

# 컴퓨터프로그래밍 기초개념

# 컴퓨팅 사고력

## ❖ CT(Computational Thinking)

- 컴퓨터처럼 생각해 현실의 복잡하고 어려운 문제를 해결하는 사고 방식
  - 컴퓨터가 문제를 해결하는 방식처럼 문제를 단순화하고 이를 논리적, 효율적으로 해결하는 능력
  - 문제 상황의 핵심 원리를 찾아내 이를 순서도를 만들어 해결하는 방식
  - *데이터를 모으고 조작하기, 큰 문제를 작은 문제들로 쪼개기, 문제를 구조화하고 추상화하기, 순서에 따라 문제 해결을 자동화하기 등*
- 창의적 사고력을 키우는 새로운 교육방법으로 주목
  - 많은 나라들이 소프트웨어 교육을 실시하는 목적: *국민을 단지 컴퓨터 코딩을 능숙하게 다루는 프로그래머로 만들자는 것이 아니라, 국민들이 모든 분야의 문제를 새로운 방향으로 생각하여 수월하게 해결할 수 있게 하고자 함*

# 컴퓨팅 사고의 활용



해결방안  
고민

컴퓨팅 사고를  
할 수 없다면?

버스가 도착할  
때까지 기다린다

컴퓨팅 사고를  
할 수 있다면?

버스 도착을 알 수  
있는 데이터 확인

문제 해결을 위한  
알고리즘 작성

알고리즘을 기반으로  
소프트웨어 개발

소프트웨어를 이용한  
편리한 사용

# 문제해결 프로세스

1	문제 파악 및 정의	해결해야 하는 문제가 무엇인지를 파악하고, 파악된 문제를 명확하게 정의, 모호함이 없이 문제의 범위 및 본질을 묘사
2	문제 해결 전략/방법 도출	문제 해결에 필요한 지식을 수집하여, 해결 전략 방법 도출
3	문제 해결 활동 수행	도출된 방법에 따라 해결 활동 수행
4	결과 검증 및 확인	정의된 문제가 해결되었는지 점검

# 문제해결 프로세스 - 단순예제

- Q) 기온이 화씨로 50도라고 하면 섭씨 온도는?

1

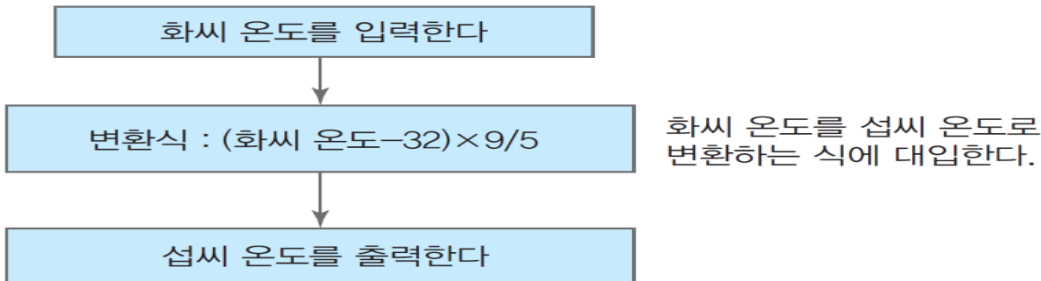
문제 파악 및 정의

문제를 파악하고 정의한다.  
화씨 온도를 섭씨 온도로 변환한다.

2

알고리즘 작성

알고리즘을 프로그래밍 언어로 변환한다.



3

프로그램 작성

알고리즘을 프로그래밍 언어로 변환한다.

```
F = int(input('화씨 온도를 입력하시오 '))  
C = (F - 32) + 9 / 5  
print('섭씨 온도는 ', C, '입니다')
```

4

프로그램 수행 및 결과 확인

프로그램을 수행하고 결과를 확인한다.

```
화씨 온도를 입력하시오 50  
섭씨 온도는 32.4 입니다  
>>>
```

# 알고리즘 (Algorithm)


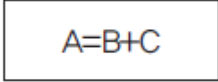
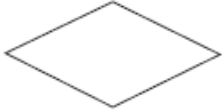
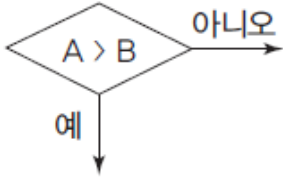

