프로그래밍 기초

컴퓨터의 구조와 개발 용어들

담당 강사 : 박종일

프로그램이란?

- 프로그램은 명령어의 집합
- 개발자가 제시한 순서에 따라 컴퓨터(cpu)가 실행해야 할 명령어 목록

- 명령어
  - 컴퓨터에게 시킬 모든 작업에 해당하는 키워드
  - 입/출력, 계산(연산)

- 프로그래밍 언어란 컴퓨터(기계)가 알 수 있는 키워드를 사용하여 컴퓨터에게 작업을 지시하기 위해 사용하는 언어
  - 컴퓨터가 알 수 있는 언어를 사용해야 함 -> 기계어(Bit)
  - 기계어는 2진법을 사용(0과 1만 사용)

컴퓨터의 역사

- 최초의 디지털 컴퓨터 콜로서스(영국) – 이미테이션 게임
  - 1943~1944년, 앨런 튜링
  - 2차 세계대전 주에 독일의 암호문 해독을 위해 개발
  - 1975년까지 1급 비밀로 공개되지 않음
  - 스위치와 플러그를 사용하여 프로그래밍 및 작동

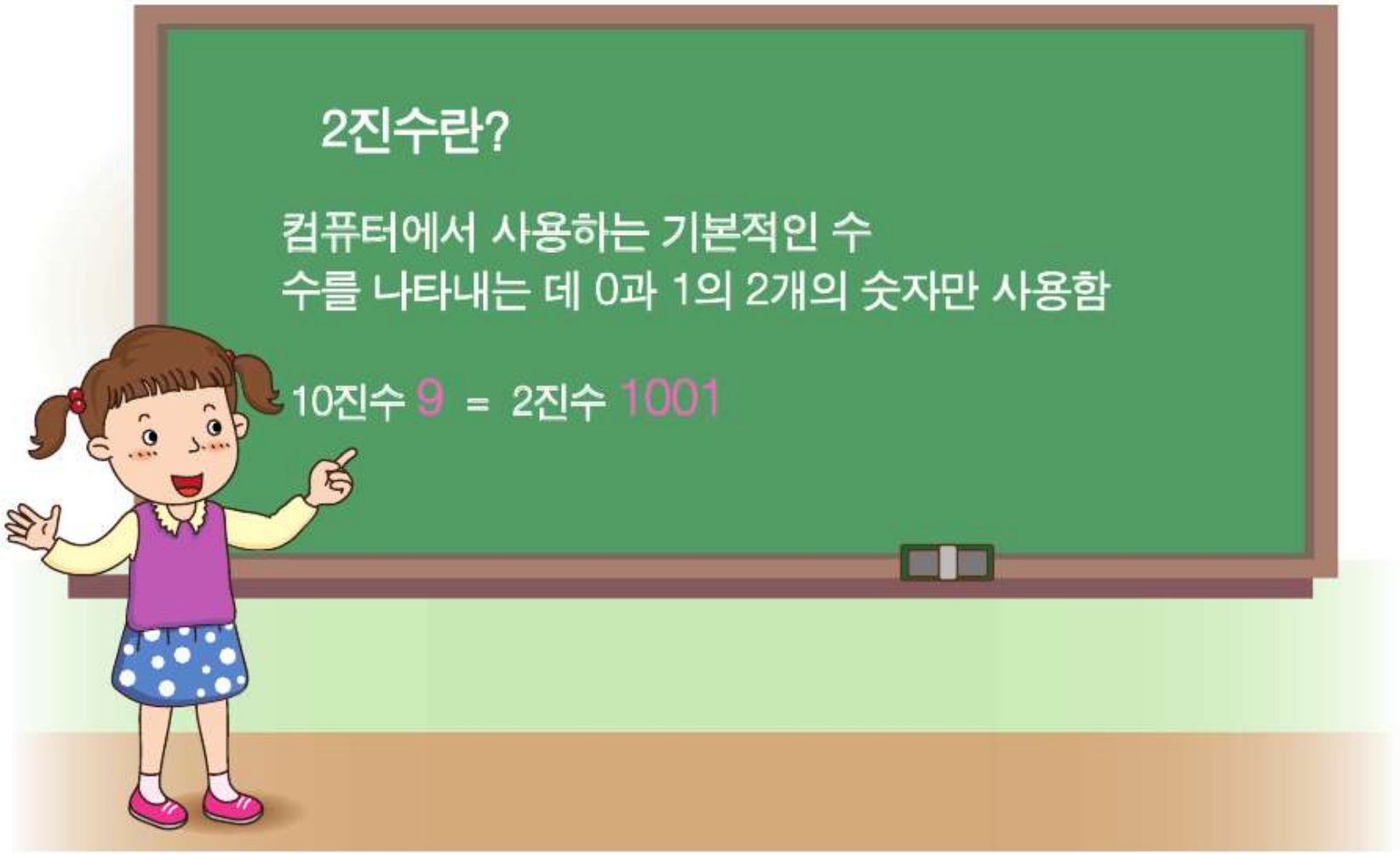
- 세간에 알려진 최초의 디지털 컴퓨터 에니악(미국)
  - 1946년, 에커트와 모클리의 공동설계
  - 탄도 계산을 위해 개발(컴퓨터라기 보다는 계산기..)
  - 배선판에 일일이 배선하는 방식(프로그램 변경은 배선판 교체)

- 두 컴퓨터의 공통점
  - 진공관(전구)을 사용 -> 2진법을 활용

참고. 수의 표현


10원이 10개이면 100원이고,
100원이 10개이면 1,000원이네.
돈은 10진법을 쓰는구나.
12개월은 1년이고,
24개월은 2년이네.
그럼 12진법이 되지.
60초는 1분이고,
60분은 1시간이네.
그럼 60진법이야.

