

1. 시작하기 전에

2018.12

ALIS중대 ALIS운영병 김재형

강의 소개

강의 기본 목표

- 파이썬으로 간단한 프로그래밍을 할 수 있다.
- 컴퓨터적 사고를 가지고 문제를 해결할 수 있다.
- 컴퓨터 구조를 간단하게 알고 이해할 수 있다.

강의 심화 목표

- git을 이해하고 사용할 수 있다.
- Python으로 데이터 수집 및 분석을 할 수 있다.

강의 소개

주 교재

- Do it! 점프 투 파이썬
- 뇌를 자극하는 파이썬 3
- 컴퓨터 사이언스 부트캠프 with 파이썬
- 컴퓨팅 사고

부 교재 및 링크

- 구름 IDE의 README.md 파일을 참조

강의 소개

강의방식

- 매주 기본과제 및 심화과제가 존재
 - 모든 과제는 구름IDE로 제출
- 코드 리뷰 및 코드 알고리즘 발표
- 중간 및 기말과제

강의시간

- 강의시간: 1시간-1시간 30분
- 코드리뷰 및 알고리즘발표, Q&A진행:
30분-1시간

프로그래밍을 하는 이유?



소프트웨어 중심사회

SW와 전 영역간 초융합



SW중심사회의 생태계

개인 : SW를 통해 문제를 해결하고 창업/취업/향유

- ◆ SW 조기교육(미/영), MOOC로 온라인 무료 대학교육(8백만명)
- ◆ 일자리의 90%가 디지털기술 필요(EU), 개인 맞춤형(아마존의 Dash)

기업 : SW로 신 사업을 창출할 수 있는 창조적 파괴

- ◆ BMW의 무인카 R&D비용의 90%가 SW
- ◆ 무한상상 제품(3D프린팅, 구글 글라스), 신비즈니스(비트코인, IoT)

정부 : SW기반으로 국가 시스템을 효율적/능동적으로 운영

- ◆ 빅데이터를 이용한 약의 효능 확인비용 절감(미, 연 0.5억불)
- ◆ 공공이 클라우드서비스 선도(미/일/영), 무인카 운행 허용(네바다주)

소프트웨어 중심사회

“

우리 사회는 이제
SW가 개인·기업·정부의 혁신을
견인하는 SW중심사회로 진입

2015년 10대 기술트렌드 중 하나로
“이제 모든 기업은 소프트웨어 기업이다.
(All companies are now software companies)”
(‘15.1월 뉴스위크)

”



4차 산업혁명

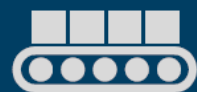
4차 산업혁명이란 무엇인가?

