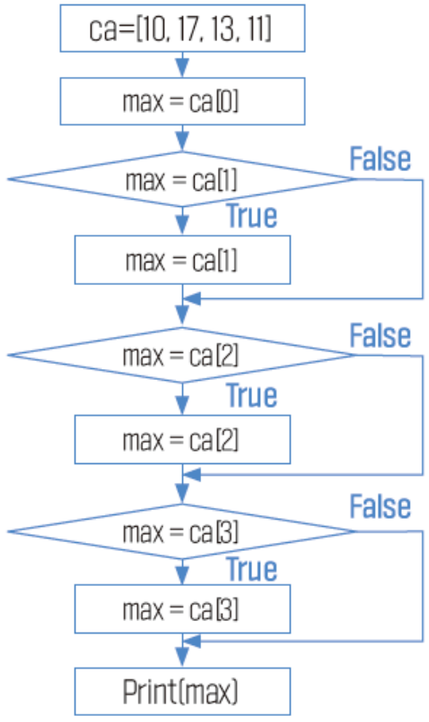
❖ 최대값 찾기

- 문제1) 주어진 리스트에서 최대값을 구하는 프로그램 작성하기
  - ca = [10, 17, 13, 11]
- 순서도에 따라 최댓값을 찾는 알고리즘 작성

```
ca=[10,17,13,11]
max=ca[0]
if max < ca[1]:
    max=ca[1]
if max < ca[2]:
    max=ca[2]
if max < ca[3]:
    max=ca[3]
print(max)
```

〈화면 출력〉

```
17
```



3. 최대값 찾기(1)

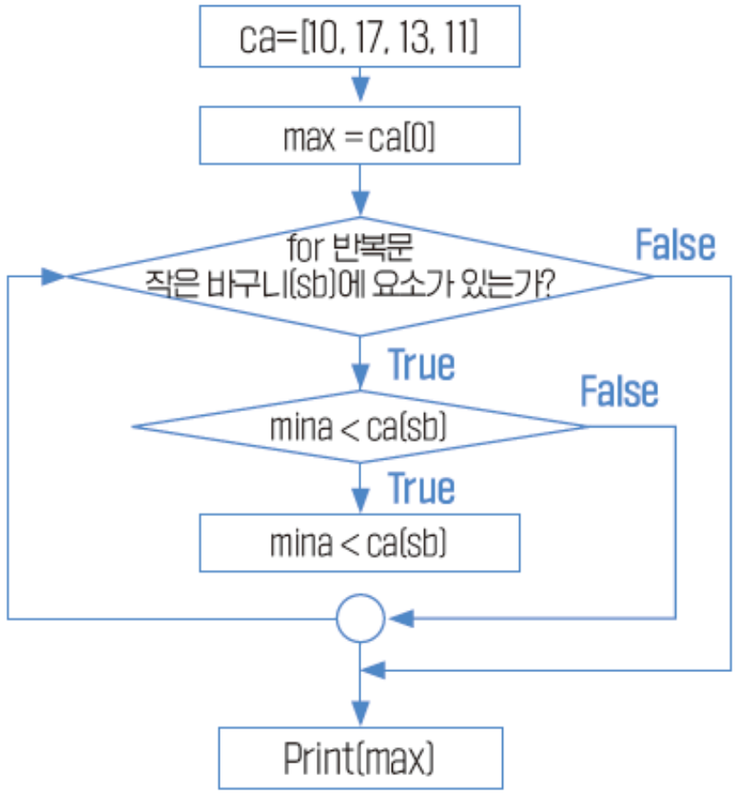
❖ 최대값 찾기

- 문제2) 리스트 변수의 규칙성을 이용, range를 사용한 for문으로 변경하기

```
ca=[10,17,13,11]
max=ca[0]
for sb in range(1, 4, 1):
    if max < ca[sb]:
        max=ca[sb]
print(max)
```

〈화면 출력〉

```
17
```



3. 최대값 찾기(1)

