



## CHAPTER 1

# 파이썬과 프로그래밍 소개

## Overview of Python & Programming



박진수 교수  
서울대학교·경영대학  
[jinsoo@snu.ac.kr](mailto:jinsoo@snu.ac.kr)



# 학습 목차

- 강의에 앞서...
- 프로그래밍이란?
- 왜 파이썬인가?
- 파이썬 특징
- 프로그램과 프로그램 개발 절차

# 강의에 앞서...

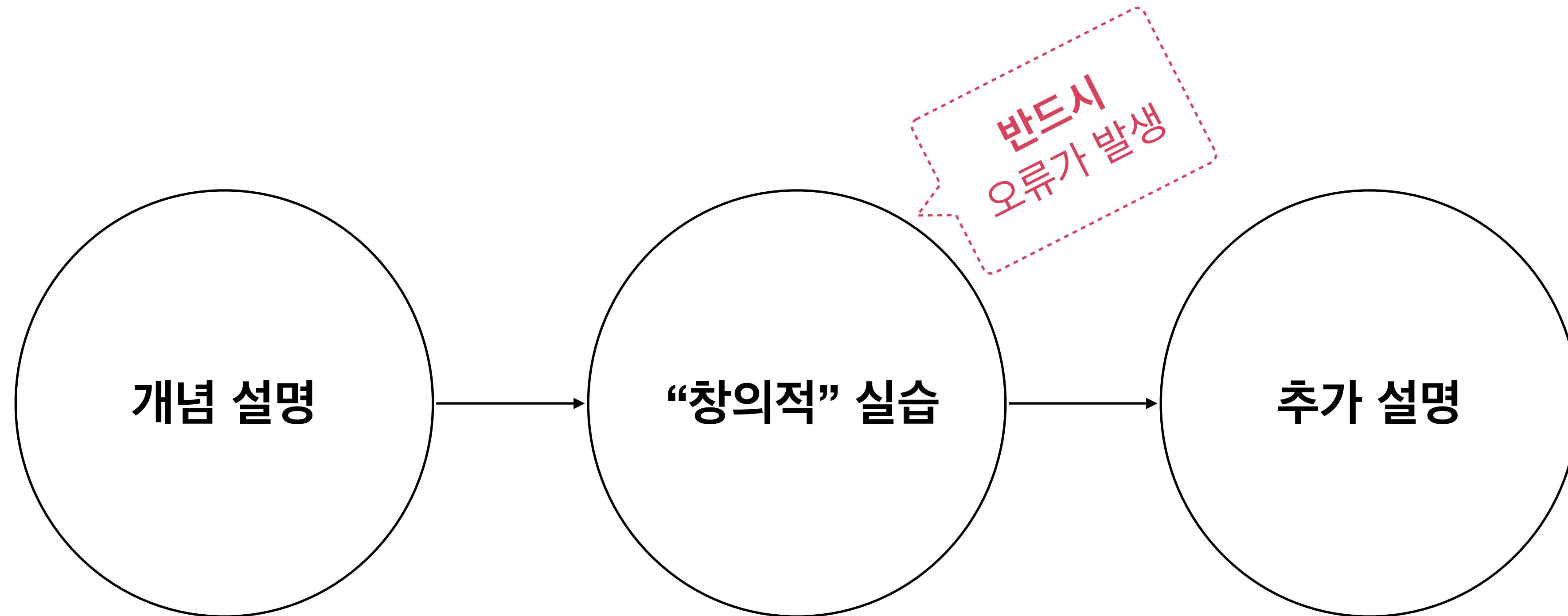
---

## Before Getting Started





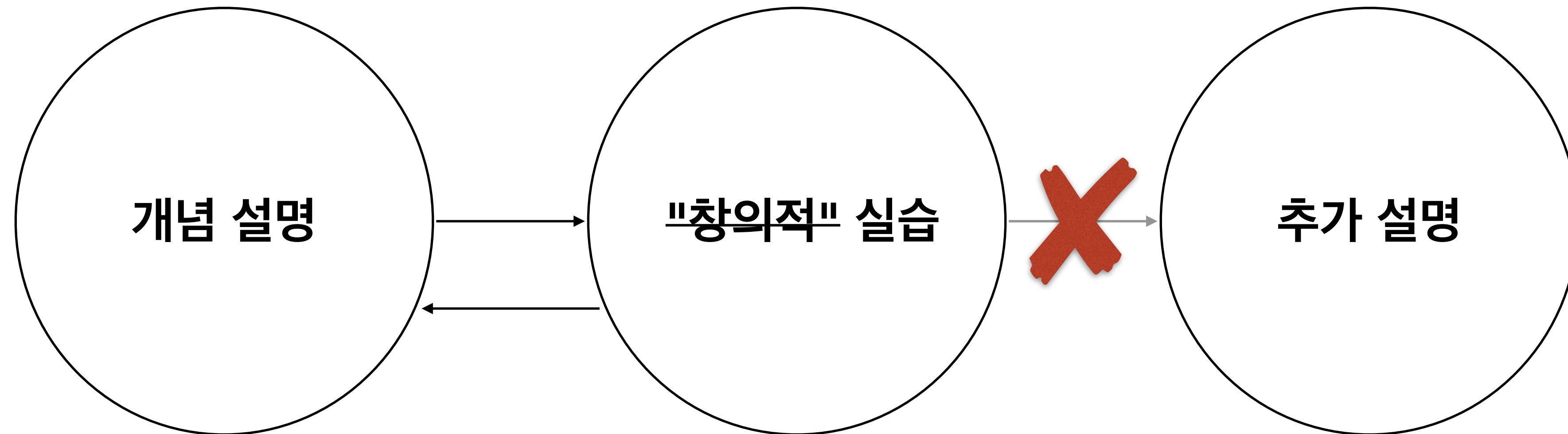
## 질문이 만들어가는 강의





질문이 만들어가는 강의

“질문이 없다면?”



- 바로 쓰는 파이썬: 기초 편 | 박진수 저 | 서울대학교 출판문화원 | 2019 | ISBN 9788952128867

- 소스 코드, 연습문제 풀이, 부록
  - [github.com/snu-python/pythonbook](https://github.com/snu-python/pythonbook)
  - [snupress.com](http://snupress.com) > 고객센터 > 자료실 > 바로 쓰는 파이썬
- SNUON 동영상 강의
  - <https://etl.snu.ac.kr/> 접속 후 회원가입
  - 로그인하고 eTL 메인 화면에서 왼쪽 메뉴 중 SNUON > 강좌 목록
  - 검색창에 ‘바로 쓰는 파이썬’ 검색
  - 강좌 클릭 후 수강신청
  - 강의실로 이동하여 동영상 강의 시청





<https://jinsooya.github.io/LN/>



## 실습(In-Class Exercise)



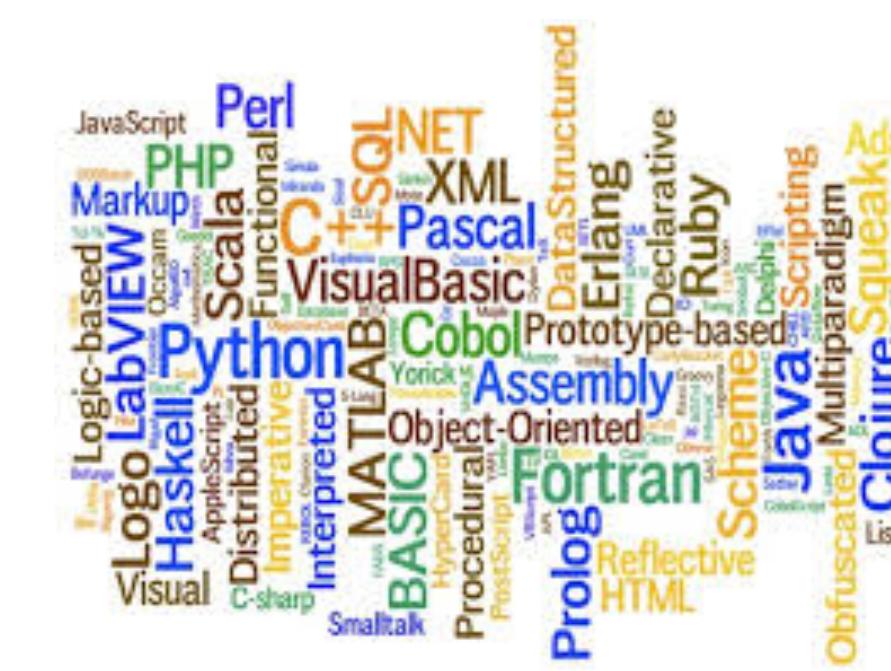
### 앞선 주제

=> 지금은 건너뛰어도 되는 주제  
(뒤에서 다루거나 수준이 올라가면 알 수 있는 내용)

# 프로그래밍이란?

---

## Programming



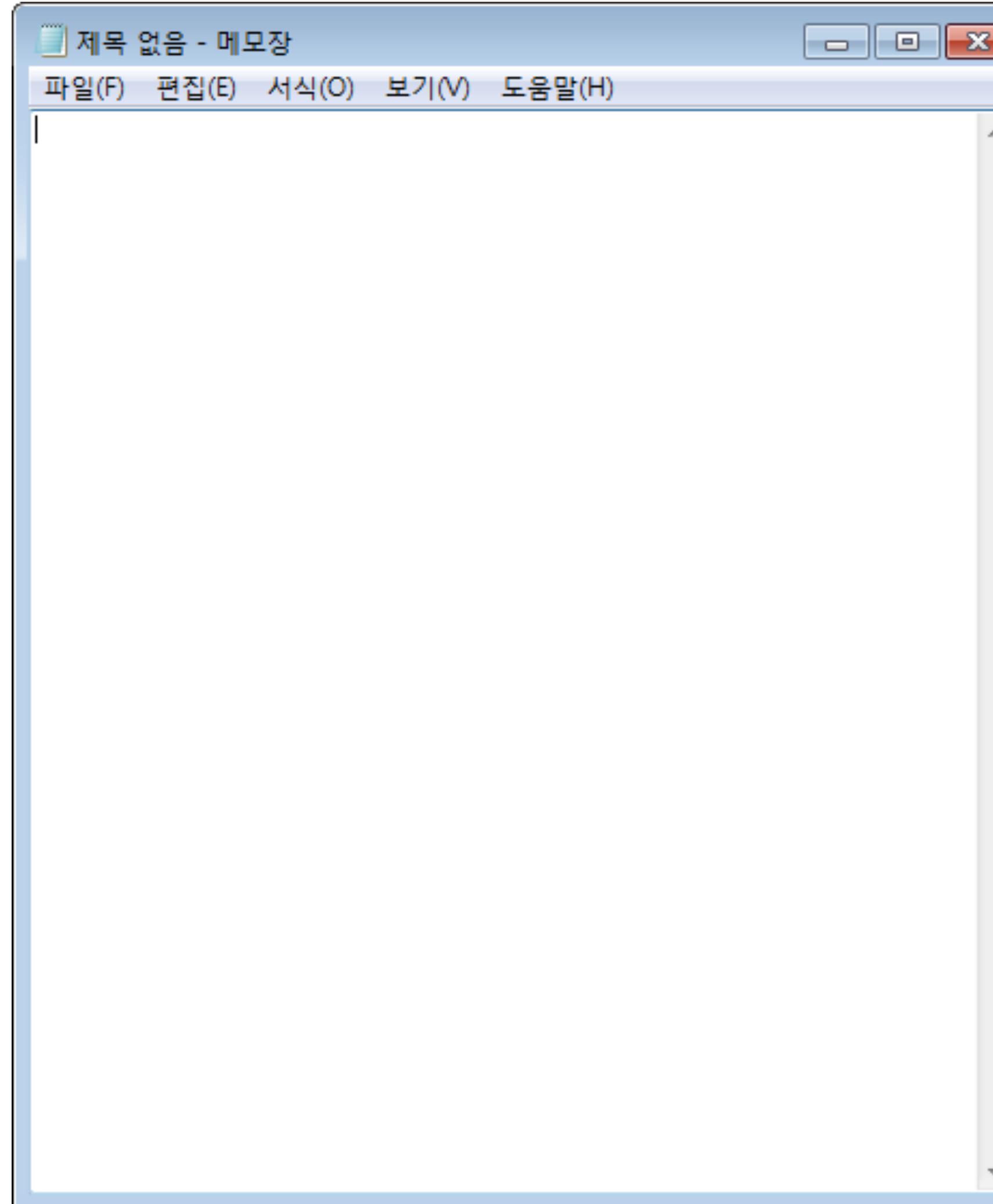


# 프로그래밍을 한다는 것은 어떤 의미일까?

오늘, 짧게나마 프로그래밍을 배우기 전에 프로그래밍을 한다는 것의 의미를 알 필요가 있다.