파괴적 기술과
역사적 산업혁명의 전개



1차 산업혁명

증기기관
18세기



2차 산업혁명

전기·내연기관
19-20세기



3차 산업혁명

컴퓨터·인터넷
20세기 후반

IoT

로봇



AI

3D
프린팅

AR/VR

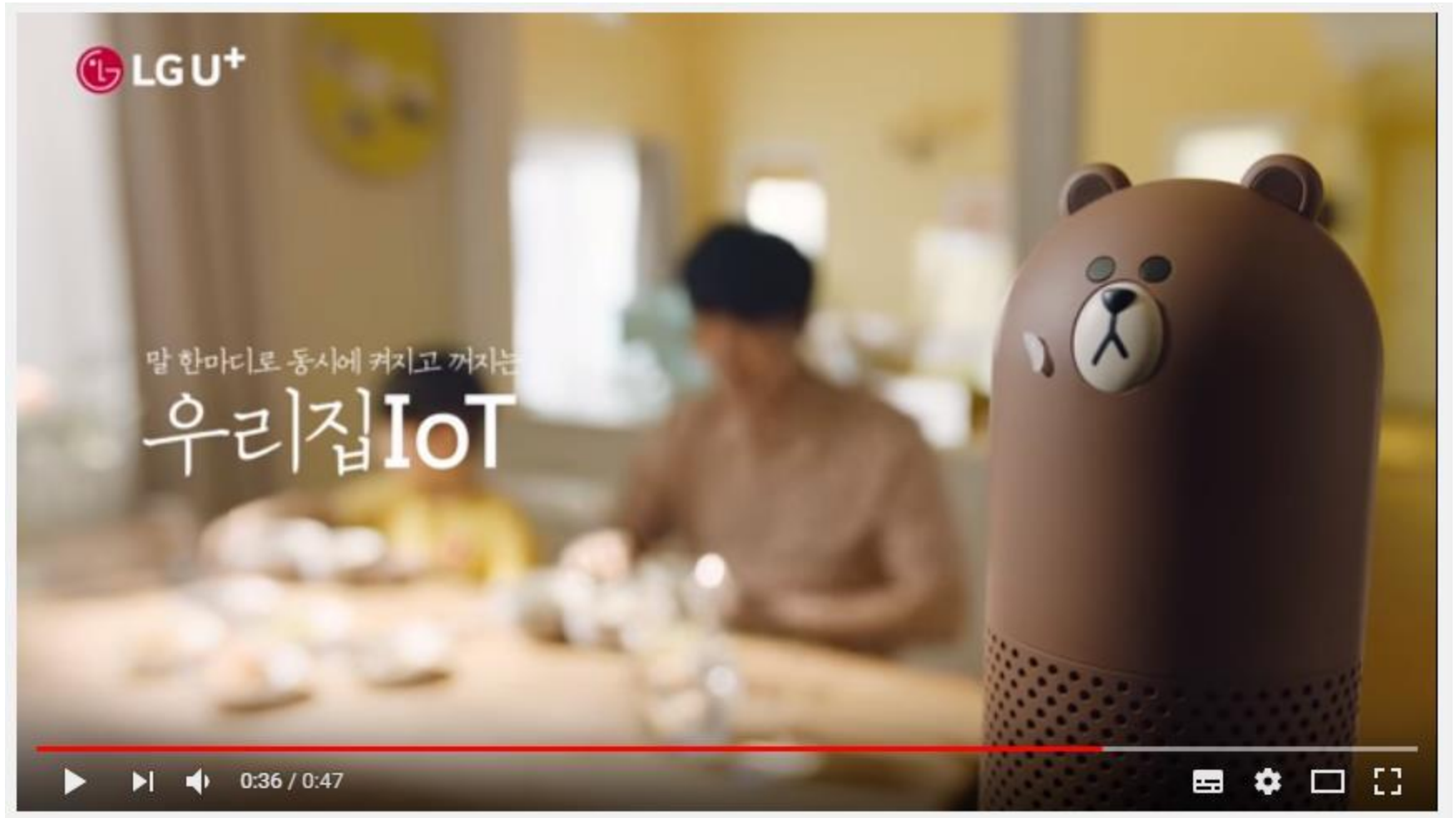
AI 기술을 핵심동인으로
상품·서비스의 생산·유통·소비 전 과정에서
모든 것이 연결되고 지능화

IoT

IoT(Internet of Things)



IoT



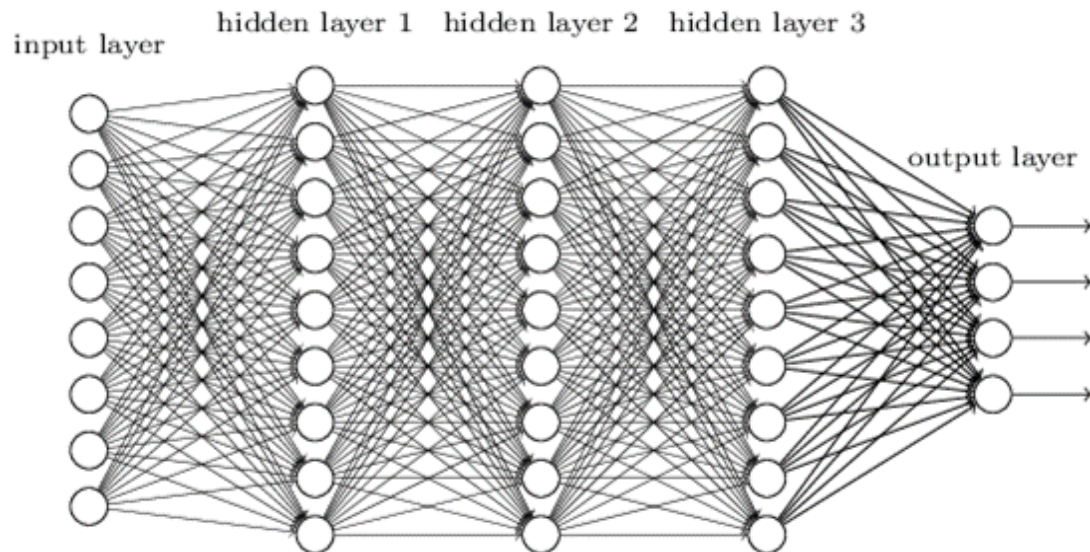
스마트그리드



딥러닝



Deep neural network



O2O



배달의민족



가장 싸게 꿈을 실현



컴퓨터?

