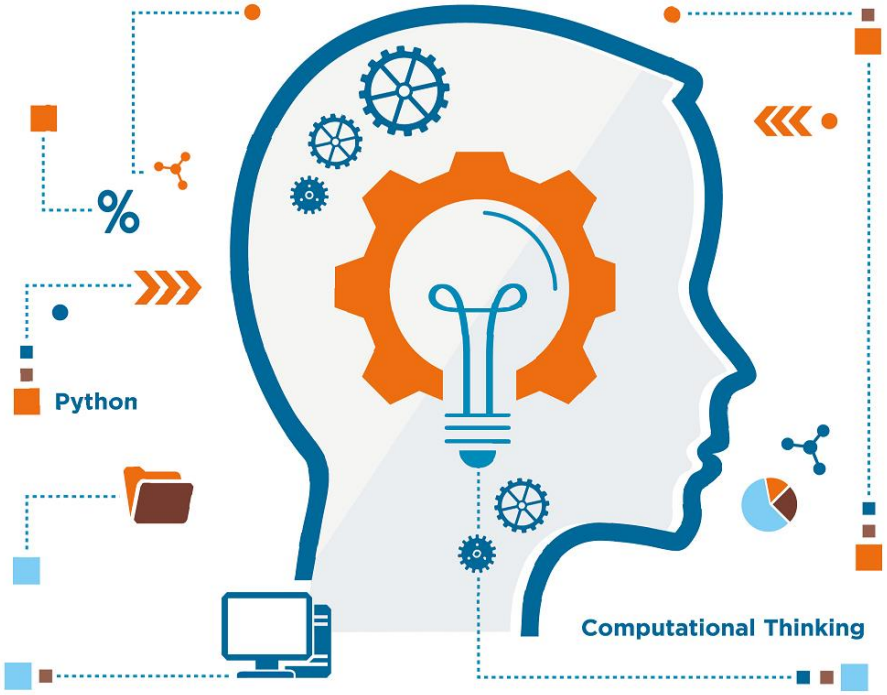


[Python]

Python으로 배우는

소프트웨어 원리

Chapter 01. 컴퓨팅 사고와 프로그래밍

목차

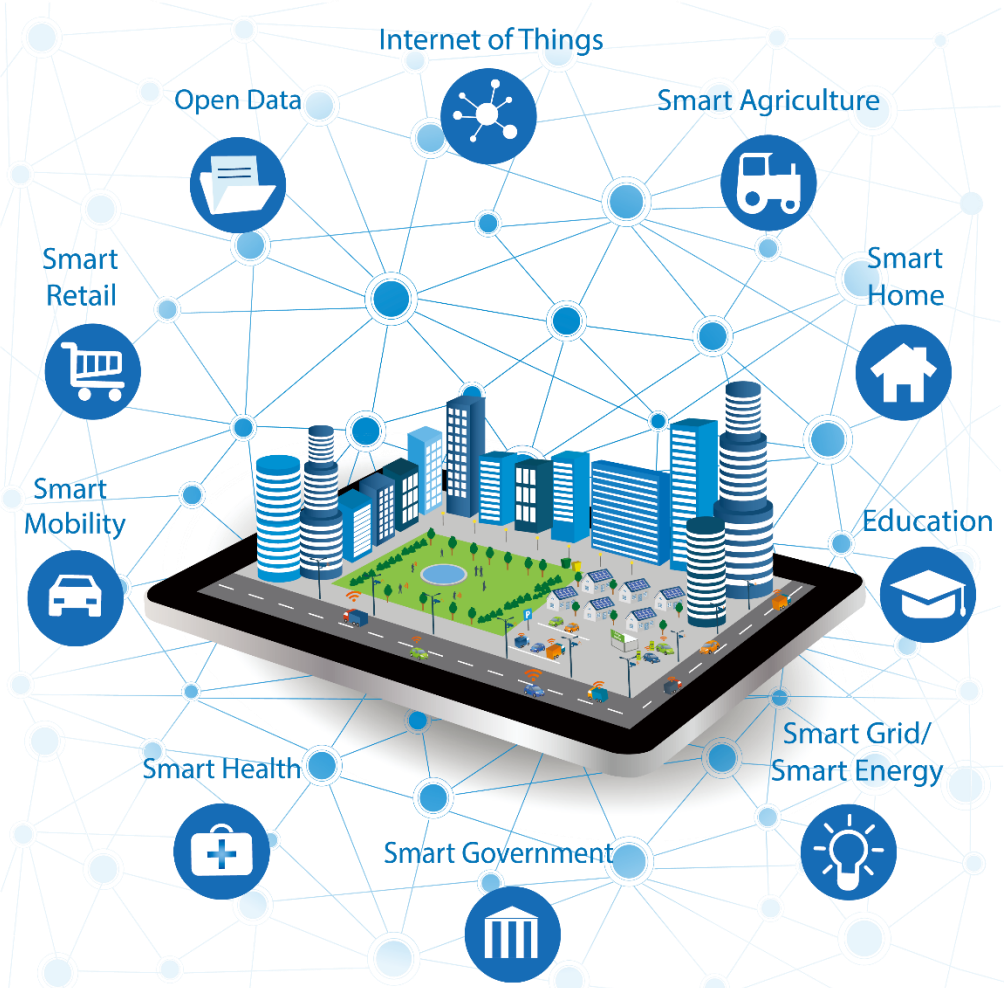
01

컴퓨팅 사고의 개념

01. 컴퓨팅 사고의 개념

[여러 분야에서 컴퓨터 활용이 증가]

- 컴퓨터 사용뿐만 아니라, '컴퓨팅 사고'에 대한 중요도도 커지고 있다.