참고. 수의 표현



Bit(비트) : 2진수로 나타내는 한자리

참고. 수의 표현

잠깐! 10진법, 2진법, 8진법, 16진법을 비교해 볼까요?

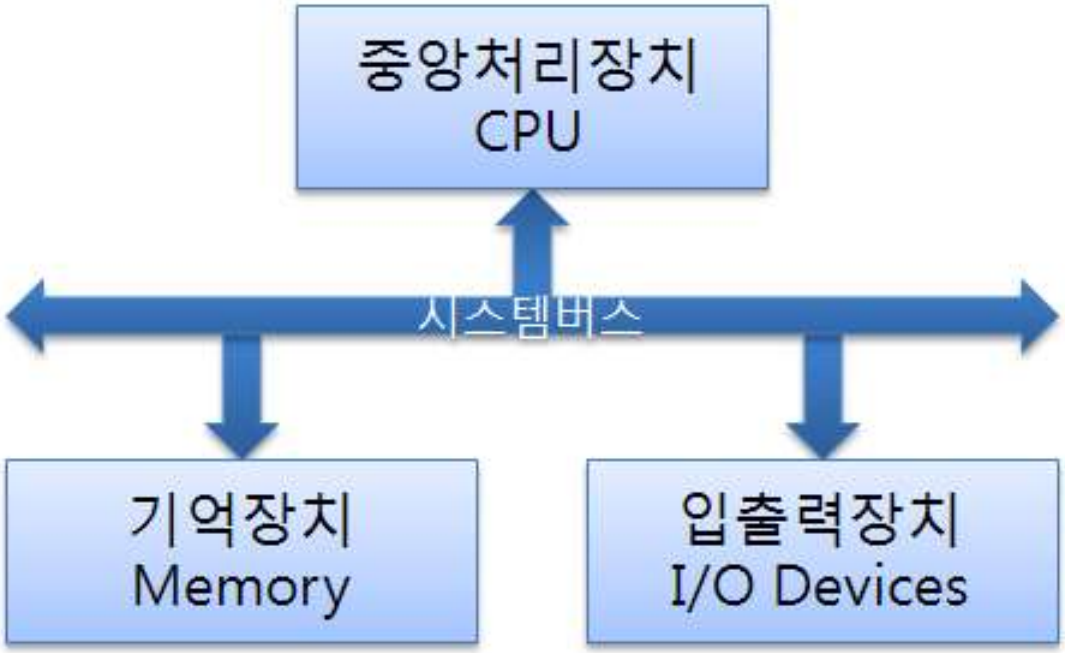
10진법	2진법	8진법	16진법
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

$$10_{(10)} = 1010_{(2)} = 12_{(8)} = A_{(16)}$$

컴퓨터의 물리적 구조

- 폰 노이만 구조
  - 존 폰 노이만 - 헝가리 출신으로 미국에서 활동한 수학자
  - 프로그램을 기억장치에 내장하는 방식의 컴퓨터를 제안

- 기본 3 요소



컴퓨터의 물리적 구조

- CPU
  - 중앙처리 장치로써 연산을 담당한다.
- Memory
  - RAM이라는 저장장치로 구성되는 메인 메모리는 컴파일이 완료된 프로그램 코드가 올라가서 실행되는 영역이다.
  - 프로그램 실행을 위해 존재하는 메모리
- System Bus
  - 컴퓨터를 구성하는 구성요소 사이에서 데이터를 주고 받기 위해 사용되는 경로
  - Address Bus, Data Bus, Control Bus 세가지로 구분된다.
  - 하드디스크, CPU, 메인 메모리등 모두가 버스에 연결되어있어 전송, 입출력을 가능하게 한다.
- Input/Output Devices
  - 기본 입력(키보드), 출력 장치(모니터)

메모리

- 폰 노이만 구조를 사용한 모든 컴퓨터(즉, 현존하는 모든 컴퓨터)에서 실행되는 모든 프로그램은 메모리(기억장치)에 내장되어 실행된다!
  - (프로그램을 구성하는) 모든 명령어는 메모리에 저장됨.
  - 입력 장치를 통해 입력된 데이터는 모두 메모리로 들어감.
  - 메모리에 저장된 모든 명령어와 데이터를 CPU로 전달됨.
  - CPU가 처리한 모든 결과는 다시 메모리에 저장됨.
  - 메모리에 저장된 결과는 출력 장치를 통해 출력됨.