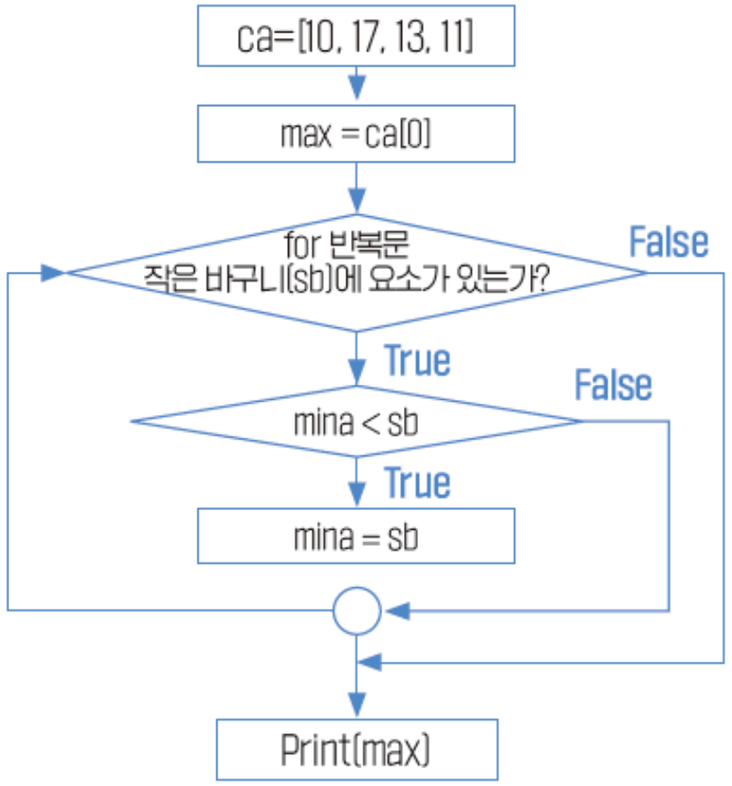
❖ 최대값 찾기

- 문제3) for 문 작성 시 리스트에서 직접 요소를 하나씩 가져와 최댓값 찾기

```
ca=[10,17,13,11]
max=ca[0]
for sb in ca:
    if max < sb:
        max=sb
print(max)
```

〈화면 출력〉

17



4. 선택 정렬

❖ 선택 정렬 알고리즘

- 주어진 리스트 요소 중 최소값을 찾아 첫 번째 위치의 요소와 서로 교환하는 과정을 모든 요소가 정렬될 때까지 반복하는 알고리즘
  - 첫 번째 수를 min 변수에 저장
  - 두 번째 수와 비교하여 두 번째 수가 더 작으면 min 변수에 저장
- 최대값을 찾아서 swap 하는 알고리즘 참고

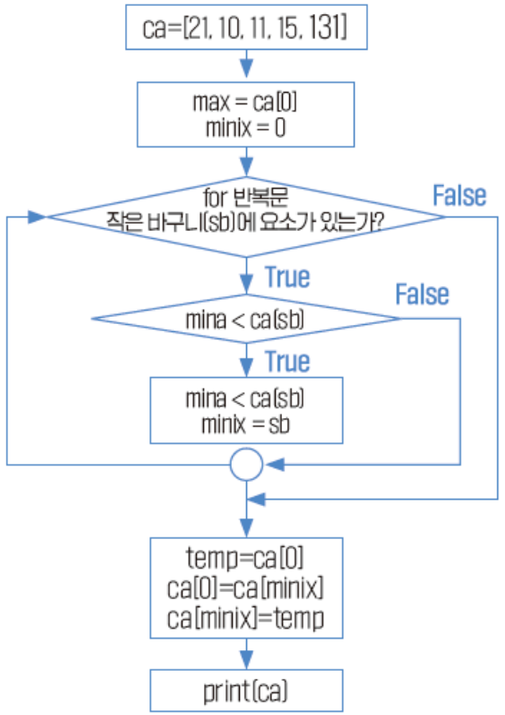
4. 선택 정렬

- **Step (1)**
  - 리스트의 요소 중 가장 작은 최소값(10)을 찾아 minix 변수에 저장하기
  - ca[0]과 ca[minix]의 값을 서로 교환하여 0번 인덱스에 최소값 위치하기

```
# 선택 정렬
ca=[21,10,11,15,13]
mina=ca[0] # ---> ①
minix=0 # --> ②
for sb in range(1, 5, 1): # --> ③
    if mina > ca[sb]: # --> ④
        mina=ca[sb]
        minix=sb

temp=ca[0] # --> ⑤
ca[0]=ca[minix]
ca[minix]=temp
print(ca)
```