01. 컴퓨팅 사고의 개념

I. 컴퓨터의 구성과 특징

- 컴퓨터 시스템은 **하드웨어**와 **소프트웨어**로 구성
- **하드웨어(Hardware)**
  - 프로그램과 데이터를 **기억장치**에 저장시켜놓고 필요 시 **중앙처리장치**로 실행하여 사용자(또는 다른 장치)의 요구사항을 처리하는 전자적/기계적 장치
  - 필요 시 외부로부터 데이터를 받아들이거나 제공하기 위한 **입력장치**와 **출력장치**를 사용

그림 1-2 컴퓨터 시스템의 구성

01. 컴퓨팅 사고의 개념

I. 컴퓨터의 구성과 특징

- 컴퓨터 시스템은 **하드웨어**와 **소프트웨어**로 구성
- **소프트웨어(Software)**
  - 소프트웨어는 하드웨어 장치가 실행하는 프로그램들을 의미
  - **시스템 소프트웨어**는 하드웨어 장치들이 프로그램들을 원활히 실행할 수 있도록 관장하는 프로그램들 → 운영체제(Window, Linux, Android, IOS 등)
  - **응용 소프트웨어**는 시스템 소프트웨어 기반으로 개발된 프로그램들을 의미

그림 1-2 컴퓨터 시스템의 구성

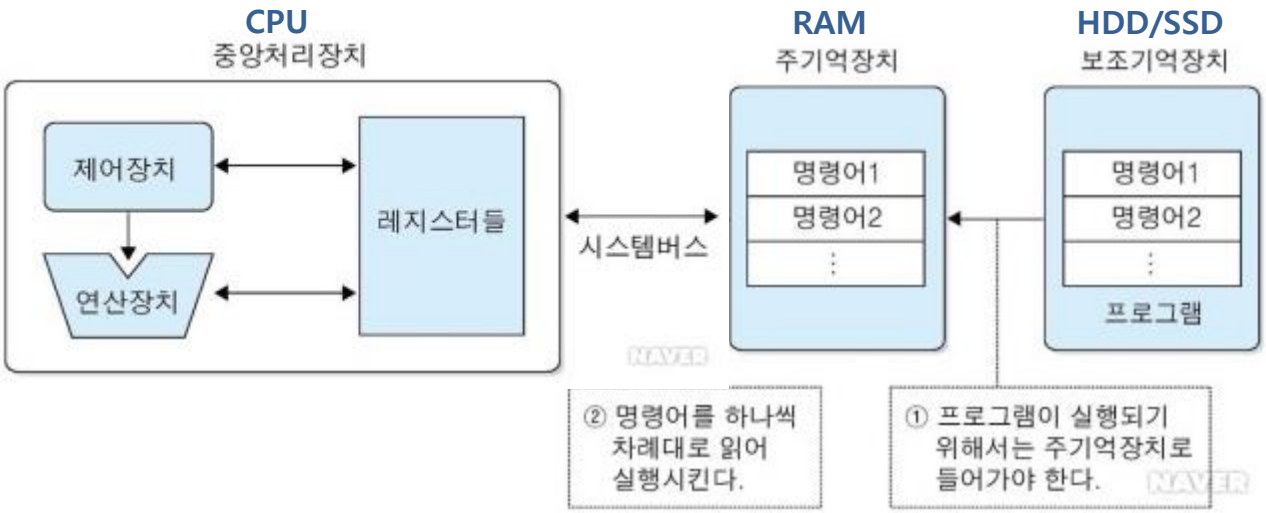
01. 컴퓨팅 사고의 개념

그림 1-10 스프레드시트 프로그램

I. 컴퓨터의 구성과 특징

▪ 컴퓨터에서 프로그램이 실행되고 있다는 의미

- 실행될 프로그램과 데이터를 주기억장치(RAM)에 올려서, 명령어를 하나씩 순차적으로 중앙처리장치의 레지스터에 올려서 CPU로 실행하는 것
- 여러 프로그램이 동시에 실행되는 것처럼 느껴지는 것은 시간을 아주 짧게 쪼개어서 (Timeslice) 그 시간 간격 동안 CPU를 번갈아 가면서 사용하기 때문에 가능하다.



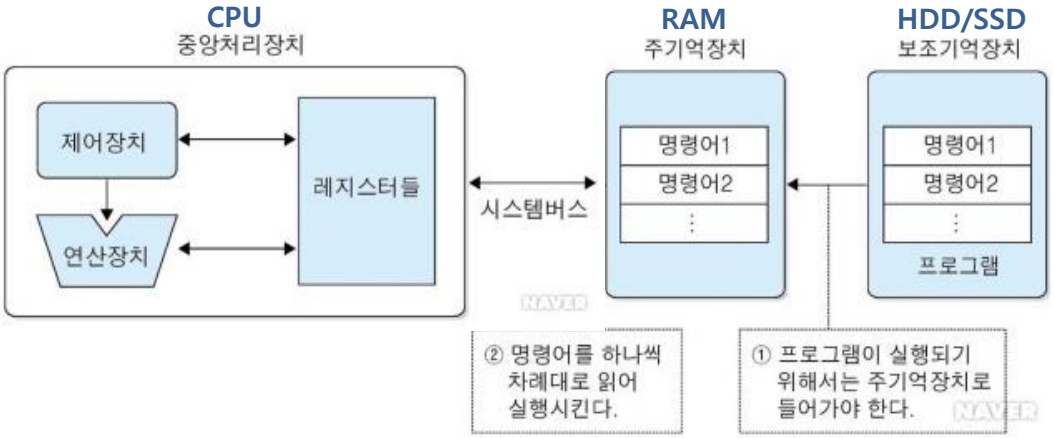
01. 컴퓨팅 사고의 개념

I. 컴퓨터의 구성과 특징

그림 1-4 마이크로프로세서

▪ [하드웨어] 중앙처리장치(CPU)