전자회로를 이용하여 정보를 전자적 형태로 저장하고 계산하고 입력된 데이터를 정해진 방법에 따라 처리하는 장치

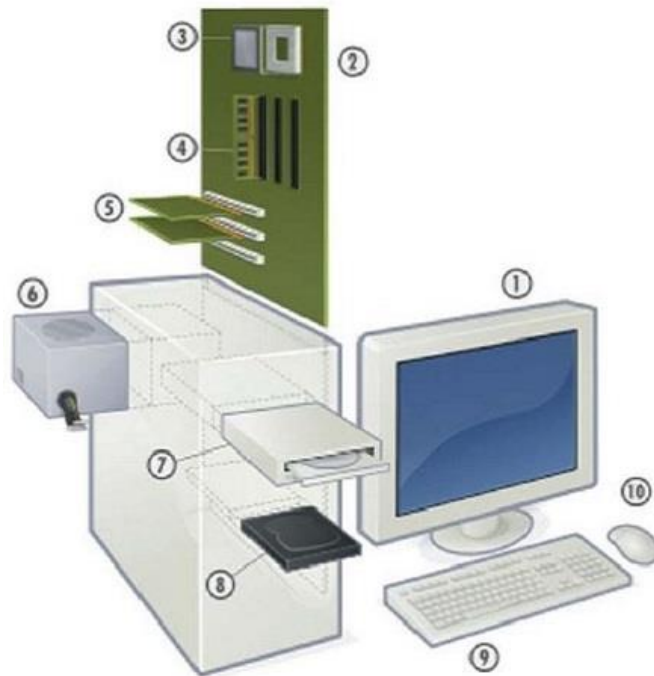
- 다만, 이 정의는 정확하지만, 범위가 너무 넓다.



컴퓨터의 구성

하드웨어(Hardware)

— 물리적 장치



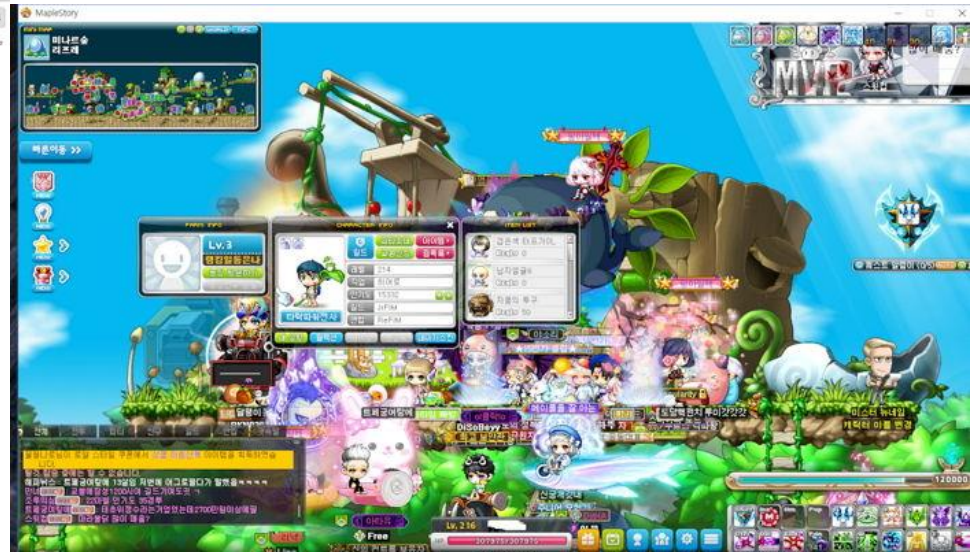
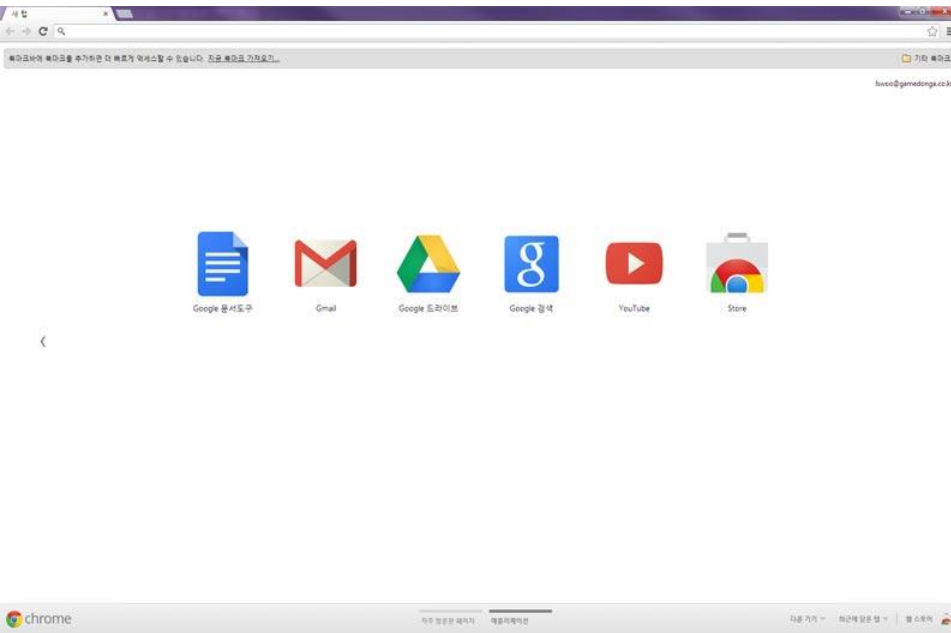
Hardware of a modern
Personal Computer:

1. Monitor
2. Motherboard
3. CPU
4. RAM
5. Expansion cards
6. Power supply
7. Optical disc drive
8. Hard disk drive
9. Keyboard
10. Mouse

