

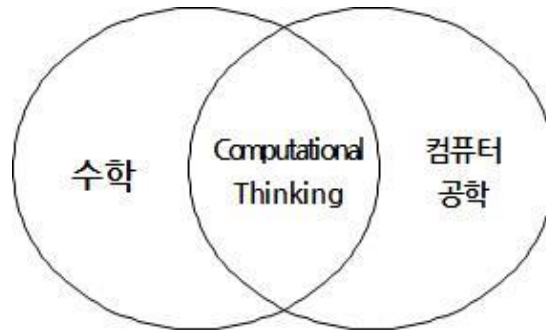
소프트웨어프로젝트 I

프로그래밍 언어와 프로그램

2022학년도 1학기

국민대학교 소프트웨어학부

지난 수업에는... 계산적 사고

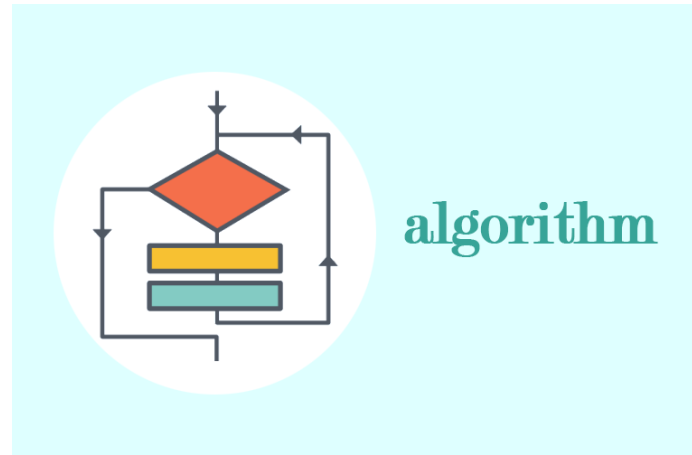


계산적 사고(Computational Thinking)는
컴퓨터(사람이나 기계)가 효과적으로 수행할 수 있도록
문제를 정의하고 그에 대한 답을 기술하는 것이
포함된 사고 과정 일체를 일컫는다.

계산적 사고 특징

- 자료를 분석하고 논리적으로 조직화
- 데이터 모형화, 자료 추상화, 모의시험
- 컴퓨터 도움을 받을 수 있도록 문제를 구성
- 가능한 해결책을 식별하고, 검증하고, 구현
- 알고리즘적 사고를 통해 해결책을 자동화
- 해당 과정을 다른 문제에 대해 일반화하고 적용

지난 수업에는... 알고리즘



알고리즘은 문제를 해결하기 위한 절차나 방법

알고리즘 조건

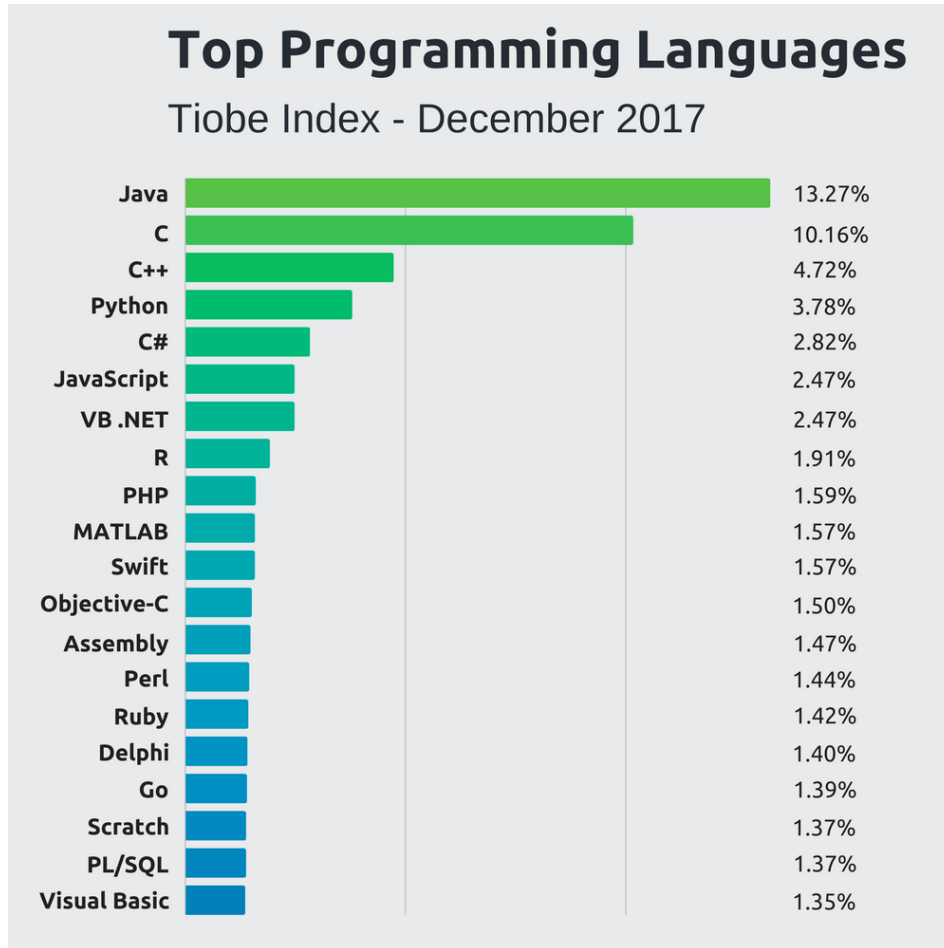
- 입력 - 0 또는 그 이상의 외부에서 제공된 자료가 존재
- 출력 - 최소 1개 이상의 결과를 가짐
- 명확성 - 각 단계는 명확하여 애매함이 없어야 함
- 유한성 - 단계들을 유한한 횟수로 거친 후 문제를 해결하고 종료해야 함
- 효과성 - 모든 연산들은 (사람이 종이와 연필을 이용하여) 유한한 시간 안에 정확하게 수행할 수 있을 정도로 충분히 단순해야 함

프로그래밍 언어



그림 출처: <https://towardsdatascience.com/top-10-in-demand-programming-languages-to-learn-in-2020-4462eb7d8d3e>

가장 많이 이용되는 프로그래밍 언어?