- 컴퓨터가 수행할 명령을 **읽고**, **해석**해서 **실행**하는 장치
- CPU는 제어장치, 연산장치, 레지스터로 구성
  - **제어 장치**는 명령어를 해석한 결과에 따라 다른 장치들에게 동작 신호를 전달
  - **연산 장치**는 명령어를 해석한 결과에 따라 연산(산술, 논리, 관계, 이동 등) 수행
  - **레지스터**는 명령어 실행 과정에서 필요한 데이터들을 잠시 기억
    - » 실행 중인 {명령어(IR), 명령어 위치(PC), 입력/결과 데이터들}을 기억



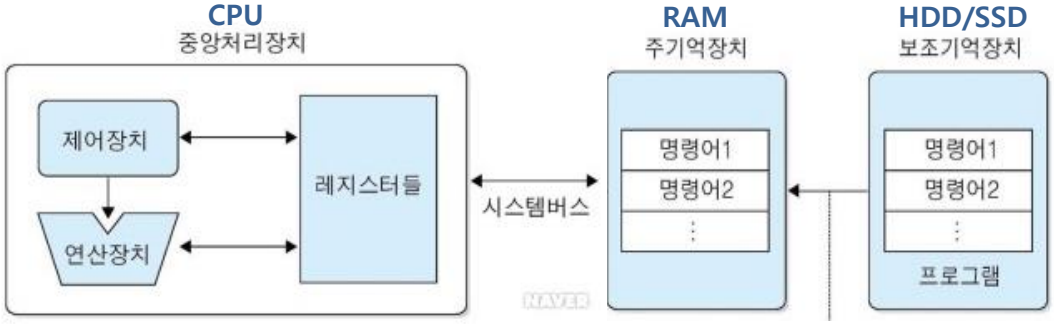
01. 컴퓨팅 사고의 개념

I. 컴퓨터의 구성과 특징

그림 1-5 하드디스크

▪ [하드웨어] 기억장치

- 주기억장치와 보조기억장치로 구성
- **주기억장치**는 RAM과 같은 휘발성 메모리
  - 전자적인 기억장치로 메모리 주소에 의해 직접 접근(random access)
  - CPU가 직접 사용하는 기억장치, 실행 중인 프로그램과 데이터를 임시로 위치시킴.
- **보조기억장치**는 하드디스크(HDD), SSD, USB와 같은 비휘발성 메모리
  - 사용하는 프로그램과 데이터들의 원본을 지속적으로 저장하기 위한 기억(저장) 장치
  - 주로 저렴하면서 대용량인 HDD(Hard Disc Drive 기계적/자기적 동작, 느림)를 사용
  - SSD(Solid State Drive)는 전자적으로 동작하기 때문에 빠름(비쌈)



01. 컴퓨팅 사고의 개념

I. 컴퓨터의 구성과 특징

▪ [하드웨어] 입력장치, 출력장치

- **입력장치** : 키보드, 마우스와 같이 외부 정보를 컴퓨터로 전달하기 위한 장치

그림 1-6 키보드

- **출력장치** : 모니터, 스피커, 프린터 등이 출력장치

그림 1-7 HMD

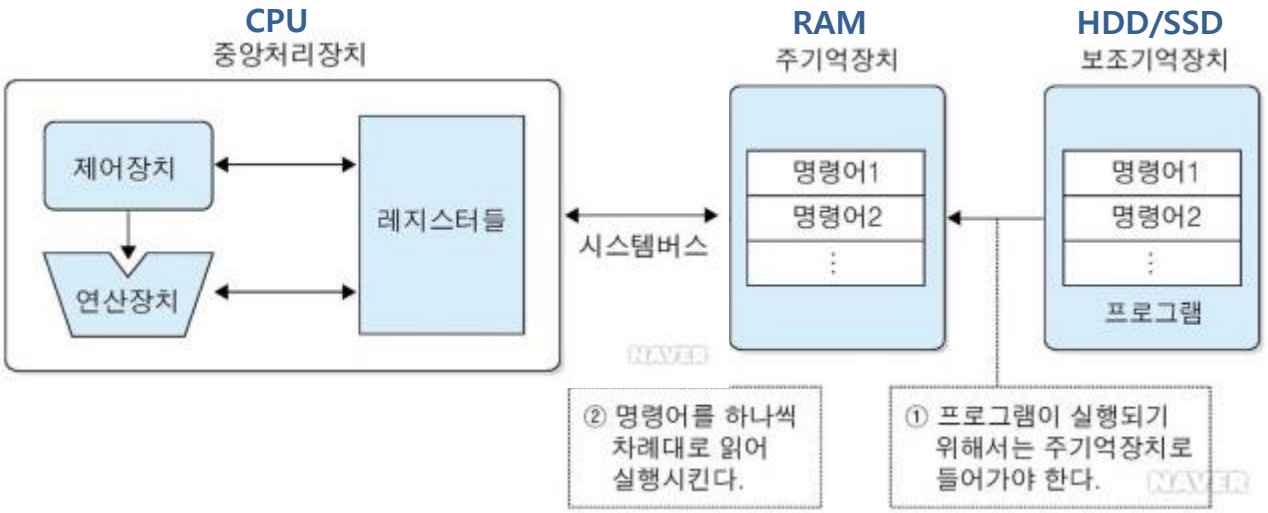
01. 컴퓨팅 사고의 개념

그림 1-10 스프레드시트 프로그램

I. 컴퓨터의 구성과 특징

▪ [컴퓨터가 프로그램을 실행하는 과정]

- [실행 요구] 저장장치(**HDD**)에 있는 **프로그램**(컴파일된 기계코드)과 데이터가 RAM에 올려짐
- [실행: **인출**] RAM에서 실행할 **명령어** 중에 하나를 CPU 레지스터(명령 레지스터)로 복사
- [실행: **해독**] 명령 레지스터(IR)의 명령문을 해석
- [실행: **실행**] 해석 결과에 따라 명령어 실행 (산술, 논리, 관계, 이동 등)
  - RAM의 용량이 작으면 생기는 문제점은?