컴퓨터의 구성

소프트웨어(Software)

- 작업을 수행하는 명령어를 모은 프로그램(program)의 통칭
- 코드(code)



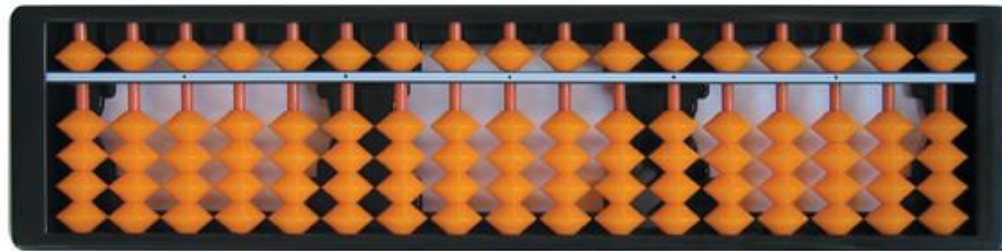
프로그래밍?



컴퓨터의 역사-계산장치

과거

—주판



—주판에서 가져온 개념

—저장소

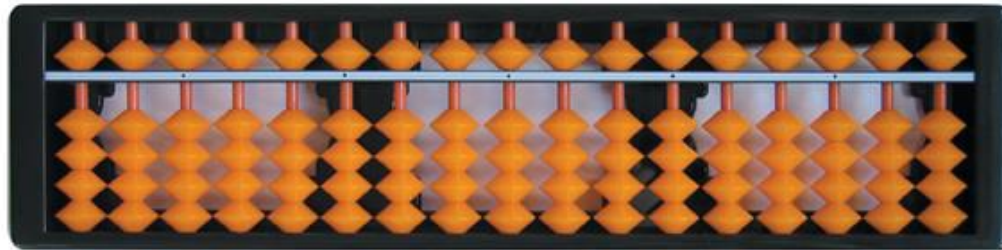
—표현방법

—계산

—사용자 인터페이스

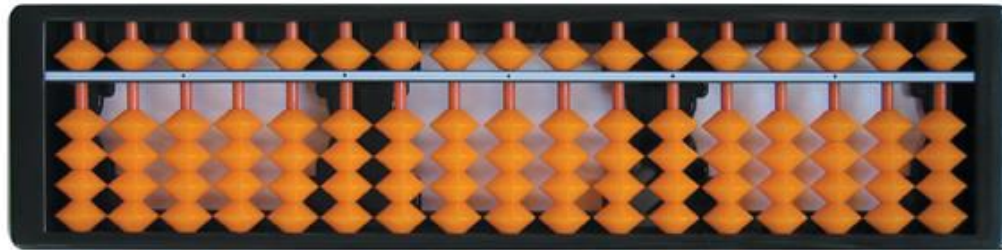
컴퓨터의 역사-계산장치

- 주판
 - 저장소
 - 동일한 숫자 값을 유지
 - 데이터: 저장하는 항목
 - 표현방법
 - 축에 있는 구슬을 사용하여 정수를 저장
 - 주판알들의 위치: 표현 방법, 수치적인 값으로 해석



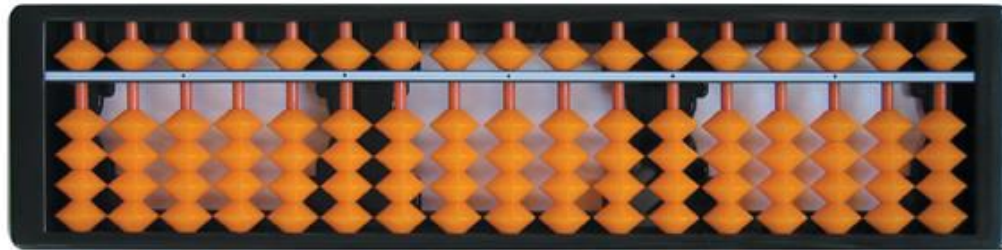
컴퓨터의 역사-계산장치

- 주판
 - 계산
 - 사람이 밀어야 가능
 - 사용자 인터페이스
 - 인간이 기계와 소통하는 방식
 - 주판알을 움직이기 위해 손가락과 엄지를 사용
 - 주판알의 위치에 따라 표현된 값을 시각적으로 해석