- 역시 Java 는 시들지 않고
- C 는 오래되었지만 임베디드 시스템 등에서 아직 널리 쓰이고
- Python 의 약진이 돋보이는 가운데
- 스타트업 등에서 가장 선호하는 것은 Ruby

데이터 제공: Tiobe (<https://www.tiobe.com/tiobe-index/>)

그림 출처: <https://techmeetups.com/most-popular-and-influential-programming-languages-of-2018/>

프로그래밍 패러다임의 종류

- 절차형 프로그래밍 (procedural programming)
 - 프로그램 실행 순서 (절차) 를 중심으로 기술
 - 대표: C, Pascal, Basic 등
- 함수형 프로그래밍 (functional programming)
 - 데이터에 대한 함수의 적용을 중심으로 기술
 - 대표: Lisp, Scala 등
- 객체지향 프로그래밍 (object-oriented programming)
 - 데이터와 데이터의 행동을 묶어 객체로 만들고 이 객체들을 조립하여 상호 작용을 기술
 - 대표: Java, C++, C# 등
- 이 구분은 프로그램의 기술 방식에 따른 것이지, 어느 프로그래밍 언어가 어디에 속한다고 말하는 것이 중요한 것이 아님

어떤 프로그래밍 언어를 택할 것인가?

- 어떤 프로그램을 만들려고 하는지에 따라 가장 좋은 선택이 달라짐
 - 웹 서버 프로그래밍: Java, PHP, Python, Ruby, ...
 - 웹 클라이언트 프로그래밍: Javascript, jQuery, ...
 - 임베디드시스템 프로그래밍: C, C++, ...
 - 모바일 앱 프로그래밍: Java, Kotlin, C#, Swift, ...
 - 통계/시뮬레이션: MatLab, R, ...

프로그래밍 언어가 왜 이리 많은가요?

혹시, 프로그래밍을 배우려는
우리들을 괴롭히기 위해서?

→ **아닙니다!**

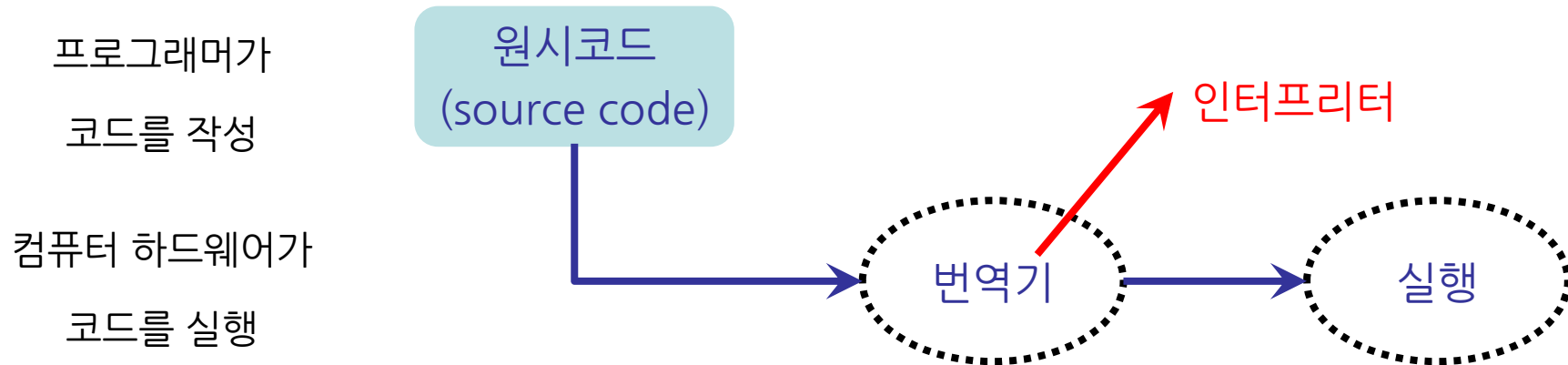
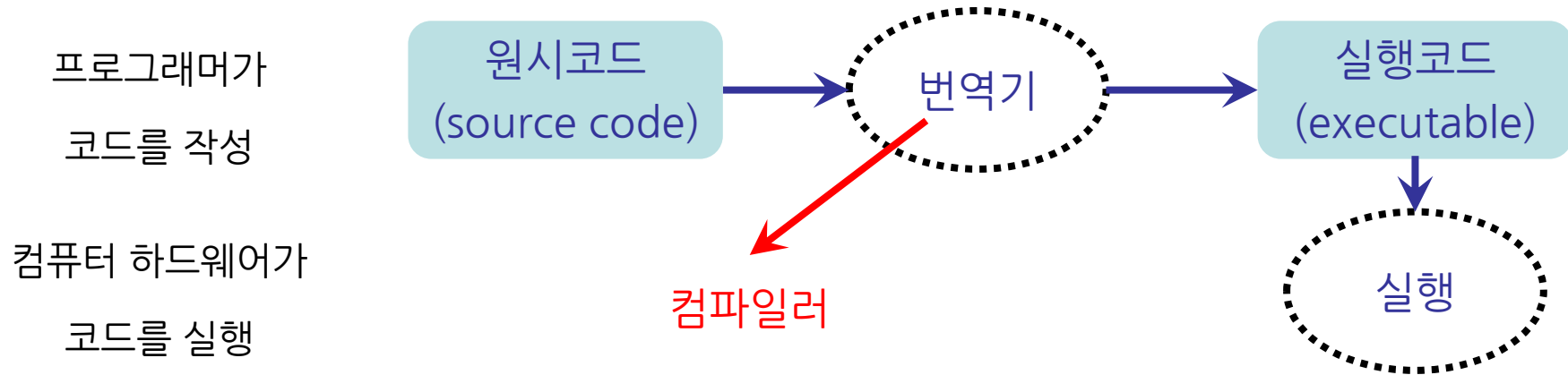


