

# 스크래치

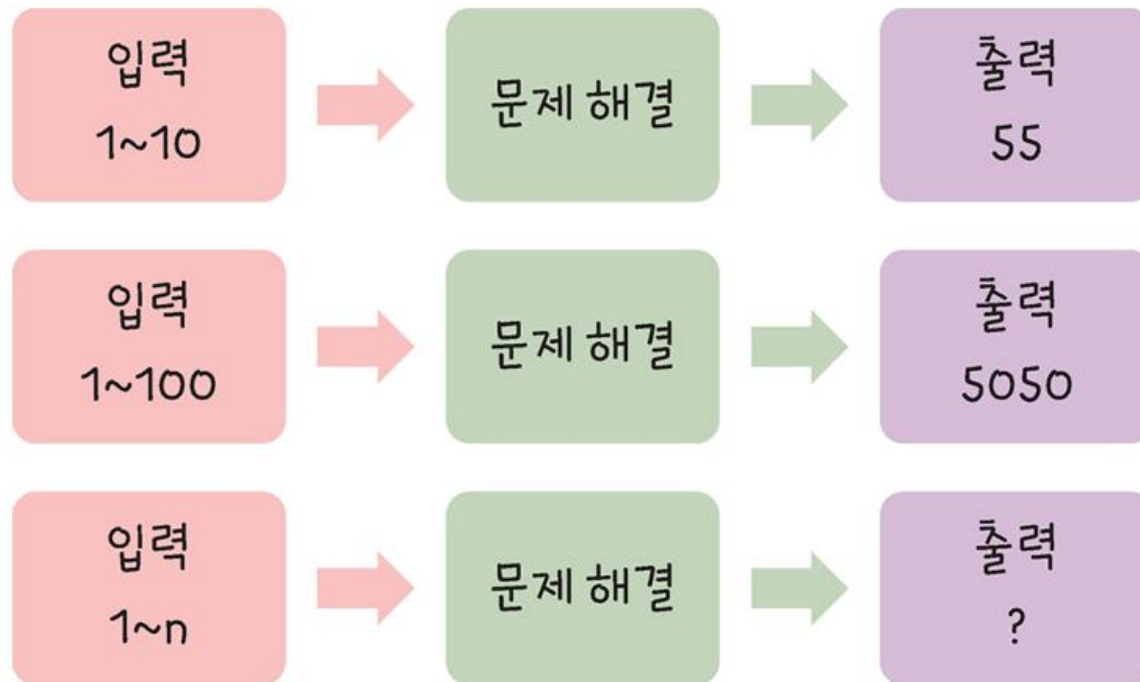
DAY 06 – 1부터 n까지 더하기

```
#include <iostream>
using namespace std;

int main() {
    cout<<"Hola Facebook\n";
    return 0;
}
```

# 1. 1부터 n까지 합 이해하기

- 1부터 10까지 합은 간단히 계산 가능
- 1부터 100까지 합은 어렵지만 어떻게든 구할 수 있음
- 그렇다면 1부터 n까지 합을 구하려면 어떻게 계산해야 할까?



# 1. 1부터 n까지 합 이해하기

- 알고리즘에서 가장 핵심: 효율성
  - 컴퓨터가 계산하더라도 수행 횟수를 줄여 좀 더 빠르게!
- 두 가지 방법
  - 1) 순서대로 더하기
  - 2) 첫째 항 1과 마지막 항  $n$ 을 더한 후  $n/2$ 을 곱하기

# 알고리즘 1. 1부터 순서대로 더하기

## ■ 1부터 10까지 합

합계(s)	1	+	2	+	3	+	...	+	10
연산 횟수	1번		2번		3번		...		10번

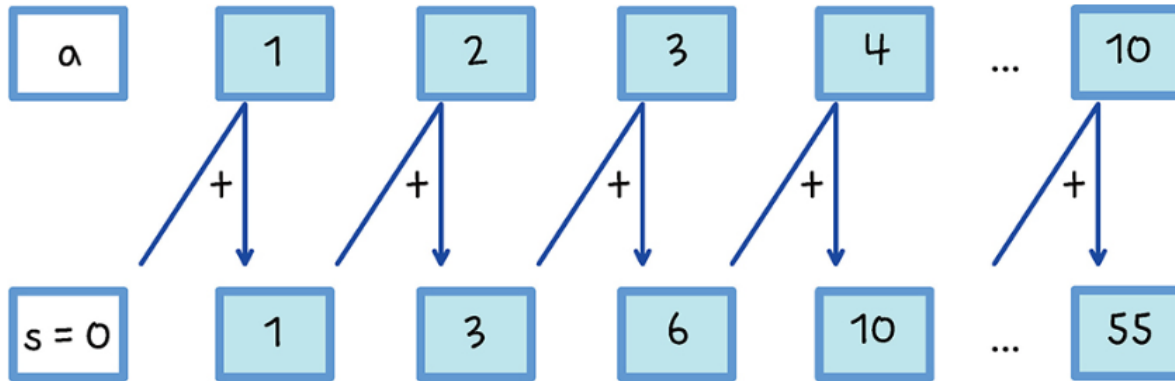
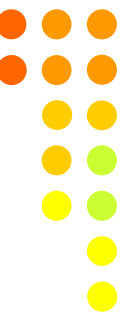
## ■ 1부터 100까지 합

합계(s)	1	+	2	+	3	+	...	+	10	+	...	+	100
연산 횟수	1번		2번		3번		...		10번		...		100번

## ■ 1부터 n까지 합

합계(s)	1	+	2	+	3	+	...	+	10	+	...	+	100	+	...	+	n
연산 횟수	1번		2번		3번		...		10번		...		100번		...		n번

# 규칙 찾아보기



- $a$ : 숫자를 1씩 증가시키는 데 필요한 변수
- $s$ :  $a$ 를 누적시키는 데 필요한 변수
- $n$ : 어디까지 더할지를 알려 주는 데 필요한 변수

초기값

$a = 1, s = 0, n = 10$

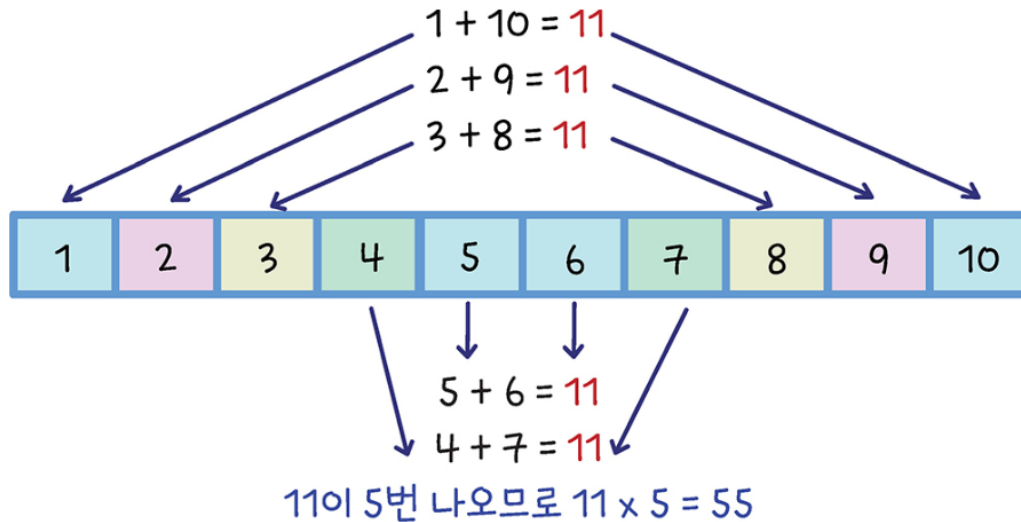
반복 실행

$a = a + 1, s = s + a$  ← 10번 반복



## 알고리즘 2. 첫째 항과 마지막 항 n을 더한 후 n/2을 곱하기

- 1부터 10까지 합을 구하는 다른 방법



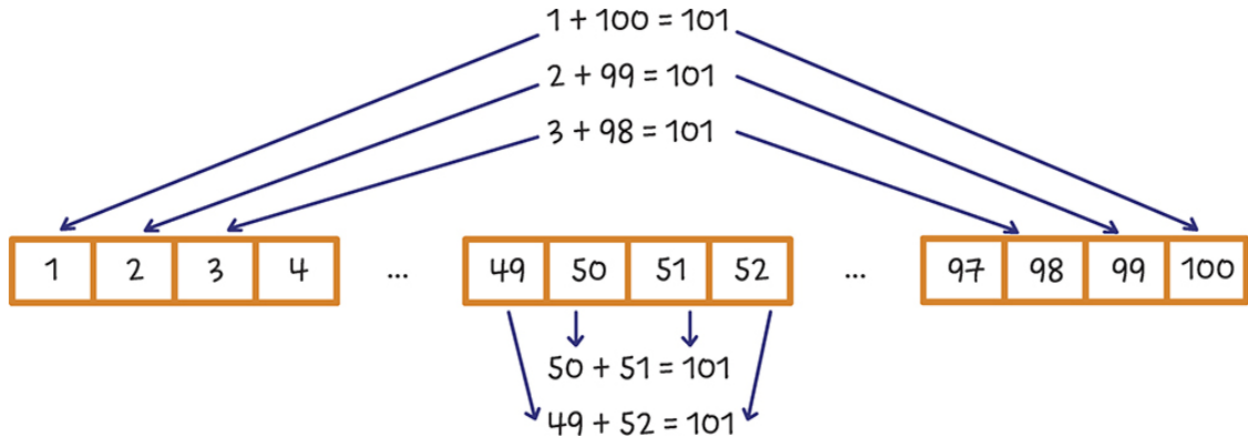
- 식으로 표현

$$10 \times (10 + 1) \div 2$$

← 덧셈 1번  
곱셈 1번 → 나눗셈 1번

# 알고리즘 2

## 1에서 100까지 합 구하기



## 식으로 표현

$$100 \times (100 + 1) \div 2$$

← 덧셈 1번  
곱셈 1번 → 나눗셈 1번

## 알고리즘 2



- 1부터 n까지 합

$$\boxed{100} \times (\boxed{100} + \boxed{1}) \div \boxed{2}$$

← 덧셈 1번  
곱셈 1번 → 나눗셈 1번



$$\boxed{n} \times (\boxed{n} + \boxed{1}) \div \boxed{2}$$

← 덧셈 1번  
곱셈 1번 → 나눗셈 1번

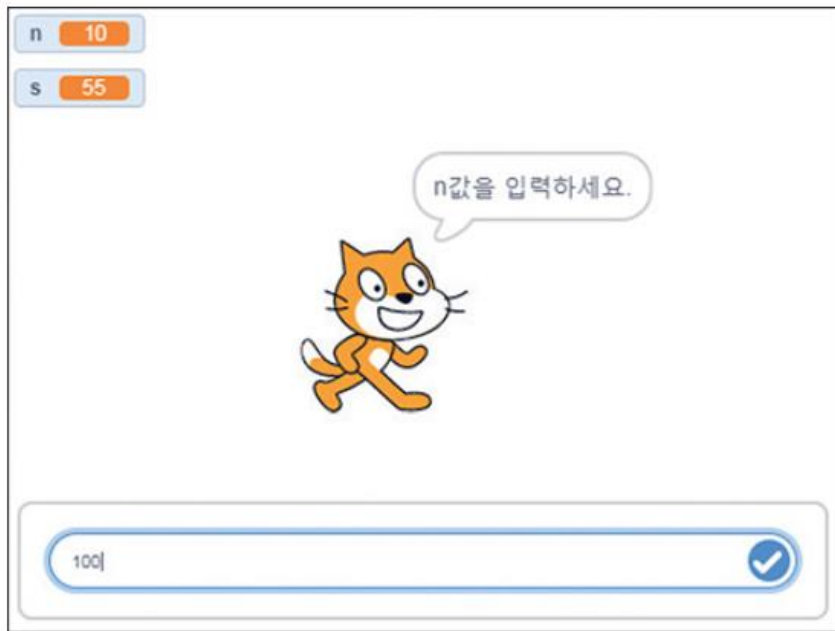
- n 변수: 몇까지 합을 구할지 결정하는 변수
- s 변수: n값을 사용하여 합계를 저장할 변수, 합계를 출력합니다.





### 3. 스크래치에서 1부터 n까지 더하기

- 스크래치에서 n값을 입력받아 1부터 n까지 합을 구해 보자. n=10 이면 1부터 10까지 합으로 55를 표시하고, n=100 이라면 1부터 100까지 합으로 5050을 표시한다. 사용자가 n 값을 자유롭게 입력하고, 그 결과로 나온 합을 표시하도록 한다.