- 프로그래밍 언어에서 '변수'는 프로그램에서 사용하는 데이터를 저장하는 메모리 공간을 말함.
  - 데이터를 처리하는 모든 명령은 반드시 변수가 필요함.
  - 입력이란 프로그램**에게** 데이터를 주는 행위
  - 출력이란 프로그램**이** 데이터를 주는 행위

자료(Data)와 정보(Information)

- Data
  - 현실 세계에서 벌어지는 일들을 단순한 방법으로 관찰하고 측정해서 얻은 값
  - 덧셈에서 더할 두 수
  - 한 사람의 이름, 키, 몸무게, 나이 등은 각각의 데이터

- Information
  - 데이터를 처리해서 얻을 수 있는 결과
  - 덧셈에서 더해진 두 수의 합
  - 이름, 키, 몸무게, 나이를 모두 묶으면 한 사람의 정보
  - 정보는 어떤 규칙에 따라 모으거나 추출되거나 생성된 데이터, 데이터 집합
  - 정보를 데이터로 하여 새로운 정보를 생성할 수도 있음.

자료(Data)와 정보(Information)

- 컴퓨터는 사람이 정보를 얻을 수 있도록 도와주는 시스템
- 프로그램은 컴퓨터가 사람이 정보를 얻을 수 있도록 돕기 위해 사용하는 도구.
  - 프로그램은 특정 정보를 얻기 위한 목적이 반영된 도구.
  - 즉, 프로그램은 사용하기 위한 목적이 존재.

- 프로그램은 기본적으로 자료를 취급하여 정보를 생성.
  - 사람은 프로그램에 자료를 입력하여 해당 결과인 정보를 획득.
  - 컴퓨터 시스템의 구조에 의해 프로그램이 동작하기 위해 입력 받은 자료를 저장하는 메모리 공간과 정보 출력을 위한 메모리 공간이 요구됨.
  - 이 공간을 변수라고 함.

Data의 처리

- 메모리(기억 장치)는 비어있는 공간
  - 계산을 할 때 공책에 숫자를 쓰듯이 컴퓨터의 메모리에 데이터를 쓴다.
  - 공책 = 메모리
- 공간의 크기는 유한함.
  - 최근에는 약 16~32(64)GB를 많이 사용.
  - 확장은 가능하지만 무한히 확장할 수 없음(물리적 한계)
- 이러한 공간의 관리를 위해 동일한 크기로 분할하여 사용.
  - 이 크기를 데이터의 최소 단위라고 함.
  - 이 공간의 관리는 운영체제가 담당.(토지관리 공사?)
  - 모든 공간은 공유지이며, 독점하여 사용할 수 없음.
  - 모든 공간에 일련번호를 지정하여 관리.

Data의 처리

- 메모리(기억 공간)를 사용하는 방법
  - 데이터를 저장하는 데 필요한 만큼 운영체제에 요청하고 할당 받아서 사용한 후 다시 반납.
  - 즉, 프로그램이 실행되면 운영체제는 프로그램이 필요한 만큼 할당해 주고 프로그램 종료 시 다시 수거함.
  - 공간 계산을 명확하기 하기 위해 모든 공간을 동일하게 분할

- 공간의 크기 = 단위
  - 컴퓨터의 최소 단위 – 1 bit(비트)
  - 데이터의 최소 단위 – 1 byte(바이트)
  - 1 Byte는 8 bit로 구성.
    - 즉 메모리는 8개의 bit 열을 방 한 칸처럼 취급

Data의 처리

- 변수(Variable)
  - 정해지지 않은 임의의 값을 대입할 수 있는 문자 – 수학
  - 메모리 상에 어떤 값을 저장하기 위한 이름을 가진 **공간**
  - 산술의 처리 과정
    - 컴퓨터에게 '두 수를 더해서 결과를 보여라'라는 일을 시킨다면
    - 단계 1. 입력 장치(키보드)를 통해 두 수를 입력
      - 문제 : 입력한 값은 어디(누구)에 기억하나?
    - 단계 2. CPU에서 두 수의 덧셈 연산을 수행
      - 문제 : 어디(누구)로 부터 두 수를 받아야 하나?
      - 문제 : 연산의 결과는 어디(누구)에 둬야하나?
    - 단계 3. 출력 장치(모니터)를 통해 결과를 출력
      - 문제 : 어디(누구)로부터 결과를 받아야 하나?

-> 메모리를 활용 **-> 문제 : 메모리가 너무 크다!**

Data의 처리

- 변수(Variable)
  - 매우 큰 기억 장치의 특정 공간에 데이터를 저장하고 가져오기 위해 이름을 지정해 놓은 것

5,616,336,108,329,144,529

$4DF1\ 3BEF\ 1428\ 90D1_{(16)}$

  - 이름을 지정하는 이유
    - 어딘가를 찾아가기 위해 사용하는 것 – 주소(Address)
    - 메모리는 동일한 크기로 공간을 분할하여 번호를 지정해 놓았음
      - **물리적 주소**
    - 물리적 주소는 2진수(또는 16진수)로 표현 -> 사람이 암기 불가

0100 1101 1111 0001 0011 1011 1110 1111 0001 0100 0010 1000 1001 0000 1101 0001

    - 프로그램이 동작 중일 때 메모리의 특정 위치에 데이터를 저장했다가 가져와서 사용하도록 컴퓨터에게 명령해야 하는데 물리적 주소로는 이를 처리하기 매우 어렵다.
      - 101번째 공간을 'a'라고 할 테니, 앞으로 'a'에 넣고 빼라고 하면 101번째 공간을 활용해라.
    - 이 이름을 '논리적 주소', **변수**라고 한다.