한두 가지 익히고 나면,
다른 것을 배우기는
생각보다 쉽습니다.

특정한 응용을 염두에 두었을
때 그에 알맞은 프로그램을
기술하기 위하여 적합한
언어들이 개발되어 왔기 때문

프로그래밍을 배우는 것은
프로그래밍 언어를 익히는 것이라기보다는
컴퓨터에게 효과적으로 일을 시킬 수 있는
어법 (프로그래밍 기법) 을 배우는 것이니까요.

프로그램의 번역과 실행



인터프리터를 이용한 프로그램 실행

Python 인터프리터를 불러내어

```
[sheayun@Sheayuns-KMU-Mac swp1]$ python
Python 2.7.16 (default, Jun 19 2019, 07:40:37)
[GCC 4.2.1 Compatible Apple LLVM 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 3 * 7
21
>>> print("Kookmin SW")
Kookmin SW
>>>
[sheayun@Sheayuns-KMU-Mac swp1]$ cat hello.py
print("Hello, world!")
[sheayun@Sheayuns-KMU-Mac swp1]$ python hello.py
Hello, world!
[sheayun@Sheayuns-KMU-Mac swp1]$
```

대화형으로 문장을 실행시킨다든지

프로그램 소스를 작성해 두고

인터프리터로 하여금 번역하여 실행하도록

컴파일러를 이용한 프로그램 번역

```
swp1 — -bash — 100x36
[sheayun@Sheayuns-KMU-Mac swp1]$ cat hello.c
#include <stdio.h>
int main()
{
    printf("Hello, world!\n");
}

[sheayun@Sheayuns-KMU-Mac swp1]$ gcc -o hello hello.c
[sheayun@Sheayuns-KMU-Mac swp1]$ ./hello
Hello, world!
[sheayun@Sheayuns-KMU-Mac swp1]$ ./hello
Hello, world!
[sheayun@Sheayuns-KMU-Mac swp1]$
```

} 프로그램 소스를 작성해 두고
(Python 에서보다 좀 길구나?)

컴파일러로 하여금 번역하여
실행 가능한 파일을 만들어 두도록 하고

프로그램의 실행은
이미 만들어진 실행 파일을 불러서
(또 실행하는 데에는 번역 작업 불필요)

프로그램과 데이터

데이터가 없으면 프로그램도 무용지물

상수 (constant):

프로그램에 직접 수를 적은 것, 프로그램에 의하여 변화할 수 없다.

변수 (variable):

수 등의 데이터를 담기 위한 저장 공간에 붙인 이름
프로그램의 실행에 의하여 동적으로 값이 변화할 수 있다.

$a = 3$

$b = a + 5$

3 과 5 는 상수 (정수형)

a 와 b 는 변수 (정수형)

대입 (assignment): 변수에 값을 저장하는 일

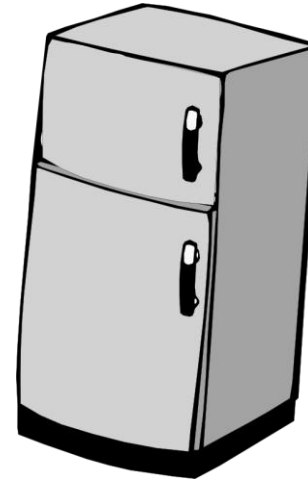
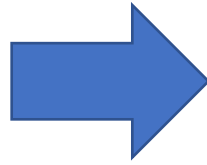
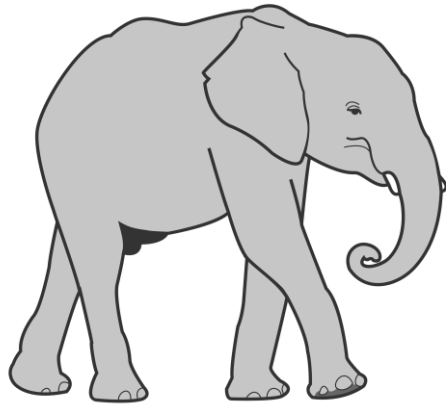
* 위 코드의 실행 결과로 b 에 대입된 값은?

데이터형 (Data Types)

- 데이터에는 타입 (type) 이 있습니다.
 - 컴퓨터 내부에서 어떻게 표현되고 어떻게 연산되는지 (지난 강의 “정보의 표현과 컴퓨터 하드웨어” 참고) 와 연관
 - 정수형 (integer), 부동소수점 실수형 (float)
 - 문자 (character), 문자열 (string)
 - 복합형 (데이터를 모아서 만든 타입)
- 프로그래밍 언어마다 가지고 있는 데이터 타입이 다름

어떤 프로그래밍 언어는 타입을 프로그래머가 일일이 지정해야 하고 (예: Java)
다른 프로그래밍 언어에서는 컴퓨터가 알아서 하기도 해요 (예: Python).