〈화면 출력〉

```
[10, 21, 11, 15, 13]
```



00

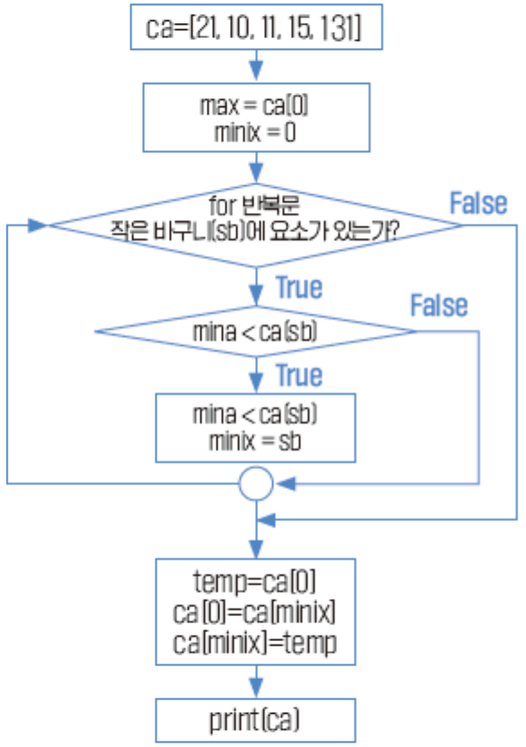
4. 선택 정렬

- **Step (2)**
  - 나머지 4개 요소 중 가장 작은 수와 ca[1]에 위치한 수를 서로 교환하는 코드 추가

```
# 선택 정렬
ca=[21,10,11,15,13]
mina=ca[0]
minix=0
for sb in range(1, 5, 1):
    if mina > ca[sb]:
        mina=ca[sb]
        minix=sb
temp=ca[0]
ca[0]=ca[minix]
ca[minix]=temp
print(ca)
mina=ca[1] # ---> ①
minix=1 # ---> ②
for sb in range(2, 5, 1): # ---> ③
    if mina > ca[sb]: # ---> ④
        mina=ca[sb]
        minix=sb

temp=ca[1] # --> ⑤
ca[1]=ca[minix]
ca[minix]=temp
print(ca)
```

〈화면 출력〉

```
[10, 21, 11, 15, 13]
[10, 11, 21, 15, 13]
```



4. 선택 정렬

- **Step (3), (4)**
  - 리스트 2번 인덱스에 2번, 3번, 4번 인덱스 값 중 최소값을 찾아 2번 인덱스와 서로 swap하는 과정 추가
  - 3번 인덱스에 3번, 4번 인덱스 값 중 최소값을 찾아 3번 인덱스와 서로 swap하는 과정 추가

```
mina = ca[2]
minix = 2
for sb in range(3, 5, 1):
    if mina > ca[sb]:
        mina = ca[sb]
        minix = sb
temp = ca[2]
ca[2] = ca[minix]
ca[minix] = temp
print(ca)
```