Data의 처리

- 변수(Variable)
  - 컴퓨터의 운영체제는 CPU의 데이터 처리량에 따라 달라짐
  - x86 vs x64(Intel CPU)
    - x86 : 일반적으로 32bit CPU를 지칭
      - 한번에 처리할 수 있는 데이터의 크기가 32bit
    - x64 : 64bit CPU를 지칭
      - 한번에 처리할 수 있는 데이터의 크기가 64bit
  - 주소도 데이터!
    - 32bit CPU는 사용할 수 있는 주소의 범위가 좁다.
    - 주소의 범위가 좁다는 것은 사용할 수 있는 메모리의 양이 작다는 것을 의미
    - 현존 컴퓨터는 64bit로 동작하며, 이 때 사용하는 메모리 주소의 길이는 2^64이다.
  - 길고 처리가 어려운 실제 주소(물리적 주소)는 컴퓨터가 내부적으로 알아서 하고 개발자는 변수를 사용하여 공간을 활용

프로그램의 실행

- (사람의 말?로..)명령어 작성(코딩)
- 모든 명령을 기계어로 번역(컴파일)
- 프로그램 실행에 필요한 (함께 작성하지 않은) 다른 기계어 명령과 지금 컴파일된 기계어 명령을 연결(링크)
- 모든 명령어 실행

프로그램의 실행

- 프로그래밍 언어
  - 진공관(전구)에서 시작한 디지털 컴퓨터는 사용할 수 있는 신호(또는 데이터)가 0과 1 뿐임.
  - 이것은 바이너리(Binary)라고 함.
  - 1 bit란 0 또는 1을 저장하는 한 공간.
  - 1 bit로는 on/off 명령 밖에 줄 수 없음.
  - 여러 개의 bit를 묶어서 하나의 의미를 부여 -> 기계어
  - 8개의 bit를 묶어서 사용 -> 1 byte

- 사람이 bit열로 구성된 기계어를 사용하기는 어려움.
  - 사람이 사용하는 언어(영단어)를 사용하여 명령어를 작성.
  - 컴퓨터가 번역을 수행하여 프로그램으로 완성.
  - 이 과정을 컴파일(Compile)이라고 함.

개발 용어(from 인프런)

- **서버(Server) vs. 클라이언트(Client)**
  - 서비스를 제공하느냐/제공받느냐에 따른 구분.
  - 외부에 필요한 서비스를 제공하도록 만든 컴퓨터나 프로그램 영역을 서버라고 한다면, 여기에 접속해 서비스를 요청하고 제공받는 유저가 클라이언트가 된다.
- **프론트엔드 개발자(Front-end Developer)**
  - 웹 브라우저를 통해 유저가 직접 마주하는 웹 서비스의 앞단(front-end)을 담당하는 개발자.
  - 클라이언트/서버를 기준으로 보면 웹 페이지 화면을 비롯한 클라이언트 영역을 프론트엔드라고 할 수 있다.

개발 용어(from 인프런)

- **백엔드 개발자(Back-end Developer)**
  - 프론트엔드 개발자의 반대 개념으로, 웹 서비스의 뒷단(Back-end)을 담당하는 개발자.
  - 주로 유저에게 보이지 않는 DB와 API로 이루어진 서버 영역을 관리, 개발하는 역할을 한다.
- **풀스택(Full-Stack)**
  - 프론트엔드 개발과 백엔드 개발 영역을 통틀어 이르는 말, 혹은 모두 다룰 수 있는 개발자.

개발 용어(from 인프런)

- **프레임워크(Framework) - 조립식 주택**
  - 프로그램을 개발하기 위한 구조를 제공하는 개발 환경.
  - 코딩을 할 때 자주 쓰이는 여러 코드를 프레임워크가 정해 둔 흐름에 맞춰 쓸 수 있도록 지원해 주기 때문에 복잡하거나 반복되는 작업에 대한 부담을 덜어줄 뿐만 아니라 개발에 들이는 시간 대비 프로그램의 질을 높일 수 있다.
- **라이브러리(Library) - 필수인테리어**
  - 프로그램을 개발하는 데 필요한 여러 기능을 활용할 수 있도록 묶어놓은 함수 또는 기능의 집합.
  - 프레임워크가 개발을 제어하는 틀을 제공해준다면, 라이브러리는 도서관에서 책을 빌리듯 개발자가 필요한 기능을 마음대로 가져다 쓸 수 있는 일종의 모음집에 가깝다.
  - 즉 정해진 방식으로 개발을 하게끔 정해주는 프레임워크와 달리 제어권이 개발자에게 주어지는 셈.