프로그램의 순차적 실행



1. 냉장고 문을 연다.
2. 코끼리를 냉장고 안에 넣는다.
3. 냉장고 문을 닫는다.

프로그래밍이 어려운 이유 중 하나는
컴퓨터에게는 이런 단계들을
매우 잘게 나누어서 일을 시켜 주어야 하기 때문

프로그램의 순차적 실행

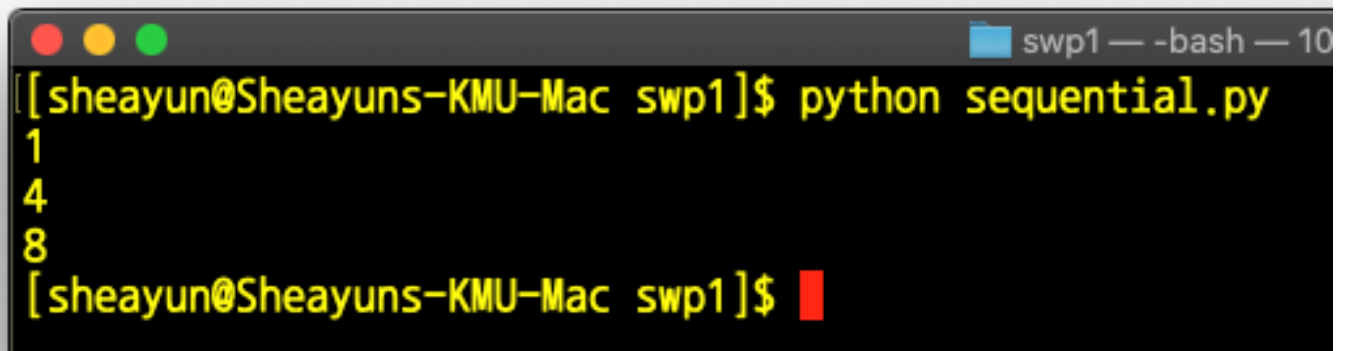
```
1 a = 1
2 print(a)
3
4 b = a + 3
5 print(b)
6
7 a = b * 2
8 print(a)
```

프로그램 소스 코드
(Python)

그런데, 이렇게
순서대로만 실행해서야
쓸모 있는 일이 될까요?

→ 걱정하지 않아도 됩니다.

실행 결과



```
swp1 — -bash — 10
[sheayun@Sheayuns-KMU-Mac swp1]$ python sequential.py
1
4
8
[sheayun@Sheayuns-KMU-Mac swp1]$
```

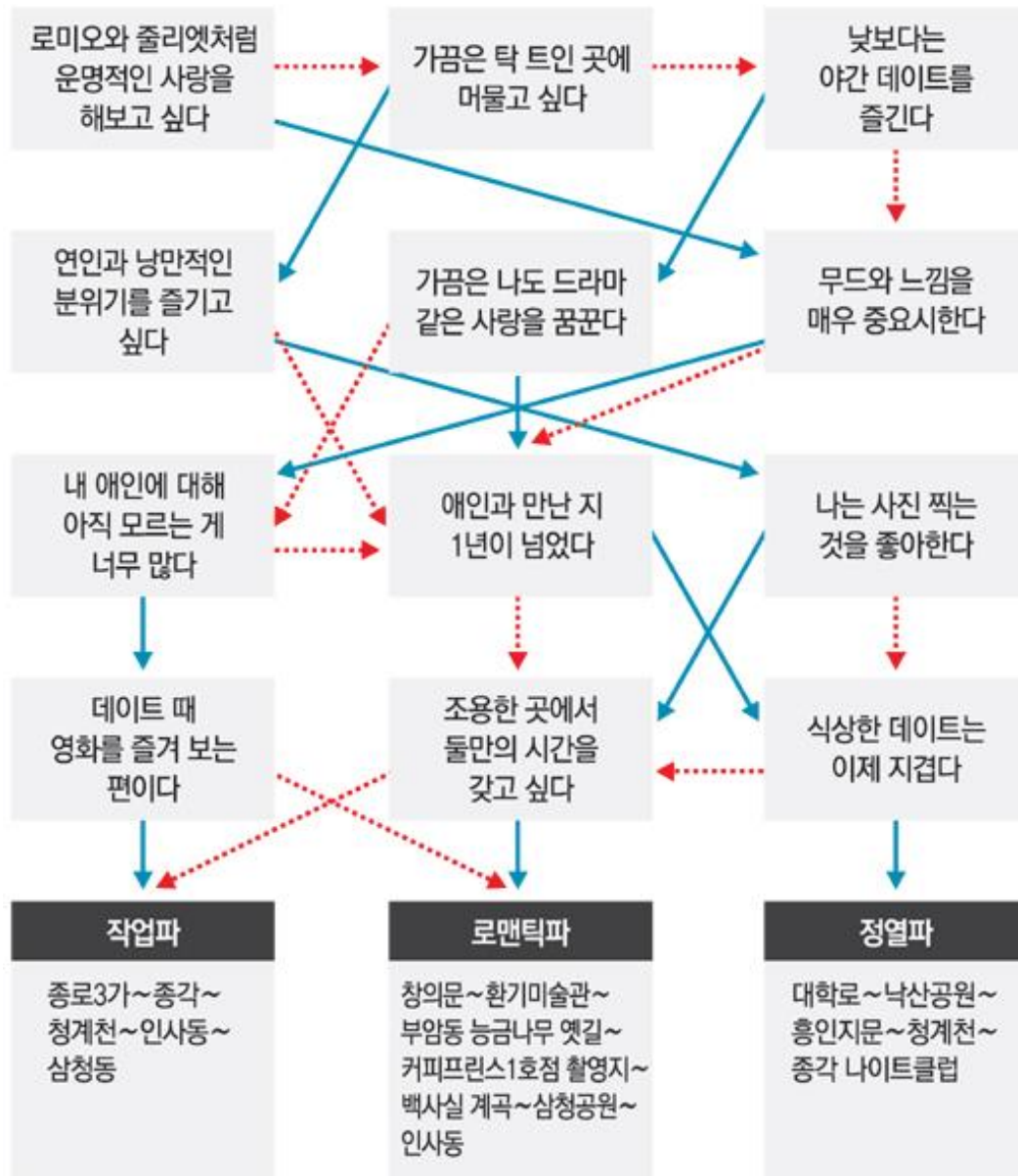
선택 (Selection)