### 3. 스크래치에서 1부터 n까지 더하기

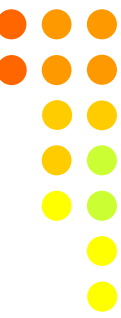
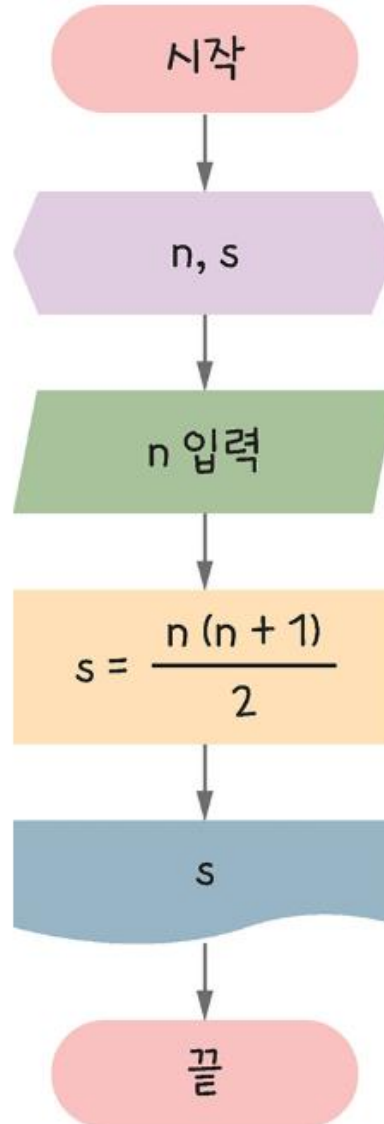
- 먼저 어떤 변수가 필요한지 고려
  - n 변수: 몇까지 합을 구할지 결정하는 변수
  - s 변수: n값을 사용하여 합계를 저장할 변수
- 실행 과정



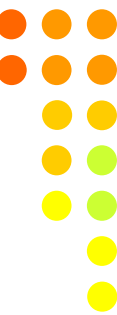
### 3. 스크래치에서 1부터 n까지 더하기



- 알고리즘을 순서도로 표현

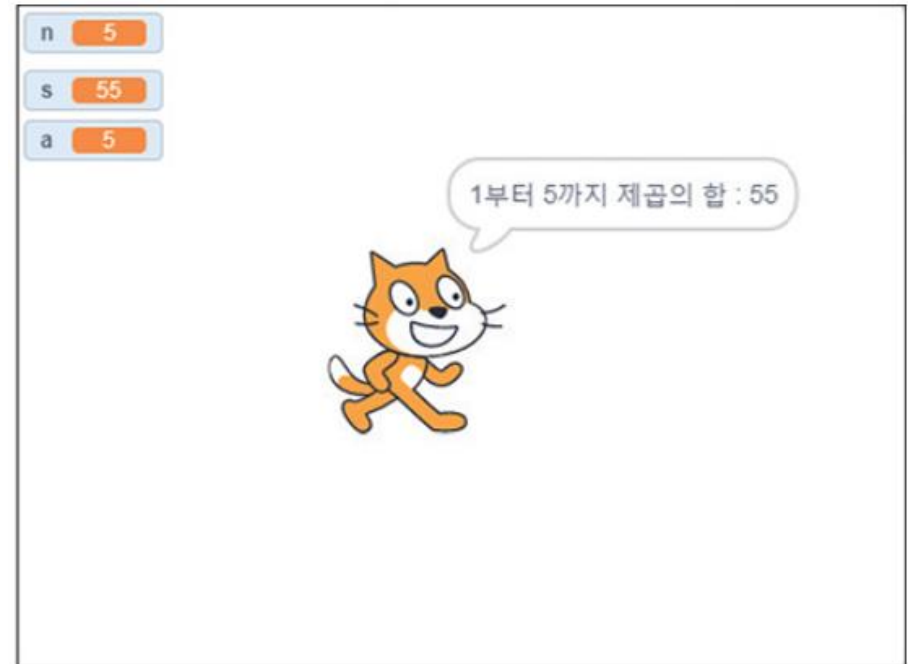
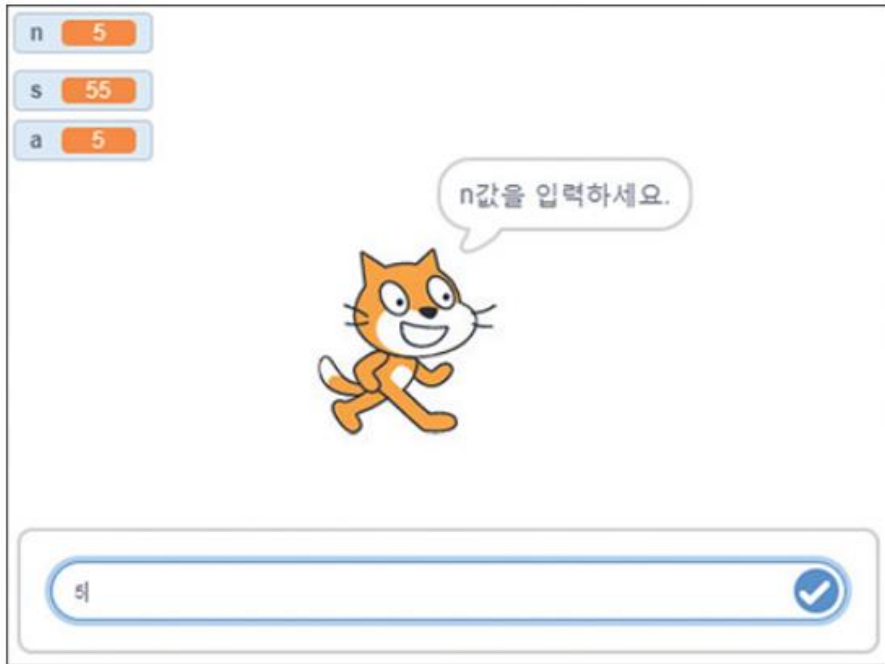


# 스크래치로 구현하기



## 4. 연습 문제

- 1부터  $n$ 까지 제곱의 합( $1^2+2^2+3^2+\dots+n^2$ )을 두 가지 방식의 알고리즘으로 풀어 보고, 그중 효율적인 알고리즘으로 구현해 보자.



- 힌트**
  - 1부터 10까지 제곱의 합은 385이다.
  - 수식은  $\frac{n(n+1)(2n+1)}{6}$  이다.

# 스크래치

## DAY 07 – 최대값 찾기

```
#include <iostream>
using namespace std;

int main() {
    cout<<"Hola Facebook\n";
    return 0;
}
```

# 1. 최댓값이란?

- 최댓값이란 주어진 숫자 중에서 가장 큰 값을 의미

35    20    10    40    15

- 주어진 데이터 5개를 저장할 공간이 필요
- 이때 변수 5개를 사용하기보다는 좀 더 간단한 **리스트**를 사용해보자.

## 2. 리스트 이해하기

- 데이터 여러 개를 저장하는 공간으로 **리스트**가 유용함

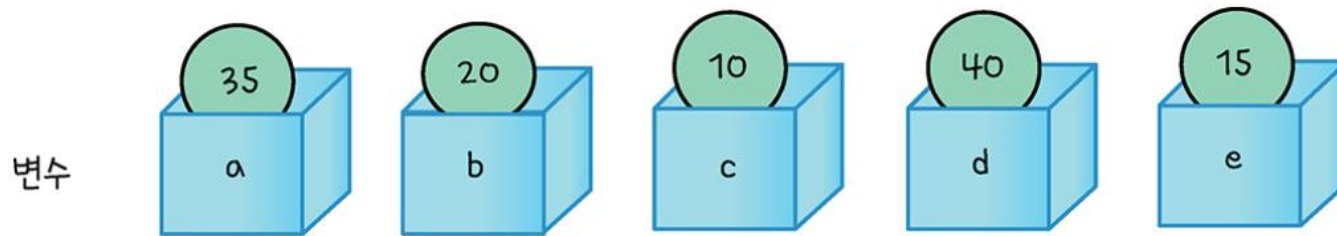
- 변수: 데이터 1개를 저장하는 공간
- 리스트: 데이터 여러 개를 저장하는 공간



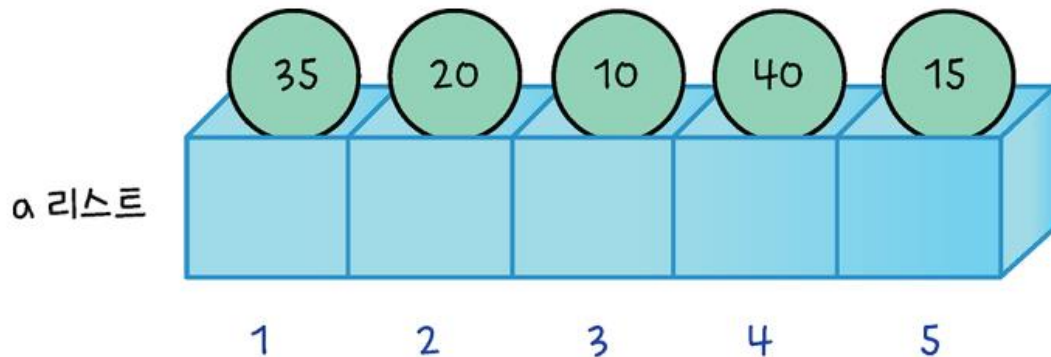


## 2. 리스트 이해하기

- 변수를 이용하여 5개 숫자 저장 → 5개 변수 필요

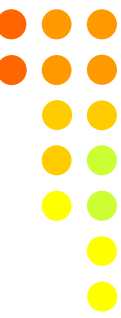


- 리스트를 사용하여 5개 숫자 저장 → 1개 리스트 필요



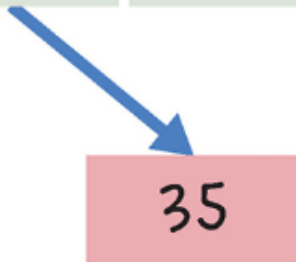
$a[i]$   
↓  
첨자  
↓  
리스트 이름

### 3. 최댓값 찾기 알고리즘



	[1]	[2]	[3]	[4]	[5]
a	35	20	10	40	15

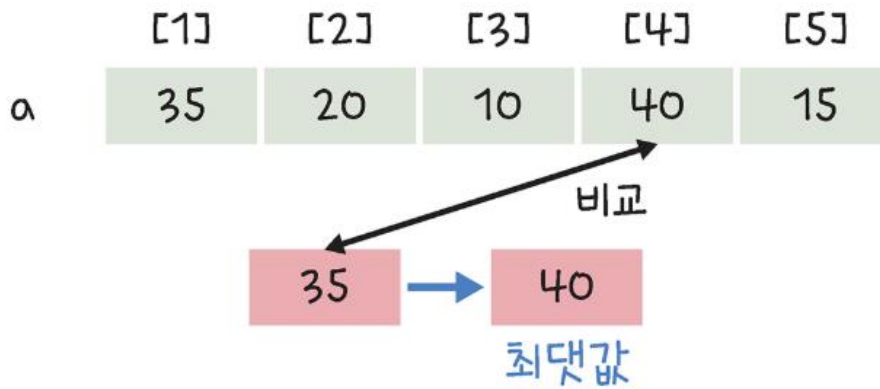
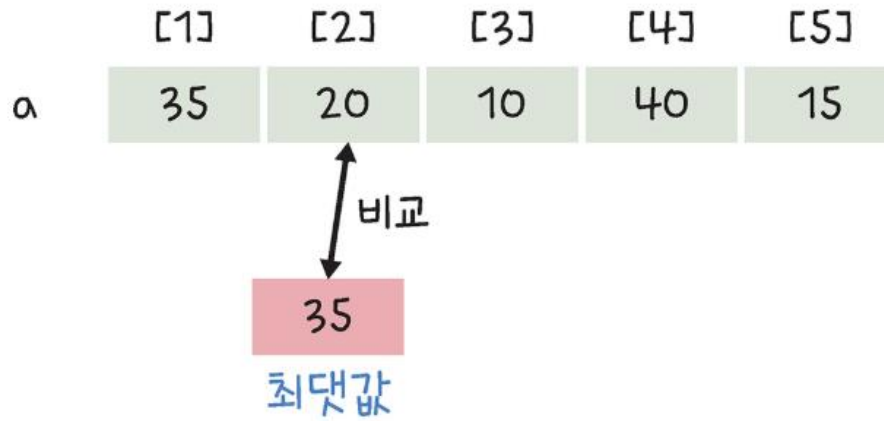
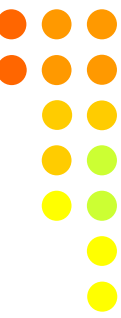
	[1]	[2]	[3]	[4]	[5]
a	35	20	10	40	15



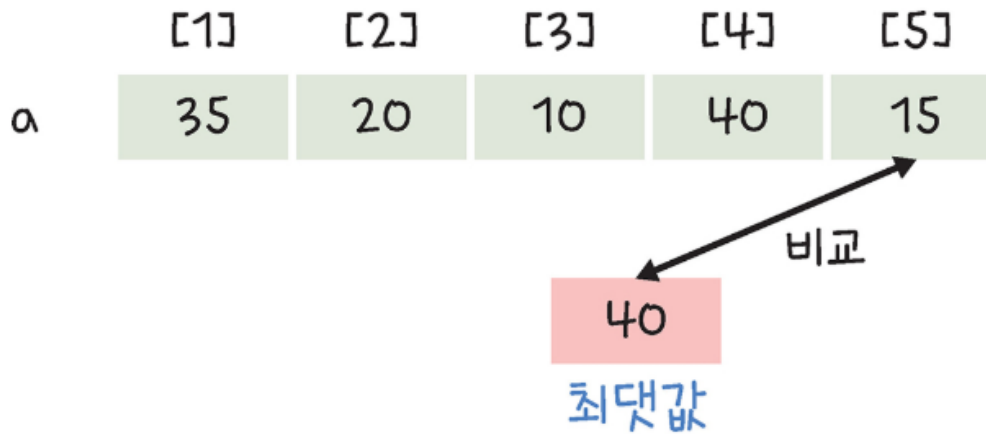
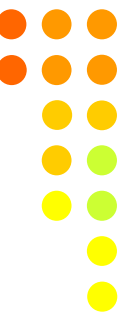
최댓값