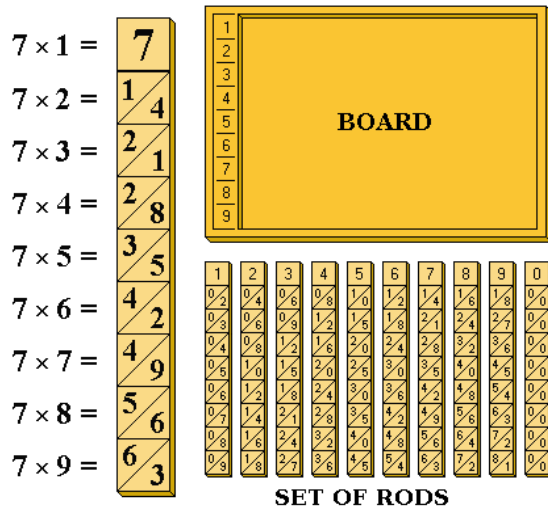


컴퓨터의 역사-계산장치

- 주판
 - 계산
 - 사람이 밀어야 가능
 - 사용자 인터페이스
 - 인간이 기계와 소통하는 방식
 - 주판알을 움직이기 위해 손가락과 엄지를 사용
 - 주판알의 위치에 따라 표현된 값을 시각적으로 해석



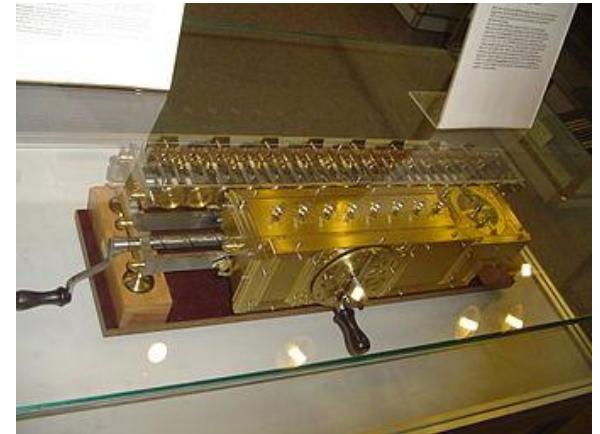
컴퓨터의 역사-계산장치



네이피어의 계산봉
(Napier's bones)



파스칼린
(Pascaline)



라이프니츠 계산기
(Leibniz' calcultor)

컴퓨터의 역사-소프트웨어

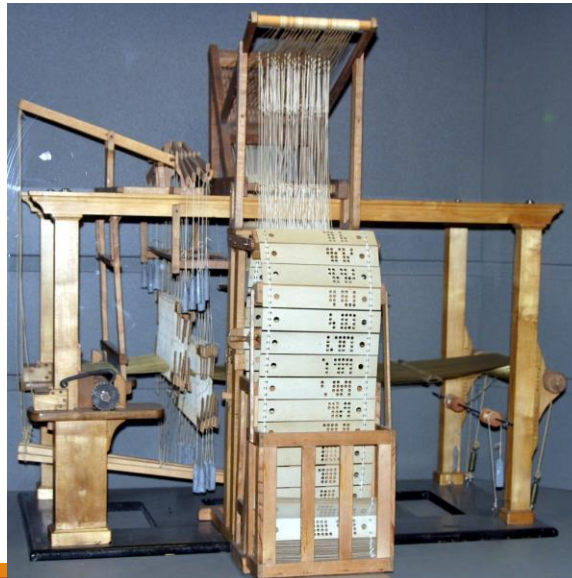
계산장치는 프로그래밍이 불가능

- 다른 계산을 수행하면 이전의 계산 설정을 잃는다.
- 재사용을 위해 프로그램이 저장되어야 한다.
- 즉, 하드웨어와 분리되어야 한다.

컴퓨터의 역사-소프트웨어

자카드 직기(Jacquard loom)

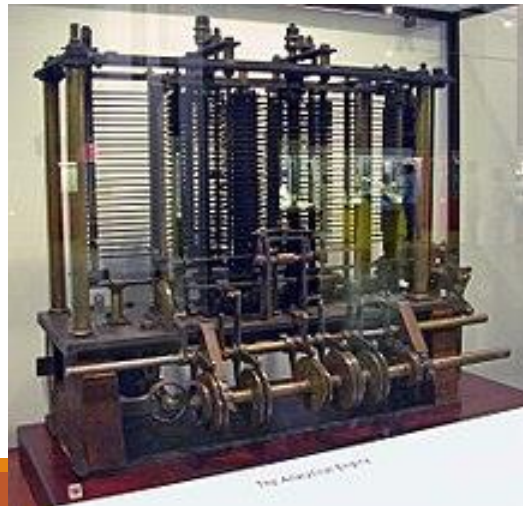
- 첫 프로그래밍이 가능한 건 천을 짜는 직기였다.
- 뽀뽀한 종이 카드 두루마기를 프로그램으로 사용
- 즉, 천공카드를 사용했다.



컴퓨터의 역사-소프트웨어

해석기관(Analytical Engine)

- 천공카드를 통해 프로그램을 하드웨어에 입력하고 저장
- 현대 컴퓨터와 동일한 방법으로 수학 연산 가능
- 그 시대에는 제조할 수 없었다.