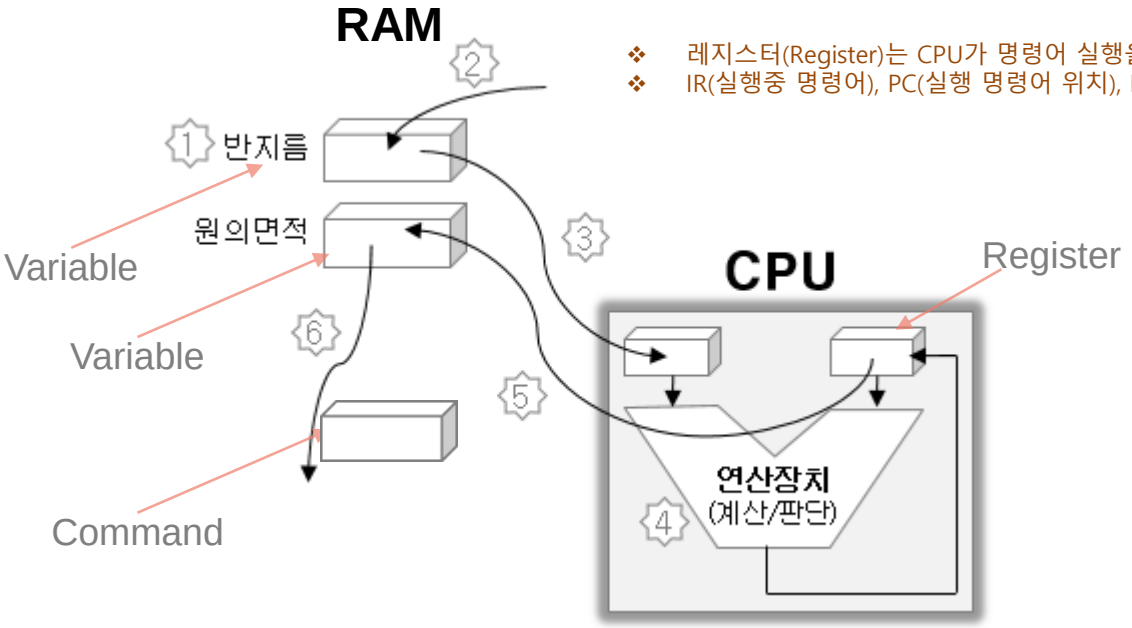


01. 컴퓨팅 사고의 개념

I. 컴퓨터의 구성과 특징

▪ [컴퓨터가 프로그램을 실행하는 과정]

- [실행 요구] 저장장치(**HDD**)에 있는 **프로그램**(컴파일된 기계코드)과 데이터가 RAM에 올려짐
- [실행: **인출**] RAM에서 실행할 **명령어** 중에 하나를 CPU 레지스터(명령 레지스터)로 복사
- [실행: **해독**] 명령 레지스터(IR)의 명령문을 해석
- [실행: **실행**] 해석 결과에 따라 명령어 실행 (산술, 논리, 관계, 이동 등)



- 레지스터(Register)는 CPU가 명령어 실행을 위해 사용하는 기억 공간
- IR(실행중 명령어), PC(실행 명령어 위치), R1/R2/AC(연산중인 데이터)

01. 컴퓨팅 사고의 개념

그림 1-9 마이크로소프트의 윈도우 운영체제

I. 컴퓨터의 구성과 특징

▪ [컴퓨터가 프로그램을 실행하는 과정]

- HDD에 있는 **프로그램**과 데이터가 RAM에 복사해놓고, 명령어 한 개씩 CPU에 가져다가 해석해서 실행(연산과 제어)함.

➢ 명령어 실행은 컴퓨터의 클럭Clock 신호의 주기에 맞추어 진행(그래야 동시에 주고, 받기가 가능)

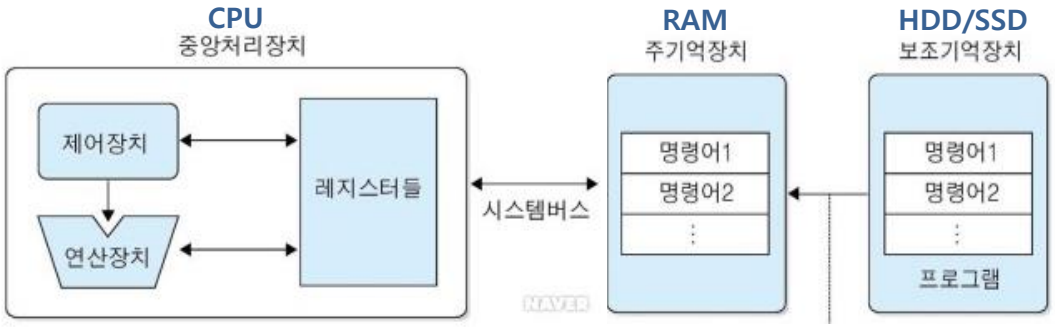
➢ 명령어 하나를 실행하는 과정에 복수 개의 클럭 주기를 사용하게 됨.

➢ 일정 시간이 흐르면 다른 프로그램에게 연산장치 사용을 양보해야함(운영체제가 관장)

✓ 양보 전에 현재 진행중인 연산 값(레지스터들에 있는)들은 잠시 RAM의 특정한 공간으로 대피시켜놓음.

