Ch1. 파이썬 소개

프로그래밍 기초 및 실습

- 프로그래밍 언어
  - 동작시키는 프로그램을 작성하기 위한 인공적인 언어

- 프로그래밍 언어의 분류
  - 사용하기 편리한 정도에 따라 분류

분류	종류
저급언어	기계어, 어셈블리어
고급언어	C, C++, Java, Python 등 대부분의 언어

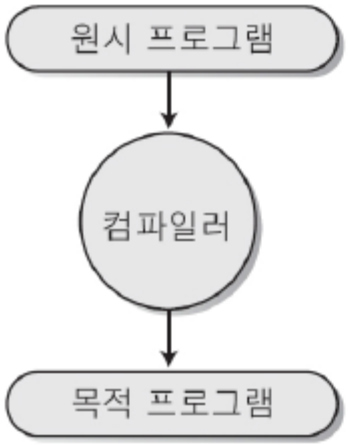
  - 저급 언어
    - 하드웨어 지향의 기계 중심 언어로, 하드웨어와 밀접한 기능 제어

```
mov ax, X
add ax, Y
mov Z, ax
```

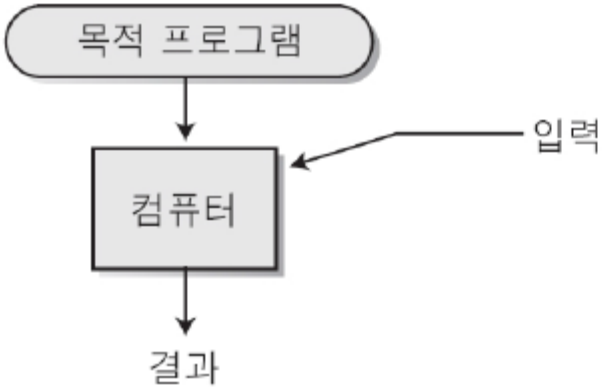
  - 고급 언어
    - 컴퓨터 기종에 따라 다르게 표현되는 저급 언어의 문제점 해결
    - 사람이 사용하는 기호 체계와 유사

```
Z = X + Y;
```

컴파일 기법: C,C++등



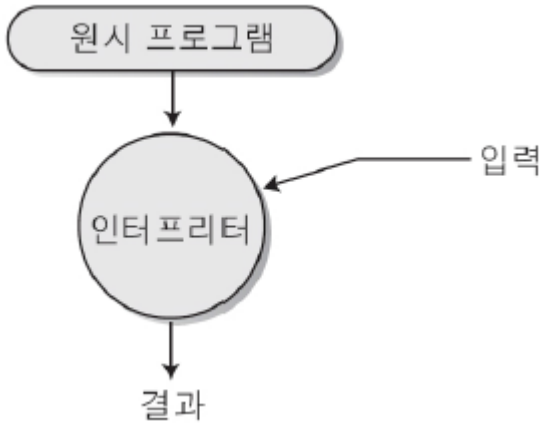

```
int main(void) {
    printf("Hello World\n");
    return 0;
}
```




```
$ gcc helloworld.c
$
```

```
$ a.out
Hello World
$
```

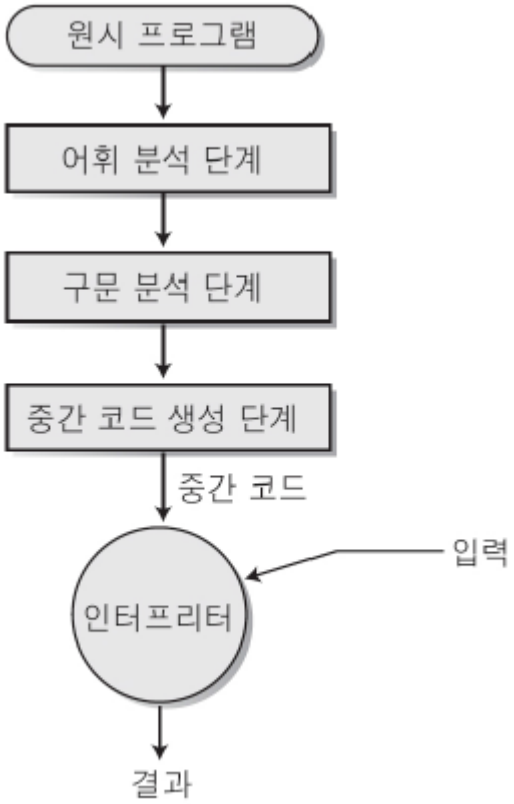
해석 기법: Scheme, Perl, Php 등



- 고급 언어로 작성된 프로그램을 바로 실행
  - 해석하는 프로그램 : 인터프리터(interpreter)
- 인터프리터
  - 고급 언어를 자신의 기계어로 취급하는 컴퓨터를 시뮬레이션 한 것
  - Scheme 인터프리터 예

```
=> (* 2 3 4)
Value: 24
```

하이브리드 기법: Java, Python등



- 컴파일 기법과 해석 기법을 혼합한 형태
  - 고급 언어로 작성된 프로그램을 쉽게 해석할 수 있도록 중간 코드 형태로 번역
  → 번역된 중간 코드 형태의 프로그램을 해석하여 실행

  – 예 : Java
  - 바이트 코드(byte code)라고 하는 중간 코드로 번역
  - 운영체제마다 별도로 존재하는 자바 가상 기계 (JavaVirtual Machine)가 바이트 코드를 실행