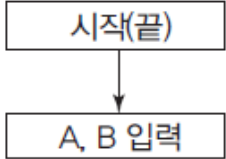
❖ 주어진 문제를 해결하기 위해 설계하는 일련의 논리적 절차

- 단계적인 해결책 / 문제의 해결을 수행하기 위한 절차적 지식
- 예) 커피 한 잔을 만드는 알고리즘

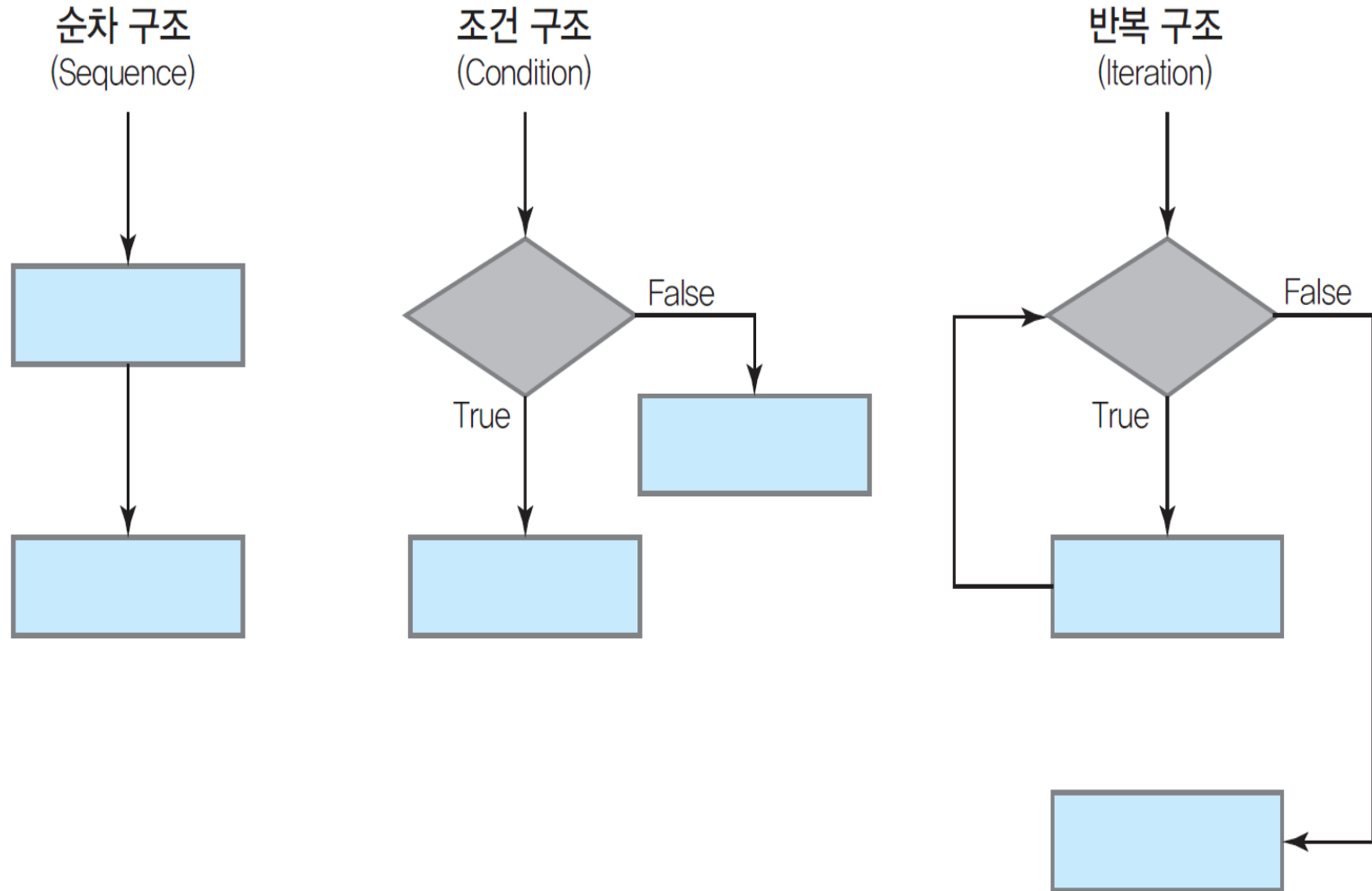
- 1 머그컵을 준비합니다.
- 2 머그컵에 인스턴트 커피 1 숟가락을 넣습니다.
- 3 주전자에 물을 넣습니다.
- 4 물을 끓입니다.
- 5 끓인 물을 머그컵의 상단 2cm 전까지 붓습니다.
- 6 냉장고에서 우유를 꺼내옵니다.
- 7 머그컵에 우유를 1cm만큼 붓습니다.
- 8 냉장고에 우유를 도로 넣습니다.

# cf) 순서도 (Flow Diagram)

❖ 알고리즘 설계를 위해 사용되는 도구

기호	기호의 설명	예
	값을 계산하거나 대입 등을 나타내는 처리 기호	
	조건이 참이면 '예', 거짓이면 '아니오'로 가는 판단 기호	
	기호를 연결하여 처리의 흐름을 나타내는 흐름 선	