### 3. 최댓값 찾기 알고리즘



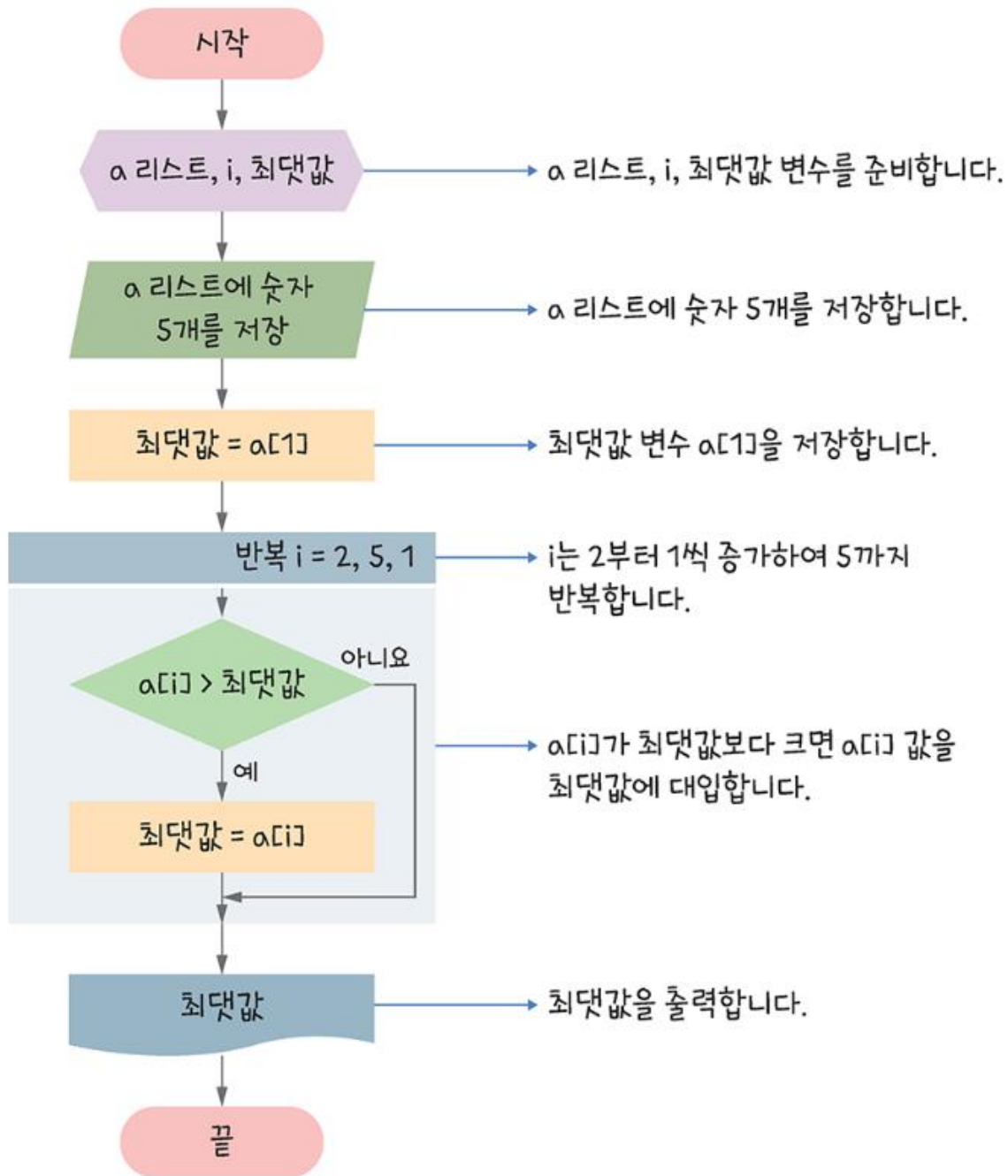
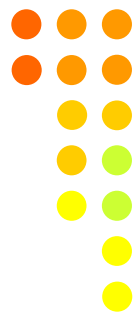
### 3. 최댓값 찾기 알고리즘



### 3. 최댓값 찾기 알고리즘

#### ■ 알고리즘 정리

- 1 | a 리스트를 만듭니다.
- 2 | a 리스트에 데이터 5개를 저장합니다.
- 3 | a[1]을 최댓값에 저장합니다.
- 4 | 다음을 반복합니다.
  - a[2]와 최댓값을 비교하여 큰 값을 최댓값에 저장합니다.
  - a[3]과 최댓값을 비교하여 큰 값을 최댓값에 저장합니다.
  - a[4]와 최댓값을 비교하여 큰 값을 최댓값에 저장합니다.
  - a[5]와 최댓값을 비교하여 큰 값을 최댓값에 저장합니다.
- 5 | 최종적으로 최댓값을 출력합니다.



## 4. 스크래치에서 리스트 만들기

- [변수] > [리스트 만들기]를 클릭하면 새로운 리스트 입력 창이 나타난다.



## 리스트의 값을 수정할 때 사용하는 블록들

항목 을(를) a ▾ 에 추가하기

1 번째 항목을 a ▾ 에서 삭제하기

a ▾ 의 항목을 모두 삭제하기

항목 을(를) a ▾ 리스트의 1 번째에 넣기

a ▾ 리스트의 1 번째 항목을 항목 으로 바꾸기

a ▾ 리스트의 1 번째 항목

a ▾ 리스트에서 항목 항목의 위치

a ▾ 의 길이

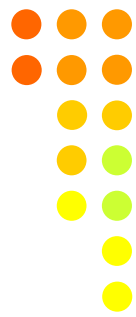
a ▾ 이(가) 항목 을(를) 포함하는가?



## 5. 스크래치에서 최댓값 찾기

- 리스트에 숫자 11, 30, 25, 40, 20을 저장하고 최댓값을 찾는 스크래치를 구현해 봅시다.



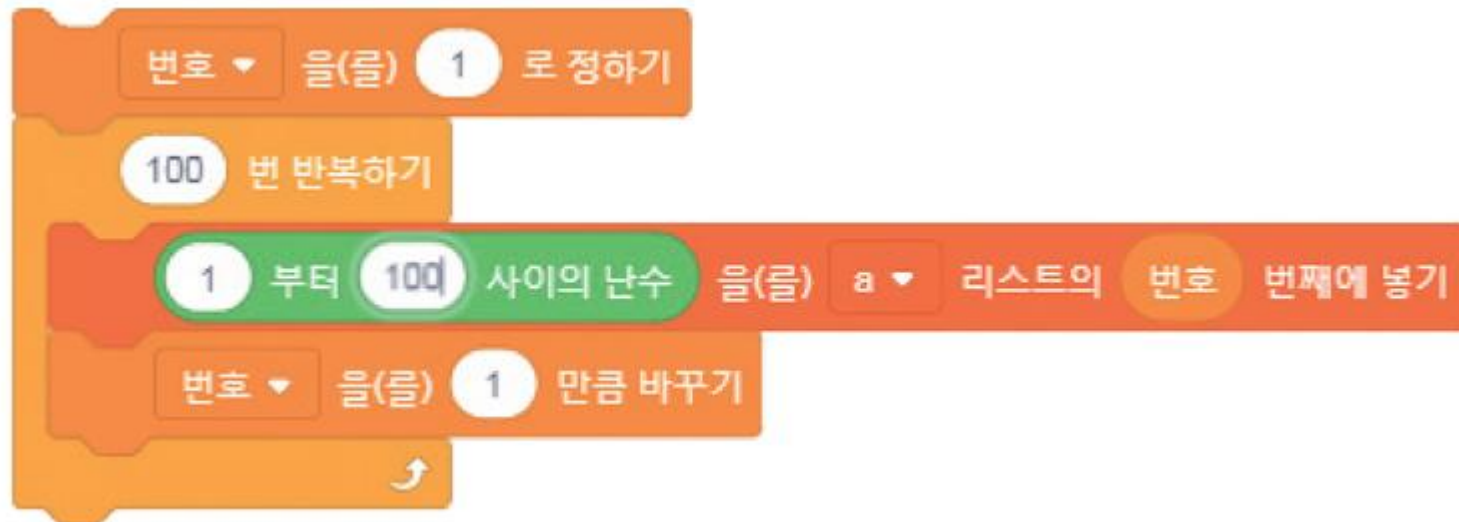


## 6. 응용하기

- 임의의 난수를 100개 마련하여 그 중에서 최댓값을 구해보자.
  - 난수는 예측할 수 없는 무작위 수를 의미
  - 1부터 100사이 난수는 1부터 100사이 숫자를 불규칙하게 생성



- 임의의 수 100개 준비하기
  - 반복문을 사용하여 간단히 구현



클릭했을 때

a 의 항목을 모두 삭제하기

다음 중 최댓값을 구하세요. 을(를) 2 초 동안 말하기

번호 을(를) 1 로 정하기

100 번 반복하기

1 부터 100 사이의 난수 을(를) a 리스트의 번호 번째에 넣기

번호 을(를) 1 만큼 바꾸기

최댓값 을(를) a 리스트의 1 번째 항목 로 정하기

번호 을(를) 2 로 정하기

번호 > a 의 길이 까지 반복하기

만약 a 리스트의 번호 번째 항목 > 최댓값 (이)라면

최댓값 을(를) a 리스트의 번호 번째 항목 로 정하기

번호 을(를) 1 만큼 바꾸기

a 와(과) 중에서 최댓값 : 와(과) 최댓값 결합하기 결합하기 말하기

## 7. 연습 문제

- 1~100까지 난수 데이터 5개를 리스트에 저장한 후 가장 작은 숫자를 찾는 알고리즘을 구현해 보세요.

a

(비어 있음)

+ 길이 0 =

최솟값 3

번호 6



다음 중 최솟값을 구하세요.

a

1 18  
2 36  
3 99  
4 39  
5 76

+ 길이 5 =

최솟값 18

번호 6



18 36 99 39 76 중에서 최솟값: 18



힌트

a ▾

 리스트의 번호 번째 항목 < 최솟값

1 부터 100 사이의 난수 을(를) a ▾ 리스트의 번호 번째에 넣기

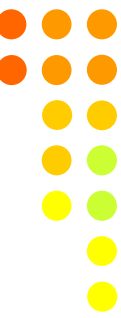
# 스크래치

## DAY 08 – 소수 구하기

```
#include <iostream>
using namespace std;

int main() {
    cout<<"Hola Facebook\n";
    return 0;
}
```