개발 용어(from 인프런)

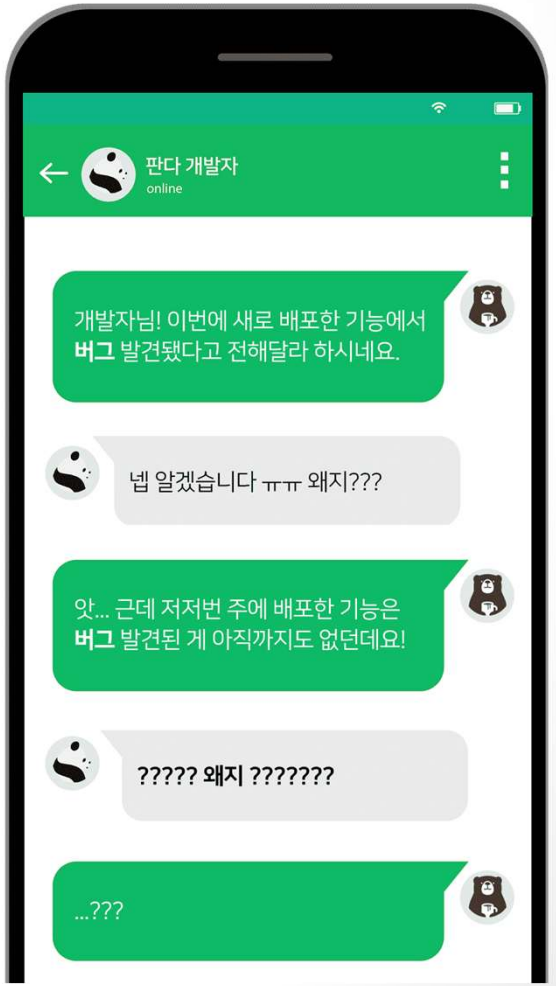
- **API(Application Programming Interface)**
  - 응용 프로그램 인터페이스.
  - 어떤 응용 프로그램에서 특정한 기능을 사용하기 위해 필요한 데이터를 주고받게끔 만든 도구나 방법을 뜻한다.
  - API가 규격에 맞게 데이터를 요청하고 받아볼 수 있도록 하는 중간 창구 역할을 하는 셈이다.
- **오픈소스(Open Source)**
  - 누구나 제한 없이 쓸 수 있는 소스 코드 혹은 소프트웨어.
  - 오픈 소스로 배포된 코드는 열람, 수정, 복제, 재배포 등이 자유롭다.
  - 한국에서는 공공데이터포털(data.go.kr)을 통해 공개된 오픈 API를 내려받을 수 있다.
  - 구글 API, 카카오 API, 네이버 API 등.

개발 용어(from 인프런)

- **플러그인(Plug-in)**
  - 어떤 특정한 기능을 해결하는 데 쓸 수 있도록 미리 만들어 놓은 코드 및 데이터의 모음.
  - 한 라이브러리나 프레임워크에서 여러 개의 플러그인을 제공한다고 볼 수 있다.
- **마크업 언어(Markup Language)**
  - 태그 등을 이용해 문서나 데이터의 구조를 표시한 언어.
  - 프로그래밍 언어와는 다르며, HTML, CSS, XML이 대표적인 마크업 언어다.

개발 용어(from 인프런)

- **버그(Bug)**
  - 소프트웨어에서 발생하는 예견치 못한 오류나 오작동.
  - 있어도 이상하고 없어도 이상한 것.
- **에러(Error)**
  - 유저가 입력한 내용이 잘못됐을 때 발생하는 문제.
  - 잘못 짠 코드처럼 내부 문제로 예상하지 못한 문제가 일어나는 버그와 달리, 유저가 오탈자를 내거나 띄어쓰기를 실수하는 등 정한 형식을 따르지 않을 때 주로 발생한다.
- **예외(Exception)**
  - 에러가 일어날 가능성을 개발자가 미리 예상하고 프로그램이 비정상적으로 종료되지 않도록 하는 것.



개발 용어(from 인프런)

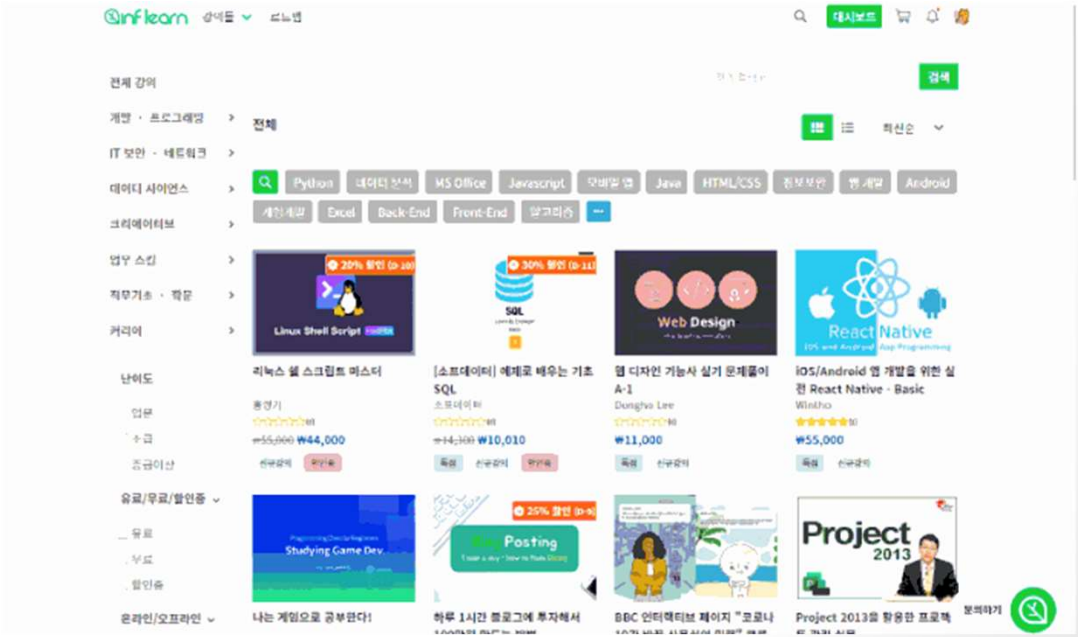
- **SDK(Software Development Kit)**
  - 소프트웨어나 시스템을 만드는 데 쓰이는 개발 도구 키트.
  - SDK 안에는 개발에 필요한 샘플 코드, 코드 편집기 같은 툴이나 콘솔, 안내 문서, API 등이 포함된다.
- **IDE(Integrated Development Environment)**
  - 프로그램을 개발하는 데 필요한 소스 코드 작성 및 편집, 컴파일, 디버깅 등 모든 작업을 한번에 할 수 있는 통합 개발 환경.
  - 비주얼 스튜디오(Visual Studio), 이클립스(Eclipse), 인텔리제이 아이디어(IntelliJ Idea) 등.