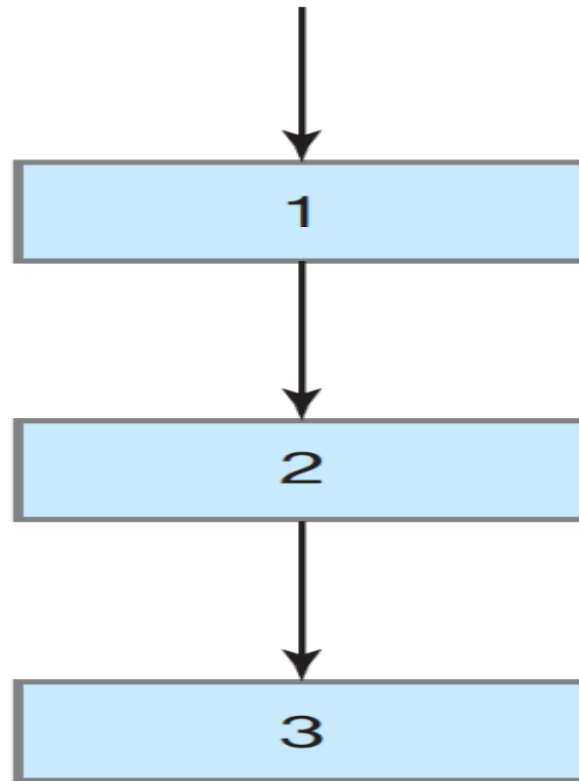
# 알고리즘의 구조





# 알고리즘의 구조: 순차 구조

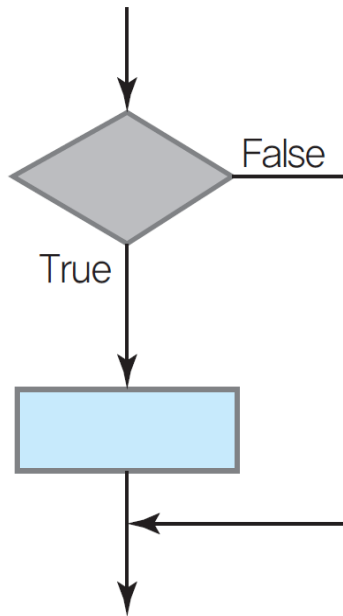
- ❖ 위에서 아래로 순서에 따라 차례대로 작업 수행



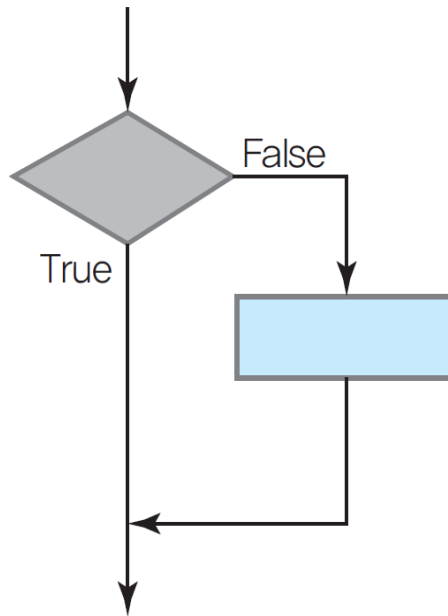
# 알고리즘의 구조: 조건 구조

- ❖ 조건에 따라 수행되는 작업들이 다른 경우
  - 조건이 맞으면 '참(True)'으로 표시된 방향의 작업 수행
  - 조건이 틀리면 '거짓(False)'으로 표시된 방향의 작업 수행

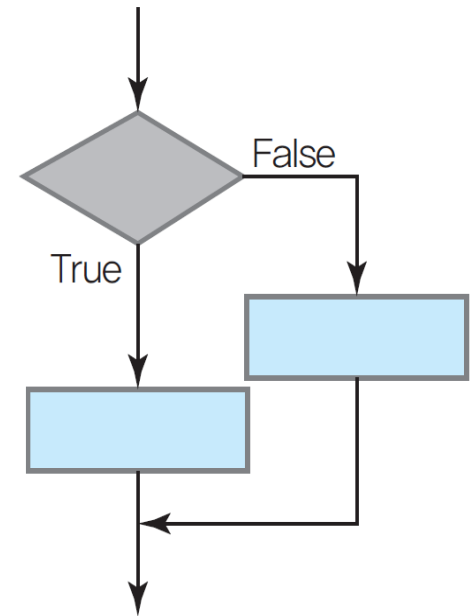
참의 경우에만  
수행되는 구조



거짓의 경우에만  
수행되는 구조

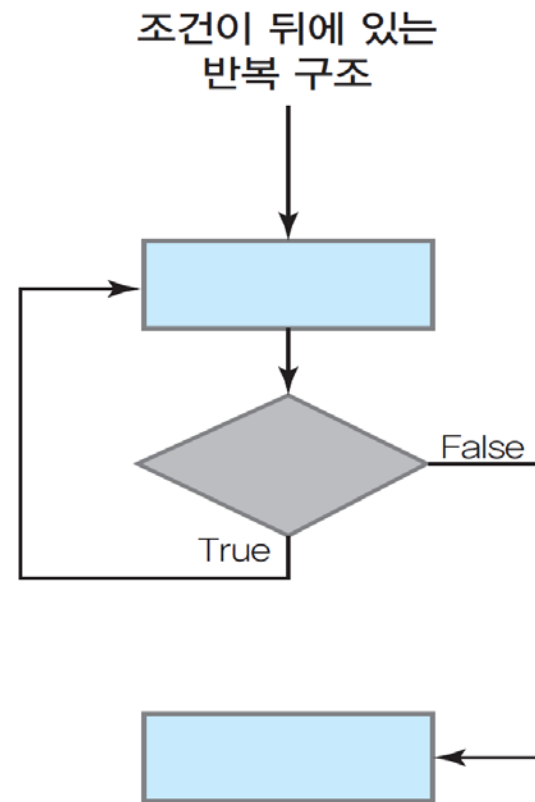
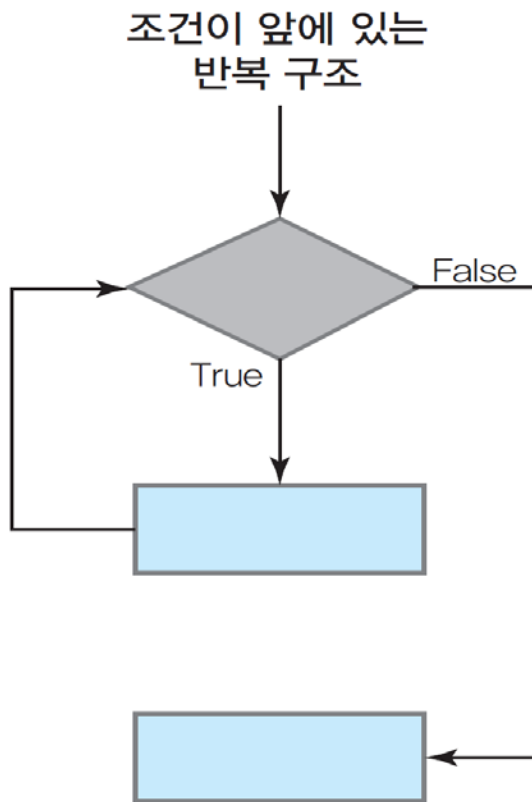