## 윈도우 기본 프로그램인 메모장(notepad)를 실행해보자.



메모장에서 우리는 하얀 배경의 텍스트 입력창에 다양한 문서를 작성할 수 있다.

그렇지만, 우리는 메모장 문서 안에 표나 그림을 넣거나 동영상을 넣을 수 없다.

마이크로소프트 워드 문서에는 표나 그림을 넣을 수 있고,

마이크로소프트 파워포인트에는 심지어 동영상도 넣을 수 있다.

왜 메모장에는 표, 그림, 동영상을 넣을 수 없을까?

답은 간단하다.

**메모장을 만든 프로그래머가 그것을 허락하지 않았기 때문이다.**



- 프로그래밍을 배우면 우리도 우리가 원하는 프로그램을 작성할 수 있다.
- 프로그래밍이 우리에게 가져다 주는 것은 크게
  - 1) 자유와
  - 2) 학습의 고통이 있다.
- 사람마다 프로그래밍을 배우는 과정에서 느끼는 재미가 크게 다른데,
- 프로그래밍을 배워서 어떤 문제를 해결할 수 있을지를 생각하며 배운다면 훨씬 큰 성취감을 느낄 수 있을 것이다.



# 아는 것의 힘 체험하기

만약 다음과 같이 데이터를 세로로 입력한 엑셀 파일을 갖고 있다고 가정해보자.  
이 데이터를 가로 배열로 옮기려면 어떻게 해야할까?

	A	B	C	D	E	F
1	Python					
2	Java					
3	C					
4	C++					
5	Pascal					
6	Fortran					
7	COBOL					
8	Basic					
9	Lisp					
10	Prolog					
11	SQL					
12	Scheme					
13	Scala					
14	Ada					
15	Bash					



# 직접 데이터 보고 입력하기

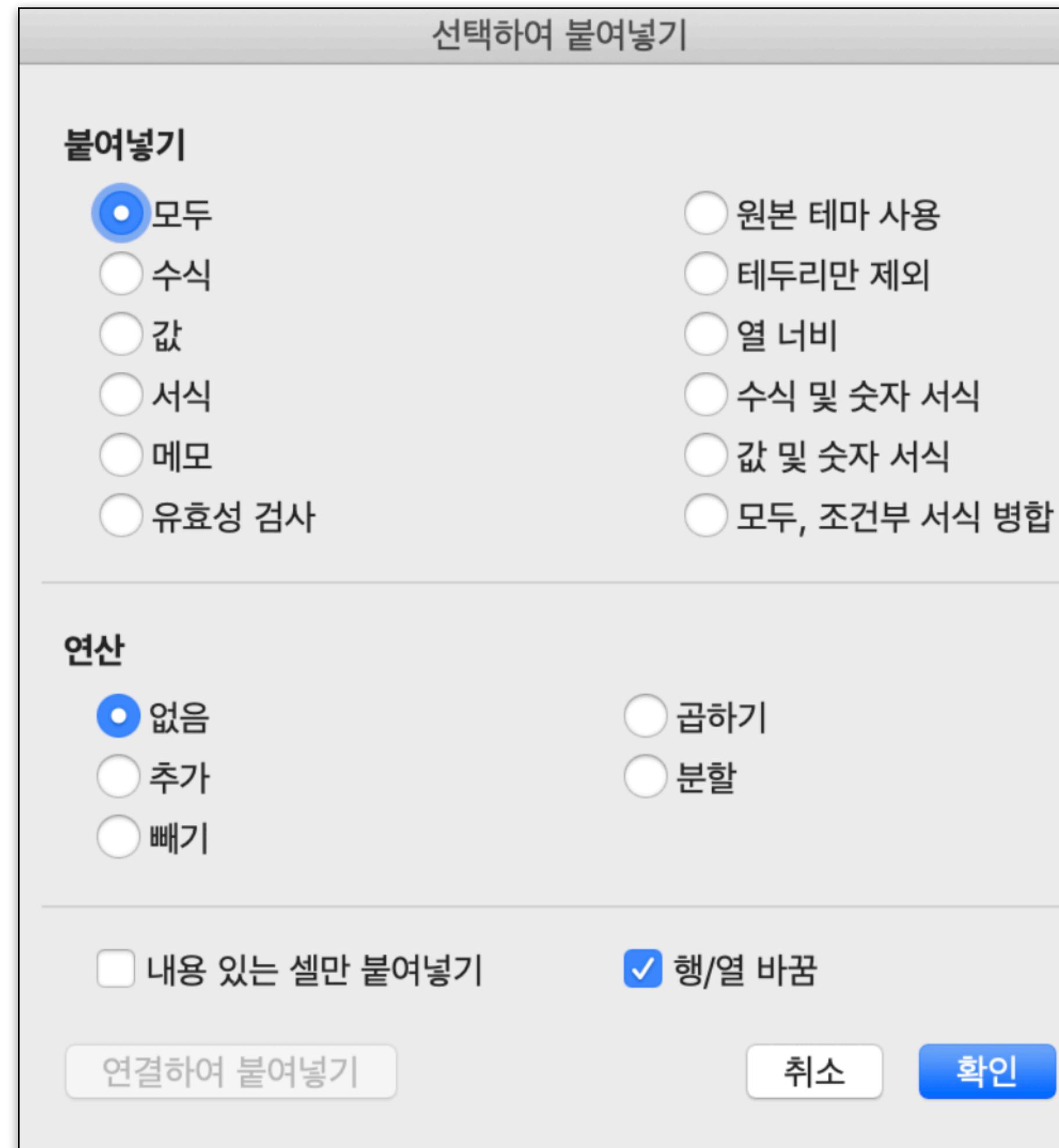
데이터가 10개, 20개라면 편리하지만 만약 데이터가 100,000개라면?

시간이 매우 오래걸리고 중간에 실수를 범할 가능성이 높다.

	A	B	C	D	E
1	Python	Java	C	C++	
2	Java				
3	C				
4	C++				
5	Pascal				



엑셀에는 이런 상황을 대비해 행/열 바꿔 붙여넣기라는 기능이 있다.  
이 기능을 활용하는 것만으로 엄청난 시간과 노력을 절약할 수 있다.





## 하지만...

우리가 겪는 모든 상황에 이렇게 딱 들어맞는 기능이 있기를 기대하는 것은 어렵다.

여러분이 프로그래밍을 할 줄 안다면 딱 맞는 기능을 직접 만들 수 있다.

하루에 32분씩 걸렸던 일을 프로그램을 작성해 2분 만에 끝낼 수 있다면  
1년에 182.5시간을 다른 일에 쓸 수 있게 되는 것이다.