    - [예] 원시 프로그램을 바이트 코드로 번역

```
$javac HelloWorld.java
$
```

    - [예] 바이트 코드를 실행

```
$java HelloWorld
Hello World
$
```

- **패러다임**
  - 계산의 본질을 보는 관점에 따라 프로그래밍 언어를 분류
  - 명령형 언어, 함수 언어, 논리 언어, 객체지향 언어

- **명령형 언어: C, Pascal 등의 대부분 언어**
  - 폰 노이만 구조라 불리는 컴퓨터 구조를 기반

  • 명령의 순차적인 실행, 메모리 위치를 의미하는 변수의 사용, 변수의 값을 바꾸는 배정문의 사용

  • 변수 x와 y를 사용하고, 배정문 '='을 사용하며, 위에서부터 아래로 순차적으로 실행

```
int main(void) {
    int x, y;
    x = 10;
    y = x + 20;
    return 0;
}
```

- **객체지향 언어: C++, Java, Python 등**

  – **명령형 언어, 함수 언어, 논리 언어 어느 것과도 결합이 가능**

  – C++와 같이 명령형 언어를 확장한 객체지향 언어가 일반적

  – 객체(object) 단위로 모든 처리를 기술

  – 객체는 데이터들과 연관된 동작들이 하나로 묶인 개념

  – 객체지향 언어는 추상 데이터 타입, 상속, 동적 바인딩 개념을 지원

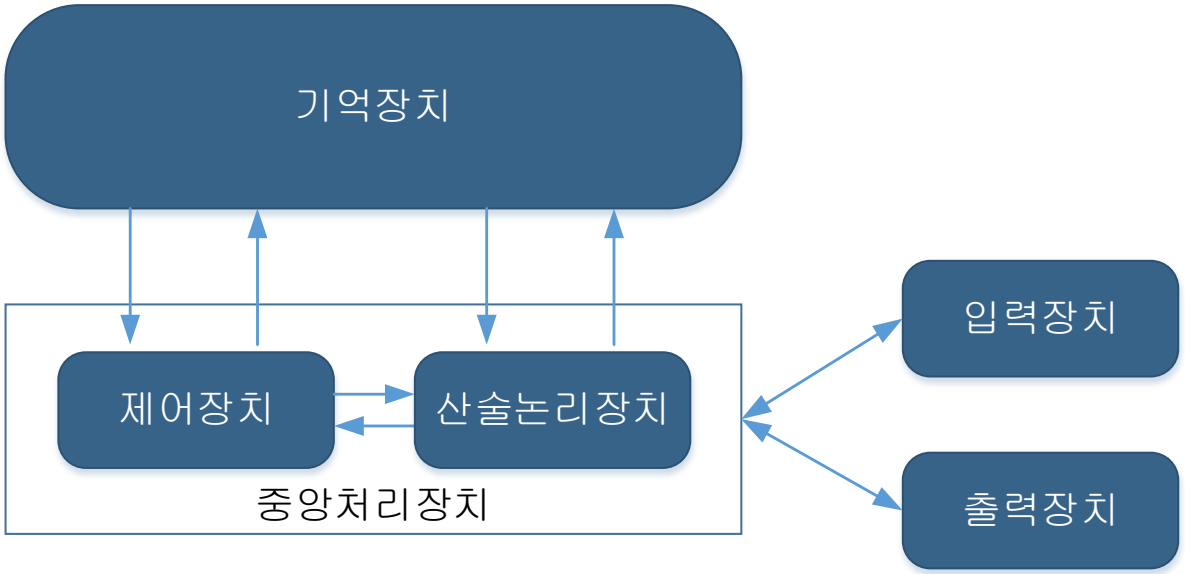
  – C++ 예

```
class employee {
private:
        char *name;
public:
        employee(char *na) {
                name = new char[strlen(na)+1];
                strcpy(name, na);
        }
        ~employee() {
                delete []name;
        }
        char* getName() {
                return name;
        }
};

employee emp("abc");
```

- **폰 노이만 구조**
  - 중앙처리장치
  - 기억장치
  - 입력과 출력
- **운영체제와 애플리케이션**
- **소프트웨어는 무엇으로 만드는가**
- **파이선 프로그래밍 언어**

- **폰 노이만 구조**
  - 오늘날의 컴퓨터들은 그 모습과 크기, 성능은 다를지 모르지만 기본적인 구조는 거의 똑같음.
  - 1945년에 존 폰 노이만이 발표한 논문 <EDVAC에 관한 보고서>에서 기술한 컴퓨터의 구조를 그대로 따르고 있기 때문



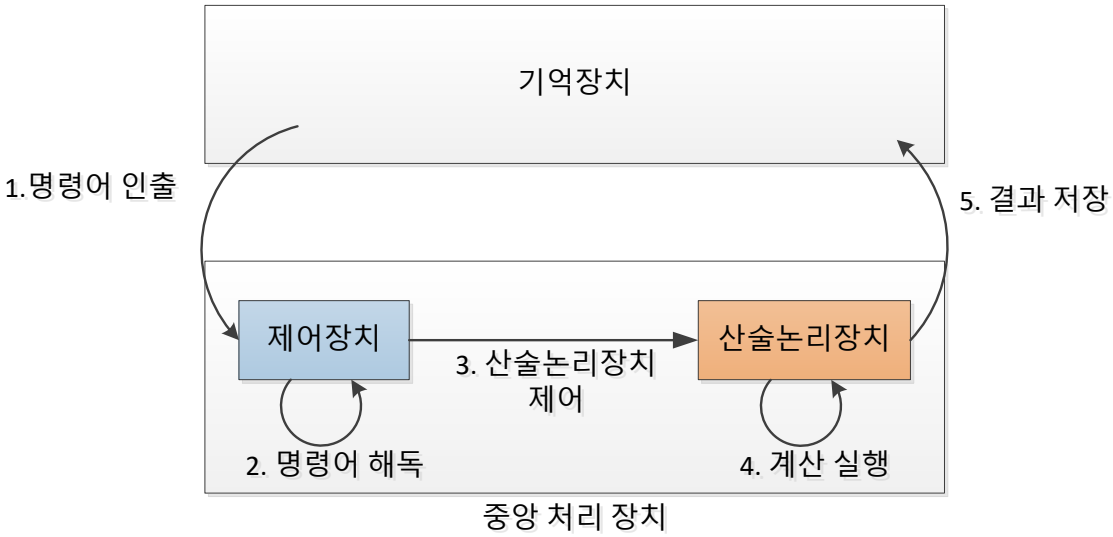
- **폰 노이만 구조**
  - EDVAC은 프로그램을 수정하는 일이 ENIAC에 비해 자유로웠음
    - EDVAC은 명령어를 기억장치에 내장하고 있었기 때문임
    - 이렇게 기억장치에 명령어를 내장하는 컴퓨터를 **내장식 컴퓨터(Stored-program computer)**라고 함
    - 내장식 컴퓨터는 폰 노이만의 논문에서 본격적으로 소개되었기 때문에 **폰 노이만 구조(von Neumann Architecture)**라는 이름으로 알려져 있음
  - PC의 폰노이만 구조
    - 입력 장치 : 키보드와 마우스
    - 출력 장치 : 모니터와 프린터
    - 중앙처리장치 : 인텔 i7
    - 기억장치 : DDR3 DRAM