# 1. 소수란?



7:  $1 \times 7$



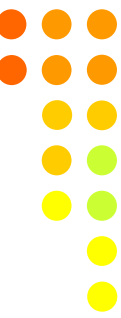
소수

8:  $1 \times 8, 2 \times 4$



소수 아님

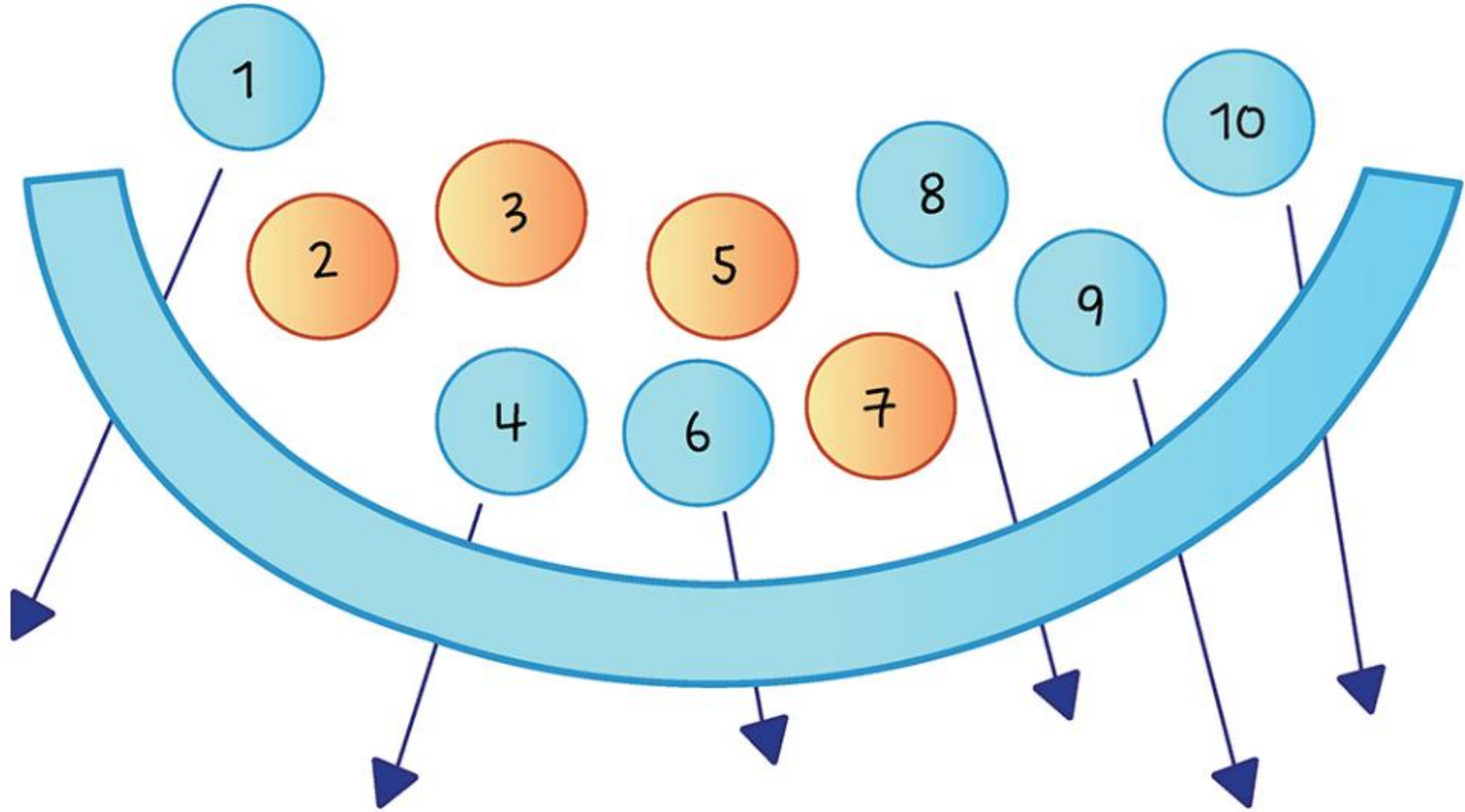




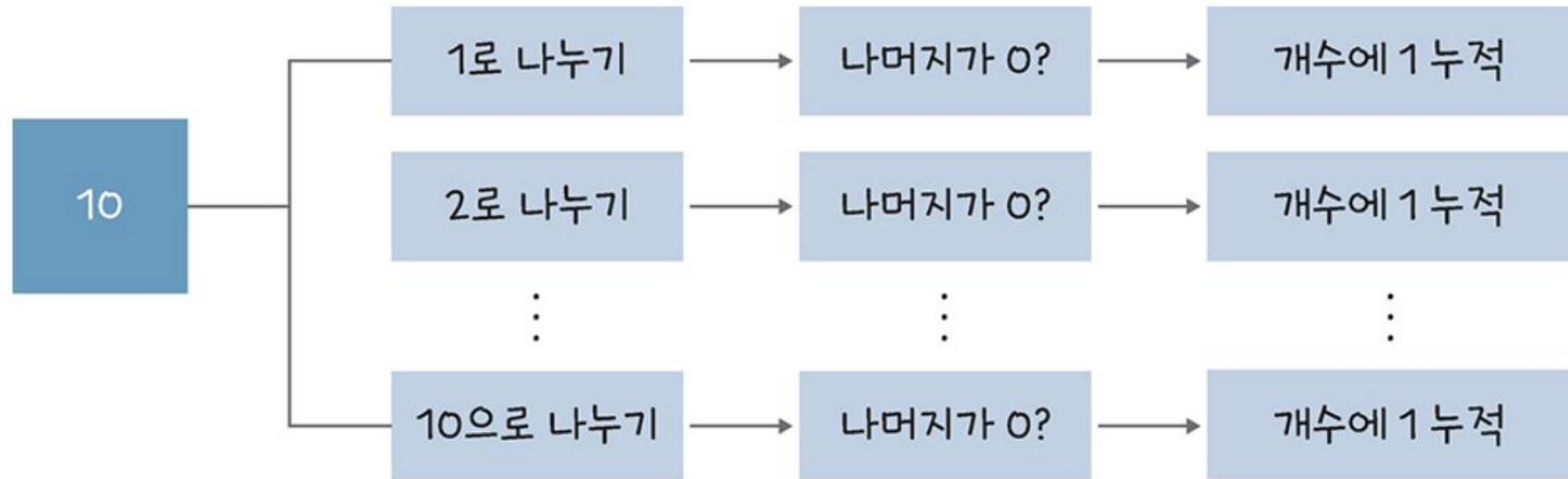
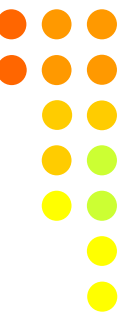
~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~

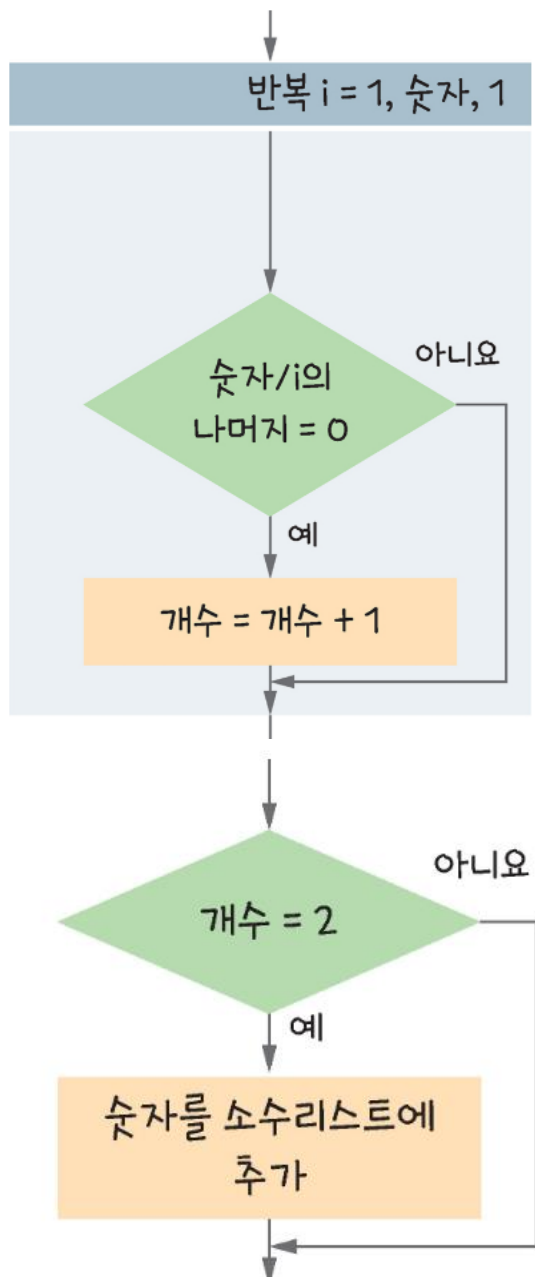
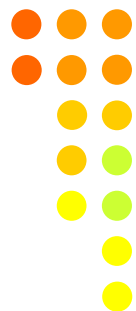


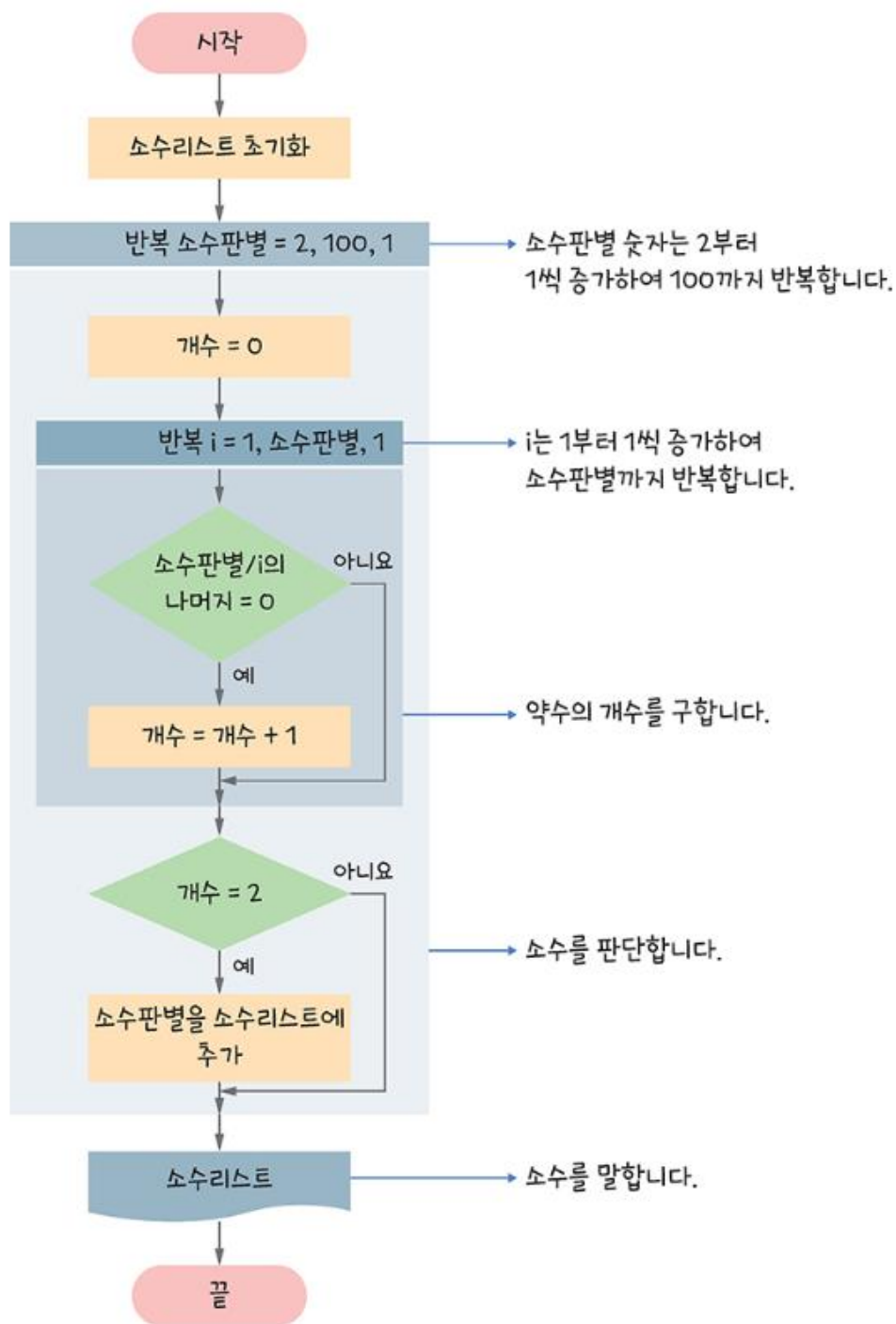




## 2. 소수 구하기 알고리즘







6은 소수일까요?

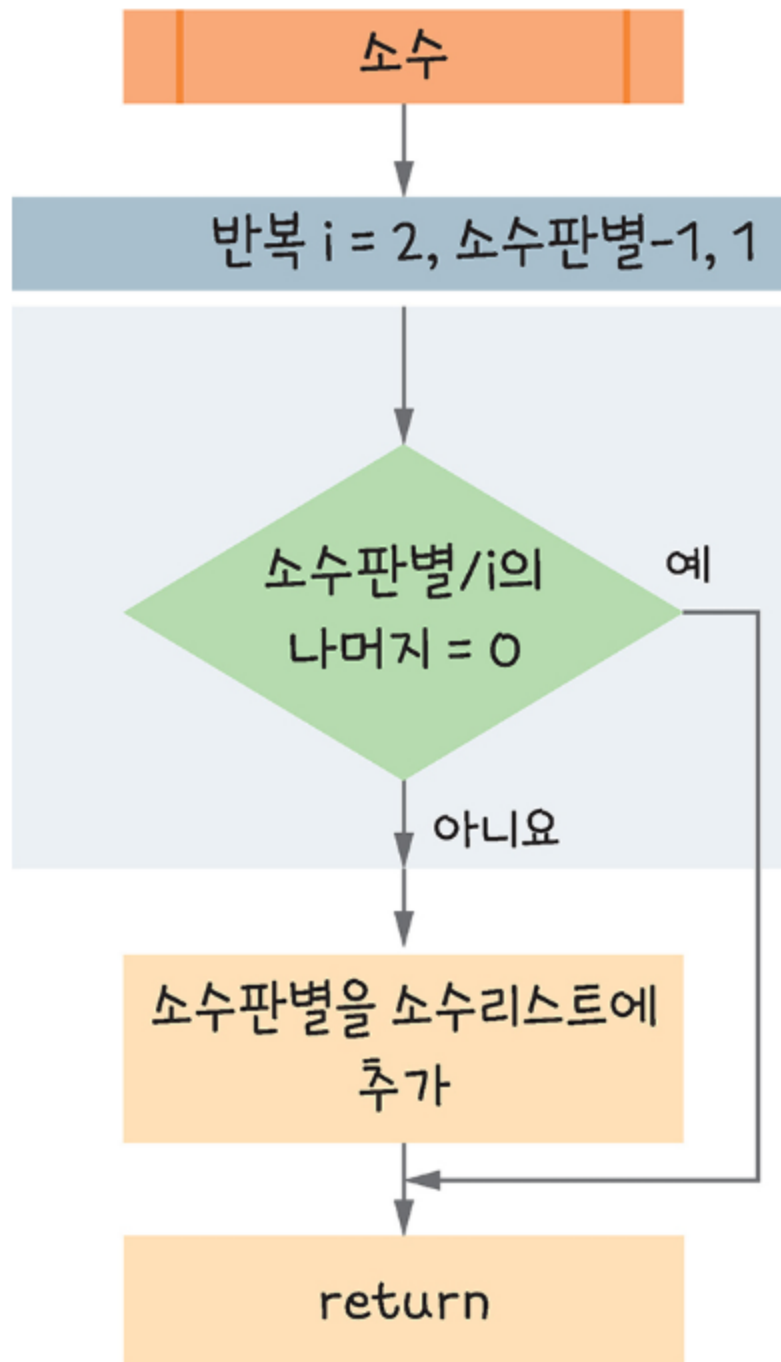
6 나누기 2

나머지 = 0

~~6 나누기 3~~

~~6 나누기 4~~

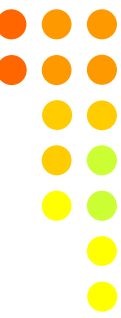
~~6 나누기 5~~



### 3. 알고리즘 선택하기



- 알고리즘 2를 이용하여 스크래치로 구현해 보겠습니다.