조건 (condition) 에 따라
서로 다른 일을 수행

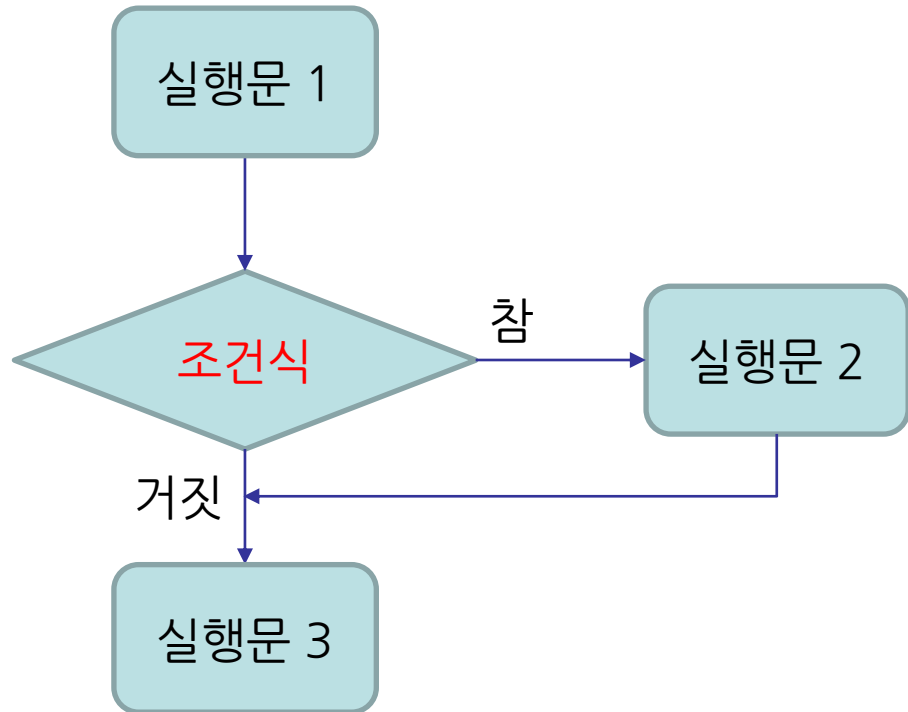
시작!

결과

종로 데이트코스 심리 테스트 YES → NO →



조건문 (Conditional Statements)



실행문 1

if 조건식:

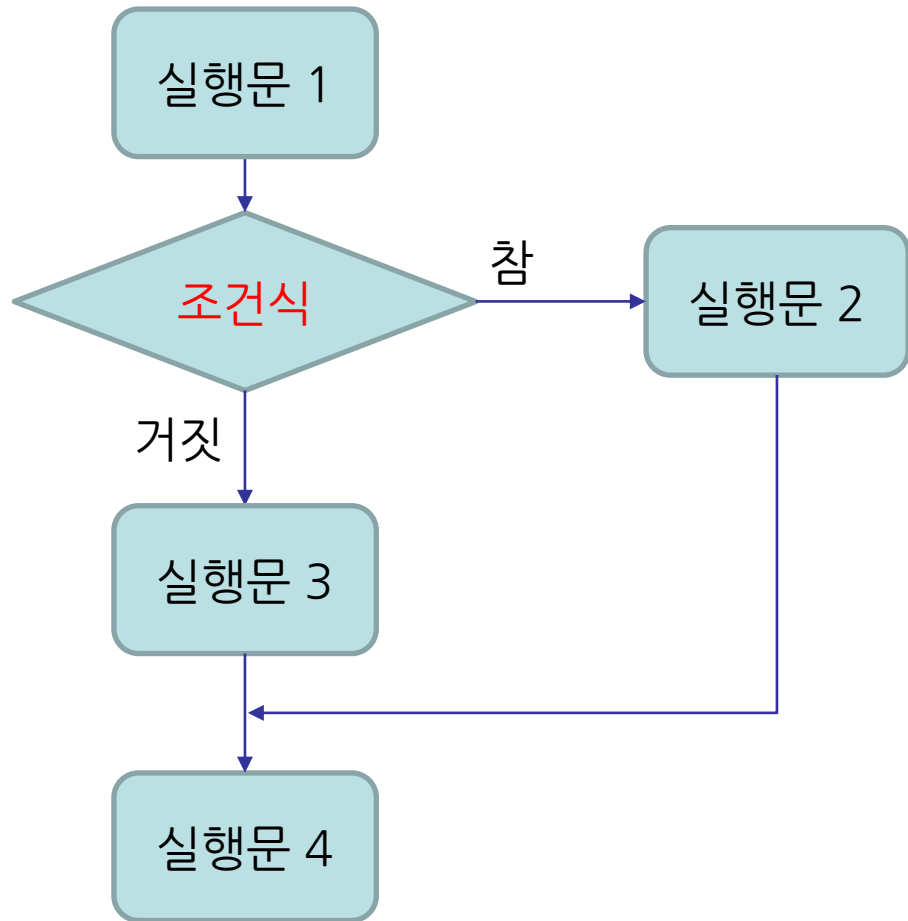
실행문 2

실행문 3

실행문 2 는 조건식이 참인 경우에만 실행
(그렇지 않은 경우에는 그냥 지나감)

1. 항공요금을 계산
2. “만약” 수하물 무게가 얼마 이상이면 부가요금을 가산
3. 요금을 승객에게 안내

조건문 (Conditional Statements)



실행문 1

if 조건식:

실행문 2

else:

실행문 3

실행문 4

조건식이 참인 경우에는 실행문 2 를,
그렇지 않은 경우에는 실행문 3 을 실행하고
그 다음 (실행문 4) 으로 진행

예 1) 눈이 오면 대중교통을 이용하고, 그렇지 않으면 자가용을 이용한다.

예 2) 성적이 60 점 이상이면 합격이고, 그렇지 않으면 불합격이다.

약간 복잡한 조건



놀이기구를 타려면,
나이 여섯 살 이상, 키 110cm 이상이어야 합니다.



나이 ≥ 6

그리고

키 ≥ 110

다른 예:

- 비가 오지 않는 휴일에는 테니스를 친다.
 - 비가 오지 않는다, 그리고, 휴일이다.
- 나이가 30세 미만이거나 TOEIC 점수가 700 점 이상이면 채용한다.
 - 나이 < 30 , 또는, TOEIC 점수 ≥ 700

반복 (Repetition)

그림 출처: 영화 “모던 타임즈 (Modern Times)” 에서 발췌



단순 반복은 사람에게는 지겨운 일이지만 (실수도 일어날 수 있음)

컴퓨터가 가장 자신 있게 할 수 있는 일이다!

→ 때로는 위대한 결과를 낳을 수도 있음

반복 (Repetition)

화면에 "환영합니다" 를 다섯 번 출력하세요!

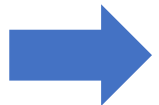
```
print("환영합니다.")  
print("환영합니다.")  
print("환영합니다.")  
print("환영합니다.")  
print("환영합니다.")
```

이것은, 의도에 정확히 맞게 동작하는
유효한 Python 프로그램입니다.

왼쪽의 프로그램은 좋은 코드인가요, 아닌가요?

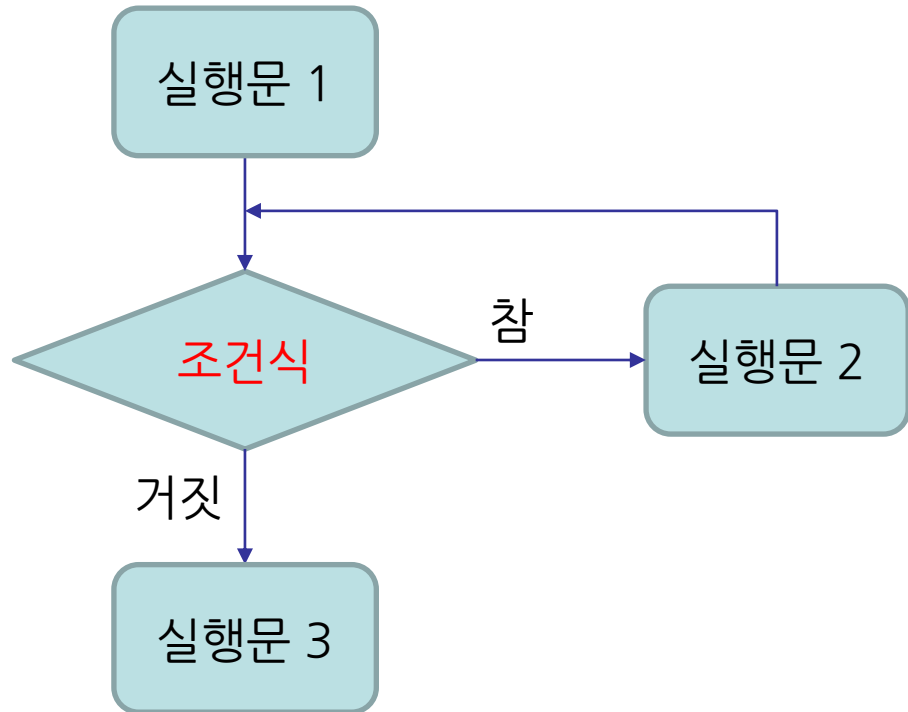
만약 아니라면, 왜 아닌가요?

1. 다섯 번이 아니라 100 번이라면? 10억 번이라면?
2. 똑같이 입력한다는 보장이 있나요? 실수 가능성은?
3. 게다가, 만약 “환영해요” 로 모두 바꾸어야 한다면?
4. “철수, 환영해요”, “미숙, 환영해요”, ...



반복하는 동작은 컴퓨터에게 시키는 것이 좋습니다.