- **중앙처리장치**
  - 컴퓨터가 수행하는 계산은 모두 이 CPU를 통해 이루어짐
    (최근에는 그래픽 카드의 처리장치를 활용하기도 함)
  - CPU를 이루는 요소
    - 산술 논리 장치(ALU : Arithmetic Logic Unit)
    - 제어 장치(CU : Control Unit)

- **산술논리장치**
  - 덧셈, 뺄셈, 곱셈, 나눗셈 등의 산술 연산과 참과 거짓(또는 0과 1)을 다루는 논리 연산을 수행하는 회로를 가짐.

- **제어장치**
  - 산술논리장치를 통제하여 계산을 수행함

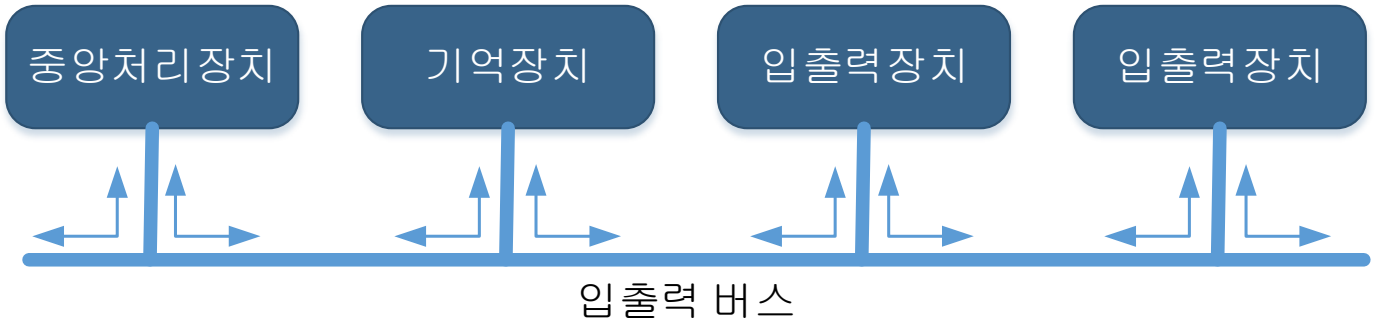
- **제어장치와 산술논리장치의 동작 과정**
  1. 먼저 제어장치가 기억장치로부터 명령어를 가져옵니다 **(Fetch)**
  2. 제어장치는 가져온 명령어를 해독**(Decode)**합니다
  3. 제어장치는 해독한 명령어에 따라 산술논리장치에 데이터를 옮기고 어떤 연산을 수행할지를 지시합니다
  4. 산술논리장치는 제어장치가 지시한 대로 계산을 **수행(Execute)**합니다
  5. 수행한 결과를 기억장치에 다시 **저장(Store)**합니다



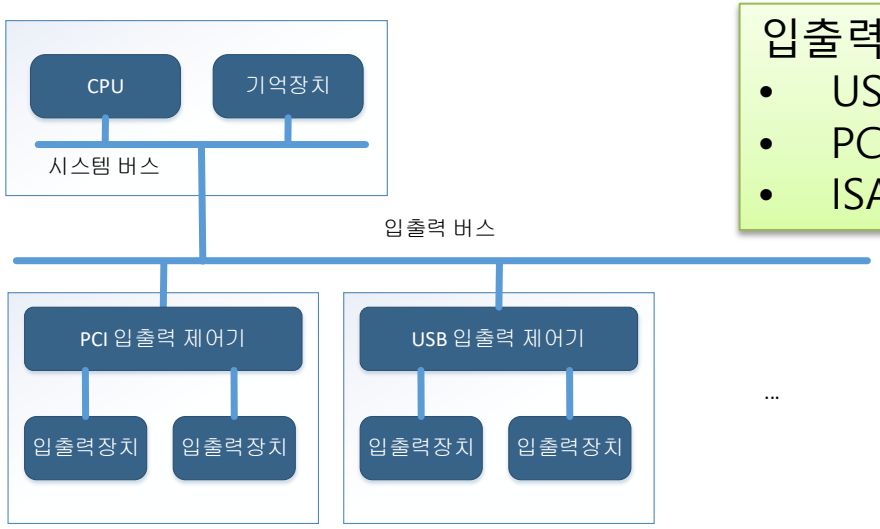
- **기억장치(Memory) : 데이터와 함께 명령어를 저장하는 역할 수행.**
- 메모리에 데이터를 기록하기 위해서는 CPU가 "쓰기 요청"을 보내고 반대로 메모리로부터 데이터를 가져오기 위해서는 "읽기 요청"을 보냄.
  - 메모리의 성능은 이 요청에 응답하기까지의 시간과 이러한 읽기/쓰기 요청을 연속적으로 처리하는 주기에 의해 결정
- **캐시(Cache) 메모리**
  - **CPU와 가장 가까이에 있는 기억장치. 성능이 좋지만 가격이 비쌈**
- **주기억 장치**
  - **캐시 메모리 아래에 위치한 기억장치. 대개 램(RAM)으로 구성**
- **보조기억장치**
  - **CPU로부터 가장 멀리 떨어져 있는 기억장치. 성능이 낮지만 가격도 낮음**
    하드디스크가 대세였지만, 메모리기술이 발전하면서 플래시 메모리와
    SSD(Solid State Drive)가 부상