# 왜 파이썬인가?

---



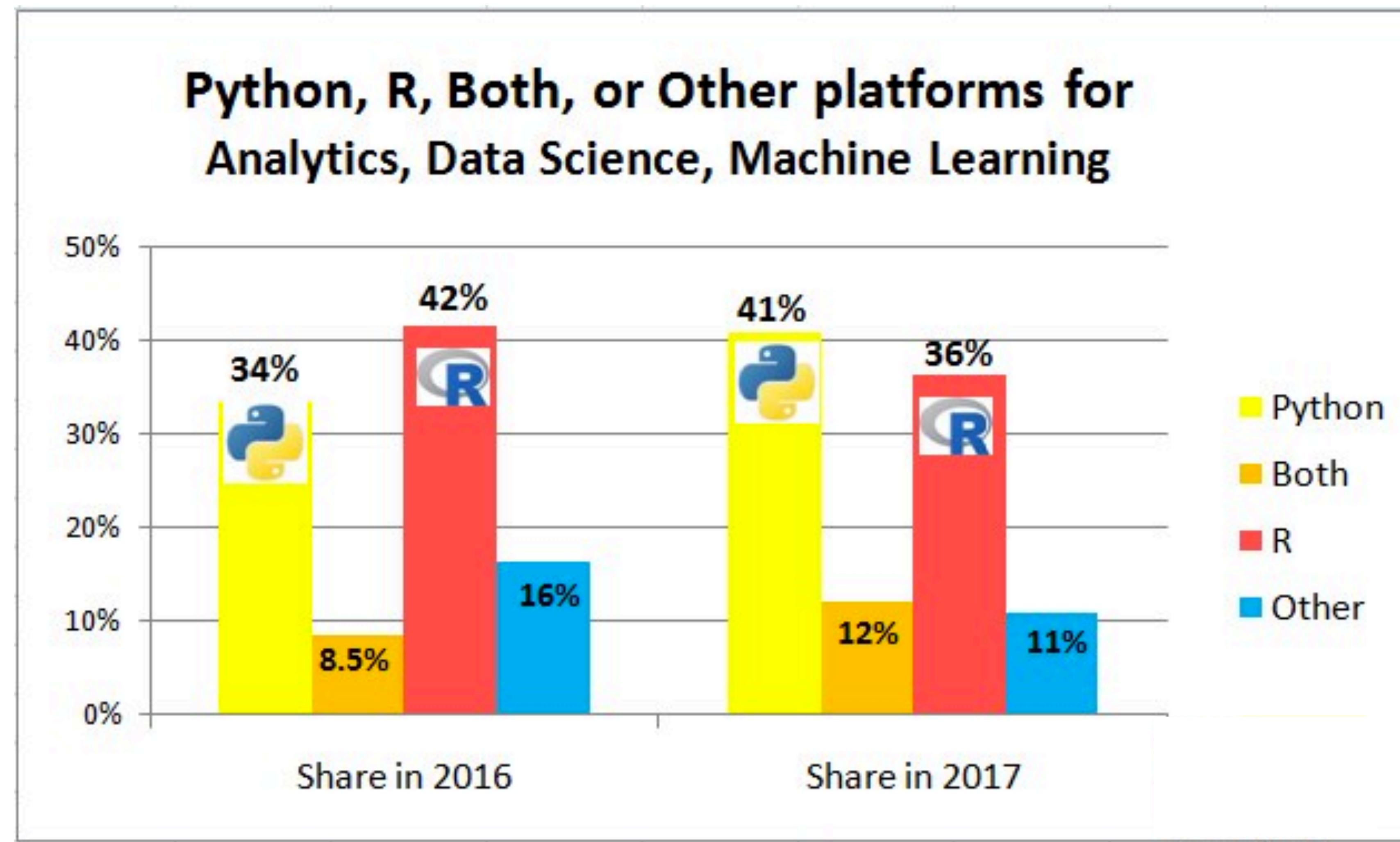
**Why Python?**





# Leader in Data Science and Machine Learning

데이터 분석을 위한 가장 강력한 프로그래밍 언어



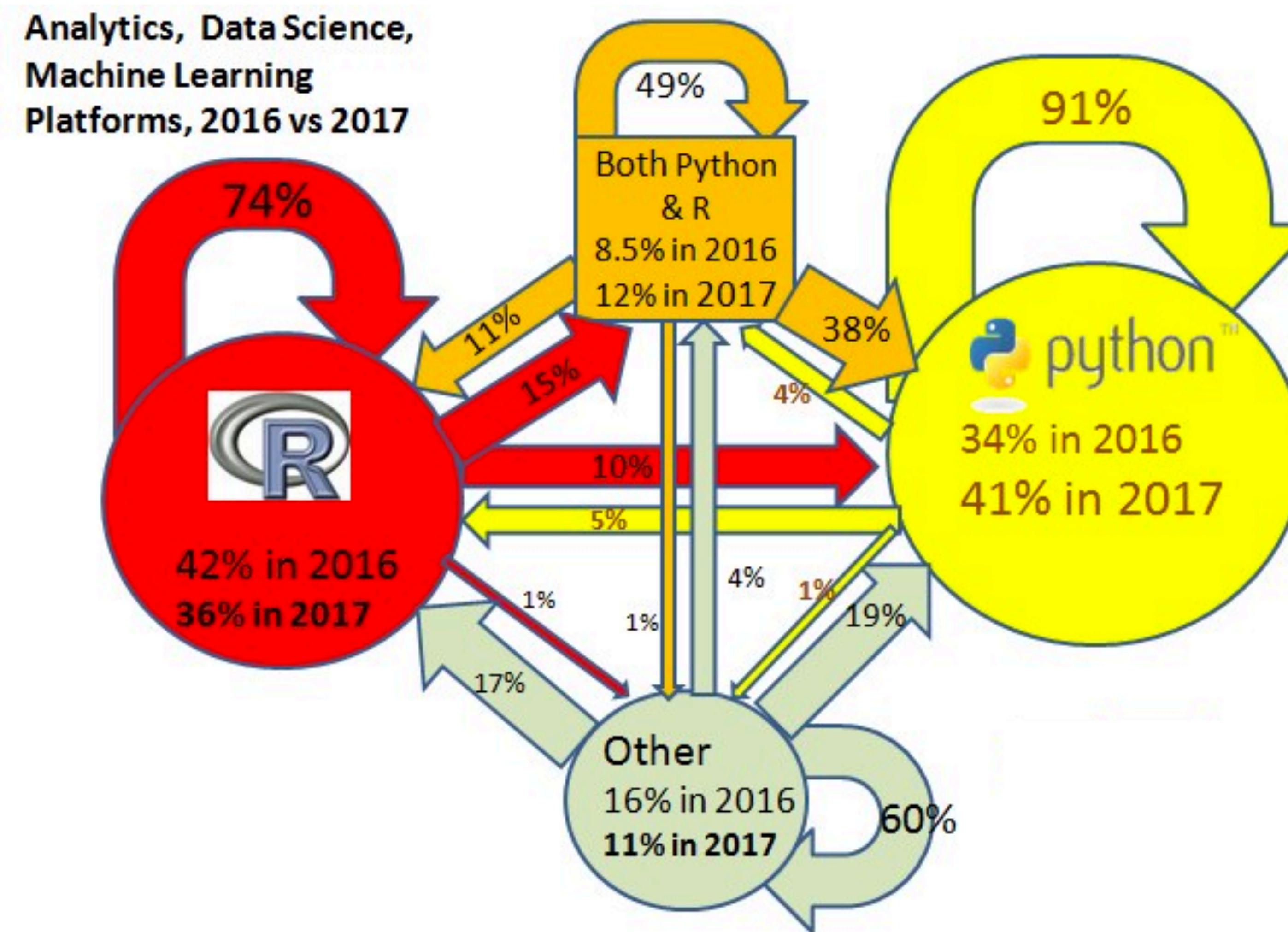
[출처: kdnuggets.com]

[출처: kdnuggets.com]



# Leader in Data Science and Machine Learning

Transitions between the different platforms



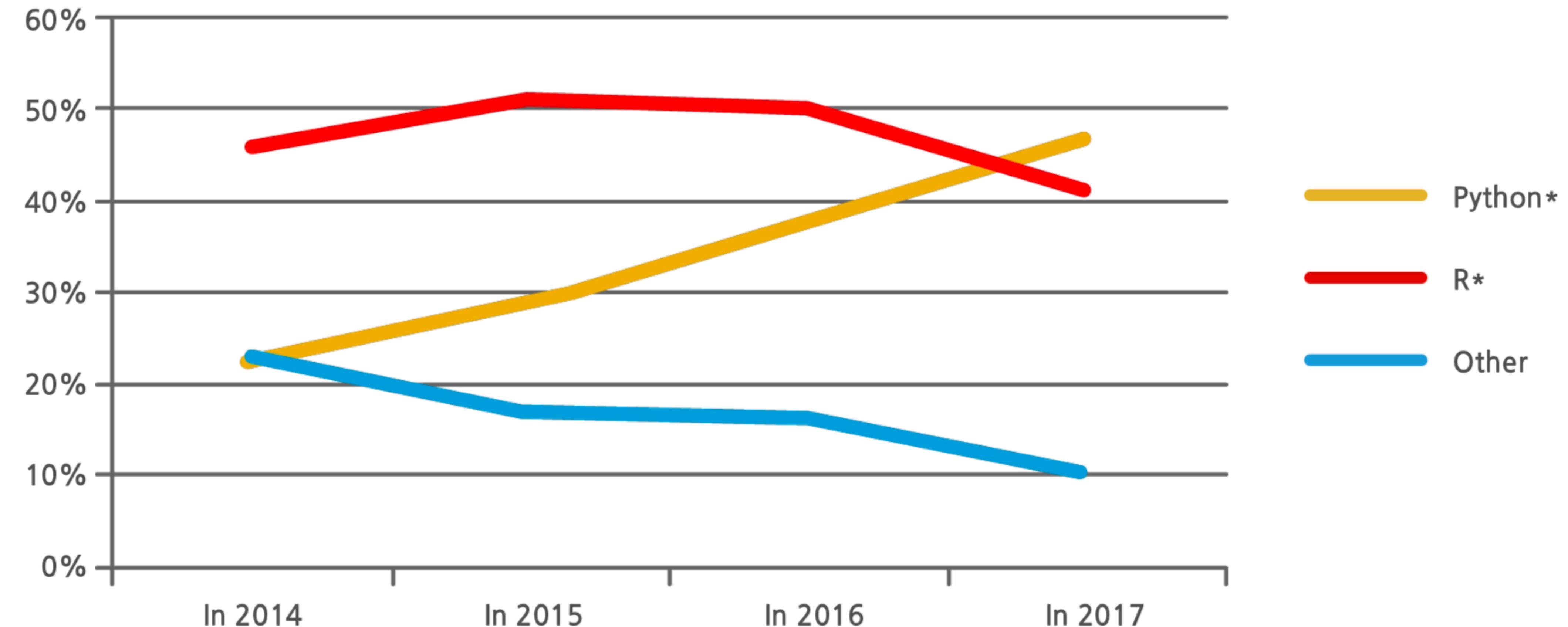
[출처: kdnuggets.com]



# Leader in Data Science and Machine Learning

Trends from 2014 to 2017

## Python vs R vs Other for Analytics & Data Science, 2014-17

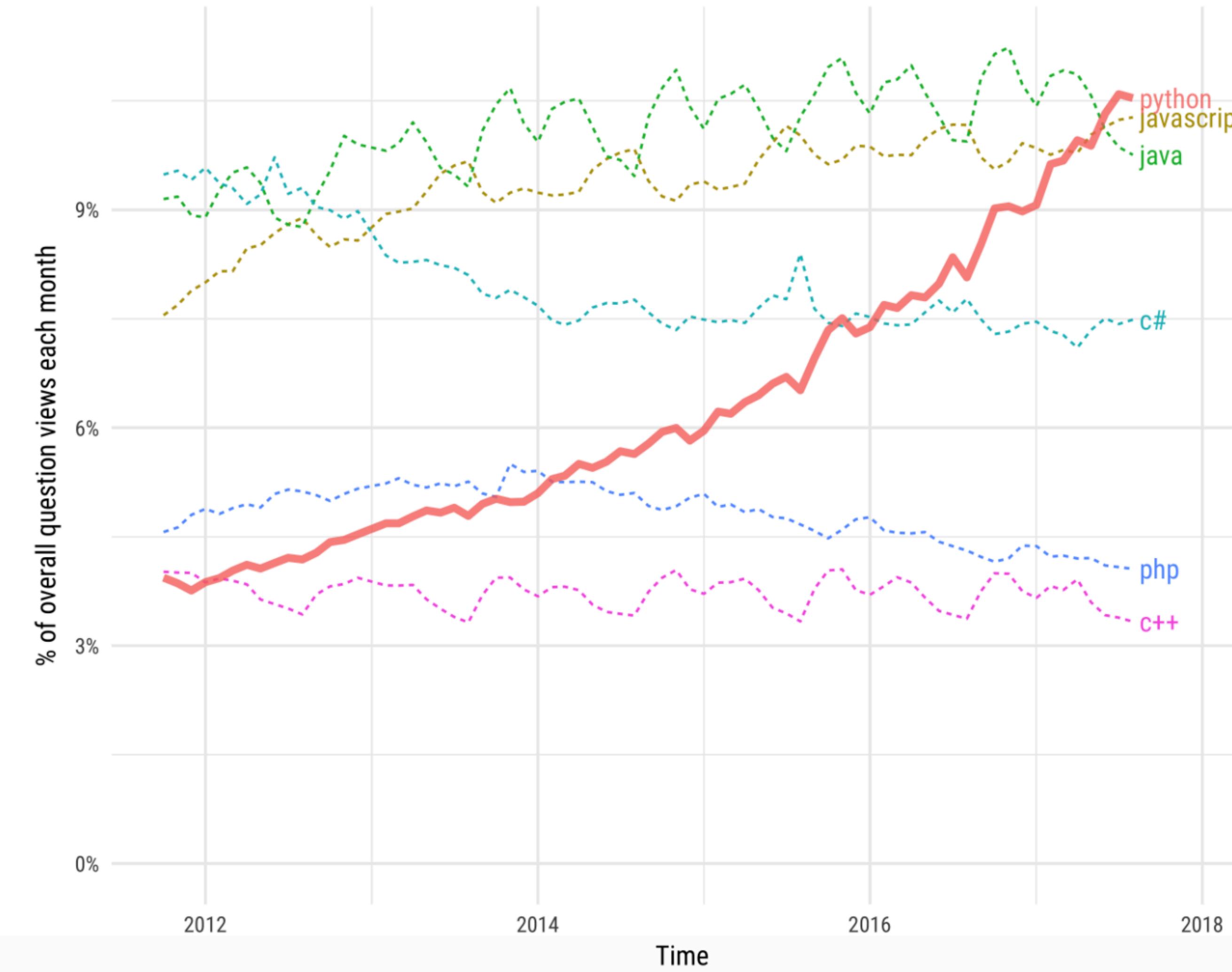


[출처: kdnuggets.com]