✓ 다시 연산 기회가 오면, 피신시켜놓은 레지스터 값들을 복구시켜서 작업을 계속함. (Context Swap)

✓ 이러한 과정을 **Timesharing**이라고 함.



01. 컴퓨팅 사고의 개념

I. 컴퓨터의 구성과 특징

▪ [컴퓨터의 속성: 인간과 비교]

- [**신뢰성**] 컴퓨터는 애매모호하지 않는 명확한 연산을 한다.
  - » 컴퓨터는 디지털 데이터(0과 1 둘중 하나만 있는)를 처리 하므로 입력 및 처리 결과에 대한 애매모호성이 없다.
  - » 입력 데이터 및 실행 프로그램(운영체제 포함)에 문제가 없다면 처리 결과를 신뢰할 수 있다.

- [**고속성**] 컴퓨터는 빠른 연산을 한다.
  - » 컴퓨터는 요구하는 목적을 달성하기 위해 일련의 명령어들을 수행하여 해결한다.
  - » 주어진 절차대해서는 빠르고 명확하게 처리, 하지만 주어진 절차 이 외에는 한계가 있다.
  - » 컴퓨터의 처리속도는 CPU와 각 장치의 성능에 의해 결정된다.

- [**대용량성**] 컴퓨터는 다량의 데이터를 보관하고 처리한다.

- [**범용성**] 컴퓨터는 문자뿐만 아니라 그림, 동영상, 소리 등과 같은 다양한 형태의 데이터를 처리한다.

01. 컴퓨팅 사고의 개념

II. 컴퓨팅 사고의 필요성

- **컴퓨팅 사고 :** 컴퓨팅 사고는 세상을 바라보고 분석하는 하나의 방법으로 문제해결을 위한 방법이나 생각을 추상화하여 명시적인 코드로 표현할 수 있는 능력

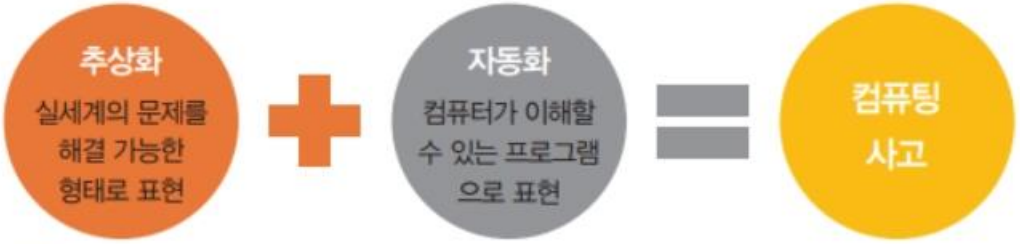


그림 1-12 컴퓨팅 사고의 개념

- **[컴퓨팅 사고**(문제 해결 방법의 추상화)**가 필요한 이유]**
  - 인간은 아날로그적 기억과 사고를 주로 하므로 애매모호한 상황이 발생
  - 컴퓨터는 2진 디지털 값을 기반으로 연산하므로 컴퓨터에게 프로그램을 통해서 작업 방식을 정해주려면 문제 해법에 대한 실행 방법에 대한 **명확**한 정의를 해주어야 함.
  - 즉, 실행 방법에 대한 명확한 정의는 불필요한 부분은 모두 제거하고 필요한 부분만 **논리적**으로 **추상화**(프로그래밍 언어 체계에 맞춰)를 해주는 과정이다.

01. 컴퓨팅 사고의 개념

III. 컴퓨팅 사고의 과정

- **분해** : 먼저 문제를 분석하여 작게 분해, 자동화가 불필요한 부분은 제거
- **추상화** : 필요한 정보를 수집하거나 반복되는 **패턴**을 찾아서 단순하게 **구조화**
- **알고리즘 작성** : 세부적인 부분을 생략하고 중요한 속성만 묘사하며, 문제를 정의할 때 중요 부분만 강조해서 표현
- **프로그래밍** : 추상화된 문제 해결책(알고리즘)을 개발환경에 맞추어 자동화된 형태로 **모델링**하고, 마지막으로는 실행할 수 있는 프로그램으로 **구현**