- **입출력장치는 컴퓨터와 바깥세계와 소통하는 수단.**
  - 입력장치의 예)
    - 마우스, 키보드, 터치스크린, 마이크, 게임패드, 동작 인식 장치
  - 출력장치의 예)
    - 모니터, 프린터, 3D 프린터, 스피커, 빔 프로젝터, E-Ink 전자 종이

- **입출력 버스(I/O BUS)**
  - 버스 : 컴퓨터의 정보 전송 회로
  - 중앙처리장치와 기억장치, 그리고 입출력 장치는 아래와 같이 버스로 연결되어 있음

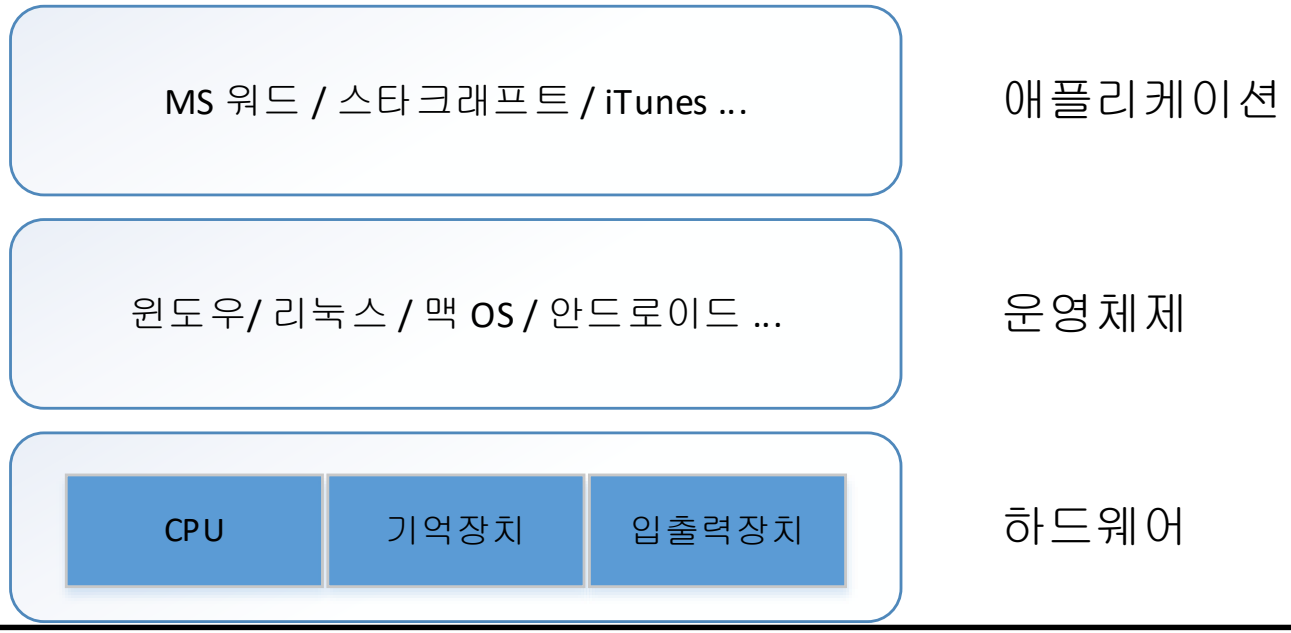


- **입출력 버스의 문제점**
  - 중앙처리장치, 기억장치, 입출력장치가 동일한 입출력버스를 사용함으로써 빠르게 동작하는 중앙처리장치가 가장 느린 입출력 장치 때문에 제 성능을 낼 수 없는 문제가 발생
  - CPU가 빠른 속도로 진화하면서 더욱 문제가 불거짐
- **개선된 입출력 버스**
  - CPU와 기억장치는 시스템 버스(System Bus)로 묶고, 그 다음 다양한 입출력장치들은 입출력 버스로 묶어 CPU의 입출력 모듈에 연결