개발 용어(from 인프런)

- **정적 웹 페이지(Static Web Page)**
  - 서버에 미리 저장된 데이터를 그대로 불러오듯 전달하는 웹 페이지.
  - 서버의 데이터가 바뀌지 않는 한 모든 사용자는 늘 고정된 웹 페이지를 받아보게 된다.
  - 주로 서비스 소개페이지처럼 내용이 자주 변경되지 않는 경우 정적 웹 페이지를 이용하는 경우가 많다.
- **동적 웹 페이지(Dynamic Web Page)**
  - 서버에 저장된 데이터를 그대로 보여주는 것이 아니라 스크립트를 통해 가공을 거쳐 출력하는 웹페이지.
  - 즉 사용자의 인터랙션에 따라 같은 페이지라도 각기 다른 결과를 받아볼 수 있게 된다.

개발 용어(from 인프런)

- **적응형 웹(Adaptive Web Design)**
  - 미리 웹 브라우저가 동작할 기기(스마트폰, PC, 태블릿...) 별로 레이아웃을 여러 개 정해놓고 조건에 따라 그 중 하나를 보여주는 웹 디자인 방식.
- **반응형 웹(Responsive Web Design)**
  - 적응형 웹과 달리 하나의 레이아웃이 웹 브라우저에 맞춰지는 형태로, 웹 브라우저의 가로폭이 바뀔 때마다 페이지 내에 있는 콘텐츠의 크기와 배치도 자동으로 맞춰진다.

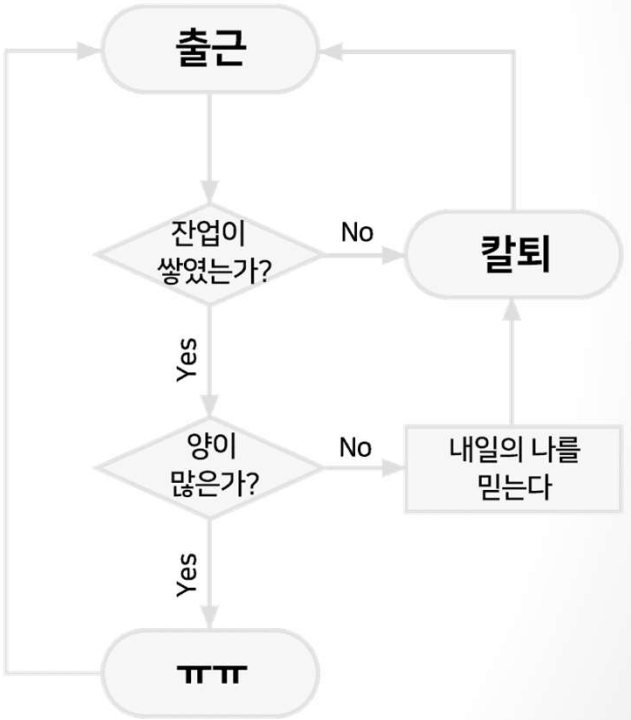


개발 용어(from 인프런)

- **UX(User eXperience)**
  - 사용자가 서비스를 이용하며 느끼고 얻는 경험.
- **UI(User Interface)**
  - 사용자가 서비스를 이용하는 환경.
  - UI는 사용자 경험(UX)에 큰 영향을 미친다.
- **GUI(Graphic User Interface)**
  - 그래픽 유저 인터페이스.
  - 사용자가 화면 안의 요소를 시각적으로 확인하고, 마우스로 아이콘을 클릭해 기능을 작동시키는 환경.
- **모달(Modal Window)**
  - UI 디자인 요소로, 웹페이지나 프로그램 화면에서 대화상자 형태로 나타나 사용자에게 동작을 요구하는 창.
  - 웹에서는 팝업(Pop-up) 창과 비슷하지만, 브라우저를 새로 띄우는 팝업과 달리 한 브라우저 화면 내부에서 레이어를 한 겹 얹듯이 새로운 창을 보여준다.