그림 1-13 컴퓨팅 사고의 과정

III. 컴퓨팅 사고의 과정

▪ 분해

- 하나의 복잡한 문제를 작은 단위로 나누는 과정

그림 1-15 탑 만들기 문제에서의 패턴 찾기

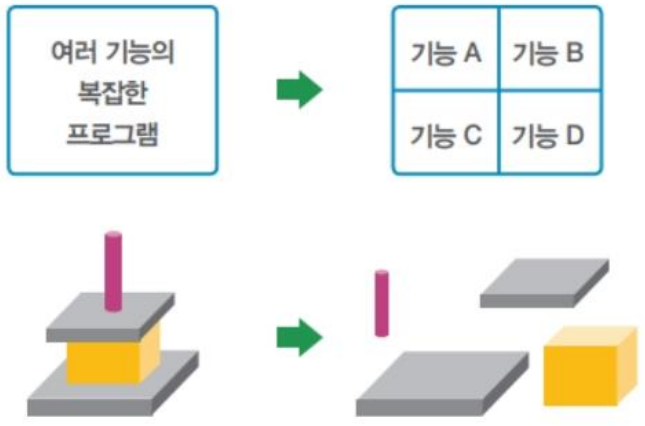


그림 1-14 문제의 분해

▪ 패턴 찾기

- 여러 번 반복되는 절차나 작업을 찾아 하나의 문제 내에서 혹은 다른 문제를 해결하는 과정에서 **공유(재사용)**하여 작업 효율을 높임

→ 나중에 함수나 클래스 형태로 만들어서 모듈이나 라이브러리로 묶어서 공유

III. 컴퓨팅 사고의 과정

▪ 알고리즘 작성

- 어떤 문제를 해결하기 위한 작업의 순서
- 순서도나 의사코드 이용

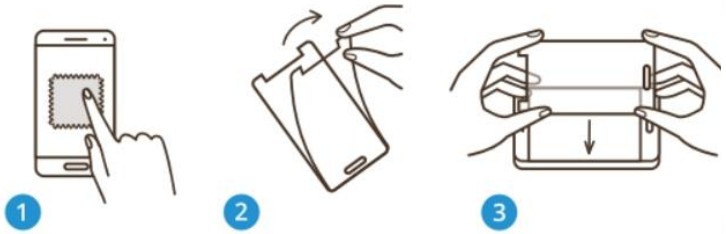
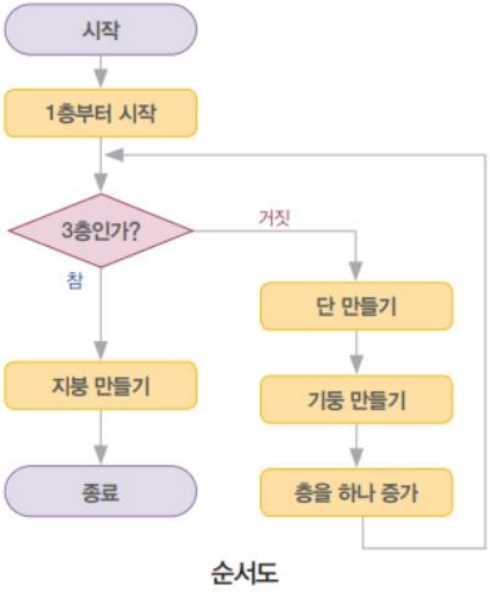


그림 1-16 일의 절차를 표현하는 설명서



1. 1층부터 시작한다.
2. 3층이 아니면,
   단과 기둥을 만든다.
   층을 하나 증가한다.
   2.를 반복한다.
3. 지붕을 만든다.

의사코드

그림 1-17 3층 높이 탑 만들기 문제의 순서도와 의사코드

▪ 프로그래밍

- 프로그래밍은 컴퓨터에게 어떤 동작을 해야 하는지 지시하는 명령들을 작성하는 과정

01. 컴퓨팅 사고의 개념

IV. 컴퓨팅 사고의 구성 요소

표 1-1 컴퓨팅 사고의 구성 요소

구성 요소	설명
자료 수집 (Data Collection)	적당한 정보를 수집하는 과정
자료 분석 (Data Analysis)	데이터의 의미를 만들고, 패턴 발견 → 결론 도출
자료 표현 (Data Representation)	적절한 그래프, 차트, 단어, 이미지 등으로 데이터를 구성하고 묘사
문제 분해 (Problem Decomposition)	관리할 수 있게 작은 단위로 작업을 분할
추상화 (Abstraction)	주요 아이디어를 단위 데이터로 정의하기 위해 복잡성 축소
알고리즘과 절차 (Algorithms & Procedures)	문제를 해결하거나 어떤 결과를 달성하기 위해 단계의 순서를 나열
자동화 (Automation)	컴퓨터나 기계를 가지고 반복적이거나 지루한 작업을 수행
시뮬레이션 (Simulation)	프로세스의 모델링 표현, 모델을 사용하여 실험을 수행하는 것을 포함
병렬화 (Parallelization)	동시에 공통의 목표에 도달하는 작업을 수행하기 위해 자원을 구성

02

프로그래밍 언어의 개념

02. 프로그래밍 언어의 개념

I. 프로그래밍 언어의 종류

- 사람이 사용하는 언어는 **자연어**, 컴퓨터가 사용하는 언어는 **기계어**
- 기계어는 0과 1로 이루어진 이진수이므로 사람과 다른 언어 사용
- 사람이 이해하기 쉽고 작성하기 좋은 형태의 프로그래밍 언어(고급 언어) 탄생
- 프로그래밍 언어로 작성된 프로그램은 컴퓨터에서 기계어로 변환되어야 실행할 수 있고, 이 과정이 **컴파일**(Compile)