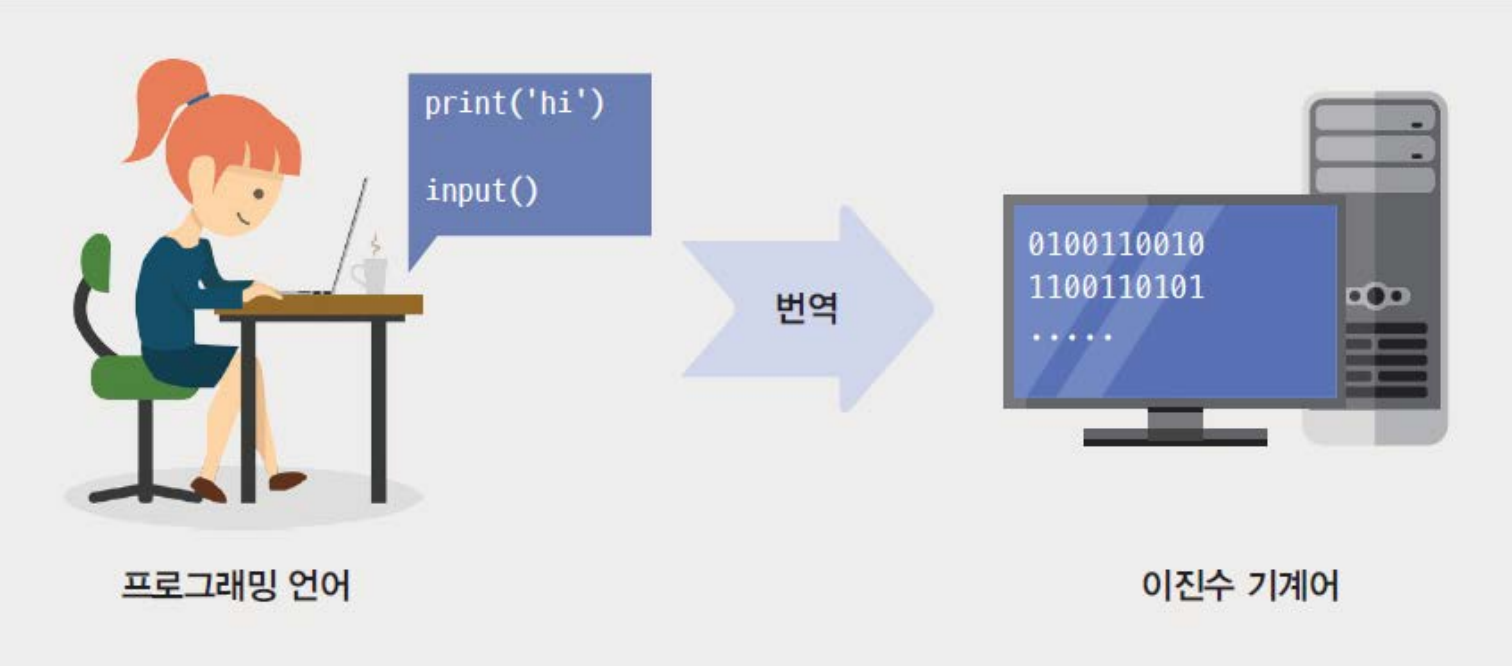


입출력 버스 표준 예:
- USB(Universal Serial Bus)
- PCI(peripheral component interconnect)
- ISA(Industry Standard Architecture)

- **운영 체제(OS)와 애플리케이션(App)**
- **운영체제(Operating System)**
  - 애플리케이션이 다양한 하드웨어(CPU, 기억장치, 입출력장치 등등) 위에서 동작할 수 있도록 하는 시스템 소프트웨어
  - 운영체제는 애플리케이션에게 API(Application Programming Interface)를 제공하여 운영체제가 제어하고 있는 하드웨어를 이용할 수 있게 함



- 번역(Compile/Interpret) : 프로그래밍 언어 -> 기계어
- 프로그래밍 언어 : 인간이 이해할 수 있는 언어
- 기계어 : 컴퓨터가 동작하는 언어로 0과 1로 이루어진 이진수
- 고수준(High Level)언어 : 인간에 가까운 언어 (Python 등)
- 저수준(Low Level)언어 : 기계에 가까운 언어 (기계어, Assembly Language)



- **컴파일 방식과 인터프리터 방식 비교**

	언어	실행 방식	특성
컴파일 방식	C, C++, C# Java	- 전체 코드 번역 후 실행 - 번역에러가 발생할 경우 전체 프로그램 실행 불가 - 정적 타이핑	- 실행속도 빠름 - 배우기 어려움 - 안정성 높음
인터프리터 방식 (스크립트 언어)	Python BASIC Javascript Ruby	- 한 줄씩 번역하면서 실행 - 에러 발생할 경우 해당 라인 직전까지 실행하고 정지 - 동적 타이핑	- 안정성 낮음 - 실행 속도 느림 - 배우기 용이함 - 안정성 낮음

- **정적 타이핑(Static Typing) : 변수를 사용하기 전에 반드시 타입을 선언해야 함**
  - **(C 언어 예, int a; a = 10;)**
- **동적 타이핑(Dynamic Typing) : 선언 필요 없이 타입을 자유롭게 변경하면서 사용**
  - **(Python 예, a = 10, a = "Hello")**

• 1990년 암스테르담의 귀도 반 로섬(Guido Van Rossum)에 의해 만들어진 인터프리터 언어

• 사전적인 뜻은 고대 신화 속의 파르나수스 산의 동굴에 살던 큰 뱀으로, 아폴로가 델파이에서 파이썬을 퇴치했다는 …

• 어디에서 많이 사용하는가?
– 구글에서 만들어진 소프트웨어의 50%이상이 파이썬으로 만들어졌다고 함
– Dropbox(파일 동기화 서비스), Django(파이썬 웹 프레임워크) 등