참과 거짓에 따라 다른  
작업을 수행되는 구조



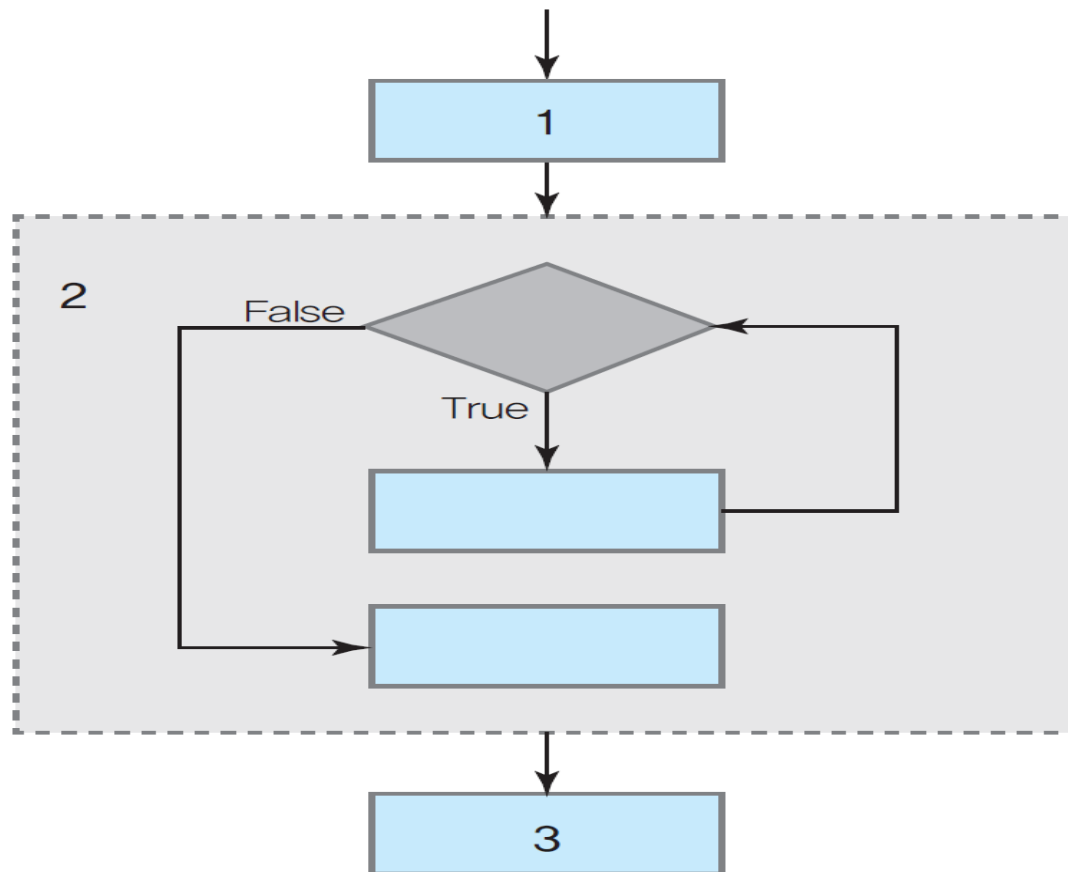
# 알고리즘의 구조: 반복 구조

- ❖ 일련의 작업들을 여러 번 반복해야 하는 경우
  - 특정한 조건을 만족하는 동안 반복



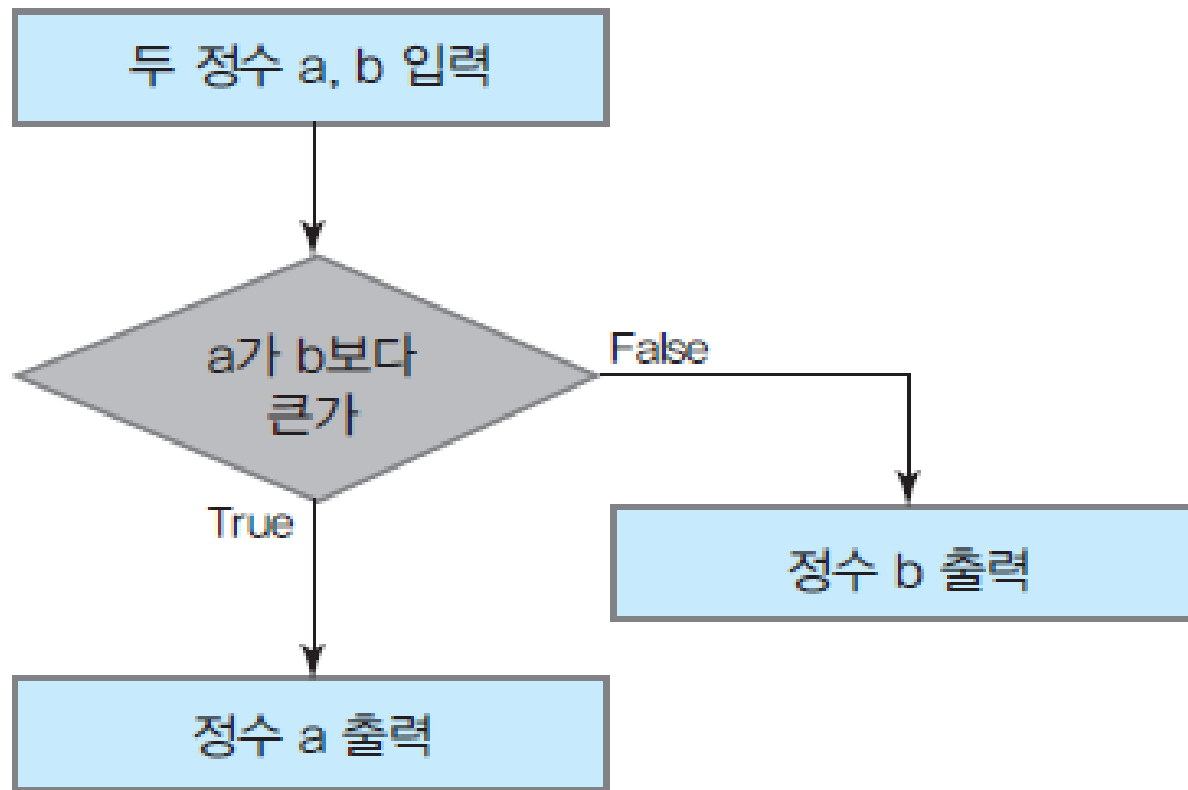
# 알고리즘의 구조: 조건 구조 + 반복 구조

❖ 조건 구조와 반복 구조를 포함하는 순차 구조



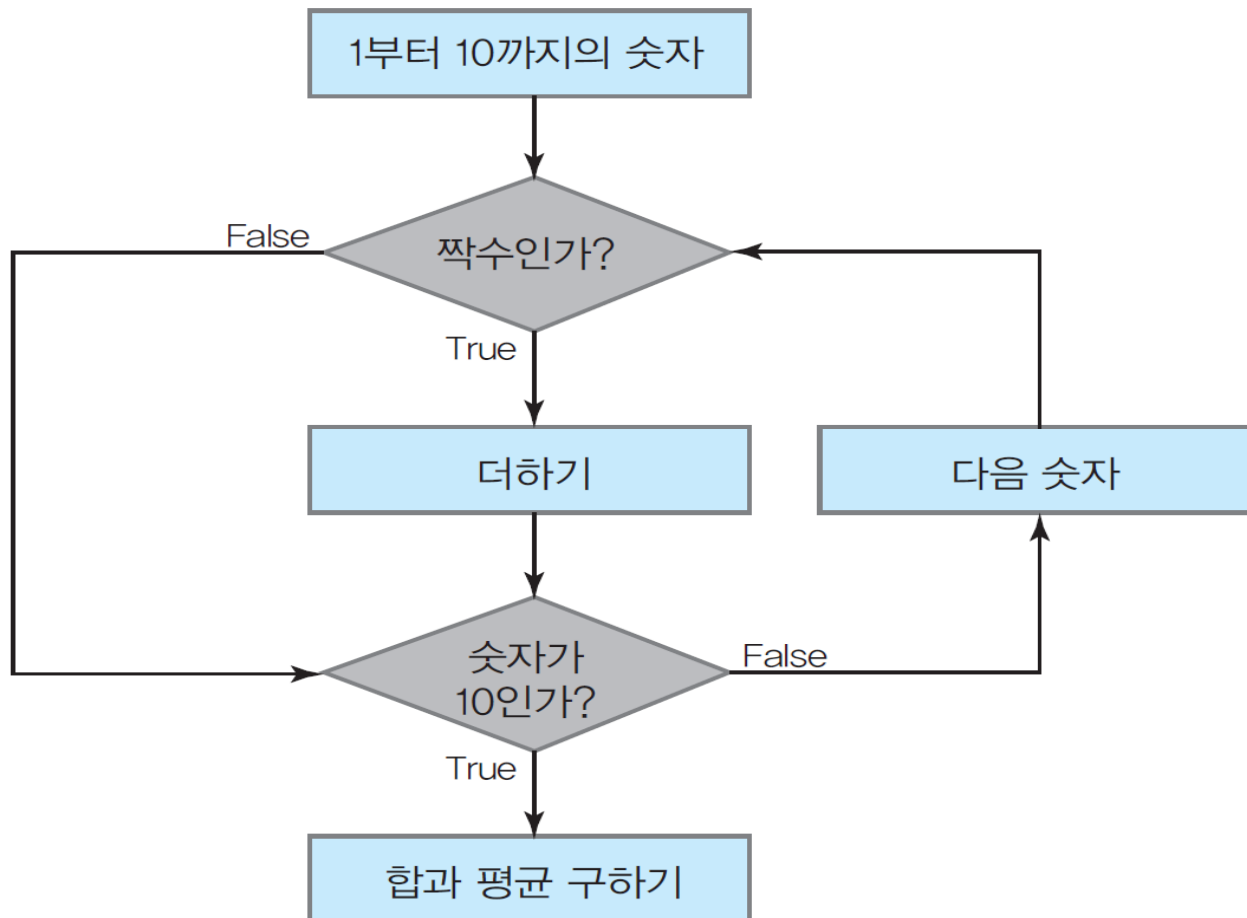
# 순서도 예제 (1)

- 두 정수를 입력했을 때, 두 정수 중 큰 수를 출력하는 순서도



# 순서도 예제 (2)

- 1부터 10까지의 숫자 중, 짝수의 합과 평균을 구하는 순서도



# 순서도 예제 (3)

- 팩토리얼 (*factorial*) 계산하는 알고리즘을 순서도로 구현해 볼 것
  - *cf)* 팩토리얼 예:  $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

- 1 변수 n에 대상 숫자를 할당합니다.
- 2 답변을 저장할 변수 ans를 생성합니다.
- 3 변수 ans에 변수 n의 값을 저장합니다.
- 4 변수 n에서 1씩 줄입니다.
- 5 반복될 때마다 ans에 n을 곱하고 그 값을 ans에 저장합니다.
- 6 n이 1이 될 때까지 반복합니다.