[출처: kdnuggets.com]



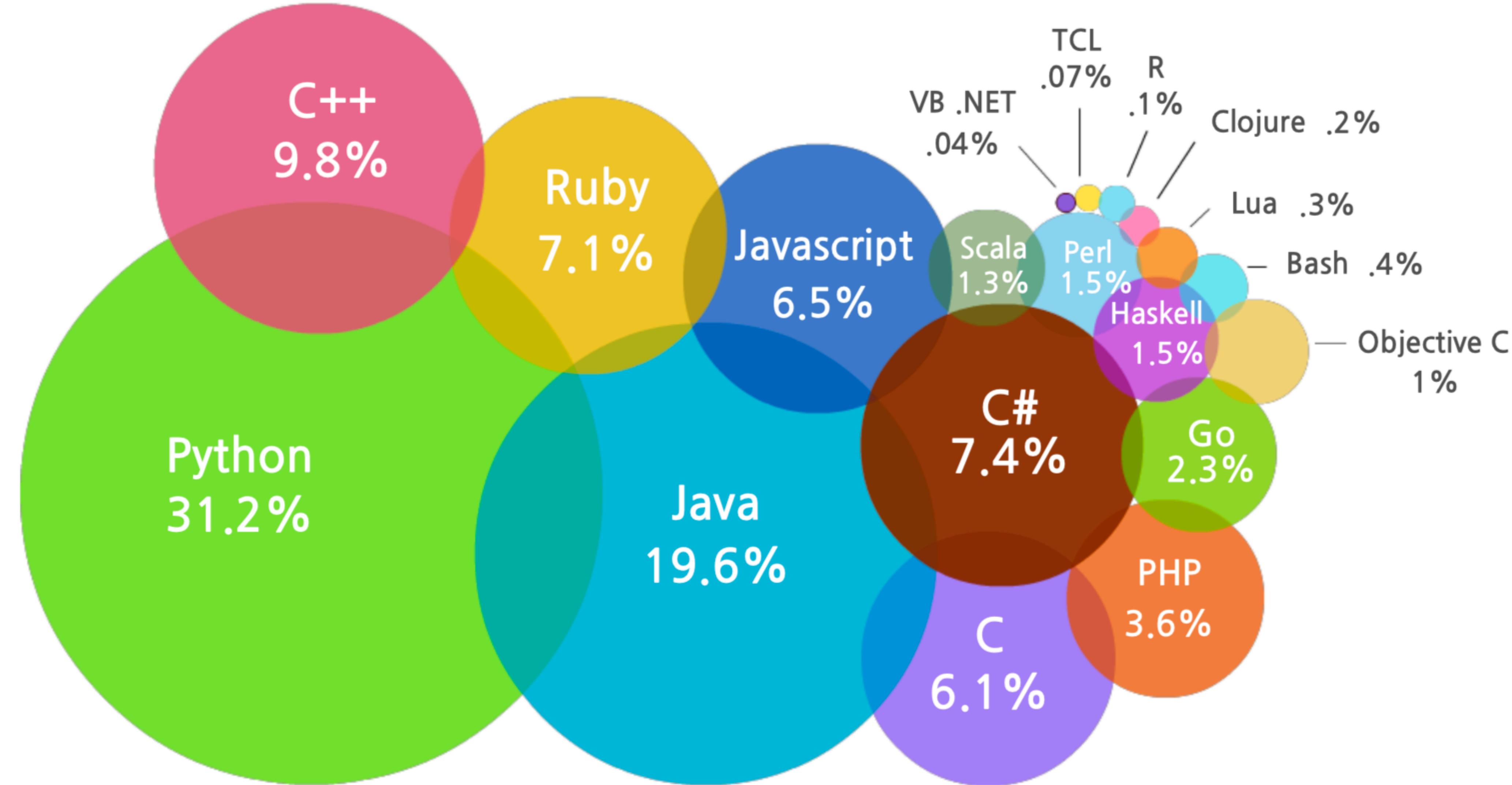
**Growth of major programming languages**  
Based on Stack Overflow question views in World Bank high-income countries



[출처: stackoverflow.blog]



# 가장 인기가 좋은 프로그래밍 언어



[출처: codeeval.com]



# AI 개발에 가장 적합한 5가지 프로그래밍 언어

1

파이썬

2

자바와 그 친구들

3

C/C++

4

자바스크립트

5

R

[출처: <http://www.itworld.co.kr/news/109189>]



## Computer Programming for Lawyers: An Introduction

Spring 2016

Paul Ohm  
Jonathan Frankle

Syllabus  
Version 0.57

### Course Description

This class provides an introduction to computer programming for law students. The programming language taught may vary from year-to-year, but it will likely be a language designed to be both easy to learn and powerful, such as Python



# 프로그래밍 언어로서의 파이썬

## ● 범용

- 자바(Java), C, C++, 베이직(Basic), ...
- **파이썬(Python)**

## ● 웹 / 스크립

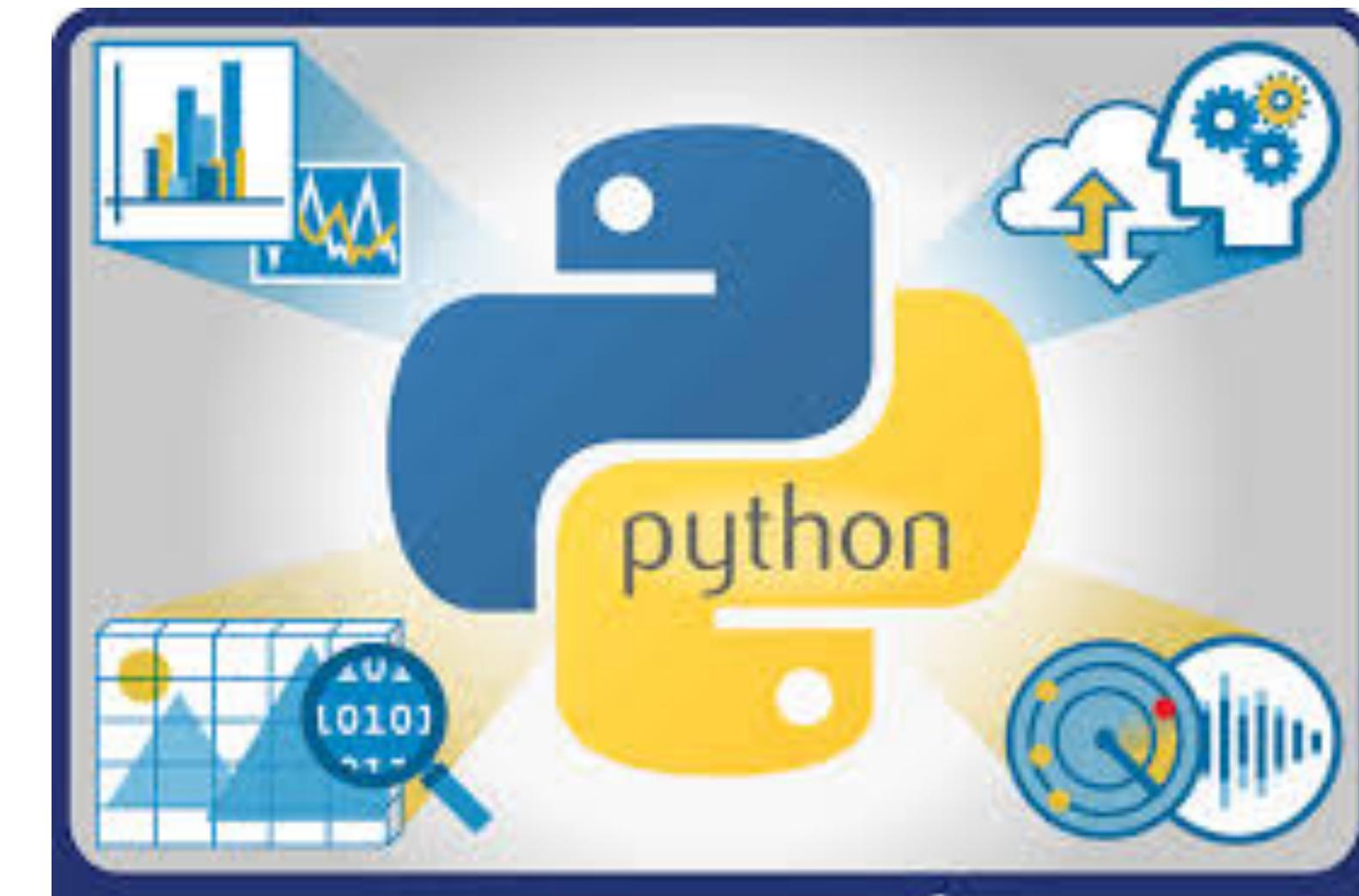
- PHP, 자바스크립트(JavaScript)
- **파이썬(Python)**, 펄(Perl), 루비(Ruby)

## ● 데이터베이스 / 데이터 파일

- SQL, SPARQL
- **파이썬(Python)**, 스칼라(Scala)
- 엑셀(Excel)

## ● 데이터 분석

- SPSS, SAS, R, **파이썬(Python)**
- 엑셀(Excel)



# 파이썬 특징

---

Features of Python





- 휘도 판 로썸(Guido Van Rossum)

- 네덜란드 출신의 개발자



- Easy to learn

- 광범위하게 사용하고 있고 초보자들도 쉽게 배울 수 있는 프로그래밍 언어
  - 다른 프로그래밍 언어보다 문법이 간단하면서도 엄격하지 않음
    - e.g., 변수를 사용하기 전에 별도의 자료형 선언을 하지 않아도 됨
  - 문법 표현이 매우 간결하기 때문에 프로그래밍 경험이 없어도 아주 짧은 기간 내 파이썬 문법을 마스터할 수 있음
    - e.g., 명령문을 구분할 때 중괄호('{' , '}') 대신 <들여쓰기(indentation)>를 사용
      - 사람이 글을 쓸 때 문단을 구분하는 것처럼 들여쓰기를 사용해서 명령문 블록을 구분하기 때문에 글을 써내려가듯이 프로그램 논리를 자연스럽게 표현할 수 있음



# 파이썬이란?

## ● Very expressive language

- 같은 작업을 하는 프로그램을 C나 자바로 작성할 때보다 파이썬으로 작성할 경우 훨씬 짧은 줄로 작성이 가능

```
public class HelloPython {  
    public static void main(String[ ] args) {  
        System.out.println("Hello Python~~~!");  
    }  
}
```

```
print('Hello Python~~~!')
```

## ● 크로스 플랫폼 언어(cross-platform language)

- 위도우즈(Windows), 맥 OS(macOS), 리눅스(Linux), 유닉스(Unix) 등 다양한 운영체제에서 실행 가능



# 파이썬이란?

## ● 해석형 언어(interpreted language)

- 고수준 언어(high-level language)로 작성한 소스코드(원시코드)를 기계어로 변환하는 컴파일 과정없이 바로 실행 가능(no compilation and linking)