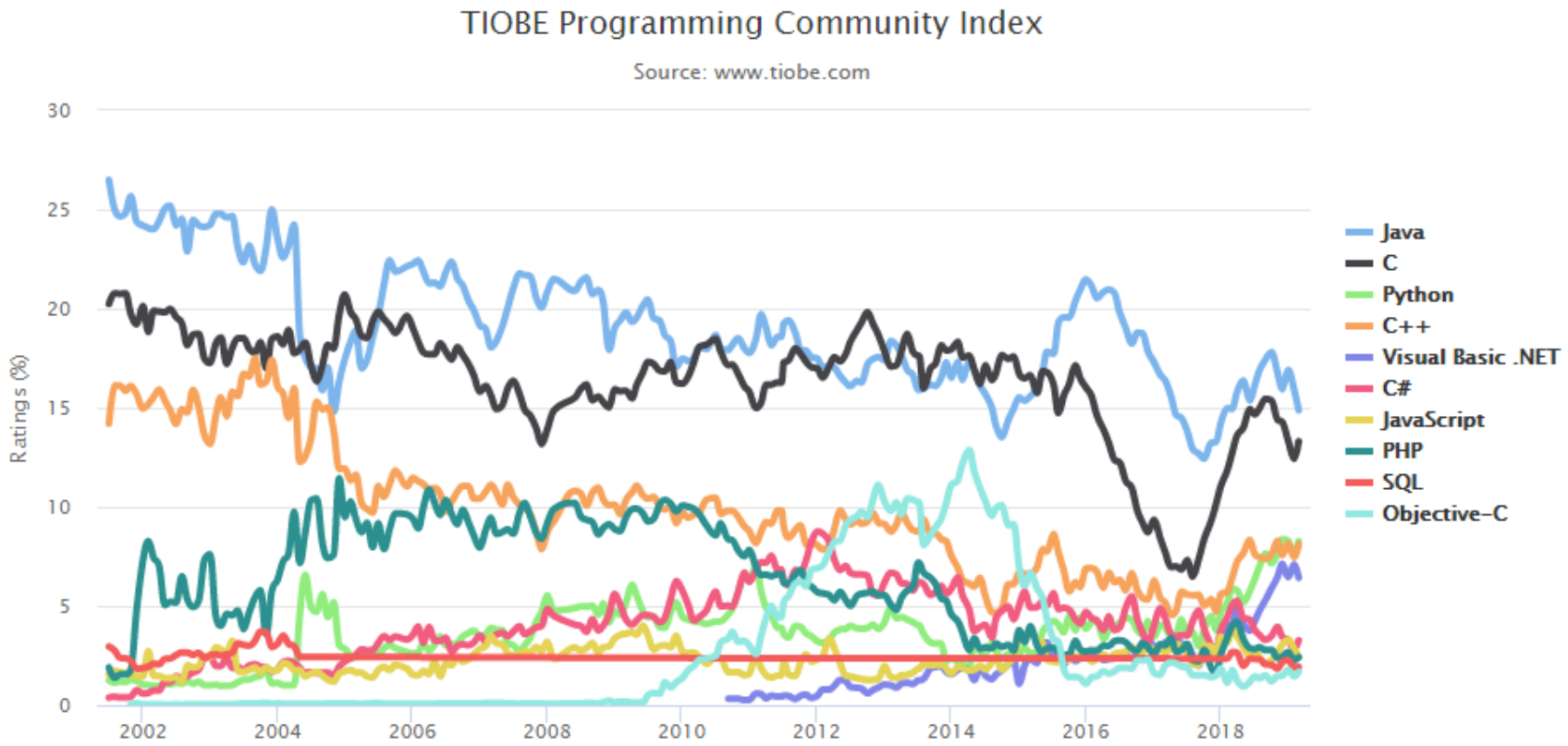
파이썬 로고(출처 : https://www.python.org)

파이썬의 창시자 귀도 반 로섬

- **파이썬 특징**

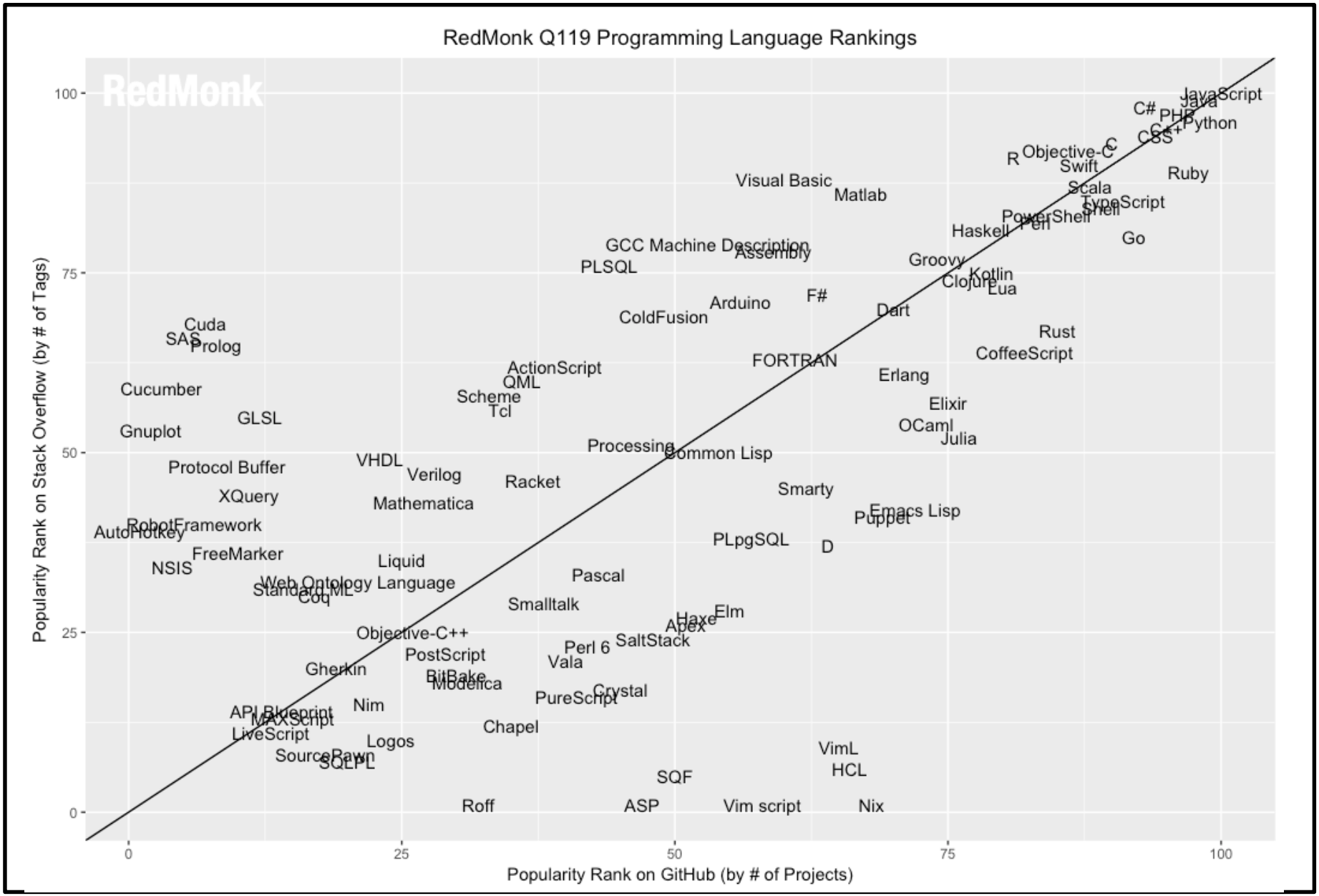
1. **인터프리터(스크립트) 언어 :** 컴파일러 언어와 달리, 소스코드 자체가 바로 실행되는 언어이다. 속도는 느리지만, 간편하게 사용할 수 있음
2. **동적 타이핑 언어 :** 사용 전에 변수의 타입을 결정하지 않고 프로그램의 실행 시점에서 각 변수의 타입이 결정되는 언어
3. **무료 :** 파이썬은 오픈 소스이며, 무료로 사용 가능. 다양한 추가 라이브러리도 무료
4. **쉽다 :** 초보자가 배우기 쉬움. 매우 간결한 코드로 알고리즘 개발에 적합. 개발시간이 단축됨. C언어에 비해 5-6배 짧은 코드 길이
5. **머신러닝 :** Google TensorFlow 등 주요 인공지능 개발 언어로 채택
6. **응용 범위** : Amazon(Web Service), Google, Raspberry PI 등 응용범위가 넓다