순환문 (Loops)



실행문 1

while 조건식:

실행문 2

실행문 3

실행문 2 는 조건식이 참인 **동안**에만 실행
(그렇지 않은 경우에는 다음으로 진행)

1. 에어컨을 켜고
2. 실내 기온이 24도 이상인 동안 계속 기온을 측정
3. 에어컨을 끈다.

예: 원리합계가 목표금액을 넘을 때까지

초기 원금은 1,000 원, 연이율은 5% 라고 할 때,
최소 몇 년이 지나야 원리합계가 2,000 원 이상 (두 배 이상) 이 될까?

초기 원금은 1,000 원이다.

year = 0

while (원리합계가 2,000 원보다 작으면):

year 를 1 만큼 증가시키고

현재 원리합계에 0.05 를 곱하여 이자를 계산한다.

이자를 현재 원리합계에 더한다.

지금까지 몇 년이 흘렀을까?

그래서, 원리합계는 얼마가 되었을까?

balance = 1000

year = 0

while balance < 2000:

year += 1

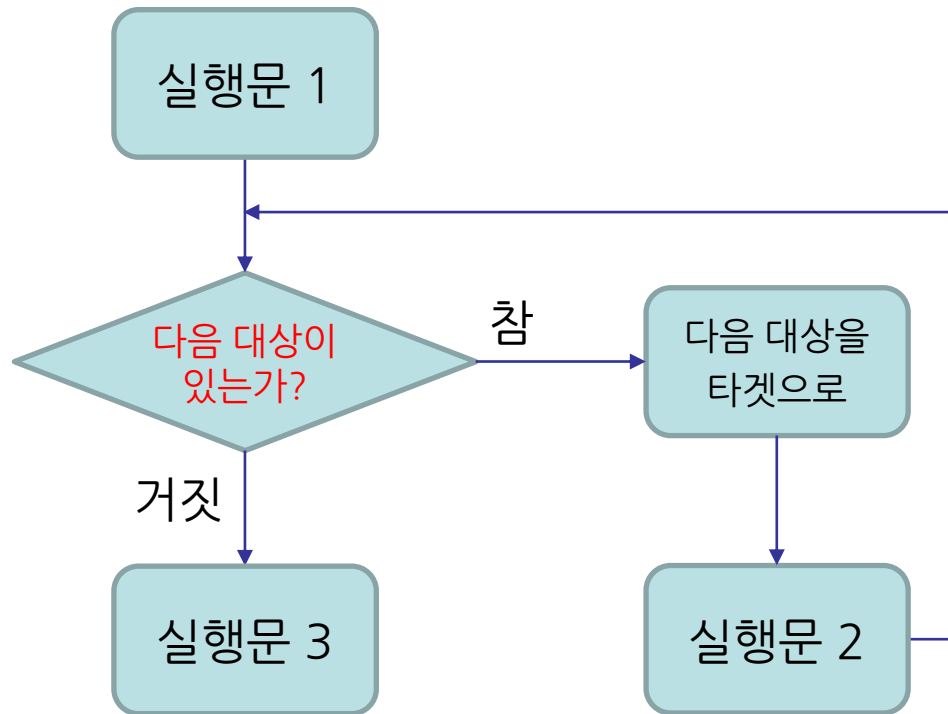
interest = balance * 0.05

balance += interest

print("기간: %d 년" % year)

print("총액: %d 원" % balance)

순환문 (Loops)



실행문 1

for 타겟 **in** 대상들:

실행문 2

실행문 3

대상들 각각에 대하여 실행문 2 를 실행
(모든 대상을 처리한 후에는 다음으로 진행)

1. 출석부를 꺼내어
2. 출석한 각 학생 이름 옆에 동그라미 표시를 하고
3. 출석부를 덮는다.

예: 자음과 모음 각각을 세어 볼까

```
문장 = "student"
```

```
vowels = 0
```

```
consonants = 0
```

```
for 글자 in 문장:
```

```
    if 글자가 모음이면:
```

```
        vowels += 1
```

```
    else:
```

```
        consonants += 1
```

```
print("vowels = %d" % vowels)
```

```
print("consonants = %d" % consonants)
```



```
vowels = 2
```

```
consonants = 5
```

프로그램의 구성

- 데이터, 데이터형, 자료 구조

- 목적에 적합한 데이터 객체를 갖추
- 매우 간단한 경우가 아니라면 기본형을 조합한 구조를 이용

Python 에는 편리한 복합 데이터 타입들이 있어요!

→ “과학과소프트웨어적사고” 에서 배우게 될 것

- 구문 구조

- 선택 (조건문) 과 반복 (순환문) 을 조합
- 데이터를 어떻게 처리할지를 기술

논리적/계산적 사고력이 필요하고

경험에 의한 언어 구사 기술도 필요합니다.