개발 용어(from 인프런)

- **알고리즘(Algorithm)**
  - 문제를 해결하기 위해 필요한 계산 절차.
  - 프로그래밍 언어를 통해 알고리즘을 프로그램으로 만들어가는 작업을 프로그래밍이라고 한다.
- **자료 구조(Data Structure)**
  - 데이터를 효과적으로 접근할 수 있도록 만들어진 데이터 체제
  - 코드상에서 자료를 저장하는 방법, 자료끼리의 관계 등을 구조적으로 표현하는 방식.
  - 어떤 자료구조를 쓰느냐에 따라 코드 효율이 달라진다.
  - List(목록), Map(맵) 등



개발 용어(from 인프런)

- **컴파일(Compile), 컴파일러(Compiler), 컴파일드 언어(Compiled Language) – 번역본**
  - 사람이 이해할 수 있는 코드를 컴퓨터가 처리할 수 있는 기계어로 **한꺼번에** 번역하는 과정을 컴파일(Compile)이라고 한다.
  - 이때 쓰이는 프로그램 또는 시스템이 컴파일러다.
  - 수정이 까다롭지만 프로그램 실행 속도가 빠르다.
- **인터프리트(Interpret), 인터프리터(Interpreter), 인터프리티드 언어(Interpreted Language) – 원서의 독해**
  - 인터프리트(interpret)는 컴파일과 달리 프로그램을 실행할 때마다 고급 프로그래밍 언어로 작성된 코드를 한 줄씩 기계어로 번역하는 방식이다.
  - 한 줄 한 줄씩 코드를 실행해내려가기 때문에 실행 속도는 느리지만 디버깅은 쉽다.
  - Javascript, Python, PHP 등이 인터프리티드 언어에 속한다.

개발 용어(from 인프런)

- **디버깅(Debugging)**
  - 디버깅 또는 디버그(Debug).
  - 프로그래밍 과정에서 발생하는 버그를 찾아 바로잡는 작업.
  - 단순히 버그를 없애는 것뿐만 아니라 문제가 발생한 근본적인 원인을 찾아 해결하는 과정이다.
- **빌드(Build), 빌더(Builder) 또는 빌드 도구**
  - 기계어(machine code)로 번역하여 컴퓨터에서 이해할 수 있는, 즉 실행 가능한 파일로 만드는 과정
  - 빌드 도구는 빌드 작업을 자동화하기 위한 프로그램.
  - 링킹(Linking. 관련 라이브러리 포함), 컴파일링, 패키징 등을 실행(처리)
  - Maven, Gradle 등

개발 용어

- **함수(Function)**
  - 프로그래밍에서 함수(function)란 하나의 특별한 목적의 작업을 수행하기 위해 독립적으로 설계된 프로그램 코드의 집합으로 정의할 수 있습니다. - 위키백과
  - 프로그래밍에서의 함수는 프로그램 소스 코드에서 일정한 동작을 수행하는 코드(들)을 말한다. - 나무위키
  - 프로그램이 동작할 때 함수의 호출 과정은 다음과 같다.
    - 코드상에서 함수가 호출된다. 이때 변수를 넘겨주기도 하는데 이를 매개변수 또는 인수(parameter)라고 한다.
    - 호출된 함수의 코드가 동작한다.
    - 함수의 동작이 끝나면 함수를 종료하고 경우에 따라 반환값을 코드에 반환한다.

개발 용어

- **객체(Object)**
  - 컴퓨터 과학에서 객체 또는 오브젝트(object)는 클래스에서 정의한 것을 토대로 메모리(실제 저장공간)에 할당된 것으로 프로그램에서 사용되는 데이터 또는 식별자에 의해 참조되는 공간을 의미하며, 변수, 자료 구조, 함수 또는 메소드가 될 수 있다. – 위키백과
  - 객체란 하나의 역할을 수행하는 '메소드와 변수(데이터)'의 묶음으로 봐야 한다. – 나무위키
  - 자바스크립트의 기본 타입(data type)
    - 이름(name)과 값(value)으로 구성된 프로퍼티(property)의 정렬되지 않은 집합
    - 프로퍼티의 값으로 함수가 올 수도 있는데, 이러한 프로퍼티를 메소드(method)