# 소프트웨어 (*software*)

- ❖ 사람이 하는 어렵고 힘든 작업을 컴퓨터가 대신 수행하도록 해주어 우리의 생활을 좀 더 편리하고 이롭게 해주는 도구
  - 예) 게임기
  - 예) 문서 편집기
  - 예) 오디오





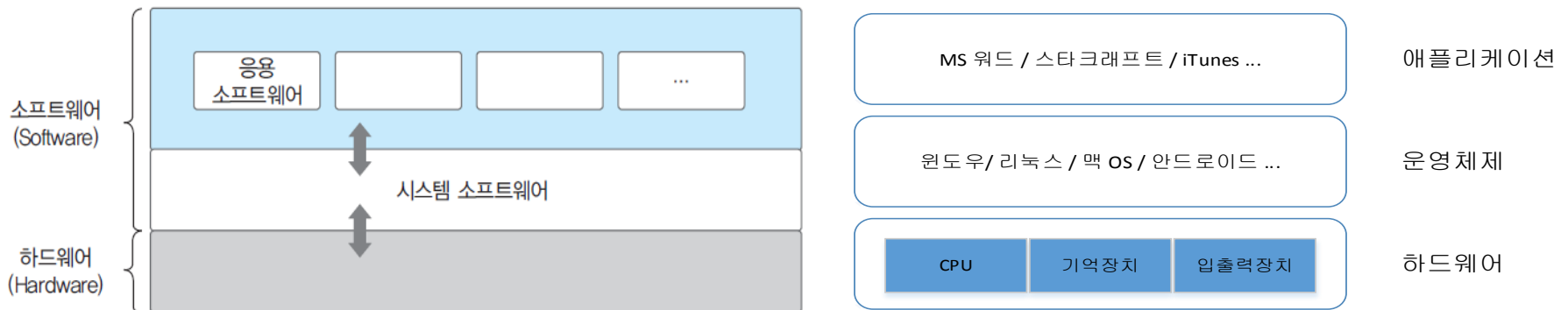
# 소프트웨어 유형

## ❖ 시스템 소프트웨어

- 컴퓨터 시스템을 운영하기 위한 운영체제 (*operating system*)
- 예) MS Windows, Mac OS 등

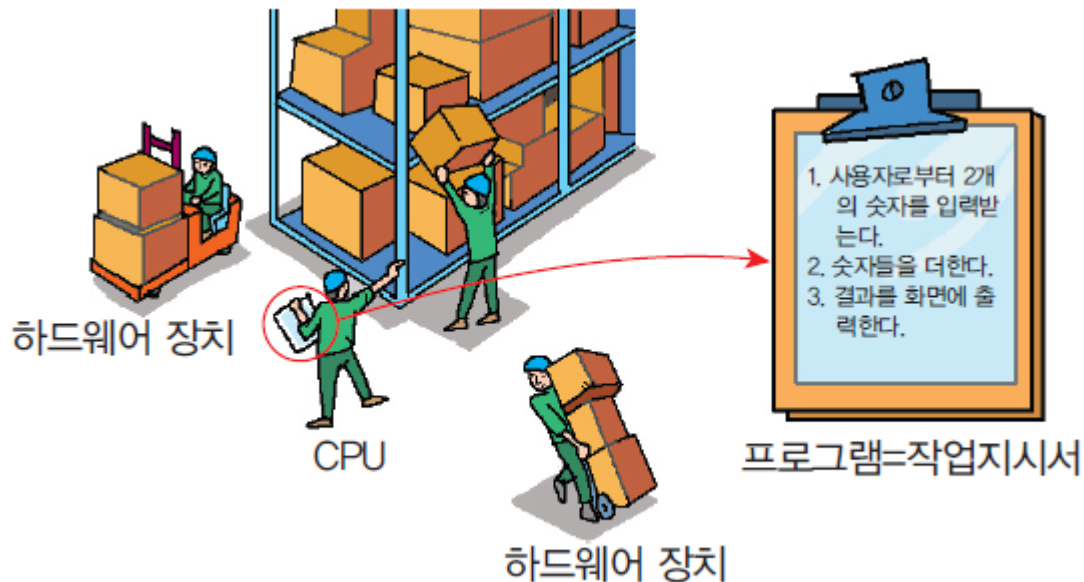
## ❖ 응용 소프트웨어

- 특정 작업을 위해 개발된 소프트웨어
- 예) Internet Explorer, MS Word 등



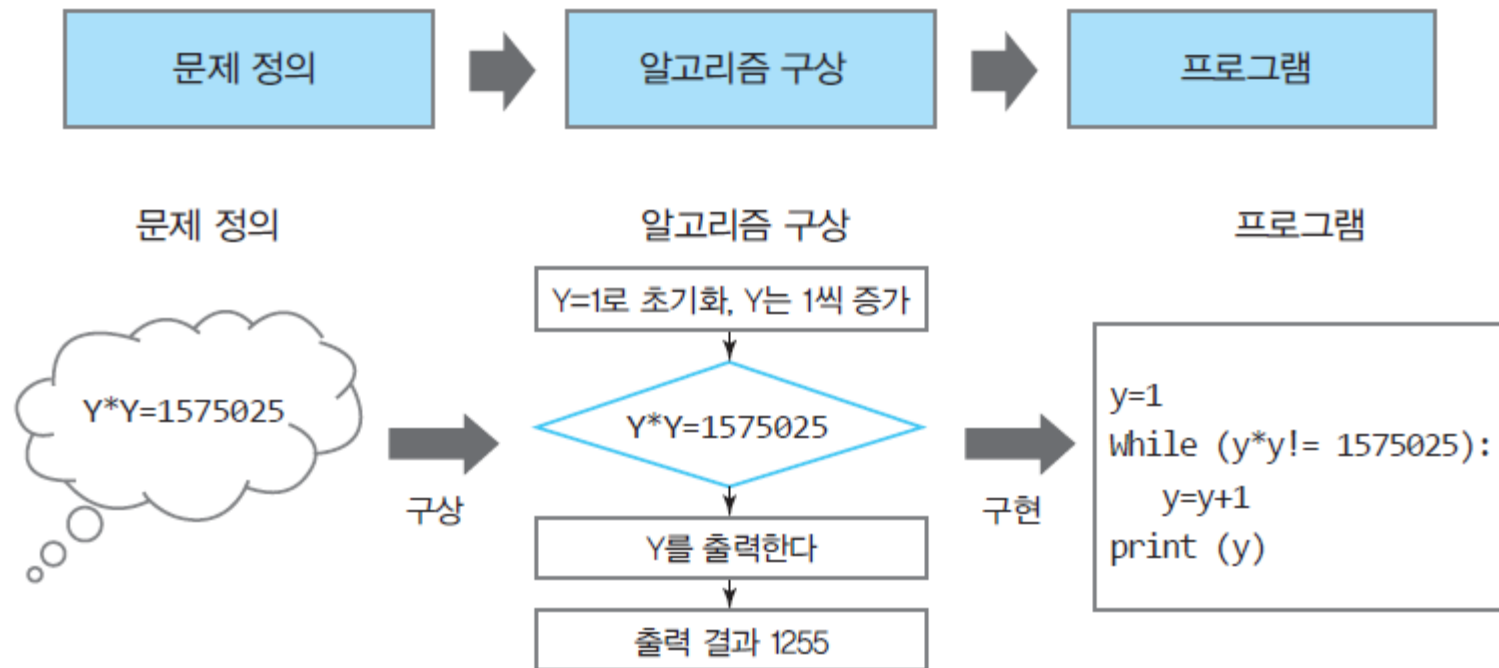
# (컴퓨터) 프로그램 (1)

- ❖ 컴퓨터에 일을 시키려면 인간이 컴퓨터에게 자세한 명령어 (*instruction*)들의 리스트를 주어야 함
  - 프로그램 : 컴퓨터가 수행할 명령어들을 적어놓은 문서



# (컴퓨터) 프로그램 (2)

- ❖ 구상한 알고리즘을 컴퓨터에 이해시키기 위해 변환한 형태
  - 설계한 알고리즘의 내용을 특정 프로그래밍 언어로 작성



# 프로그래밍 언어 (1)

- ❖ 컴퓨터에서 작동하는 소프트웨어(엑셀, 한글, 인터넷 익스플로러 등)를 만들기 위한 도구
  - 프로그래밍 언어를 사용해 소프트웨어나 앱을 만드는 사람을 프로그래머라 함
  - 프로그램을 작성하는 일을 흔히 코딩한다고 표현



# 프로그래밍 언어 (2)

## ❖ 컴퓨터와 소통하기 위해 사용하는 언어

### ■ 기계어 (*Low-level or Machine Language*)

- 컴퓨터가 이해하는 실질적인 언어
- 0과 1로만 구성됨

### ■ 고급언어 (*High-level Language*)

- 사람이 이해하기 어려운 기계어 대신 편하게 표현할 수 있는 언어

```
0100011101011010100110100101010100010  
10110101001101110010101101001010100101  
1010001010110101001101001010101000101  
0110101001101001010101000101011011101  
0011010010101010001010110101001101001  
0101111010001010110101001101001010101  
00010101101010011010010101
```