컴퓨터의 역사-현대적 컴퓨터

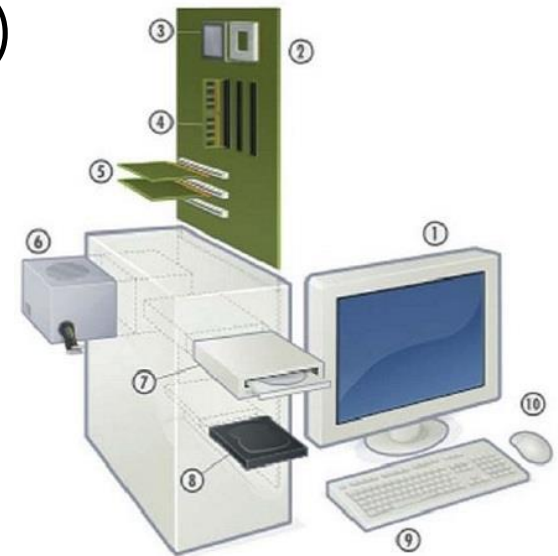
현대적인 컴퓨터(modern computer)

1. 전용 기계식이 아닌 전자식이어야 한다.
2. 아날로그가 아닌 디지털이어야 한다.
3. 내장 프로그램 개념(stored program concept)을 가진다.
 - 전자식 저장장치에 프로그램 명령어를 저장한다.
 - 다른 작업을 하게 만들기 위해 하드웨어를 바꾸지 않는다.
 - 이전에는 탄도학 계산 컴퓨터, 암호해독 컴퓨터 등이 따로 있었다면, 이 개념으로 통해 컴퓨터 하나로 여러가지를 할 수 있게 되었다.

컴퓨터의 기능 및 구조

폰 노이만 구조(현대 컴퓨터 구조)

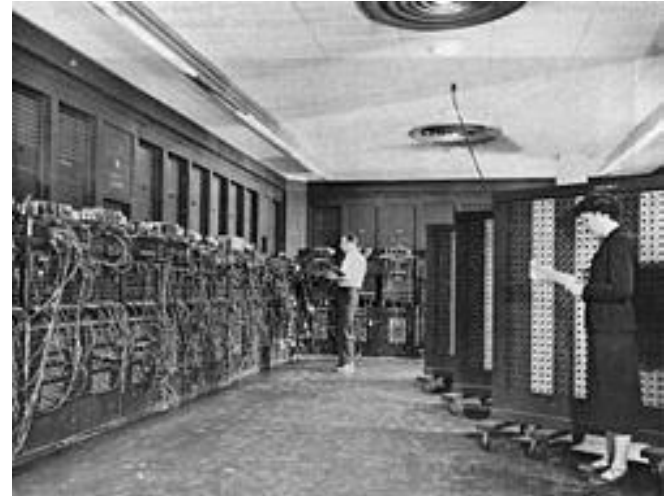
- 입력(input)장치
 - 키보드, 마우스
- 출력(output)장치
 - 모니터
- 메모리(mememory)
 - 데이터와 명령어를 저장
- 중앙 처리 장치(CPU, Central Processing Unit)
 - 데이터의 가공을 담당



컴퓨터의 역사-현대적 컴퓨터

에니악(ENIAC)

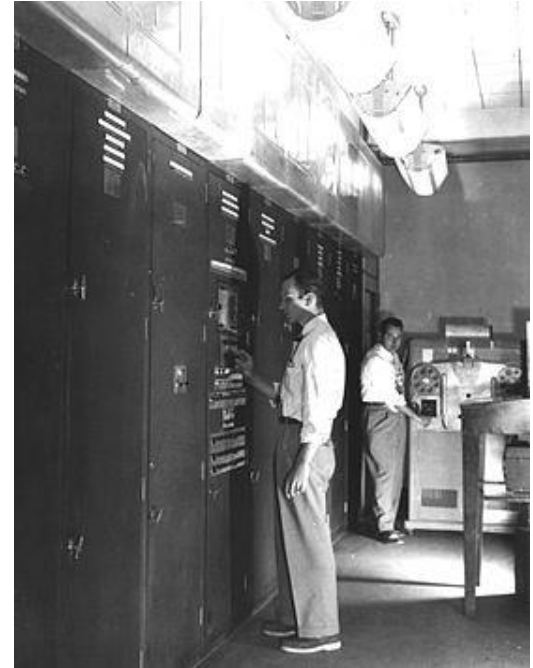
- 에커트와 모클리
- 현대적인 컴퓨터의 초기의 기준 만족
- 초기버전은 내장 프로그램이 아니었으나, 나중에 반영되었다.
- 그러나 특허권을 얻지 못한 초기 연구가 있어 특허를 무효화되어 최초의 현대적인 컴퓨터로 간주되지 않았다.