## ● 대화형 언어(interactive mode)

- 코드를 대화하듯 한 줄 입력하고 실행한 후 결과를 바로 확인 할 수 있기 때문에 매우 편리
- 언어가 제공하는 다양한 기능들을 실험적으로 사용하고 테스트해 볼 수 있음
- 대화형 모드 예시

```
>>> 2 + 2      ← 사용자가 입력(input)
        4          ← 컴퓨터가 대답(output)
```

## ● 멀티파러다임(multiparadigm) 프로그래밍 언어

- 객체지향형 언어(e.g., Java, Smalltalk, Ruby, Scala)
- 절차형 언어(e.g., C, Fortran, Pascal, COBOL, MATLAB)
- 함수형 언어 (e.g., Scheme, LISP)



# 파이썬이란?

## ● 다양한 고급 프로그래밍 기능 제공

- 숫자(정수, 실수, 복소수 등) 또는 문자열(ASCII, Unicode) 등 다양한 내장(built-in) 기본 자료형과 리스트, 튜플, 사전과 같은 내장 복합 자료형을 지원
- 클래스, 다중 상속(multiple inheritance), 모듈, 패키지, 예외처리(exception handling), 자동 메모리 관리 등을 지원
- 생성자(generator), 리스트/세트/딕셔너리 축약(list/set/dictionary comprehension) 등 강력한 고급 프로그래밍 기능을 지원

## ● 강력하고 풍부한 라이브러리(library) 제공 : 개발 속도가 빠름

- 프로그래밍에 자주 사용되는 다양한 기능이 장착된 표준 라이브러리(standard library)를 기본적으로 제공
  - 라이브러리(library)
    - 다양한 기능을 제공하는 함수나 모듈을 프로그램에서 쉽게 불러 사용할 수 있게 미리 작성해서 컴파일해 놓은 코드(모듈)들의 모음
    - e.g., 웹 서버 연결, 정규표현식(regular expression)을 통한 문자 검색, 파일 읽고 쓰기 등
  - 기본으로 제공되는 라이브러리 뿐만 아니라, 개인이나 단체에서 만들어서 배포하는 수천개가 넘는 다양하면서도 기능이 풍부한 라이브러리가 존재하면 필요시 언제든 사용 가능
  - [pypi.python.org/pypi](https://pypi.python.org/pypi)



# 파이썬이란?

## ● 엄청난 사용자 커뮤니티

- 대규모의 사용자 커뮤니티가 존재
- 도움을 손쉽게 받을 수 있을뿐 아니라 참고자료도 풍부
  - e.g., 스택오버플로우(<https://stackoverflow.com/>)에서 파이썬 관련 많은 도움을 받을 수 있음

## ● 뛰어난 확장성

- C, C++에서 작성된 모듈을 불러와서 사용하는 것이 가능
  - 모듈 : 미리 작성되어 편리하게 가져다 쓸 수 있는 코드를 뜻하며 여러 개의 모듈이 모이면 라이브러리가 됨
- 프로그래밍 가능한 인터페이스를 제공하기 때문에 다른 프로그래밍 언어에서 불러 사용하는 것이 가능

## ● 무료(free) 소프트웨어

- 무료로 다운로드 및 사용이 가능
- 개인이 제작한 애플리케이션에 탑재시킬 때도 무료
- 파이썬은 오픈소스 소프트웨어이기 때문에 자유(free)롭게 수정 및 재배포가 가능





하지만...



# 파이썬은 프로그래밍 언어라고 하기 보다는

다양한 모형을 조립할 수 있는 블록완구에 더 가깝다고 보는게





파이썬에 대한 올바르고 공정한 평가라 할 수 있을 것 같다



# 파이썬은 간결하며



다른 프로그래밍 언어보다 훨씬 빠르게 배울 수가 있다



따라서...



다른 프로그래밍 언어처럼 많은 노력을 쏟아 붓지 않아도



# 논리적인 사고로 표현만 할 수 있다면



# 마치 블록완구로 원하는 모형을 만들 수 있듯이



# 파이썬이 제공하는 다양한 기능들을 잘 조립만 한다면



# 나머지는 파이썬이 알아서 처리해준다

```
stream.filter(track=['Keanu', 'Captain America'])
```



## PEP 20 - The Zen of Python

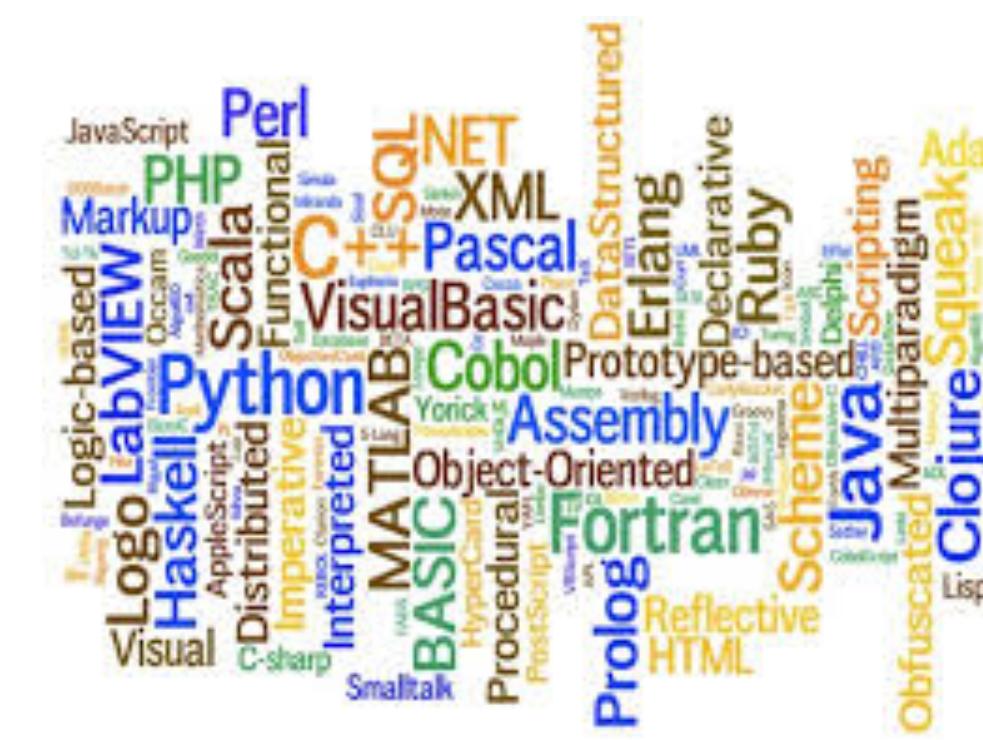
<https://www.python.org/dev/peps/pep-0020/>

- "아름다운게 추한 것보다 좋다." (**Beautiful is better than ugly.**)
- "명시적인 것이 암시적인 것보다 좋다." (**Explicit is better than implicit.**)
- "단순한 것이 복잡한 것보다 좋다." (**Simple is better than complex.**)
- "복잡한 것이 난해한 것보다 좋다." (**Complex is better than complicated.**)
- "평평한 것이 중첩한 것보다 다 좋다." (**Flat is better than nested.**)
- "듬성듬성 여유로운 것이 촘촘하게 밀집한 것보다 좋다." (**Sparse is better than dense.**)
- "가독성은 중요하다." (**Readability counts.**)
- ...

# 프로그래밍과 프로그램 개발 절차

---

## Program & Program Development Process





## ● 프로그램(program)이란?

- 특정한 작업을 어떻게 수행해야 하는지 그 순서를 일련의 명령어로 나열한 것

## ● 프로그램 언어의 공통 기본기능

- 파이썬(Python), 자바(Java), C 언어를 포함한 모든 프로그래밍 언어는 공통적으로 다음과 같은 기본 기능들을 제공

### ● **입력**

- 키보드나 파일 또는 별도의 장치로부터 데이터를 입력

### ● **출력**

- 데이터를 컴퓨터 화면에 보여주거나 파일 또는 별도의 장치로 전송

### ● **처리**

- **연산 처리** : 더하기, 빼기, 곱하기, 나누기 같은 기본적인 수학적 연산을 수행

- **순차 처리** : 순서대로 작업을 수행

- **선택(조건) 처리** : 특정한 조건에 따라 그에 맞는 적절한 코드를 선택해서 실행

- **반복 처리** : 주어진 횟수나 조건에 따라 특정한 작업을 (주로 약간의 변화를 주면서) 반복적으로 수행



## 예시 : 1학년 학생들의 평균 점수를 계산하여 화면에 출력하는 프로그램

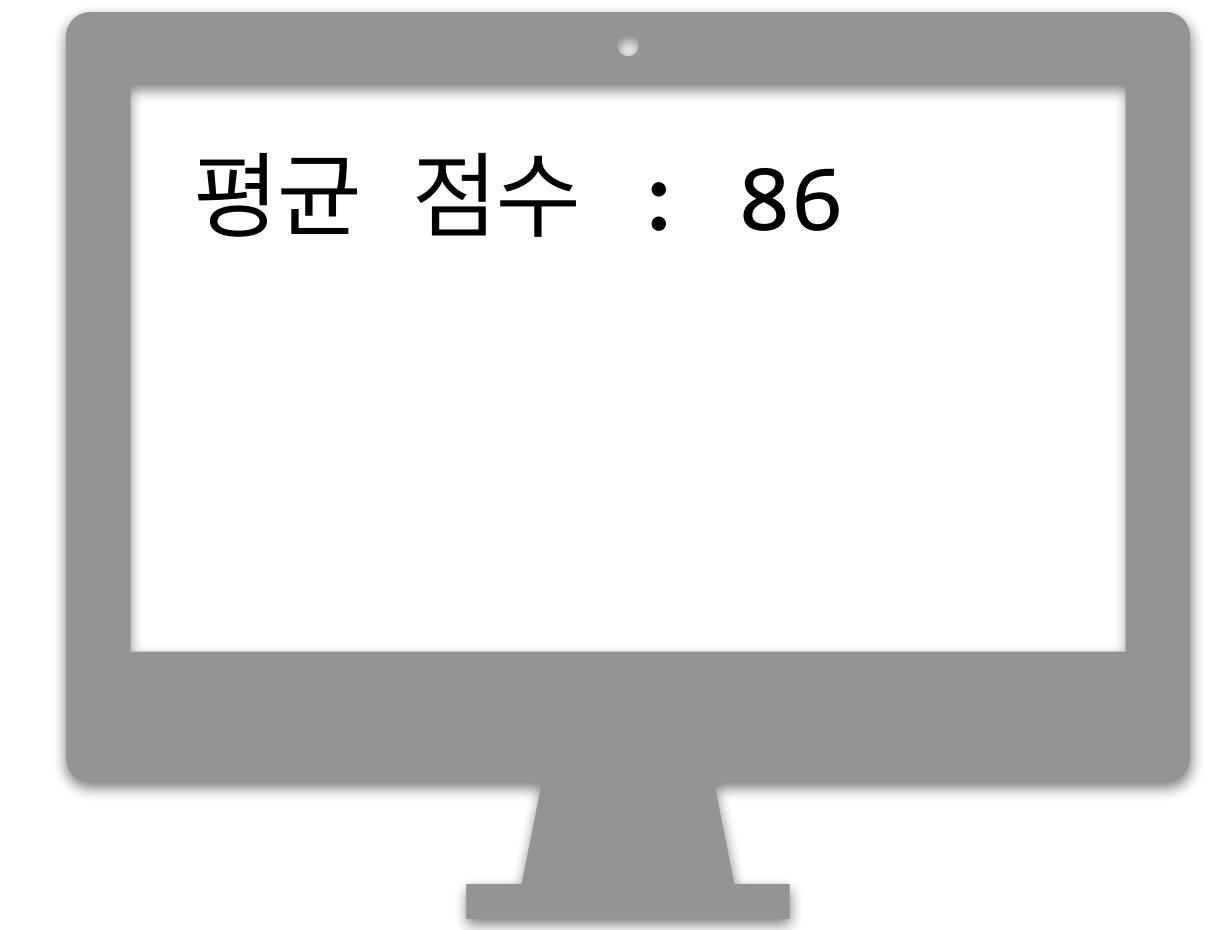
이름	학년	점수
Hong	2	83
Kim	1	90
Park	2	95
Lee	1	80
Hahn	1	88

grade.csv

목표: grade.csv의 표에서 1학년 학생들의 평균 점수를 계산한다.