Most Popular Programing Languages

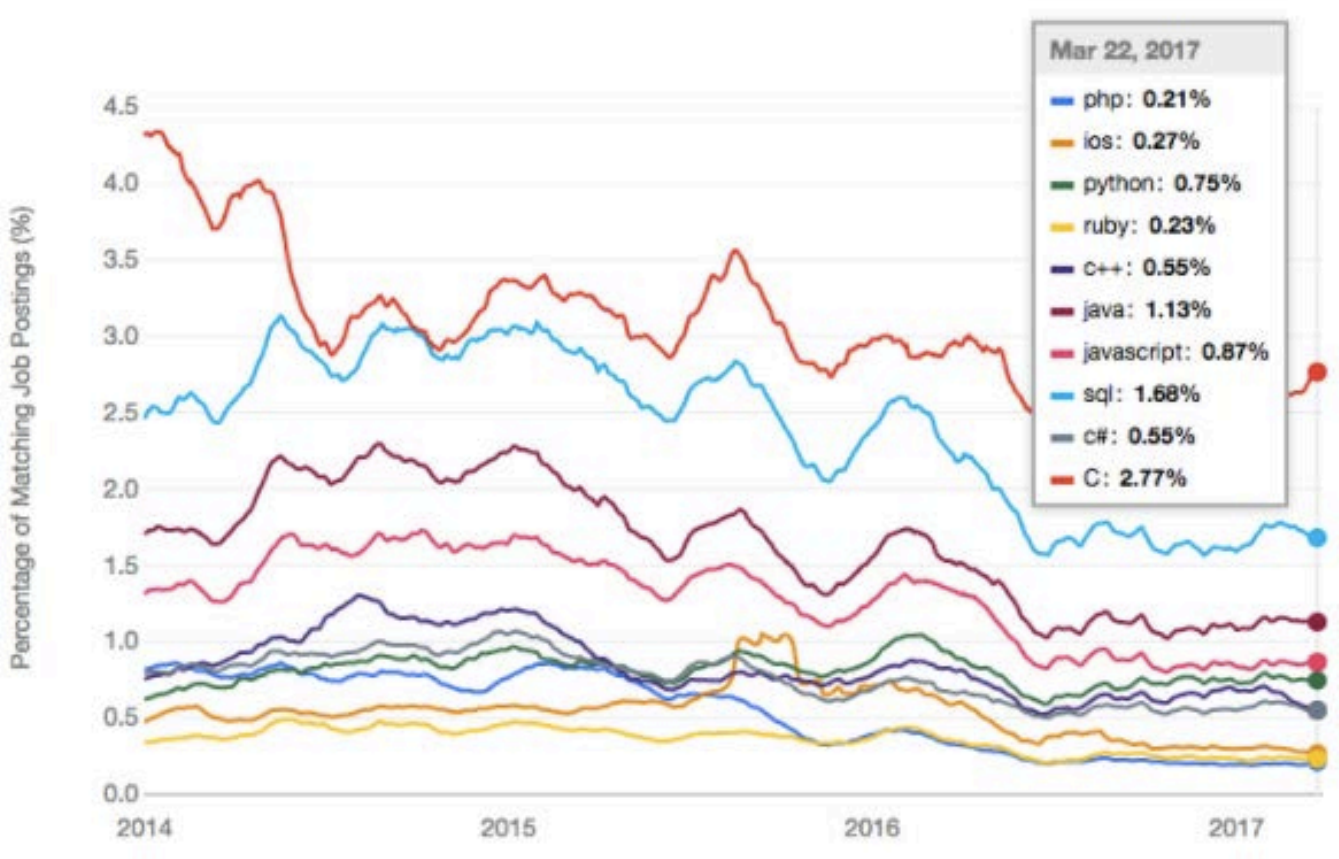


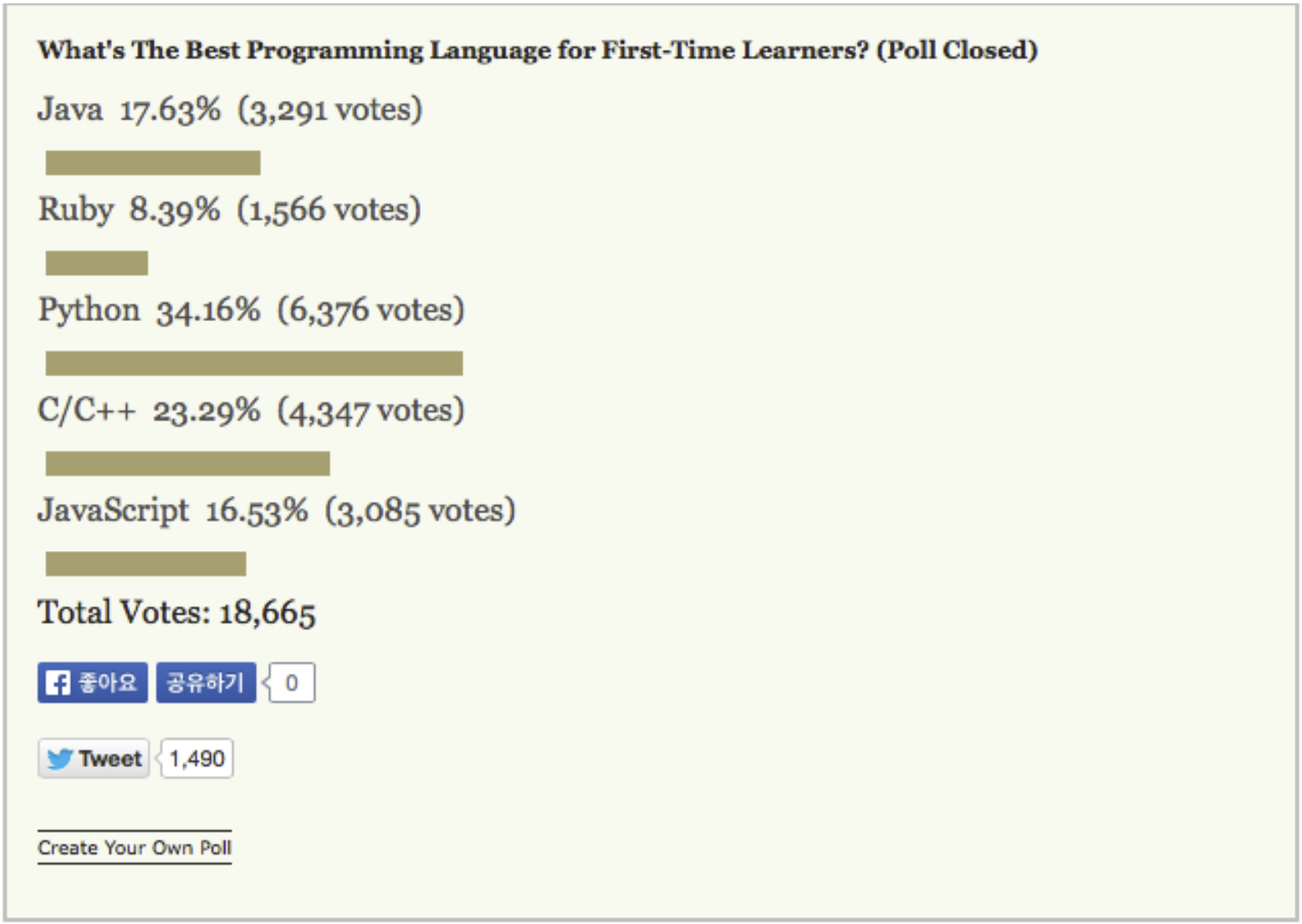
Most Popular Programing Languages

Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	**4**	▲	**Python**	**10.020%**	**+3.03%**
4	3	▼	C++	6.057%	-1.41%
5	6	▲	C#	3.842%	+0.30%
6	5	▼	Visual Basic .NET	3.695%	-1.07%
7	8	▲	JavaScript	2.258%	-0.15%
8	7	▼	PHP	2.075%	-0.85%
9	14	⇈	Objective-C	1.690%	+0.33%
10	9	▼	SQL	1.625%	-0.69%
11	15	⇈	Ruby	1.316%	+0.13%
12	13	▲	MATLAB	1.274%	-0.09%
13	44	⇈	Groovy	1.225%	+1.04%
14	12	▼	Delphi/Object Pascal	1.194%	-0.18%
15	10	⇊	Assembly language	1.114%	-0.30%


RedMonk Q119 Programming Language Rankings
RedMonk
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)
0
25
50
75
100
JavaScript
Java
Python
PHP
C#
C++
CSS
C
Objective-C
R
Swift
Ruby
Scala
TypeScript
Shell
PowerShell
Perl
Haskell
Go
Visual Basic
Matlab
GCC Machine Description
Assembly
PLSQL