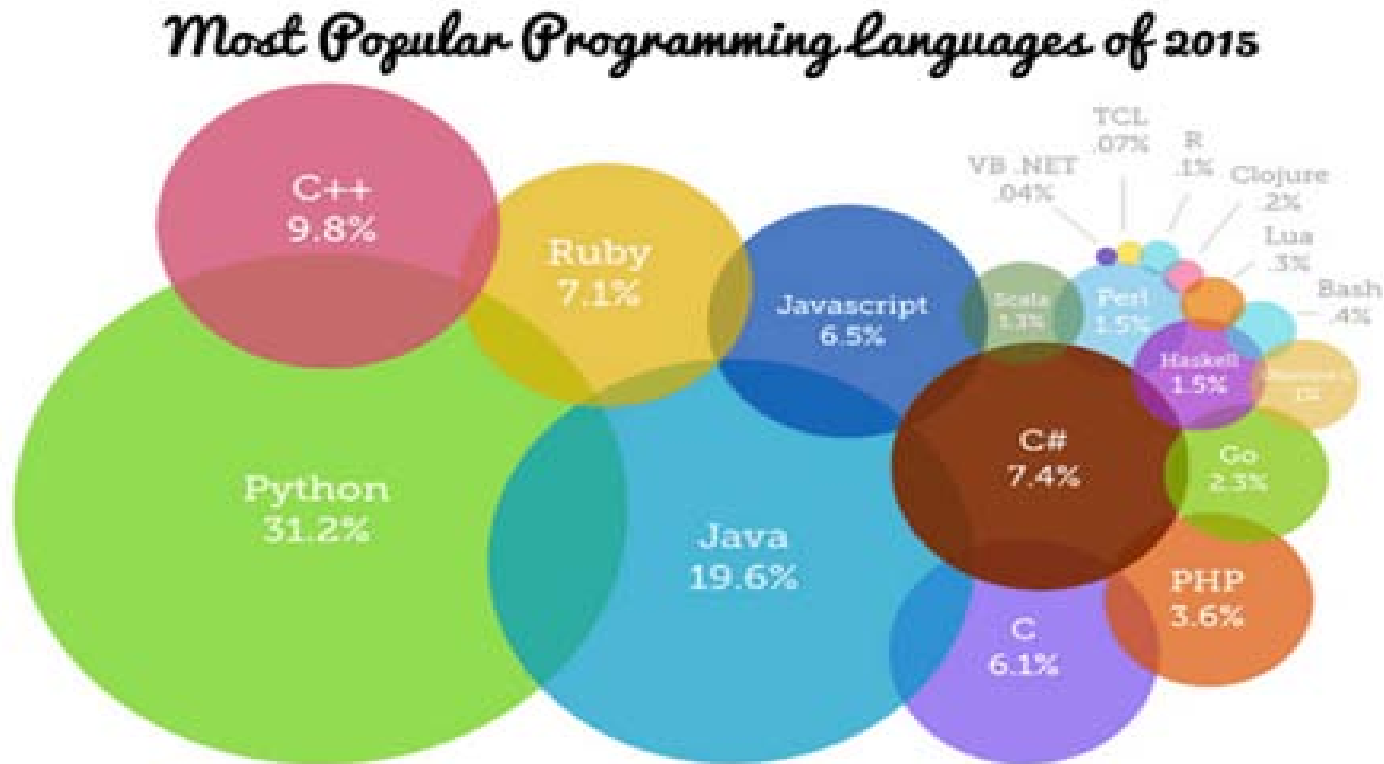
기계어

```
def uChoice():  
    print("다음 중 하나를 고르시오")  
  
    print("0 : 가위")  
    print("1 : 바위")  
    print("2 : 보")  
    uChoice = int(input())  
    return uChoice
```

고급언어

# 고급 프로그래밍 언어

❖ 많이 사용되는 고급언어들



# 컴파일러 / 구문

## ❖ Compiler

- 모든 고급언어들은 컴퓨터가 이해할 수 있는 언어인 기계어로 번역하는 소프트웨어를 필요로 함
- 이러한 소프트웨어를 컴파일러(*compiler*)라고 함

## ❖ Syntax

- 모든 프로그래밍 언어는 각각의 고유한 문법을 가짐
- 이러한 사용법을 구문(*syntax*)이라고 부름

# 프로그래밍 오류(*error*) 유형

## ❖ 구문오류

- 명령어들을 해당언어의 구문에 맞도록 컴퓨터에 입력해야 함
- 프로그래머가 프로그래밍 언어를 부정확하게 사용하면 컴파일러는 이를 구문오류(*syntax error*)라고 알려줌

## ❖ 실행오류

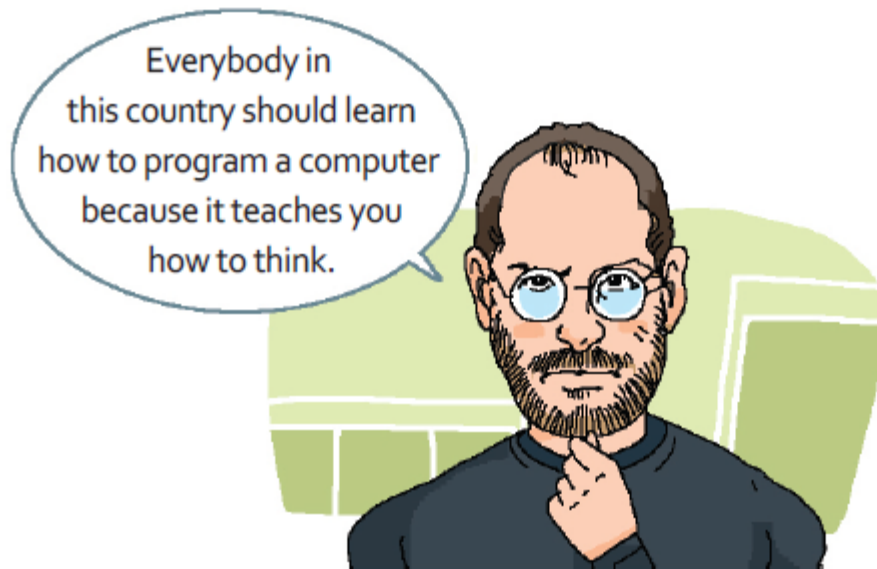
- 입력된 명령어들이 성공적으로 기계어로 컴파일된 이후에야 해당 프로그램이 수행(*run*)/실행(*execute*) 가능
- 실행시 발생하는 에러를 실행오류(*run-time error*)라고 함

## ❖ 결과오류



# 프로그래밍의 중요성

- ❖ “논리적으로 문제를 해결”하는 능력을 배양



이 나라 모든 사람들이 컴퓨터 프로그래밍을  
배워야 하는 이유는 사고하는 법을 가르쳐주기 때문입니다.  
- 스티브 잡스(Steve Jobs)