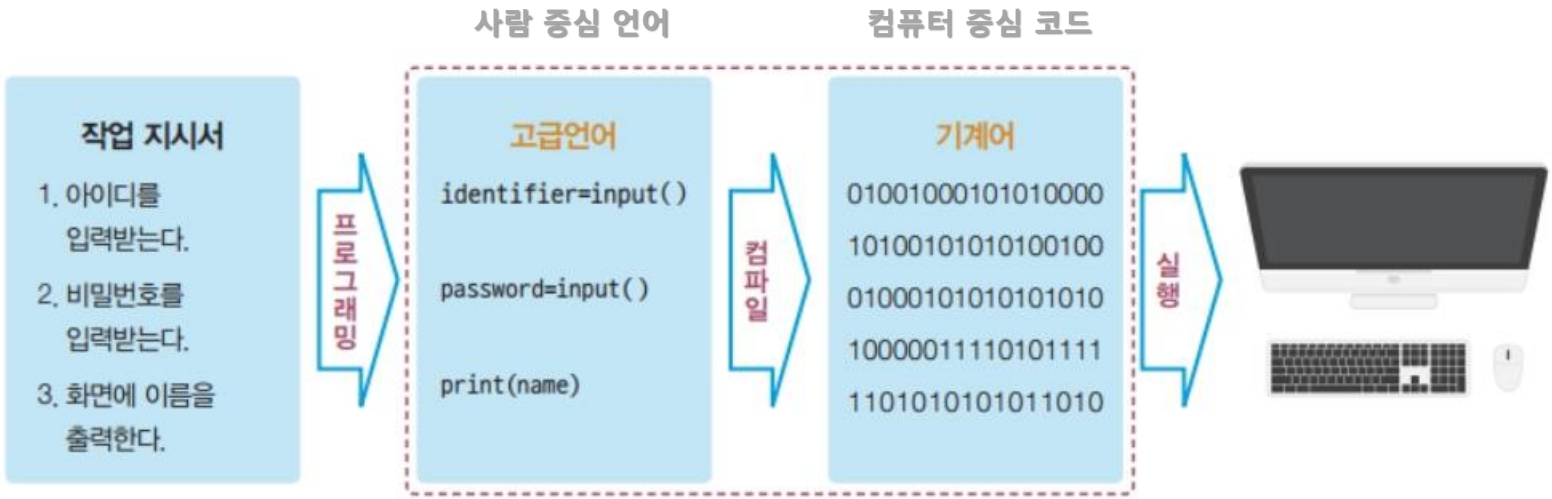


그림 1-19 컴파일의 개념

02. 프로그래밍 언어의 개념

I. 프로그래밍 언어의 종류

❖ **고급언어**는 인간에게 프로그램 개발이 용이하도록 구성한 언어 체계

- 다양한 종류의 **명령**이 존재 (산술, 논리, 관계, 이동 등)
- 데이터는 **상수**와 **변수**를 사용

❖ **기계어**는 중앙처리장치가 연산을 하기에 용이하도록 구성된 언어 체계

- 간단한 명령만 존재(더하기, 비교하기, 이동하기 등)
- 데이터는 기준점을 기준으로 하는 상대적인 **위치 값**으로 사용 → 어느 RAM에 올려지더라도 실행 가능

C 언어	**어셈블리 언어**(x86)

int CAS(int* pos, int oldval, int newval)
{
int oldpos = *pos;
if(*pos == oldval)
* pos = newval;
return oldpos;
}
 	
int a, b;
b = CAS(&a, 10, 20);
	CAS:
	mov ecx, dword ptr[esp + 4]
	mov eax, dword ptr[ecx]
	mov ebx, dword ptr[esp + 8];
	mov edx, dword ptr[esp + 12];
	cmpxchg ebx, edx;
	mov dword ptr[ecx], esp;
	ret 16;

II. 파이썬 소개

- **쉬운 난이도** : 컴퓨터 프로그래밍 학습을 위한 입문용 언어로 많이 사용
- 실행 과정이 간단한 **인터프리터 언어** : 명령어 단위로 번역해서 파일을 만들지 않고 바로 실행하는 인터프리터 방식
- 높은 **확장성**과 **생산성** : 인공지능, 빅데이터, 데이터 시각화, 게임 등 다양한 영역에서 활용할 수 있는 **오픈소스**와 **라이브러리**를 간단하게 설치하고 활용

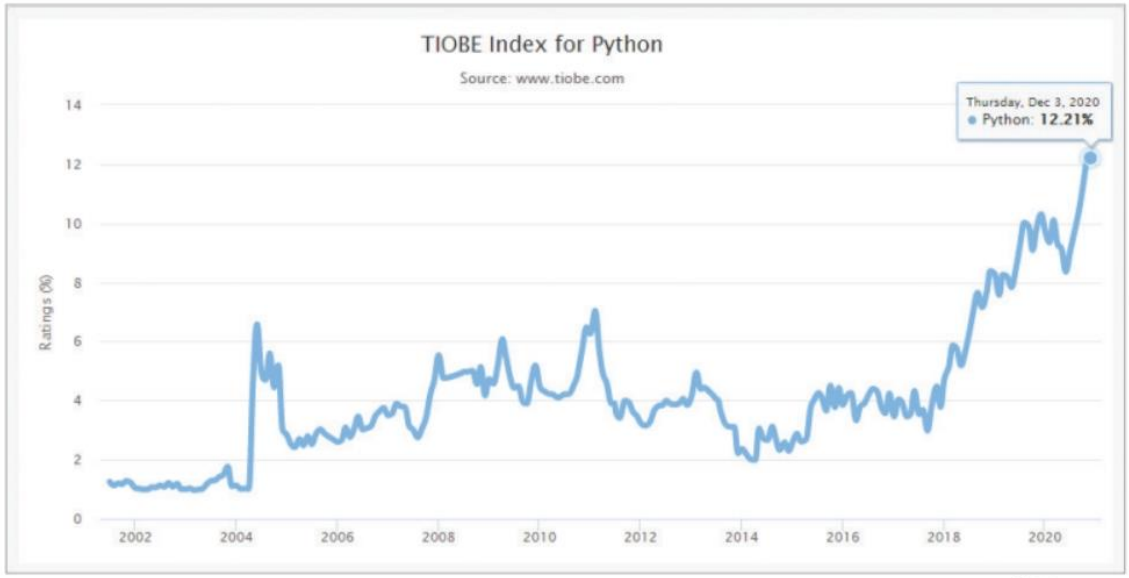
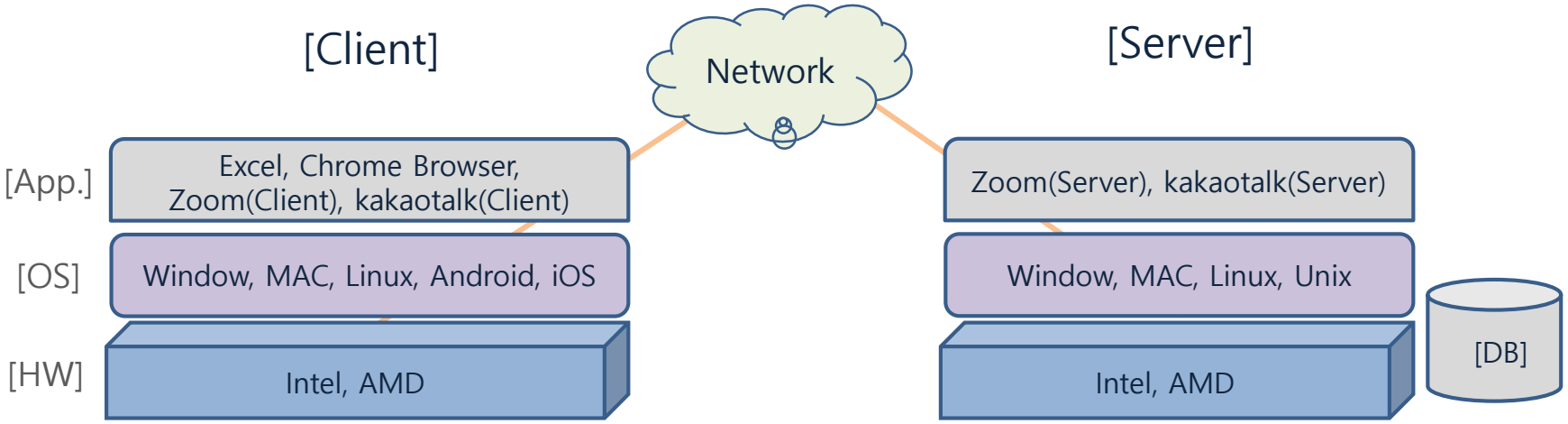