Groovy
Kotlin
Clojure
Lua
Arduino
F#
Dart
ColdFusion
Rust
CoffeeScript
FORTRAN
Erlang
ActionScript
QML
Scheme
Tcl
Elixir
OCaml
Julia
Cuda
SAS
Prolog
Cucumber
GLSL
Gnuplot
Processing
Common Lisp
VHDL
Verilog
Racket
Protocol Buffer
XQuery
Mathematica
Smarty
Emacs Lisp
Puppet
RobotFramework
AutoHotkey
PLpgSQL
D
FreeMarker
NSIS
Liquid
Pascal
Web Ontology Language
Standard ML
Coq
Smalltalk
Elm
Haxe
Apex
Objective-C++
SaltStack
Perl 6
PostScript
Vala
Gherkin
BitBake
Modelica
Crystal
PureScript
Nim
API Blueprint
MAXScript
LiveScript
Chapel
Logos
SourcePawn
SQLPL
VimL
HCL
SQF
Roff
ASP
Vim script
Nix

Most Popular Programing Languages




What's The Best Programming Language for First-Time Learners? (Poll Closed)
Java 17.63% (3,291 votes)
Ruby 8.39% (1,566 votes)
Python 34.16% (6,376 votes)
C/C++ 23.29% (4,347 votes)
JavaScript 16.53% (3,085 votes)
Total Votes: 18,665
좋아요
공유하기
0
Tweet
1,490
Create Your Own Poll