## 4. 스크래치에서 소수 구하기

n 5

숫자 6

소수리스트

1	2
2	3
3	5

+ 길이 3 =

a 5

몇 이하의 소수를 구할까요?(2이상)



n 10

숫자 11

소수리스트

1	2
2	3
3	5
4	7

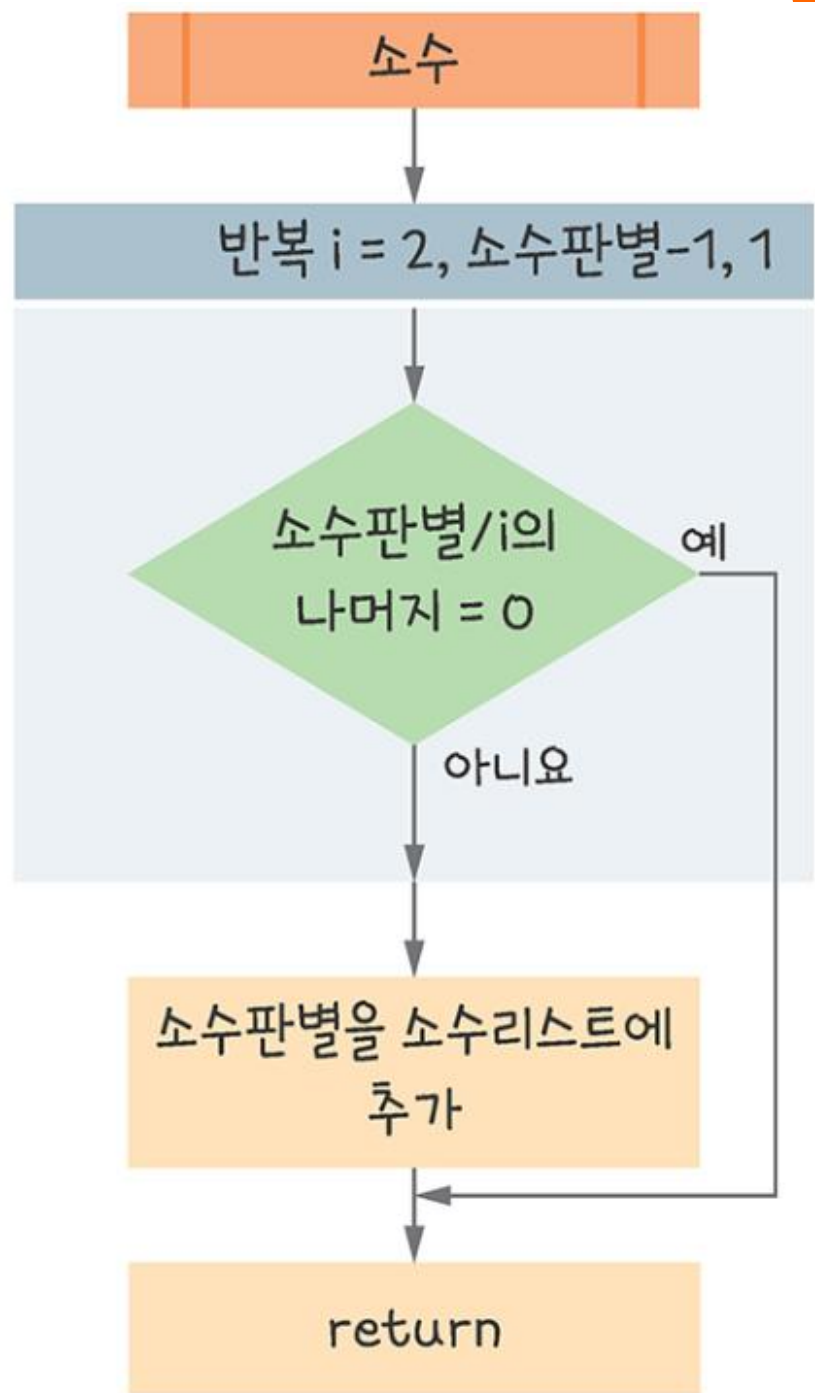
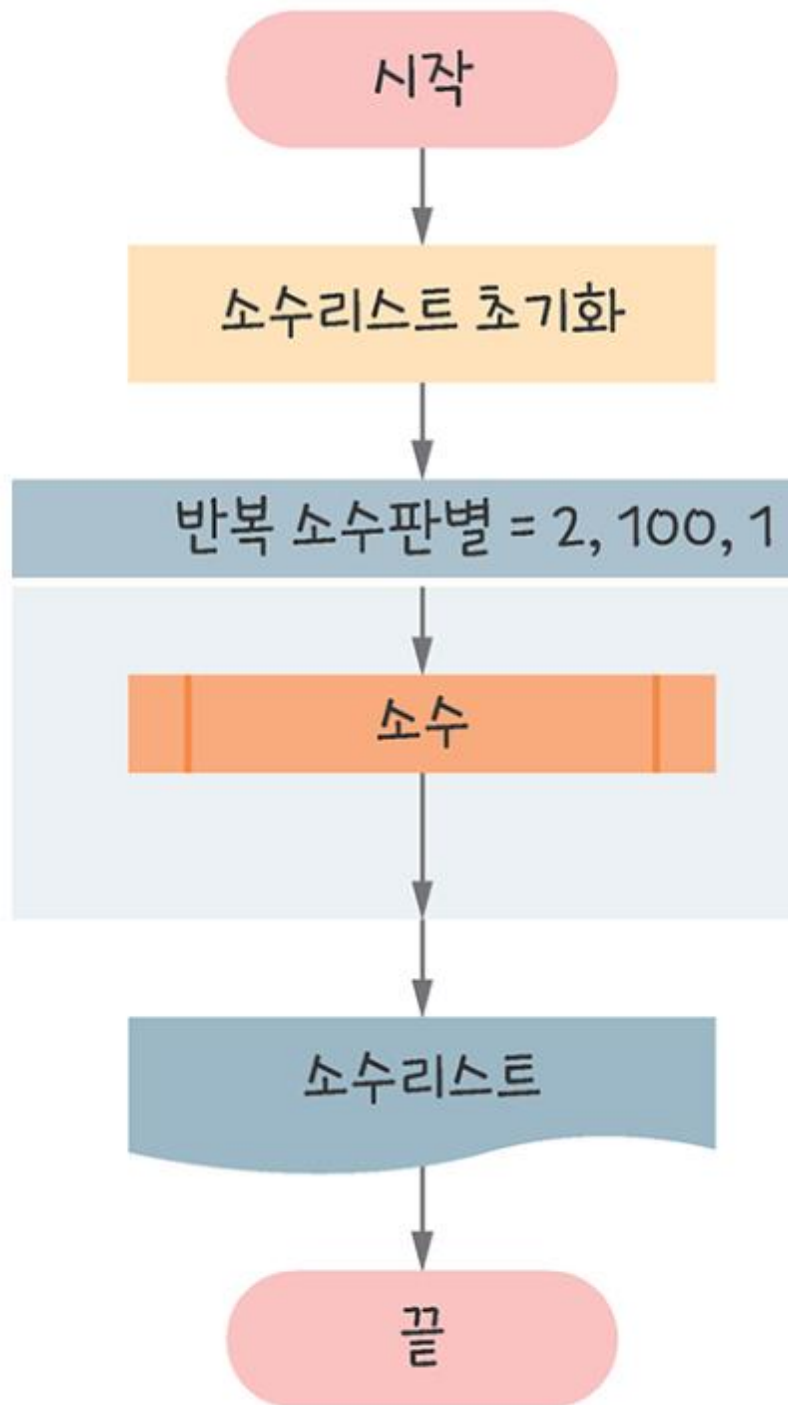
+ 길이 4 =

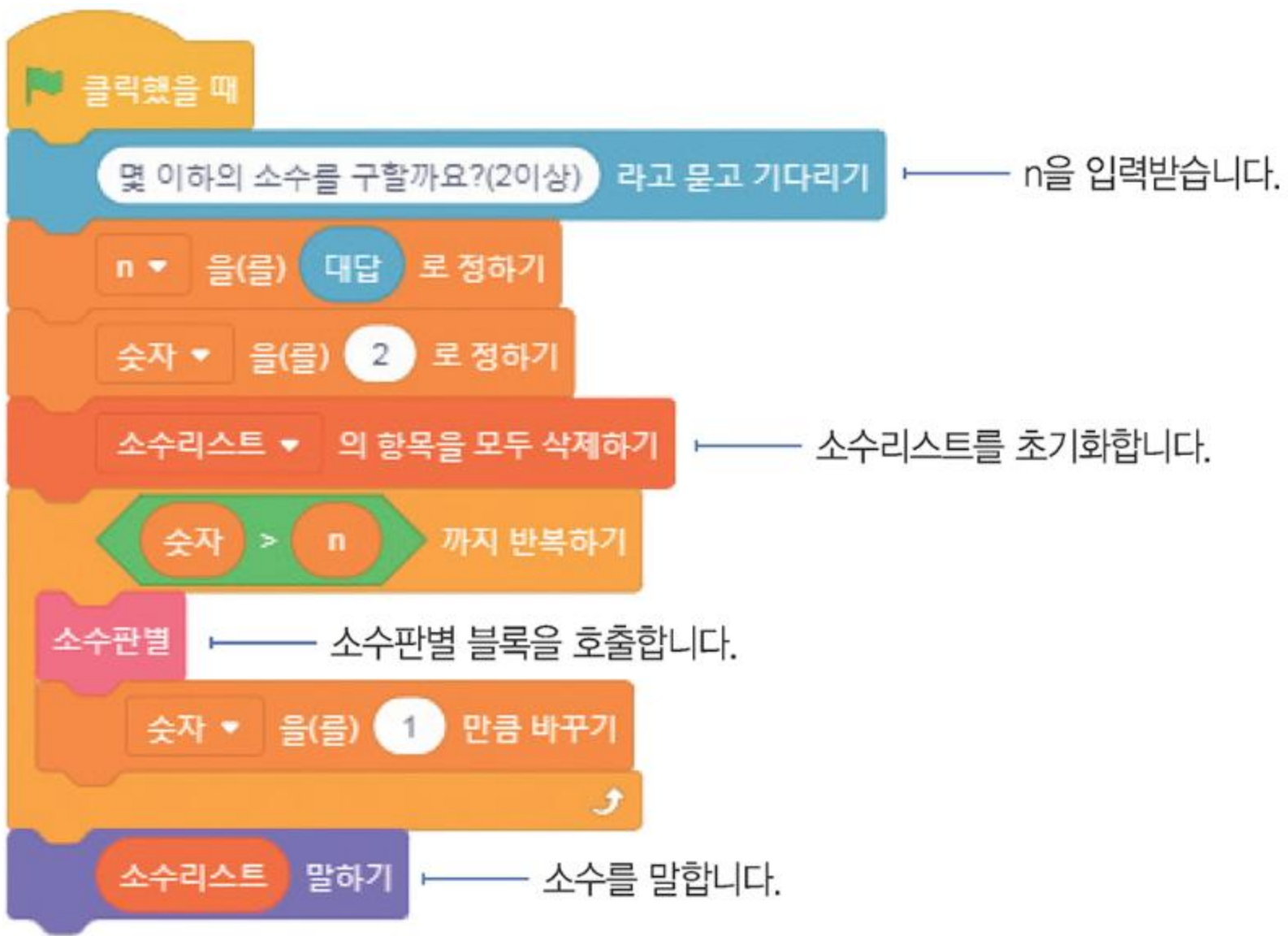
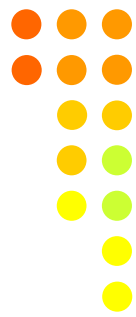
a 2

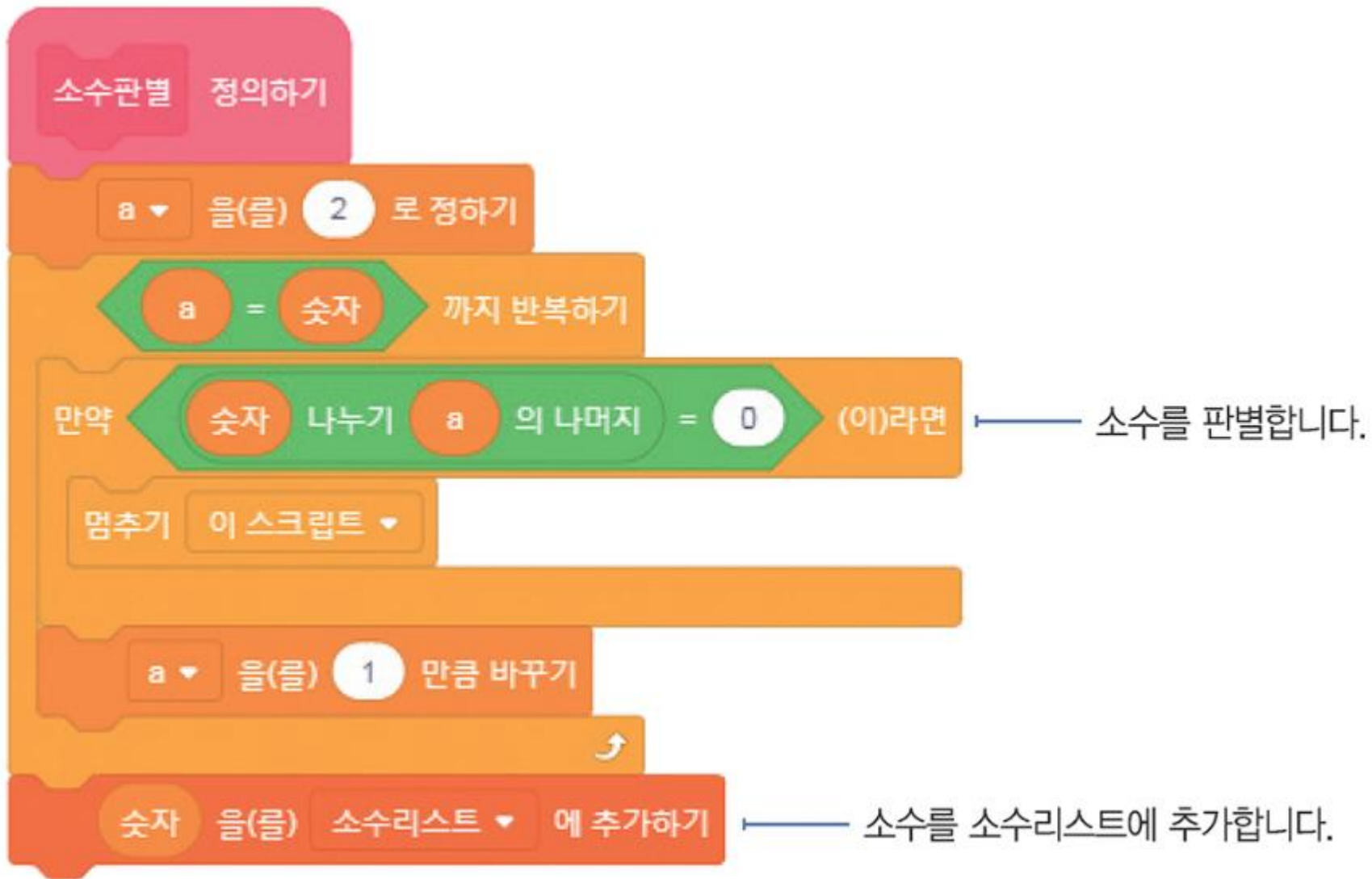
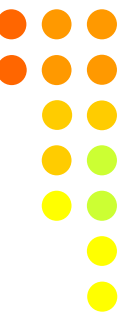
2 3 5 7