그림 1-21 파이썬 언어의 TIOBE 인덱스

❖ 컴퓨터 시스템

[컴퓨터 시스템의 구성]

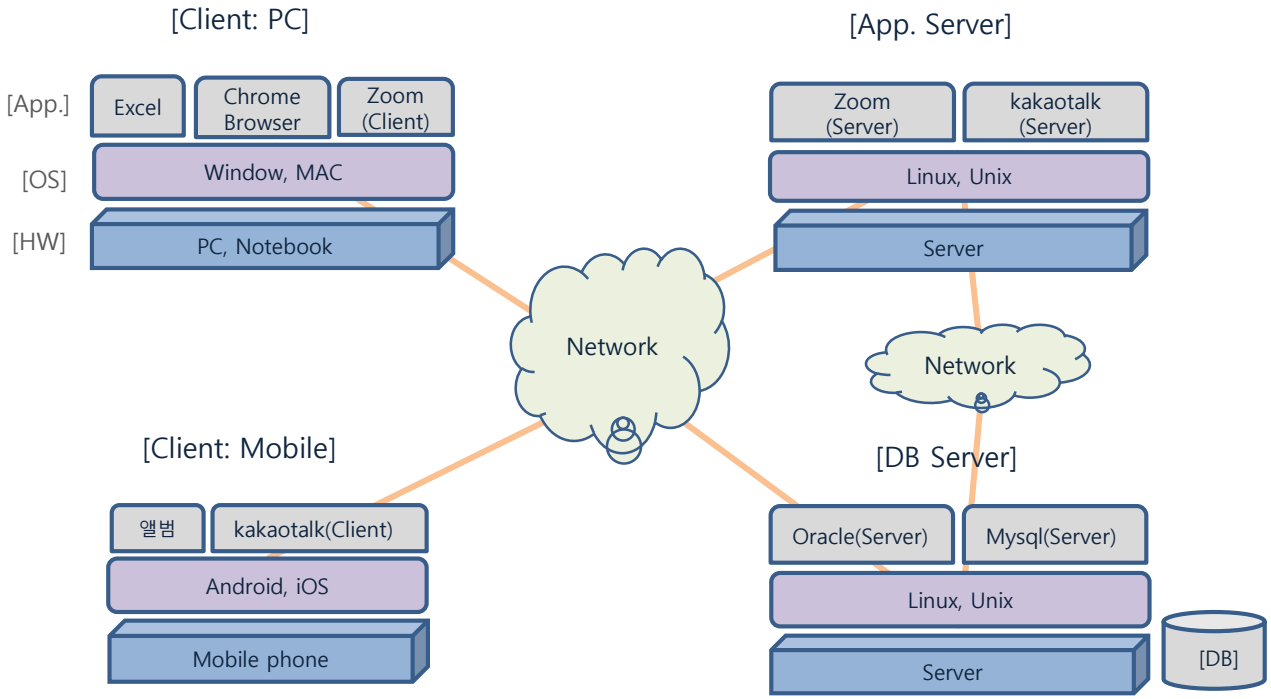
- 컴퓨터는 **하드웨어**Hardware와 **소프트웨어**Software로 구성
- 소프트웨어는 **운영체제**OperatingSystem와 **응용프로그램**ApplicationProgram으로 구성
- 운영체제는 응용프로그램들에게 컴퓨터 자원(CPU, RAM, HDD 등)을 효율적으로 사용할 수 있도록 자원관리 기능 담당
- 응용프로그램은 운영체제에게 자원 사용을 허락 받아 원하는 서비스 제공



❖ 컴퓨터 시스템

[응용프로그램의 상호 동작]

- 응용 프로그램의 역할 분담
  - **독자적 동작**: 개인용 컴퓨터에서 독자적으로 동작하는 프로그램(메모장, Excel, 지뢰게임)
  - **서버와 상호 동작**: Clien용 프로그램이 Server용 프로그램에게 작업 요청을 하여 동작
    - 반드시 Client 프로그램 설치가 필요 (Zoom, Kakaotalk)
  - 서버와 상호 동작을 하나 Client 프로그램 설치가 필요 없는 경우
    - **웹브라우저**를 Client 프로그램으로 사용하는 방식



Thank You !