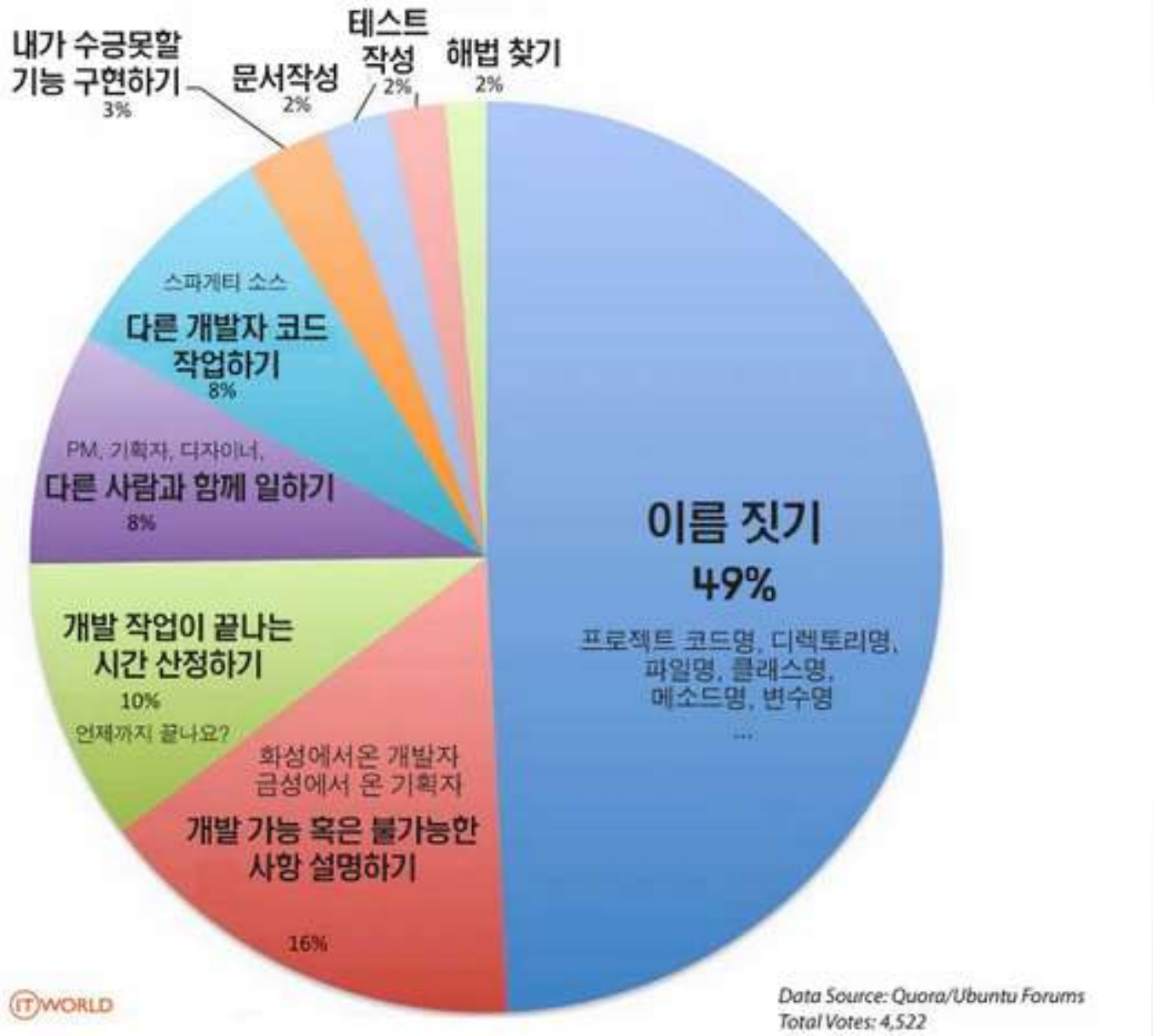
개발 용어

- **루틴(Routine)**
  - 어떤 프로그램이 실행될 때 불려지거나 반복해서 사용되도록 만들어진 일련의 코드들
  - 메인 루틴
    - 프로그램의 주된 흐름에 해당하는 루틴
  - 서브 루틴
    - 간단한 작업을 처리하기 위한 부분 루틴
    - 메인 루틴에서 사용(호출, Call)
- **모듈(Module)**
  - 프로그램의 기능을 독립적인 부품으로 분리한 것을 모듈이라고 한다.
  - 일반적으로 서브루틴과 데이터 구조의 집합체로서, 그 자체로서 컴파일 가능한 단위이며, 재사용 가능하고 동시에 여러 다른 모듈의 개발에 사용될 수 있다.

개발 용어

- **유효범위, 스코프(Scope)**
  - 변수는 자신이 선언된 위치에 의해 자신이 유효한 범위, 즉 다른 코드가 변수 자신을 참조(활용)할 수 있는 범위가 결정된다.
  - 모든 식별자(변수 이름, 함수 이름, 클래스 이름 등)는 자신이 선언된 위치에 의해 다른 코드가 식별자 자신을 참조할 수 있는 유효 범위가 결정된다.
    - 서울에 사는 홍길동과 부산에 사는 홍길동은 다른 사람.
  - 코드의 모든 부분에서 사용 가능함 – 전역(Global)
  - 특정 영역 내부에서만 사용 가능함 – 지역(Local)
- **카멜케이스(Camel Case)**
  - 낙타의 형태와 유사하여 이름이 붙은 표기형식
  - 컴퓨터가 띄어쓰기를 인식하지 못하기 때문에 두 단어 이상을 조합한 식별자를 작성할 경우 두번째 또는 그 다음 단어의 첫글자를 대문자로 작성하는 형식
  - public data -> publicData

프로그래머가 가장 힘들어하는 일은?



당부하고 싶은 말들...

- 규칙에 맞게 작성하자. (문법, 순서, 범위)
  - 모르겠거나 헷갈리면 **외우자!**
  - 관습도 따르자.(대부분의 개발자는 이렇게 합니다)
- 처음부터 프로그램의 모든 부분을 코딩하는 것은 어렵다.
  - 프로그램의 기능을 생각하여 작은 부분으로 분할하자.
  - 모든 프로그램은 크게 3가지로 구성된다.
    - 입력, 처리, 출력
- 프로그램 코드는 재사용하자.
  - 복사 붙여넣기 보다는 모듈화!
  - 오픈 소스 코드는 내 코드다!
    - 100시간에 100% 내가 해서 1개의 프로그램을 만든 것보다 같은 시간에 10% 내가 해서 10개의 프로그램을 만든 것이 인정받는다.
  - 구글링의 생활화!(검색 많이 하세요~)
- 작성한 코드(프로그램)는 잘 정리해 둘 것!(재산)