- 1) 'grade.csv' 파일을 읽는다.
- 2) 표의 데이터를 한 줄씩 읽으면서,
- 3) 학년이 1학년이라면,
- 4) 해당 점수를 총 점수 값에 더하고, 학생 수를 1 증가시킨다.
- 5) 평균 점수를 계산(총 점수 ÷ 학생 수)하여 화면에 출력한다.

평균 점수 : 86





# 예시 : 1학년 학생들의 평균 점수를 계산하여 화면에 출력하는 프로그램

이름	학년	점수
Hong	2	83
Kim	1	90
Park	2	95
Lee	1	80
Hahn	1	88

grade.csv

순차 처리

```
total = 0
freshmen = 0

data = open('grade.csv')      입력

for row in data: 반복 처리
    if row['학년'] == 1: 선택 처리
        total += row['점수'] 연산 처리
        freshmen += 1      연산 처리

    else:
        avg = total / freshmen 연산 처리

print('평균 점수 : ', avg)    출력
```

평균 점수 : 86



- 프로그램 개발 과정

- 프로그램 개발은 다음과 같은 과정을 거침

1	프로그램 논리 설계 및 개발
---	-----------------

2	프로그램 코드화(coding)
---	------------------

3	기계어(machine language)로 변환(컴파일)
---	--------------------------------

4	프로그램 실행과 검증
---	-------------



# Step 1 : 프로그램 논리 설계 및 개발

## 핵심 단계

- 프로그래밍 개발 단계 중 가장 중요한 핵심 단계
- 알고리즘(algorithm) 개발이라고도 함
- 어떤 작업 절차를 어떤 순서로 실행할지 결정

## 논리 설계에 사용되는 툴(tool)

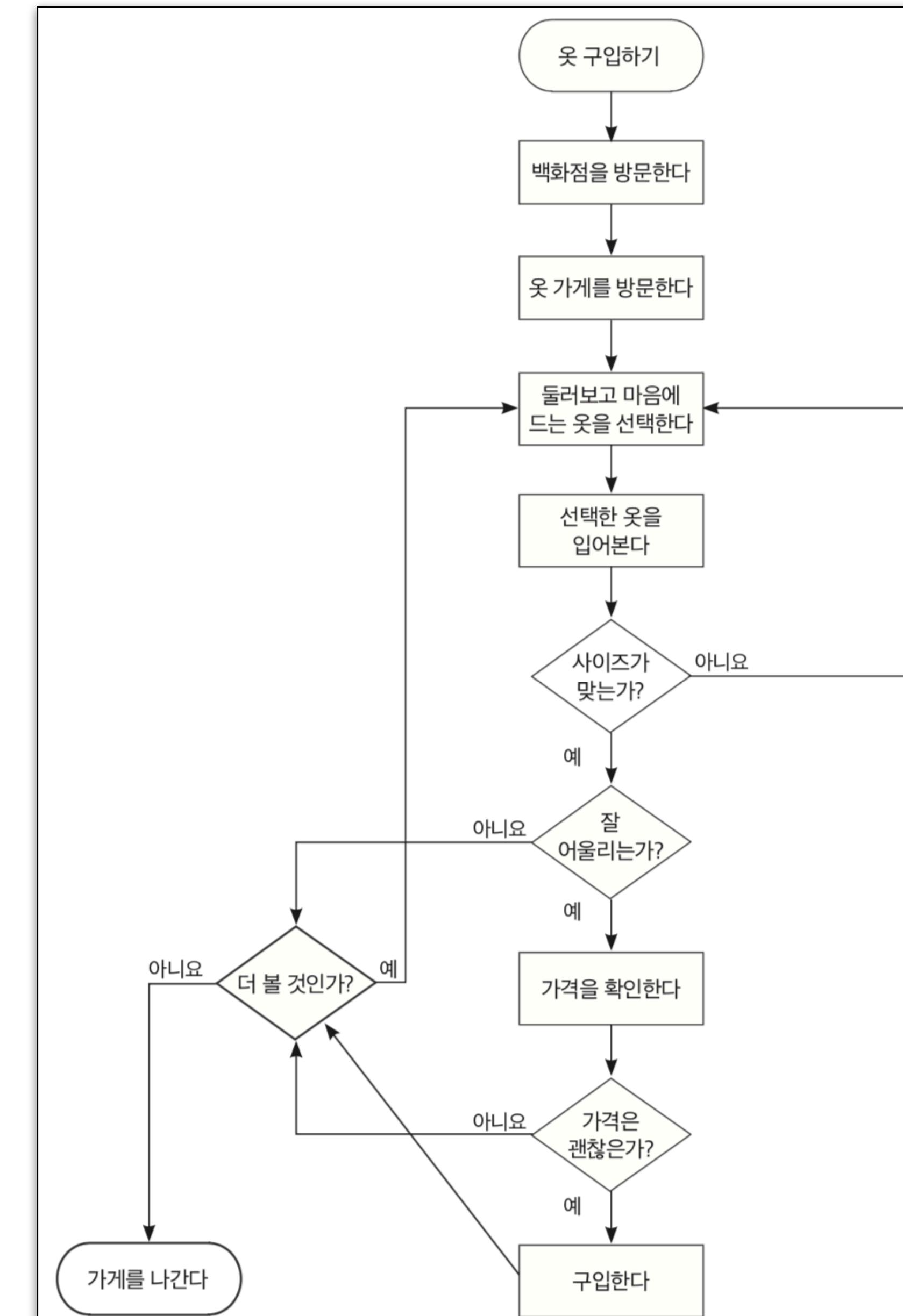
- 흐름도(flowcharts) / 순서도
- 의사(擬似)코드(슈도코드, pseudocode)
- 일상어(자연어)

## 탁상 검사(desk-checking)

- 눈으로 프로그램의 논리를 검사하는 것
- 언어적 문법은 이 단계에서 고려하지 않음

## 논리적 오류(logic(al) errors)

- 프로그램이 정상적으로 실행되는 것 같지만 잘못된 결과가 나오는 경우





# 프로그램 논리 설계 : 컴퓨터처럼 생각하는 방법

## ● 흐름도(flowchart, 순서도)

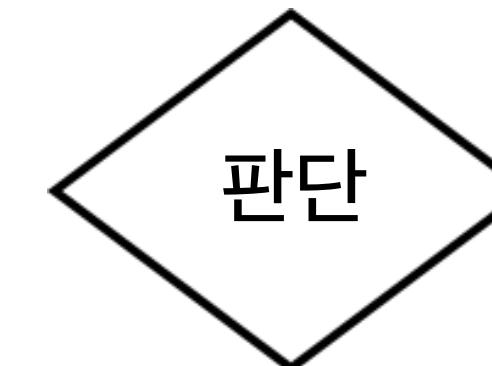
- 프로그래밍이 처리되는 과정을 시각적으로 나타낸 도표

## ● 단말기호(terminal symbol)

- 프로그램의 시작과 끝을 나타냄



## ● 비교/판단기호(decision symbol)



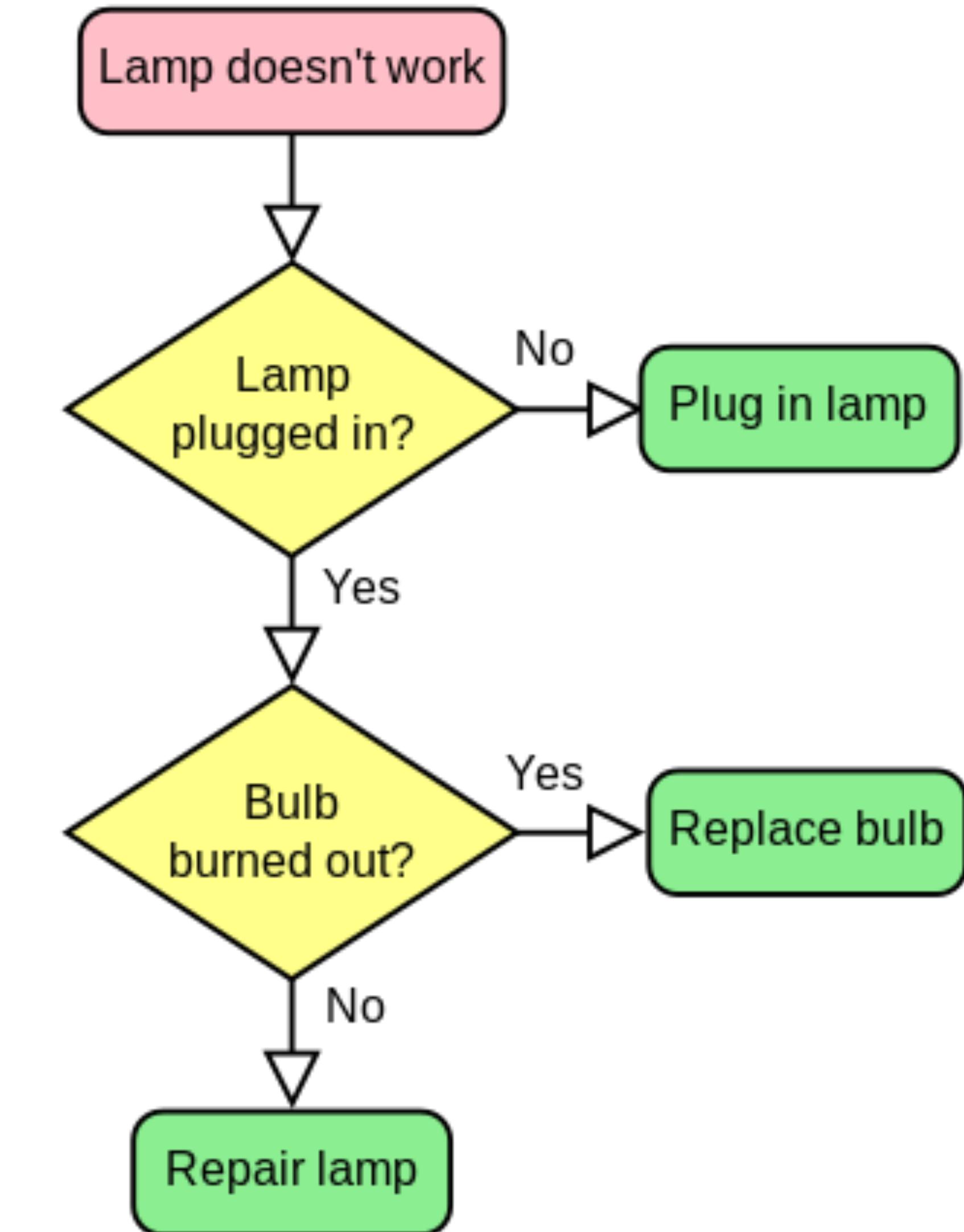
## ● 입출력기호(input and output symbol)