컴퓨터의 역사-현대적 컴퓨터

에드박(EDVAC)

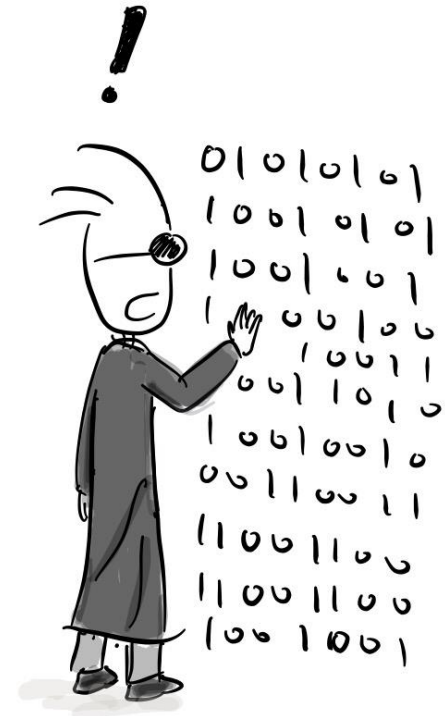
- 에커트와 모클리가 제안
- 에니악이 가지고 있지 않은 내장 프로그램 개념을 가지고 설계되었다.
- 단, 폰 노이만의 <First Draft of a Report on the EDVAC> 보고서가 폰 노이만의 이름만 단 상태에서 퍼졌고 결론적으로 폰 노이만 구조가 되었다.



프로그래밍 언어-저급언어

기계어

- CPU가 직접 해독하고 실행할 수 있는 언어이다.
- 0과 1로만 이루어진 언어이다.

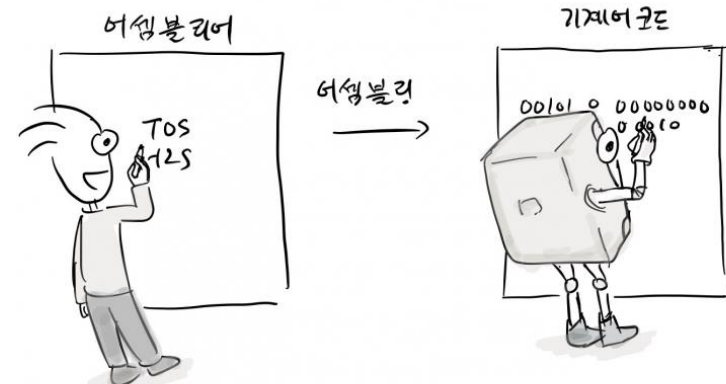


프로그래밍 언어-저급언어

어셈블리어

- 기계어와 1:1로 매칭된다.
- 어셈블러를 통해 기계로 바꾼다
- 그나마 사람이 읽을 수 있다.

| Assembly 명령어 | 설 명 |
|---|-------------------------------------|
| ADD A B | A의 내용에 B의 내용을 더해서 그 결과를 A에 저장한다. |
| SUB A B | A에서 B를 빼고 그 결과를 A에 저장한다. |
| IML A B | A와 B를 곱하고 그 결과를 A에 저장한다. |
| LEA A B | A의 값을 B의 값으로 만든다. (레지스터에서 주로 사용한다.) |
| MOVE A B | B의 값을 A로 복사한다. |
| TEST A B | A와 B를 AND 연산한다. |
| 연산 결과 값이 A에 저장되지 않지만 ZF 플래그를 설정에 영향을 준다. 연산 결과가 0이면 ZF가 1이 되고 연산 결과가 0이 아니면 ZF는 0이 된다. | |
| AND A B | A와 B를 AND 연산한다. |
| 연산 결과 값은 A에 저장되고 ZF 플래그를 설정에 영향을 준다. 연산 결과가 0이면 ZF가 1이 되고 연산 결과가 0이 아니면 ZF는 0이 된다. | |
| CMP A B | 비교구문으로 A와 B의 값이 같은지 판단한다. |
| 같은 경우 ZF는 1이 되고 다를 경우 ZF는 0이 된다. | |
| XCHG A B | A에 있는 값과 B에 있는 값을 바꾼다. |



<https://joone.net/2016/12/05/4->

[%EC%B4%88%EC%B0%BD%EA%B8%B0-](https://joone.net/2016/12/05/4-%EC%B4%88%EC%B0%BD%EA%B8%B0-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D/)

[%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D/](https://joone.net/2016/12/05/4-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D/)

프로그래밍 언어-저급언어

어셈블리어

- 폰 노이만의 일화
- 기계어와 1:1 대응하는 어셈블리어를 제자가 만들어 옴
- 어디서 컴퓨터느님의 연산속도를 깎아먹어?
- 이 일화는 사실이 아닐 가능성이 높다 한다.



John von Neumann

프로그래밍 언어-중급언어

C언어

- 훨씬 사람이 읽기 좋다.
- 고급 언어중 하나이다.
- 요즘에는 더 사용자 친화적인 언어가 많아 중급언어라 구분하기도 한다.
- 낮은 부분까지 제어할 수 있다.
- (A라는 언어를 B로 옮기는) 컴파일러를 통해 실행 파일을 만든다.