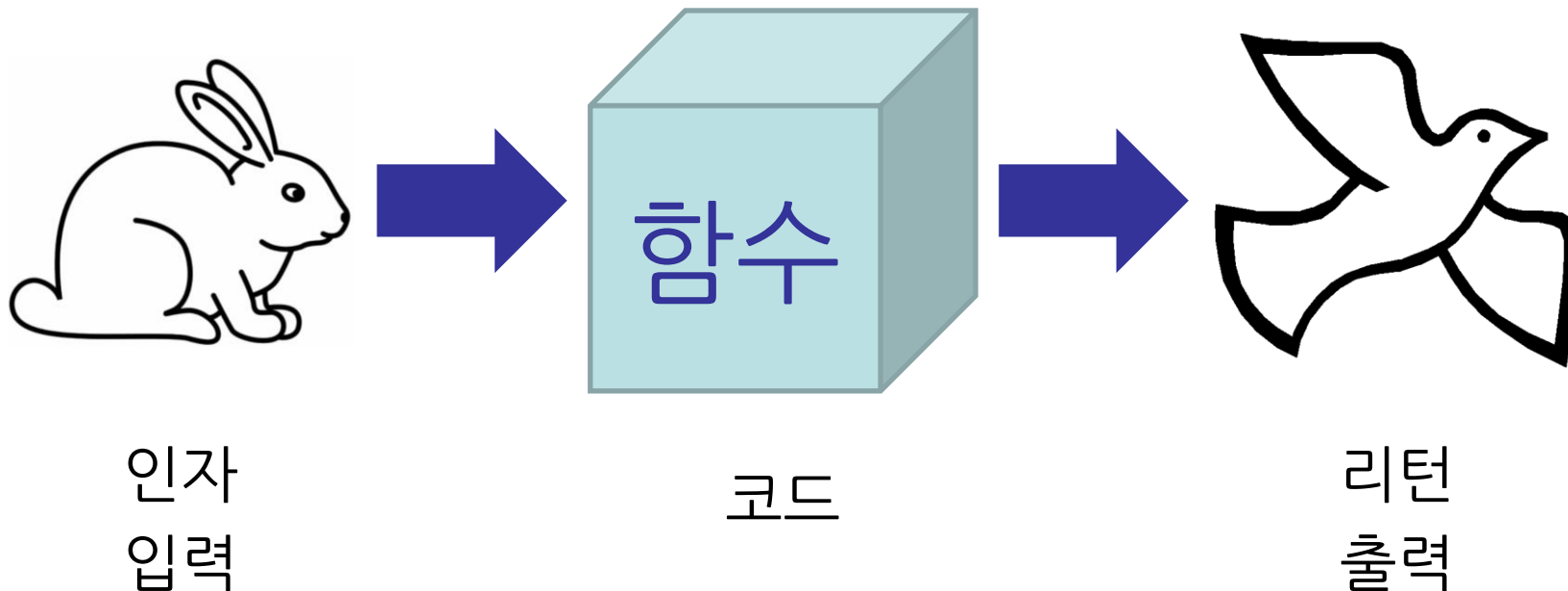
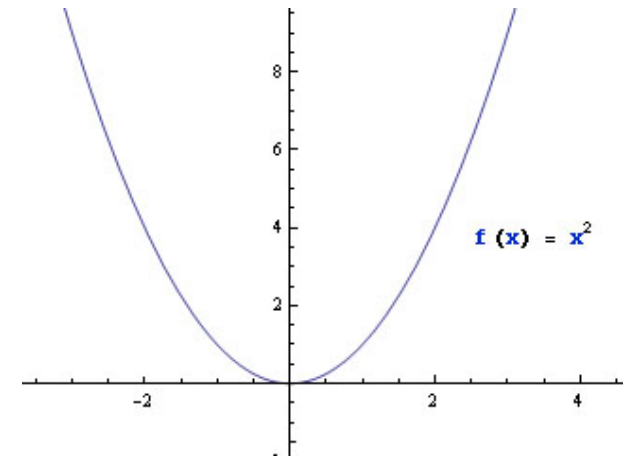
구조화된 프로그래밍

- 동일한 코드의 반복은 좋지 않은 프로그래밍
 - 코드의 낭비가 심하다, 라는 점 외에도, 더 안 좋은 점은:
 - 동일한 코드를 동일하게 다 고쳐야 할 경우가 생기면?
- 기능에 따라 코드를 잘게 나누면
 - 논리적, 구조적 사고에 도움이 되고, 이 구조를 프로그래밍 언어로 표현하는 데에도 도움이 되며
 - 코드의 가독성 (readability) 과 유지보수성 (maintainability) 이 높아지는 장점도 있음

함수 (Functions)

함수 (函數) 란?
상자

$$f(x) = x^2$$



함수의 이용

함수의 정의 (definition)

함수 `congrats` (인자로 “선수” 를 받음):

선수의 목에 메달을 걸어 준다

선수에게 꽃다발을 준다

선수와 악수한다

여러 선수에게 메달과 꽃다발을 수여하고
그들과 악수를 하게 될 터이니,
이 행동의 묶음을 “함수” 로 만들어 두자!

정의해 둔 함수를
호출 (call) 하여 이용

for player in team:

`congrats(player)`

이 코드를 매우 간단하게 만들 수 있었고
선수를 축하하는 절차가 변경되어도
이 쪽에는 아무런 수정이 필요하지 않음

요약 (1)

- 프로그램이란
 - 컴퓨터에게 쓸모 있는 일을 시키기 위한 작업 지시서
 - (계산적) 사고의 흐름을 프로그래밍 언어로 기술한 것
- 프로그램을 만들기 위해서는
 - 해결해야 할 문제를 “잘” 정의하고
 - 이에 맞는 프로그래밍 언어를 선정하여
 - 코드를 작성, 번역하여 실행할 수 있도록 개발

요약 (2)

- 프로그램의 구성
 - 코드와 데이터로 이루어져 있음
- 소프트웨어를 만들 (프로그램을 개발할) 수 있으려면
 - 데이터를 다루는 방법을 익혀야 하고
 - 프로그램 구문 구조를 활용할 수 있어야 하며 (선택, 반복)
 - (덩치가 커지면) 구조화된 설계를 할 수 있어야 함