## ● 처리기호(processing symbol)



- 각 기호들은 화살표로 연결되어 프로그램의 처리 과정을 나타냄



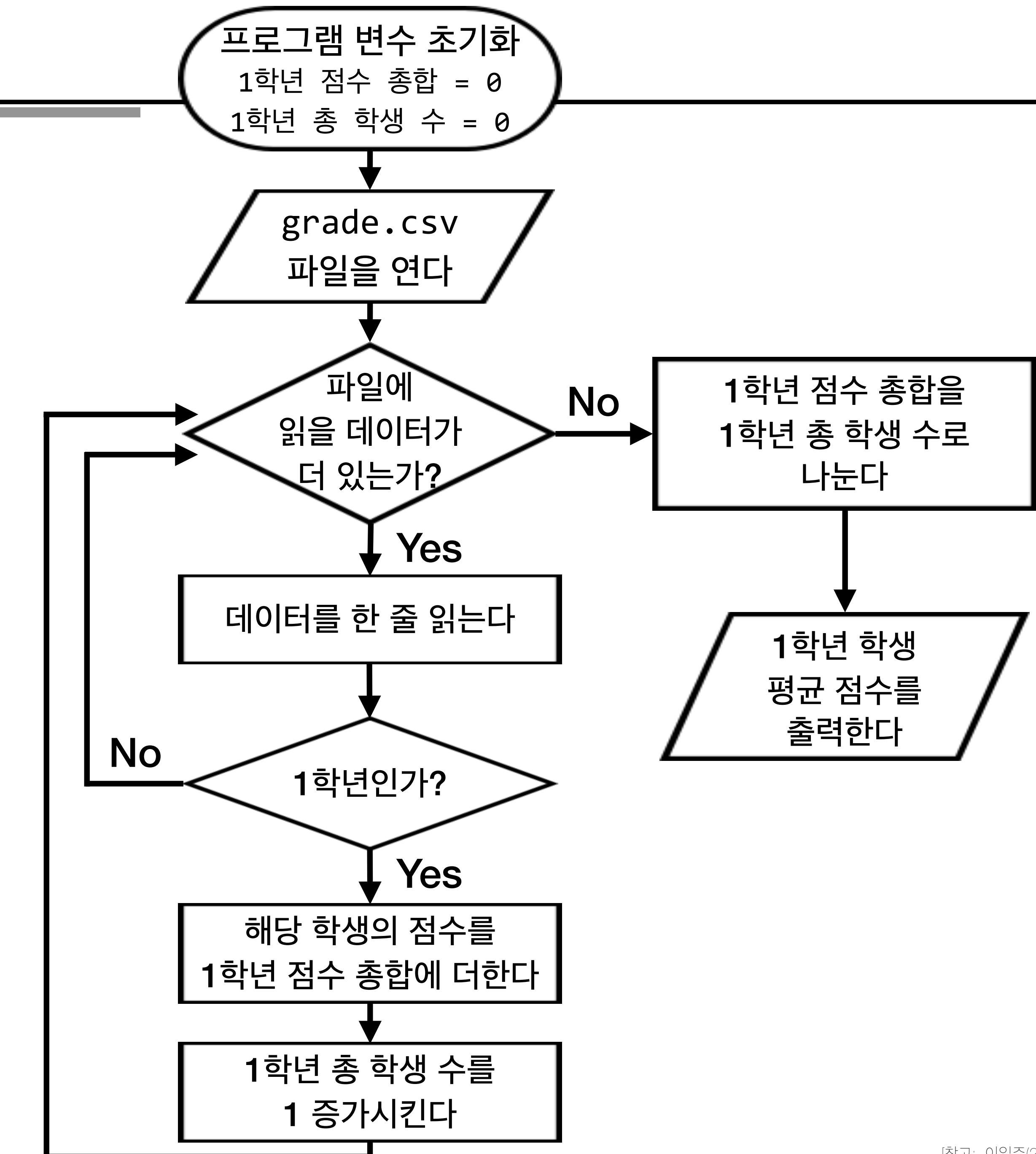
[출처: <https://en.wikipedia.org/wiki/Flowchart>]



# 예시 : 흐름도

목표: grade.csv의 표에서 1학년 학생들의 평균 점수를 계산한다.

- 1) 'grade.csv' 파일을 읽는다.
- 2) 표의 데이터를 한 줄씩 읽으면서,
- 3) 학년이 1학년이라면,
- 4) 해당 점수를 총 점수 값에 더하고, 학생 수를 1 증가시킨다.
- 5) 평균 점수를 계산(총 점수 ÷ 학생 수)하여 화면에 출력한다.



[참고: 이일주(2021)]



# 프로그램 논리 설계 : 컴퓨터처럼 생각하는 방법

## ● 특정한 작업을 수행하기 위한 필요한 과정을 순서대로 나열해 봄

- 반드시 거쳐야 하는 과정들을 논리적 순서대로 정리해서 **알고리즘**을 생성함

☞ **알고리즘(algorithm)** : 특정한 작업을 수행하기 위해 필요한 과정들을 논리적으로 나열한 것

## ● 의사(擬似)코드(슈도코드, pseudocode)

- 실제 프로그래밍 언어가 아닌 유사한 형태의 언어나 일상어 형태로 작성한 코드

- 프로그램 설계 시에만 사용하므로 문법(구문) 오류를 신경 쓸 필요가 없음

☞ 컴파일 또는 실행에 사용하는 코드가 아님(가짜 코드)

- 의사코드는 어떤 프로그램 언어로든지 실제 코드로 옮겨 적는 것이 가능

- e.g., 녹차 만들기 (일상어 형태)

☞ 주전자에 물을 넣는다;  
☞ 주전자의 물을 끓인다;  
☞ 주전자의 물이 끓으면 불을 끈다;  
☞ 컵에 티백을 넣는다;  
☞ 컵에 물을 붓는다;  
☞ 약간 기다린 뒤 티백을 꺼낸다;  
☞ 마신다;

### 1학년 학생들의 평균 성적 구하기 의사코드

- 'grade.csv' 파일을 읽는다.
- 표의 데이터를 한 줄씩 읽으면서,
- 학년이 1학년이라면,
- 해당 점수를 총 점수 값에 더하고, 학생 수를 1 증가시킨다.
- 평균 점수를 계산(총 점수 ÷ 학생 수)하여 화면에 출력한다.



## ● 제어문 의사(擬似)코드(슈도코드, pseudocode) 작성법

- 정해진 규칙은 없지만
- 최소한 프로그래밍 언어의 일반적으로 사용하는 키워드를 사용하여 표현하고
- 블록은 들여쓰기를 할 것을 권장

```
if (condition)
    statements...
else-if (condition)
    statements...
else
    statements...
end if
```

```
while (condition)
    statements...
break if (condition)
    statements...
end while
```

```
for (condition)
    statements...
break if (condition)
    statements...
end for
```

```
function name(input1, input2...)
    statements...
return (output1, output2...)
```

```
class name(input1, input2...)
    statements...
return (output1, output2...)
```

```
output1, output2 = call name(input1, input2...)
```



## Step 2 : 프로그램 코드화

### ● 프로그래밍(programming)

- 파이썬(Python), 자바(Java), R, 루비(Ruby), C, C++, 비주얼 베이직(Visual Basic), SQL 등의 해당 프로그래밍 언어 문법에 맞추어 코드를 작성
- 인간 언어처럼 프로그래밍 언어도 각기 다른 표현과 문법을 가지고 있음
- 일반적으로 프로그램 논리개발보다 코드화 작업이 더 쉬움

```
inputs = Input(shape=(28, 28))
flatten = Flatten()(inputs)
hidden = Dense(256, activation=relu)(flatten)
hidden = Dense(64, activation=relu)(hidden)
hidden = Dense(32, activation=relu)(hidden)
outputs = Dense(10, activation=softmax)(hidden)
model = Model(inputs, outputs)

model.compile(optimizer=Adam(), loss=CategoricalCrossentropy(), metrics=CategoricalAccuracy())
model.fit(X_train, y_train, validation_split=0.2, epochs=100)
```



# 코드화의 기본 작업

- **선언(declaration)** : 프로그램에서 사용될 자료형(data type)를 컴퓨터에게 알려주는 단계

- 데이터는 사용하기 전에 반드시 미리 선언 (일반적으로 코드화 작업 앞부분에서 선언)
  - 자료형을 선언함으로써 메모리에 차지할 저장 공간과 쓰임새를 컴퓨터에게 미리 알려줌

- **입력(input)** : 컴퓨터에 데이터를 입력하는 것

- 사람으로부터 : 키보드, 마우스, 펜 등
  - 파일로부터 : 텍스트 파일, 데이터베이스, 웹 페이지 등
  - 센서로부터 : 빛, 동작 감지, 생체인식 등

- **처리(processing)** : 프로그램에 의해 수행되는 작업

- e.g., 급여 계산, 체스 게임에서 말의 이동

- **출력(output)** : 결과 또는 답

- 화면에 표시
  - 파일로 저장
  - 종이에 출력

```
total = 0  
freshmen = 0  
  
data = open('grade.csv')  
  
for row in data:  
    if row['학년'] == 1:  
        total += row['점수']  
        freshmen += 1  
    else:  
        avg = total / freshmen  
  
print('평균 점수 : ', avg)
```