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int iAny;
6     int iNum=1;
7
8     while(iNum<10)
9     {
10         iAny=1;
11         while(iAny<10)
12         {
13             printf("%d X %d = %d\n", iNum, iAny, iAny*iNum);
14             iAny = iAny + 1;
15         }
16         iNum = iNum+1;
17     }
18     return 0;
19 }
20 }
```

프로그래밍 언어-고급언어

사람이 이해하기 쉽게 작성된 언어
가독성이 높고 생산성이 좋다.

예: 파이썬, JAVA, C#, 등등

컴퓨터 언어의 분류

컴파일 방식

- 소스코드를 컴파일하여 실행파일로 만든다.
- CPU가 실행할 수 있는 기계어로 되어있어 실행속도가 빠르다.
- 컴파일 시간이 있어 수정후 바로 볼 수 없다.
- 이식성이 낮다.
CPU/운영체제가 바뀌면 새로 컴파일해야 한다.
(단, 가상머신 기반일 경우에는 이식성이 높다.
- C, C++ 파스칼 등

컴퓨터 언어의 분류

인터프리트 방식

- 일반적으로, 전체를 읽어 실행파일을 만드는 것이 아니라 한 줄씩 읽어 파일을 실행한다.
- 인터프리터가 소스코드를 기계어로 번역함으로 실행속도가 느리다.
- 오류를 수정한 후 바로 실행하여 확인할 수 있다.
- 이식성이 높다.
인터프리터가 대상 cpu/운영체제를 지원하면 코드 변경없이 어떤 환경에서나 실행한다.
- 베이직, 파이썬, 루비, 펄 등

Python?

1989년 귀도 반 로섬(Guido van Rossum)



<Monty Python's Flying Circus>



Python의 장점

인간다운 언어이다.

```
if 4 in [1, 2, 3, 4]: print ("4가 있습니다.")
```

Python의 장점

문법이 매우 쉽다

별명: '실행할 수 있는 의사코드'
(Executable pseudocode)'

미국 최고 대학들의 컴퓨터 과학 입문과정에서
가장 인기있는 언어

Python의 장점

코드를 읽기 쉽고, 만들기 쉽다.

c언어

python

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Hello World\n");
6
7     return 0;
8 }
```

```
>>> print ("Hello World")
```

Python의 장점

ex)

```
# simple.py
```

```
languages=['python', 'c']
```

```
for lang in lanuages:
```

```
    if lang in ['python']
```

```
        print("%6s need interpreter" % lang)
```

```
    else:
```

```
        print("%6s need compiler" % lang)
```

Python의 장점

다양한 패키지

- '배터리 포함((batteries included))'
- 사용자가 바로 사용할 수 있는 라이브러리와 통합 환경이 배포판과 함께 제공

오픈 소스-무료

Python의 단점

일반적으로 C나 C++보다 느리다.
(단순히 100만까지 숫자만 출력해도 출력 속도의 차이가 보임.)

구조상 멀티스레딩 문제가 있다.

매우 무거운 프로그램은 C언어로 짜고 python에 붙여 사용한다.

실제 python의 활용도?

TIOBE Index(2018.04)

| Apr 2018 | Apr 2017 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|----------------------|---------|--------|
| 1 | 1 | | Java | 15.777% | +0.21% |
| 2 | 2 | | C | 13.589% | +6.62% |
| 3 | 3 | | C++ | 7.218% | +2.66% |
| 4 | 5 | ▲ | Python | 5.803% | +2.35% |
| 5 | 4 | ▼ | C# | 5.265% | +1.69% |

RedMonk Programming Language Rankings(2018.01)

| Jan 2018 | Jan 2017 | Change | Programming Language |
|----------|----------|--------|----------------------|
| 1 | 1 | — | JavaScript |
| 2 | 2 | — | Java |
| 3 | 3 | — | Python |
| 4 | 4 | — | PHP |
| 5 | 5 | — | C# |

Python의 철학

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Python의 철학

Pythonic

www.emmasaying.com

Python2와 Python3





현재 사용할 수 있는 python 버전은 2와 3이다.

python 2의 최신 버전은 2.7버전으로, 더 이상의 개발은 중지되었다.

python 3의 최신 버전은 3.7.1(2018.11.25) 버전이다.

Looking for a specific release?

Python releases by version number:

| Release version | Release date | Click for more | |
|------------------------------|--------------|--|-------------------------------|
| Python 3.7.1 | 2018-10-20 |  Download | Release Notes |
| Python 3.6.7 | 2018-10-20 |  Download | Release Notes |
| Python 3.5.6 | 2018-08-02 |  Download | Release Notes |
| Python 3.4.9 | 2018-08-02 |  Download | Release Notes |
| Python 3.7.0 | 2018-06-27 |  Download | Release Notes |
| Python 3.6.6 | 2018-06-27 |  Download | Release Notes |

Python2와 Python3

양쪽은 비슷하나, 차이점이 있어 코드가 호환되지 않는다.

- print문의 호출
- 유니코드 문자처리 등

본 수업에서는 python3를 사용한다.

Python

"Life is too short, You need python."
(인생은 너무 짧으니 파이썬이 필요해.)