# 5. 연습 문제

- 1부터 100까지 숫자 중 소수를 모두 나열하세요.

n5

숫자6

소수리스트

12

23

35

+ 길이 3 =

a3

몇 이하의 소수를 구할까요?(2이상)

n10

숫자11

소수리스트

12

23

35

47

+ 길이 4 =

a2

2 3 5 7

힌트

a > 제공된 ( 숫자 )

# 스크래치

## DAY 09 – 피보나치 수열

```
#include <iostream>
using namespace std;

int main() {
    cout<<"Hola Facebook\n";
    return 0;
}
```

# 1. 피보나치 수열이란?

■ 피보나치 수열: 1, 1, 2, 3, 5, 8, 13, ...인 수열을 의미

- 피보나치 생성 규칙: 처음 두 항이 1이고, 세 번째 항부터는 바로 앞 두 항의 합

(n-2)번째 피보나치 수

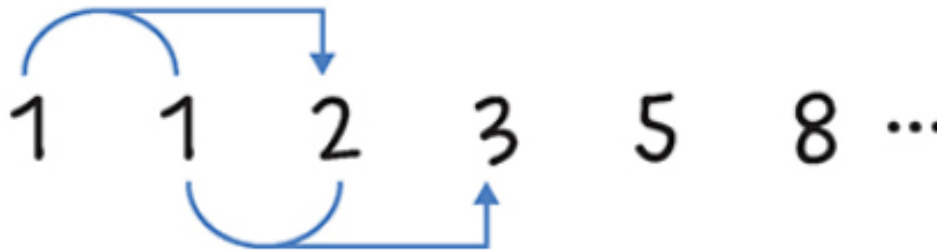


(n-1)번째 피보나치 수



n번째 피보나치 수

- **n번째 항**은 (n-2)번째의 피보나치 수와 (n-1)번째의 피보나치 수를 더한 것



## 2. 리스트로 구현한 피보나치 수열

- 1항과 2항은 1로 정해져 있으니 3항부터 전 항과 그 이전 항을 더하여 구함
- 전 항은 a로 표시하고, 이전 항은 b로 표시한다면 새로운 피보나치 값은  $b+a$

항	피보나치 값
1	1
2	1 1
3	1 1 2
4	1 1 2 3
5	1 1 2 3 5
6	1 1 2 3 5 8
7	1 1 2 3 5 8 13
8	1 1 2 3 5 8 13 21
9	1 1 2 3 5 8 13 21 34
10	1 1 2 3 5 8 13 21 34 55

# 알고리즘

## 알고리즘

1부터 10항까지 피보나치 수열을 나타내도록 순서대로 정리하면 다음과 같습니다.

- 1 | 피보나치 리스트를 초기화합니다.
- 2 | 첫째 항(a)과 둘째 항(b)에 1을 저장합니다.
- 3 | a 값과 b 값을 피보나치 리스트에 추가합니다.
- 4 | i는 3부터 10까지 반복합니다(i=3부터 10까지 1씩 증가).

$c = a + b$

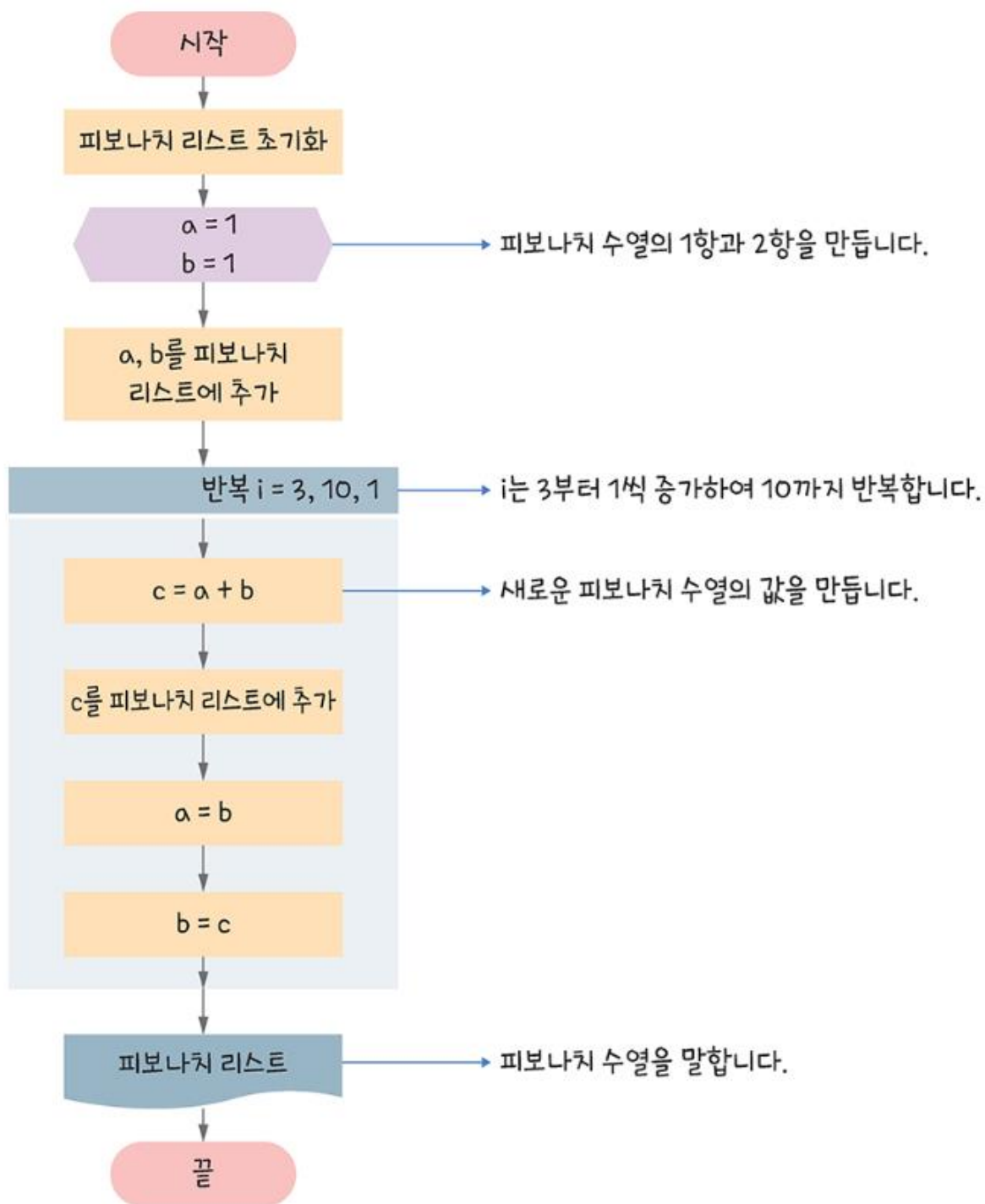
c를 피보나치 리스트에 추가합니다.

$a = b$

$b = c$

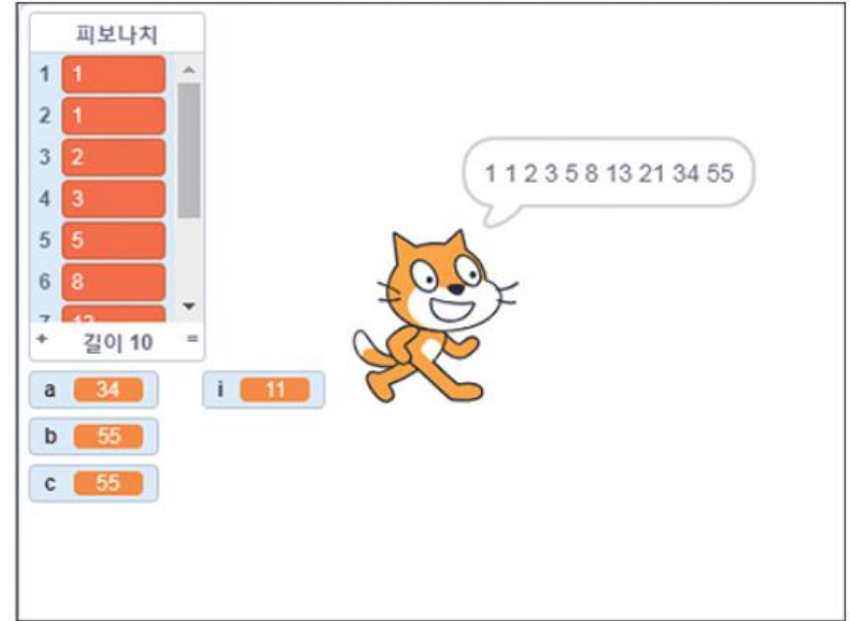
- 5 | 피보나치 리스트를 표시합니다.





# 스크래치 구현하기

- 1부터 10항까지 피보나치 수열을 구하는 것을 스크래치로 구현
  - 예제 파일: 알고리즘09-1.sb3



클릭했을 때

피보나치 수열 10항까지 구하겠습니다. 을(를) 2 초 동안 말하기

피보나치 의 항목을 모두 삭제하기 — 피보나치 리스트를 초기화합니다.

a 을(를) 1 로 정하기

b 을(를) 1 로 정하기

피보나치 수열의 1항과 2항을 만듭니다.

a 을(를) 피보나치 에 추가하기

b 을(를) 피보나치 에 추가하기

i 을(를) 3 로 정하기

i > 10 까지 반복하기

c 을(를) a + b 로 정하기 — 새로운 피보나치 수열의 값을 만듭니다.

c 을(를) 피보나치 에 추가하기

a 을(를) b 로 정하기

b 을(를) c 로 정하기

i 을(를) 1 만큼 바꾸기

피보나치 말하기 — 피보나치 수열을 말합니다.

### 3. 블록을 이용한 피보나치 수열

#### 알고리즘

먼저 진행 순서를 정리하면 다음과 같습니다.

- 1 | n 값을 입력받습니다.
- 2 | 피보나치 수열 블록을 호출합니다.
- 3 | 피보나치 리스트를 표시합니다.

피보나치 수열의 블록은 다음과 같습니다.

- 1 | 피보나치 수열을 저장할 리스트를 초기화합니다.
- 2 | 첫째 항(a)과 둘째 항(b)에 1을 저장합니다.
- 3 | a 값과 b 값을 피보나치 리스트에 추가합니다.
- 4 | i는 3부터 10까지 반복합니다.

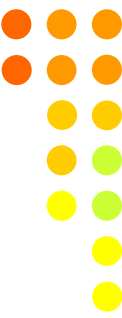
$c = a + b$

c를 피보나치 수열에 추가합니다.