선언

입력

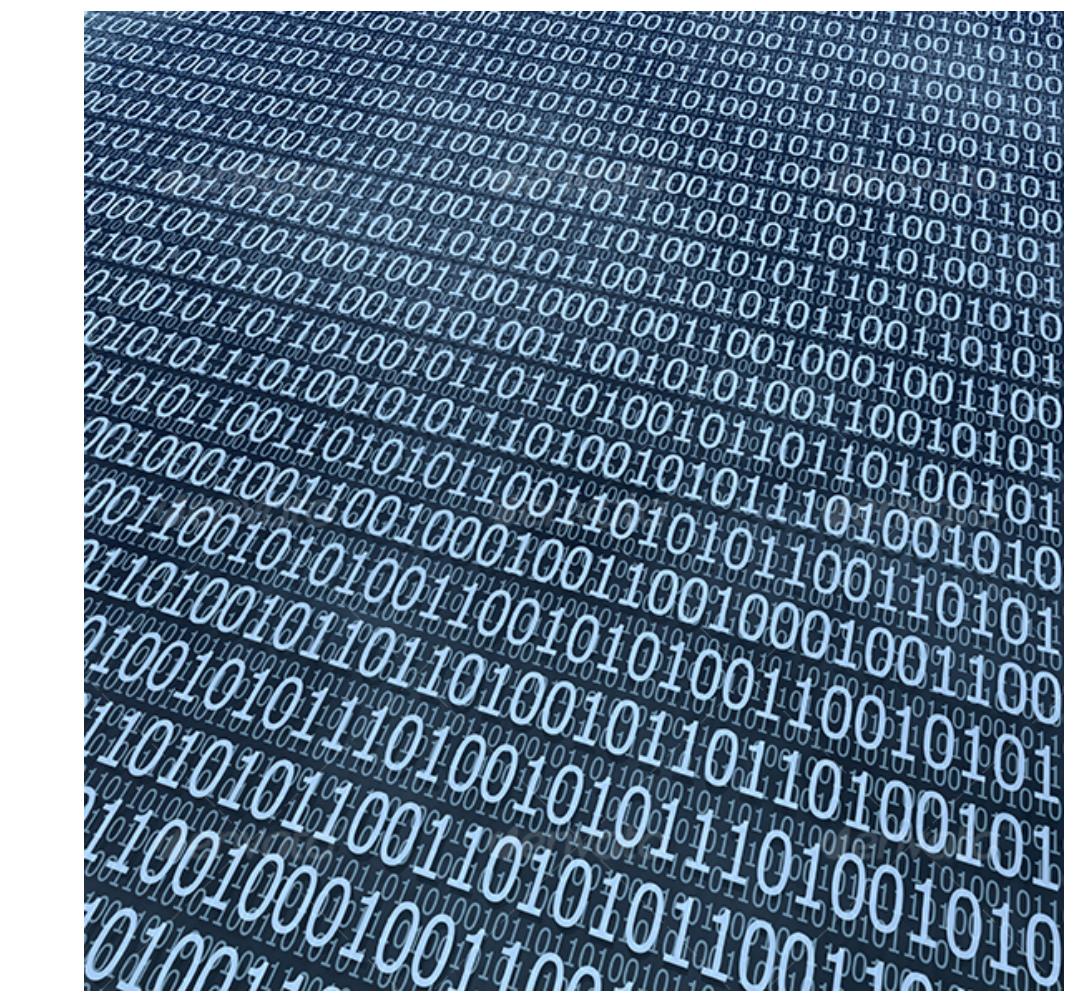
처리

출력



# Step 3 : 기계어(machine language)로 변환(컴파일)

- 컴퓨터는 1 또는 0 밖에 모른다
  - 소프트웨어를 통해 프로그램을 컴퓨터가 이해할 수 있는 기계어(0과 1)로 변환
  - 다양한 프로그래밍 언어가 있지만 컴퓨터는 오로지 1 또는 0만 이해할 수 있기 때문에 어떤 언어든지 기계어로 변환이 필요
- 컴파일러(compiler)와 인터프리터(interpreter)
  - 일상어와 유사한 형태를 지닌 고수준 프로그래밍 언어를 저수준 기계어로 변환해주는 방식
- 문법 오류(syntax error)
  - 컴파일러가 코드를 변환할 수 없을 때 문법 오류가 발생
  - 원인
    - 오타
    - 잘못된 문법
  - 코드에서 잘못된 부분을 수정한 뒤 다시 변환(recompile)해야 함



A large grid of binary code (0s and 1s) representing machine language. The grid is composed of many rows and columns of binary digits, forming a pattern that represents instructions for a computer's processor.

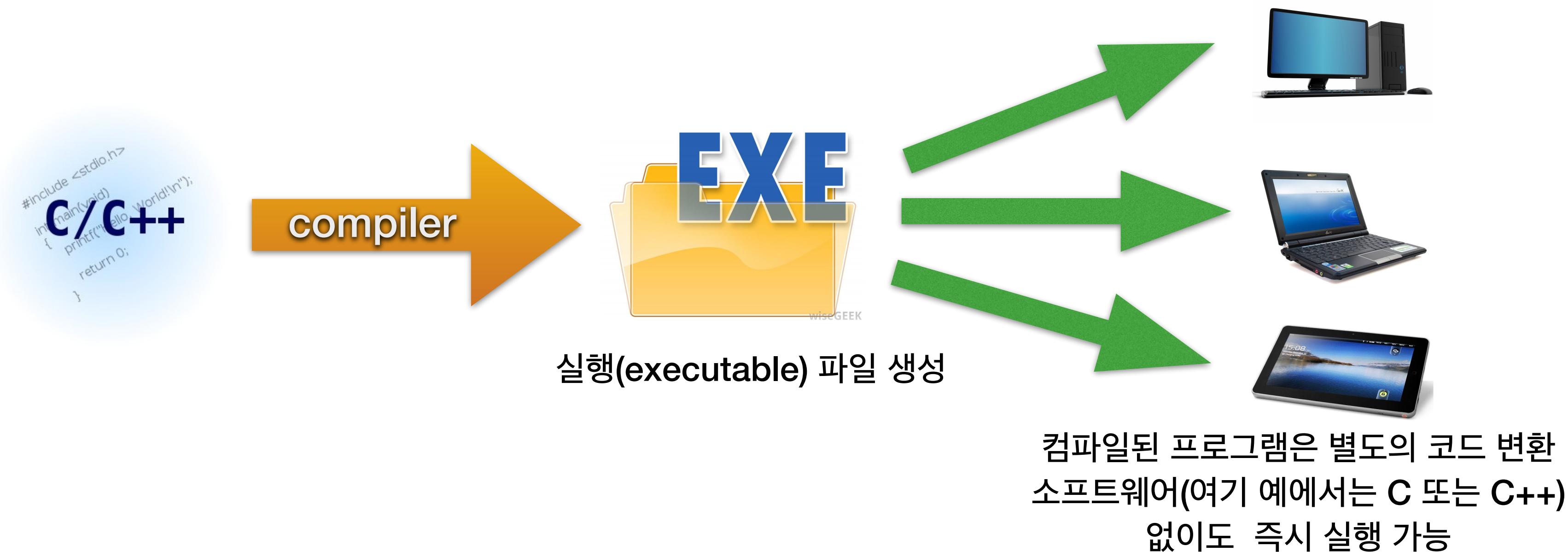


## 컴파일러

- 고수준 언어로 작성된 프로그램을 컴퓨터가 ‘이해’할 수 있는 별도의 기계어 프로그램(1 또는 0으로 구성된)으로 변환해주는 프로그램
- 고수준 언어 프로그램은 실행하기 전에 기계어로 변환하는 과정(compile)이 필요한 반면, 기계어(원시코드, native code) 프로그램은 바로 실행이 가능

## 컴파일러의 장점

- 미리 기계어로 변환이 되었기 때문에 **빠른 실행**이 가능
- 기계어로 변환된 프로그램은 역설계(reverse-engineer)을 통해 원본 코드(source code)가 유출되는 것을 어렵게 함



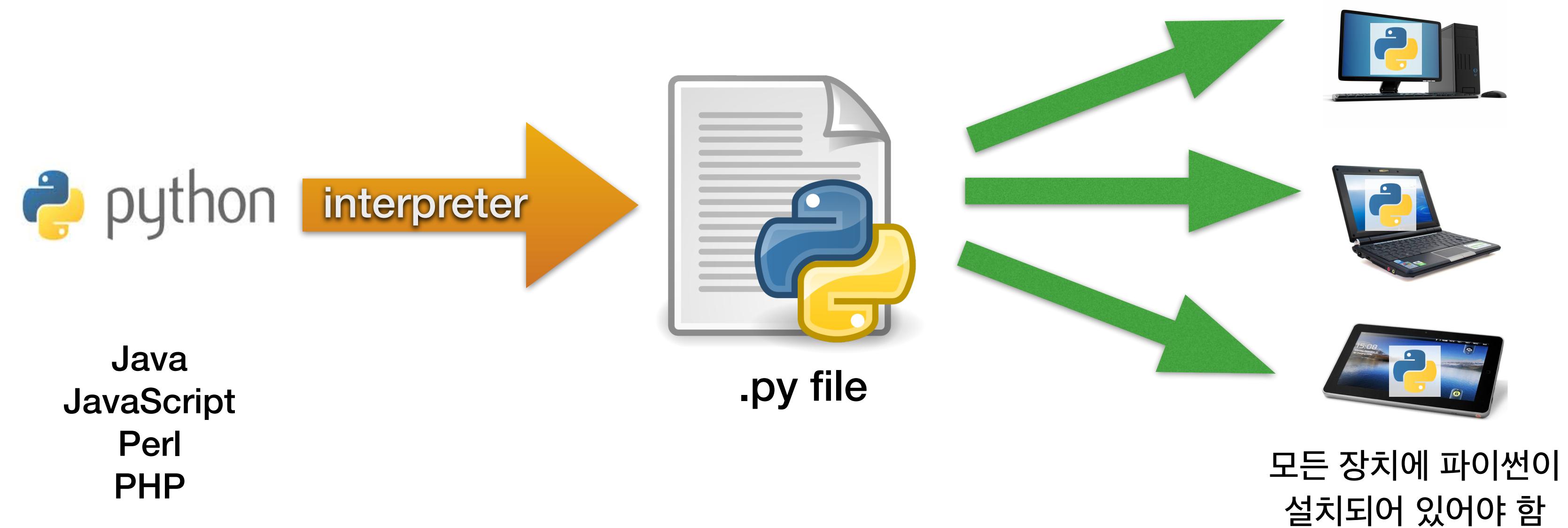


## 인터프리터

- 고수준 언어 프로그램의 명령어들을 변환하고 실행하는 프로그램
- 한번에 한 줄씩 명령문을 해석함(코드가 실시간으로 한 줄씩 실행됨)
- 컴파일러와 달리 기계어로 변환된 별도의 실행(executable) 파일이 생성되지 않음(즉, 프로그램이 원시코드(native code)로 변환되지는 않음)
- 따라서 프로그램을 실행하려는 장치에 반드시 인터프리터가 설치되어 있어야 함

## 인터프리터의 장점

- 인터프리터만 설치되어 있으면 어떤 장치에서도 실행이 가능(platform independence)
- 인터프리터로 작성된 프로그램은 상대적으로 파일 크기가 작음





# Step 4 : 프로그램 실행과 검증

## ● 프로그램 오류의 종류와 디버깅(debugging)

### ● 문법 오류(syntax error)

• 오류 수정이 쉬움(즉, 버그 잡기가 쉬움)

### ● 런타임 오류(runtime error)

• 예외(exception) 처리를 해야 함

### ● 논리 오류(logical error)

• 오류 없이 프로그램이 실행되지만 예상한 결과가 나오지 않는 경우

•  $1 / 2 * 3$

• 문법 오류는 컴파일 단계에서, 런타임 오류와 논리 오류는 실행 단계에서 발생



## ● 실행 단계에서의 오류

• 문법 오류가 없는 프로그램이라도 실행단계에서 오류가 발생하지 않는다고 말할 수 없음

• 실행 단계에서 오류를 피하기 위해서는 샘플 데이터를 넣어 테스트 해보는 것이 좋음

• 샘플 데이터는 신중히 선택하고 오류 발생 가능한 다양한 경우의 수를 고려 —> be a creative pessimist!

• 발견된 오류로 인해 프로그램 논리를 변경해야 할 수도 있음



# 프로그램 논리 설계

Lab Exercises





# Lab : 의사코드를 파이썬 코드로 변환



- 의사코드 예시

- 사용자가 입력한 값이 짝수인지 홀수인지 구하기

- 파이썬 코드로 변환 한 예시