〈화면 출력〉

```
[10, 21, 11, 15, 13]
[10, 11, 21, 15, 13]
[10, 11, 13, 15, 21]
```

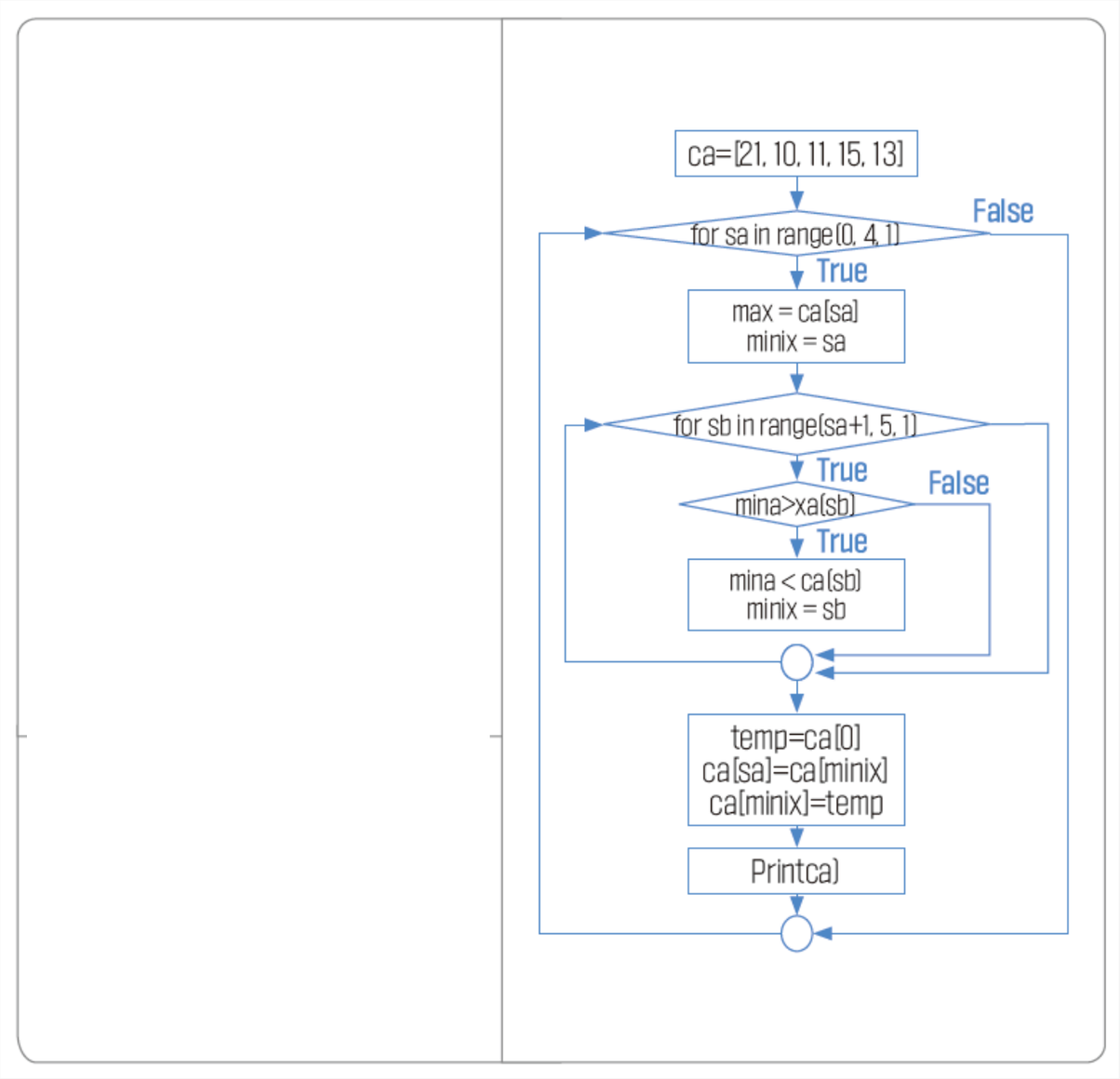
```
mina = ca[3]
minix = 3
for sb in range(4, 5, 1):
    if mina > ca[sb]:
        mina = ca[sb]
        minix = sb
temp = ca[3]
ca[3] = ca[minix]
ca[minix] = temp
print(ca)
```

〈화면 출력〉

```
[10, 21, 11, 15, 13]
[10, 11, 21, 15, 13]
[10, 11, 13, 15, 21]
[10, 11, 13, 15, 21]
```

4. 선택 정렬

- **Step (5)**
  - **중첩된 for 문으로 변경하기**



4. 선택 정렬

- **Step (6)**
  - **앞서 작성한 프로그램에서 선택 정렬 부분을 fselsort 함수로 정의하고 호출하여 데이터를 정렬하는 프로그램 완성하기**

```
ca=[21,10,11,15,13]
fselsort(ca)
```

5. 최대값 찾기(2)

❖ 최대값 찾기

- 최대값 반환 프로그램 작성
  - 데이터를 리스트로 정의
  - 최대값 반환 함수 fmax 정의 및 호출

```
su=[5, 4, 7, 10, 6]
```

5. 최대값 찾기(2)

- **매개변수와 인수 (1)**
  - **규칙성을 이용하기 위해 매개변수를 리스트로 정의**

```
su=[5, 4, 7, 10, 6]

def fmax(a, b, c, d, e):
    max=a
    if max < b:
        max=b
    if max < c:
        max=c
    if max < d:
        max=d
    if max < e:
        max = e
    return max
a=su[0]
b=su[1]
c=su[2]
d=su[3]
e=su[4]

max=fmax(su[0],su[1],su[2],su[3],su[4])
print("최대값 max는", max)
```

```
su=[5, 4, 7, 10, 6]

def fmax(a, b, c, d, e):
    fu=[a, b, c, d, e]
    max=fu[0]
    if max < fu[1]:
        max=fu[1]
    if max < fu[2]:
        max=fu[2]
    if max < fu[3]:
        max=fu[3]
    if max < fu[4]:
        max = fu[4]
    return max
a=su[0]
b=su[1]
c=su[2]
d=su[3]
e=su[4]
max=fmax(su[0],su[1],su[2],su[3],su[4])
print("최대값 max는", max)
```

00

5. 최대값 찾기(2)