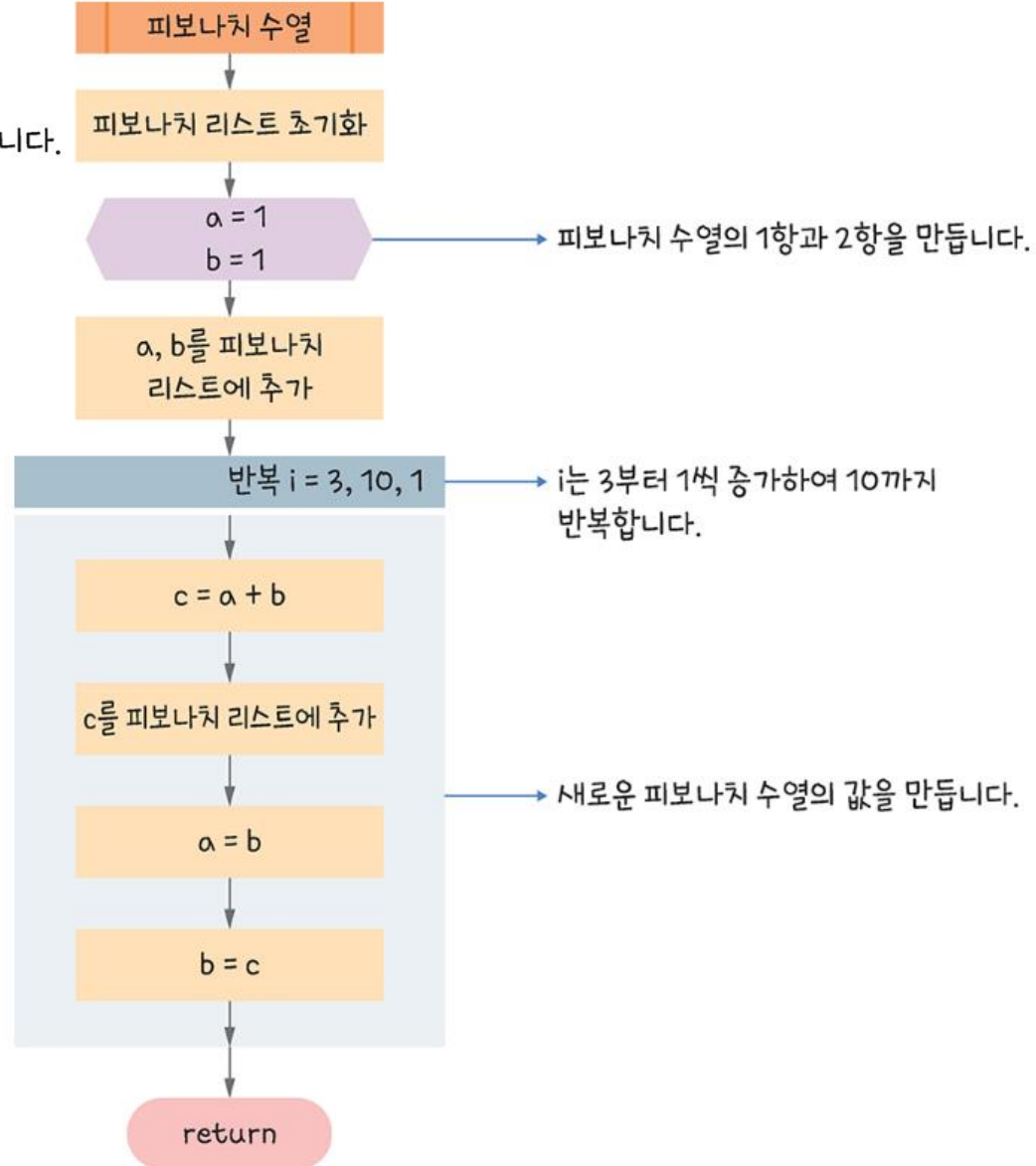
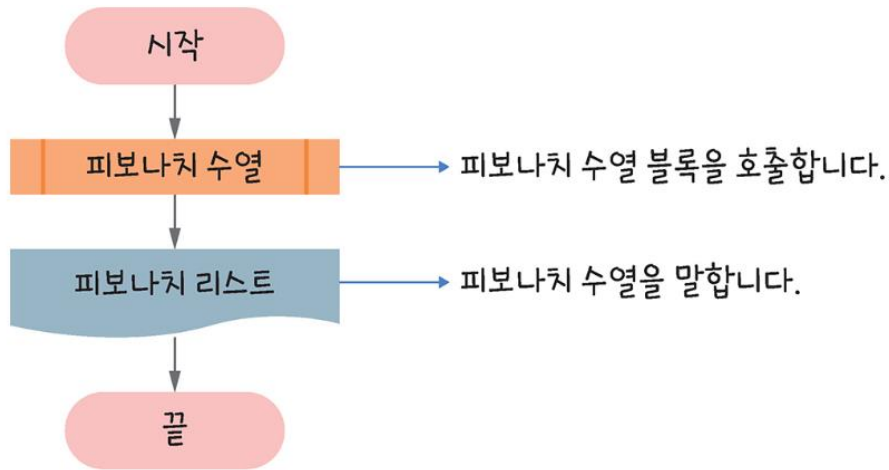
$a = b$

$b = c$

이것을 순서도로 표현하면 다음과 같습니다.

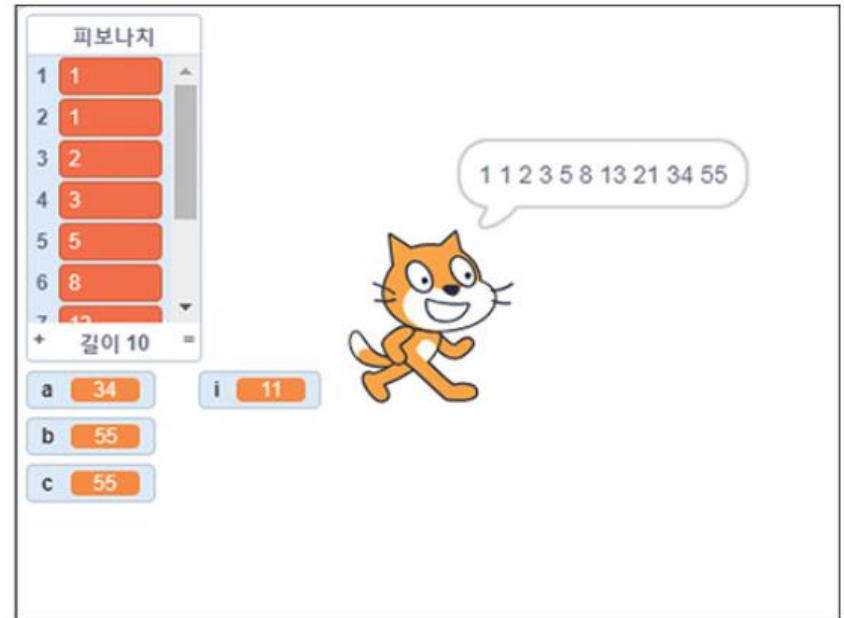


### 3. 블록을 이용한 피보나치 수열



# 스크래치에서 구현하기

- 1부터 10항까지 피보나치 수열을 구하는 것을 스크래치로 구현
  - 예제 파일: 알고리즘09-2.sb3



# 스크래치에서 구현하기

피보나치 수열 10항까지 구하겠습니다. 을(를) 10 초 동안 말하기

피보나치수열

피보나치 말하기

피보나치 수열 블록을 호출합니다.

피보나치수열 정의하기

피보나치 의 항목을 모두 삭제하기

a 을(를) 1 로 정하기

b 을(를) 1 로 정하기

피보나치 수열의 1항과 2항을 만듭니다.

a 을(를) 피보나치 에 추가하기

b 을(를) 피보나치 에 추가하기

i 을(를) 3 로 정하기

i > 10 까지 반복하기 10항까지 피보나치 수열을 구합니다.

c 을(를) a + b 로 정하기 새로운 피보나치 수열의 값을 구합니다.

c 을(를) 피보나치 에 추가하기

a 을(를) b 로 정하기

b 을(를) c 로 정하기

i 을(를) 1 만큼 바꾸기

## 4. 연습 문제

- n 값을 입력받아 1부터 n항까지 피보나치 수열을 구하는 내용을 구현해 보세요.
  - 예제 파일: 알고리즘09-3.sb3

피보나치

1	1
2	1
3	2
4	3
5	5

+ 길이 5 =

a 3 i 6

b 5 n 5

c 5

피보나치 수열 몇 항까지 구할까요?(3이상 입력)



피보나치

1	1
2	1
3	2
4	3
5	5
6	8
7	13


+ 길이 20 =

a 4181 i 21

b 6765 n 20

c 6765

1 1 2 3 5 8 13 21 34 55 89  
144 233 377 610 987 1597  
2584 4181 6765





# 스크래치

## DAY 10 – 최대공약수

```
#include <iostream>
using namespace std;

int main() {
    cout<<"Hola Facebook\n";
    return 0;
}
```

# 1. 최대공약수란?

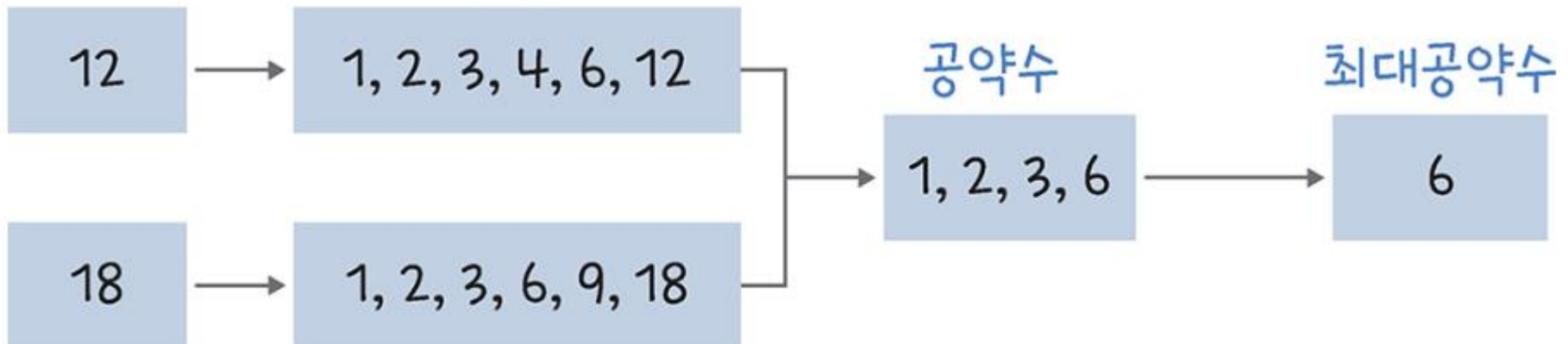
■ 최대공약수: 공통된 약수 중에서 가장 큰 공약수

- 약수: 어떤 정수를 나누어지게 하면서 0이 아닌 정수
- 공약수: 두 정수에서 공통의 약수가 되는 정수, 나누어 떨어지는 공통의 수
- 최대공약수: 공약수 중에서 가장 큰 수



- 숫자 12와 18 예

약수



## 2. 소수의 곱셈으로 나타내기

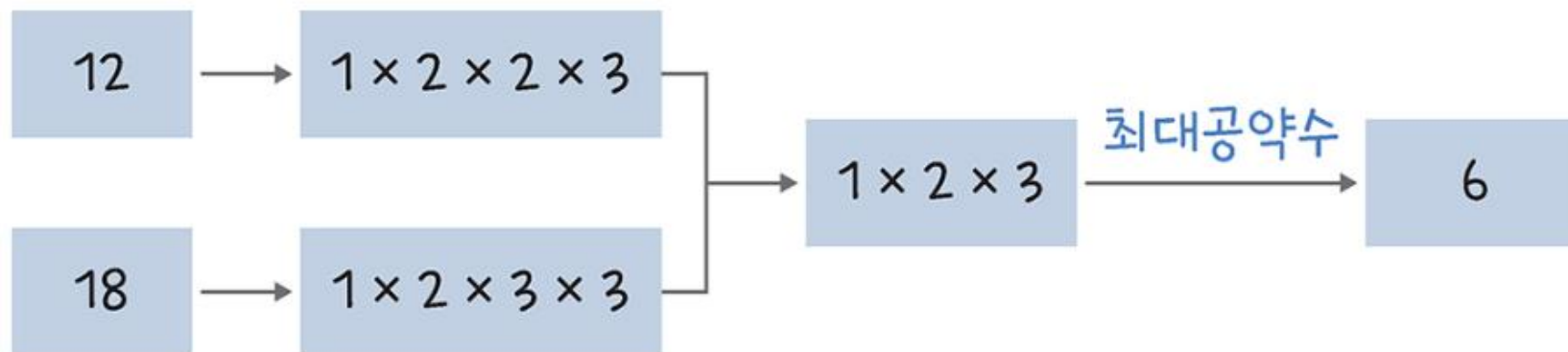
- 임의의 숫자를 소수의 곱으로 나타내기

$$\begin{array}{l} 12 = 1 \times 12 \\ \quad \downarrow \\ \quad \hline 2 \times 6 \\ \quad \quad \downarrow \\ \quad \quad \hline \quad 2 \times 3 \end{array} \quad \left. \vphantom{\begin{array}{l} 12 = 1 \times 12 \\ \quad \downarrow \\ \quad \hline 2 \times 6 \\ \quad \quad \downarrow \\ \quad \quad \hline \quad 2 \times 3 \end{array}} \right\} 12 = 1 \times 2 \times 2 \times 3$$

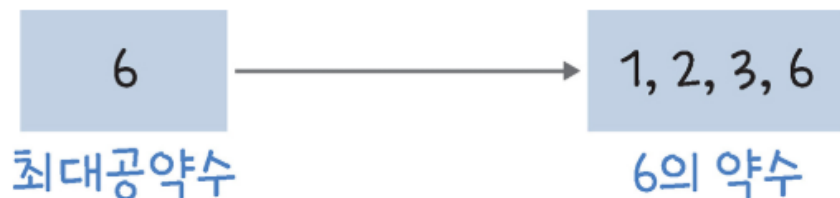
$$\begin{array}{l} 18 = 1 \times 18 \\ \quad \downarrow \\ \quad \hline 2 \times 9 \\ \quad \quad \downarrow \\ \quad \quad \hline \quad 3 \times 3 \end{array} \quad \left. \vphantom{\begin{array}{l} 18 = 1 \times 18 \\ \quad \downarrow \\ \quad \hline 2 \times 9 \\ \quad \quad \downarrow \\ \quad \quad \hline \quad 3 \times 3 \end{array}} \right\} 18 = 1 \times 2 \times 3 \times 3$$

## 2. 소수의 곱셈으로 나타내기

- 그러면 공통 부분은  $1 \times 2 \times 3 = 6$  입니다. 이 수가 12와 18의 최대공약수가 됩니다.
- 즉, 12와 18의 최대공약수는 6입니다.



- 최대공약수를 알면 공약수도 쉽게 구할 수 있음.
  - 최대공약수의 약수가 공약수가 됨.
  - 최대공약수는 6이고, 6의 약수는 1, 2, 3, 6이며 이것이 공약수가 됨



### 3. 최대공약수로 나누기

■ 12와 18의 최대공약수를 구하기 위해 소수로 나누어 보자.

- 12와 18의 공약수인 2로 나누면 몫이 6과 9가 됨
- 다시 6과 9를 3으로 나누면 몫이 2와 3이 됨
- 이제 공약수가 1밖에 없는 상태이니 멈춤
- **왼쪽에 쓴 나누는 수**는 2와 3이고, 이 수를 곱한  $2 \times 3 = 6$ 이 최대공약수가 됨

$$\begin{array}{r} 2 \overline{) 12 \quad 18} \end{array} \longrightarrow 12 \text{와 } 18 \text{의 공약수 중 } 1 \text{ 이외의 공약수는 } 2, 3, 6 \text{입니다.}$$

$$\begin{array}{r} 3 \overline{) 6 \quad 9} \end{array} \longrightarrow 6 \text{과 } 9 \text{의 공약수 중 } 1 \text{ 이외의 공약수는 } 3 \text{입니다.}$$

$$\begin{array}{r} 2 \quad 3 \end{array} \longrightarrow 2 \text{와 } 3 \text{은 } 1 \text{ 이외의 공약수가 없습니다.}$$

$$2 \times 3 = 6$$

12와 18의 최대공약수

### 3. 최대공약수로 나누기

- 나누는 수는 소수가 아니어도 됨
- 12와 18의 공약수 중에 6으로 나누었더니 계산이 좀 더 짧아짐.

$$\left. \begin{array}{r} 6 \overline{) 12 \quad 18} \\ \underline{2 \quad 3} \end{array} \right\} \text{최대공약수는 6}$$

- 그렇다면 3과 18의 최대공약수는 무엇?
- 18을 3으로 나누면 나머지가 0이 되어서 바로 3이 최대공약수라는 것을 알 수 있음

$$\begin{array}{r} 3 \overline{) 3 \quad 18} \\ \underline{1 \quad 6} \end{array} \qquad \begin{array}{r} 6 \\ 3 \overline{) 18} \\ \underline{18} \\ 0 \end{array}$$

### 3. 최대공약수로 나누기

- 18과 6의 최대공약수는?
- 18을 6으로 나누어 나머지가 0이 되므로 최대공약수는 6

$$\begin{array}{r} 3 \\ 6 \overline{) 18} \\ \underline{18} \\ 0 \end{array}$$

- 12와 18의 최대공약수는?
- 18 (a값)을 12(b값)로 나누면 나머지는 0이 아니다.
- 몫은 1, 나머지 6. 그러면 작은 수였던 12와 나머지였던 6을 가지고 다시 나누기 진행
- 12 나누기 6을 하면 나머지 0이 되어 최대공약수 6

$$\begin{array}{r} 6 \overline{) 18} \quad 6 \\ \underline{12} \quad 6 \\ 6 \end{array}$$

$$\begin{array}{r} 1 \\ 12 \overline{) 18} \\ \underline{12} \\ 6 \end{array}$$
  
$$\begin{array}{r} 2 \\ 6 \overline{) 12} \\ \underline{12} \\ 0 \end{array}$$

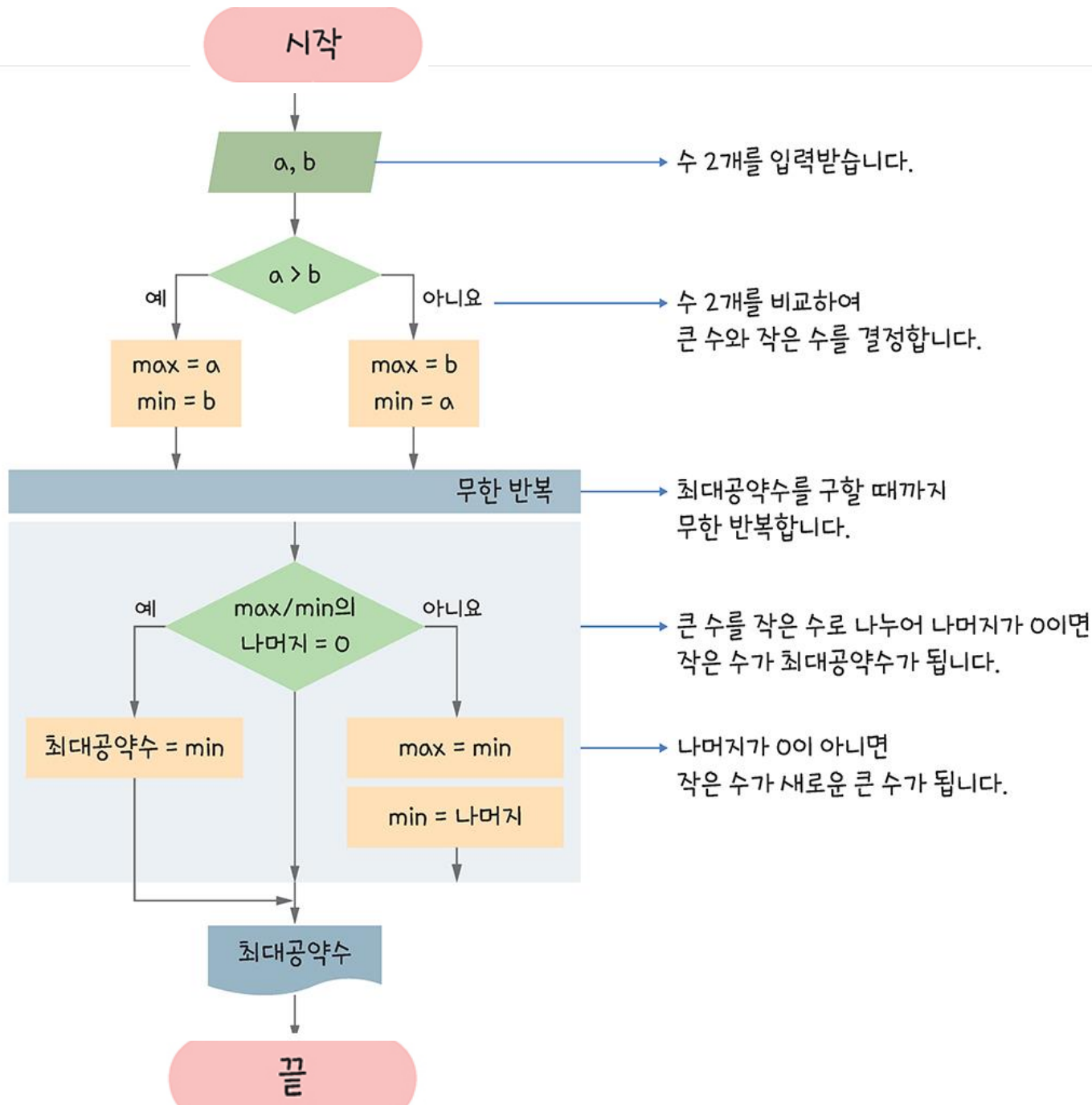
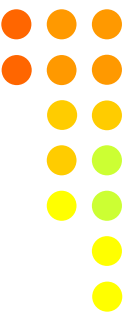
## 4. 알고리즘

### ■ 유클리드 호제법

앞서 나열한 내용을 정리하면 다음과 같습니다.

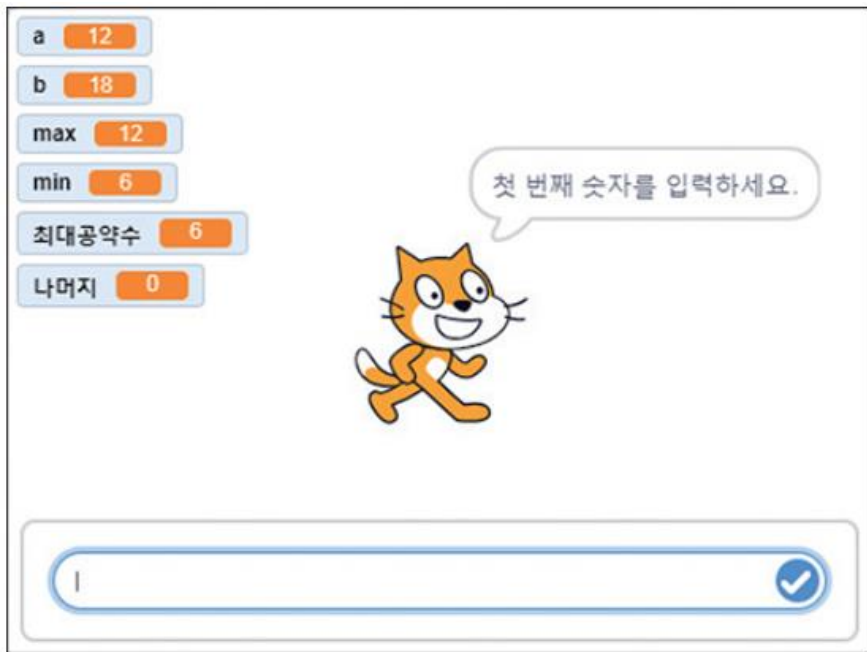
- 1| 수 2개를 입력받습니다.
- 2| 수 2개 중에서 큰 수와 작은 수를 결정합니다.
- 3| 큰 수 나누기 작은 수를 하여 나머지를 구합니다.
- 4| 나머지가 0이면 작은 수가 최대공약수가 되고, 프로그램이 종료됩니다.
- 5| 나머지가 0이 아니면 작은 수가 새로운 큰 수가 되고, 나머지는 새로운 작은 수가 되어 프로그램을 반복합니다.

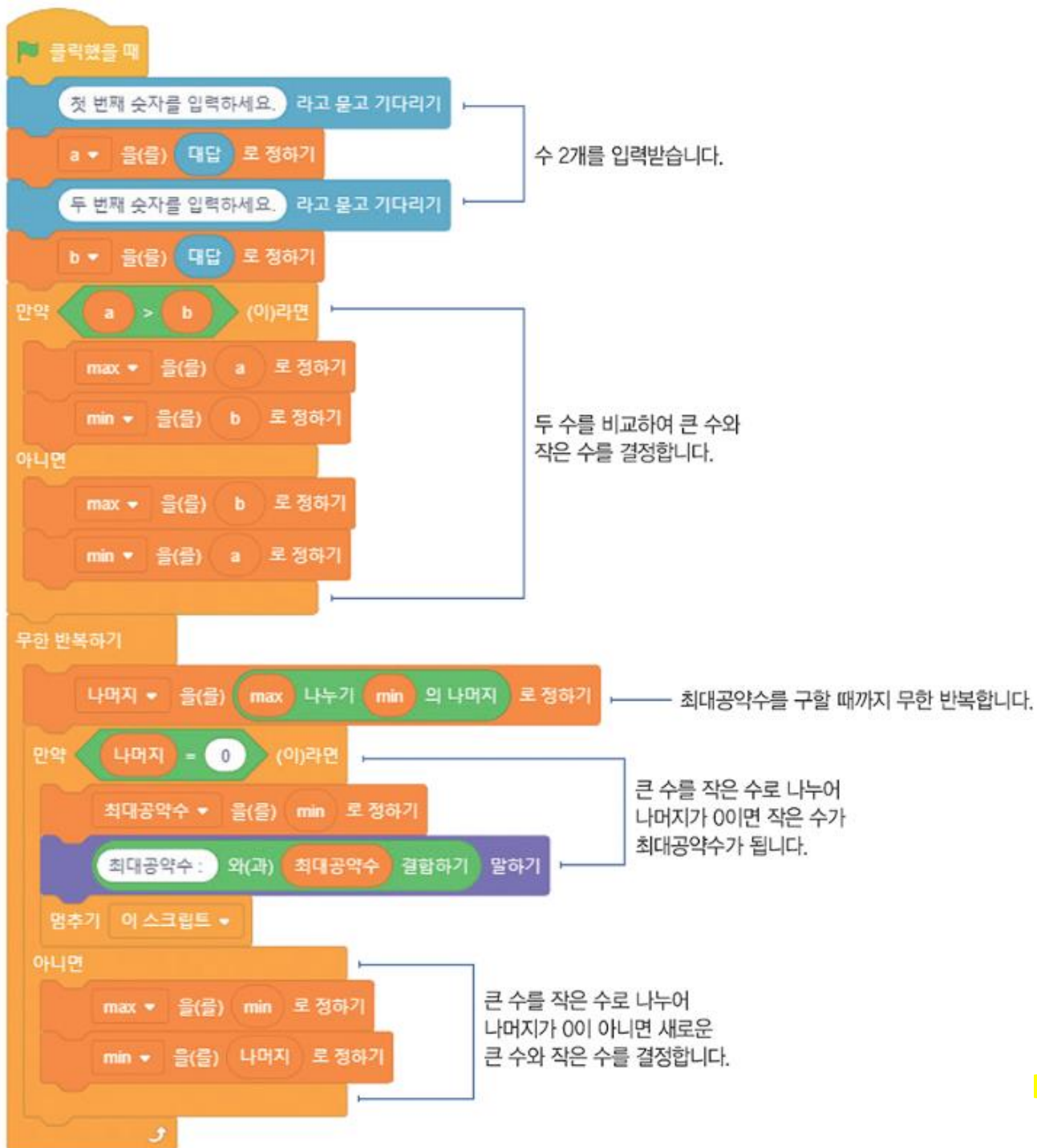




## 5. 스크래치에서 최대공약수 구하기

- 숫자 2개를 입력받아 최대공약수를 구하는 내용을 스크래치로 구현해보자.
  - 예제 파일: 알고리즘10-1.sb3





수 2개를 입력받습니다.

두 수를 비교하여 큰 수와 작은 수를 결정합니다.

최대공약수를 구할 때까지 무한 반복합니다.

큰 수를 작은 수로 나누어 나머지가 0이면 작은 수가 최대공약수가 됩니다.

큰 수를 작은 수로 나누어 나머지가 0이 아니면 새로운 큰 수와 작은 수를 결정합니다.

## 6. 연습 문제

- 두 수의 최대공약수를 구한 결과 최대공약수가 1이면 특별히 '서로소' 라고 합니다. 수 2개를 입력받아 최대공약수를 구하고, 최대공약수가 1이라면 '서로소' 라고 표시하는 내용을 스크래치로 구현해 보세요.

- 예제 파일: 알고리즘10-2.sb3

<조건>

최대공약수를 구하는 부분을 나만의 블록으로 처리하세요.



Variables:

- a: 15
- b: 17
- max: 2
- min: 1
- 최대공약수: 1
- 나머지: 0

Dialogue: 두 수는 서로소입니다.



Variables:

- a: 12
- b: 18
- max: 12
- min: 6
- 최대공약수: 6
- 나머지: 0

Dialogue: 최대공약수 : 6