- 매개변수와 인수 (1)
  - 리스트로 만들어진 규칙성을 활용하여 for 문으로 변경
  - range 함수를 활용 => 리스트 요소 직접 이용

```
su=[5, 4, 7, 10, 6]

def fmax(a, b, c, d, e):
    fu=[a, b, c, d, e]
    max=fu[0]
    for i in range(1, 5, 1):
        if max < fu[i]:
            max = fu[i]
    return max
a=su[0]
b=su[1]
c=su[2]
d=su[3]
e=su[4]
max=fmax(su[0],su[1],su[2],su[3],su[4])
print("최대값 max는", max)
```

```
su=[5, 4, 7, 10, 6]

def fmax(a, b, c, d, e):
    fu=[a, b, c, d, e]
    max=fu[0]
    for sfu in fu:
        if max < sfu:
            max = sfu
    return max
a=su[0]
b=su[1]
c=su[2]
d=su[3]
e=su[4]
max=fmax(su[0],su[1],su[2],su[3],su[4])
print("최대값 max는", max)
```

5. 최대값 찾기(2)

- 매개변수와 인수 (2)
  - 빈 리스트를 정의하고 매개변수 값을 리스트에 추가하는 프로그램 작성하기
  - 매개변수로 직접 리스트를 입력 받도록 수정

```
su=[5, 4, 7, 10, 6]

def fmax(a, b, c, d, e):
    fu=[]
    fu.append(a)
    fu.append(b)
    fu.append(c)
    fu.append(d)
    fu.append(e)
    max=fu[0]
    for sfu in fu:
        if max < sfu:
            max=sfu
    return max

max=fmax(su[0],su[1],su[2],su[3],su[4])
print("최대값 max는", max)
```

```
su=[5, 4, 7, 10, 6]

def fmax(fu):
    max=fu[0]
    for sfu in fu:
        if max < sfu:
            max=sfu
    return max

max=fmax(su)
print("최대값 max는", max)
```

00

5. 최대값 찾기(2)

- 매개변수와 인수 (2)
  - range 함수를 이용한 for문으로 수정
    - fu = su 는 불필요한 코드이나 시각적 이해를 돕기 위해 추가됨

```
su=[5, 4, 7, 10, 6]
def fmax(fu):
    max=fu[0]
    for i in range(1, 5, 1):
        if max < fu[i]:
            max=fu[i]
    return max
fu = su
max=fmax(su)
print("최대값 max는", max)
```

00

5. 최대값 찾기(2)

- 문제) 주어진 리스트를 사용하여 최소값 구하는 함수 fmin을 정의 및 호출하기

```
su = [5, 4, 7, 10, 6]
# 함수 정의 작성




mina = fmin(su)
print("최소값 mina는", mina)
```

6. 딕셔너리 매개변수 활용

❖ 딕셔너리를 매개변수로 사용하는 코드 (1)

- 딕셔너리 데이터 할당에 따른 값 변화 확인

```
a={1:"월", "tue":"화", "wed":"수"}
print(a)
x=a # ----------> ①
print(x) # -----> ②
x[1]="일" # ----> ③
print(x) # -----> ④
print(a) # -----> ⑤
```

〈화면 출력〉

```
{1: '월', 'tue': '화', 'wed': '수'}
{1: '월', 'tue': '화', 'wed': '수'}
{1: '일', 'tue': '화', 'wed': '수'}
{1: '일', 'tue': '화', 'wed': '수'}
```

- ① a 변수 주소를 x 변수에 할당하여 x도 a 데이터를 가리키도록 함
- ② x와 a가 같은 데이터를 참조하는지 확인
- ③ 딕셔너리 x에서 key가 1인 value “월” 을 “일” 로 변경
- ④, ⑤에서 변경된 것을 확인 가능

6. 딕셔너리 매개변수 활용

❖ 딕셔너리를 매개변수로 사용하는 코드 (2)

- 딕셔너리 자료형을 함수의 매개변수로 사용하기

```
def fch(x):
    x[1]="일" # ----> ③
a={1:"월", "tue":"화", "wed":"수"}
print(a)
fch(a)
print(a)
```

〈화면 출력〉

```
{1: '월', 'tue': '화', 'wed': '수'}
{1: '일', 'tue': '화', 'wed': '수'}
```

Notice

❖ 과제

- 1. 리스트 데이터 스왑 코드 작성하기
- 2. id 함수 사용 결과 확인하기
- 3. 최대값 찾는 알고리즘 코드 작성하기
- 4. 선택 정렬 알고리즘 코드 작성하기
- 5. 딕셔너리 매개변수 활용 코드 작성하기

❖ 다음 수업 내용

- 클래스
  - 클래스 구성, 정의, 객체, self, __init__() 메서드, 내장 클래스