Programming Languages

Cole Crawford
Software Engineer
Arts and Humanities Research Computing

A programming language is a formal language comprising a set of strings that produce various kinds of machine code output

Programming languages are …

- Difficult
- Intimidating
- Confusing

Programming languages are ***just like other languages***

Programming Languages

Compiled Languages

- “Built” translation
- Converted into machine code

Interpreted Languages

- “Live” translation
- Run line-by-line

Making Software

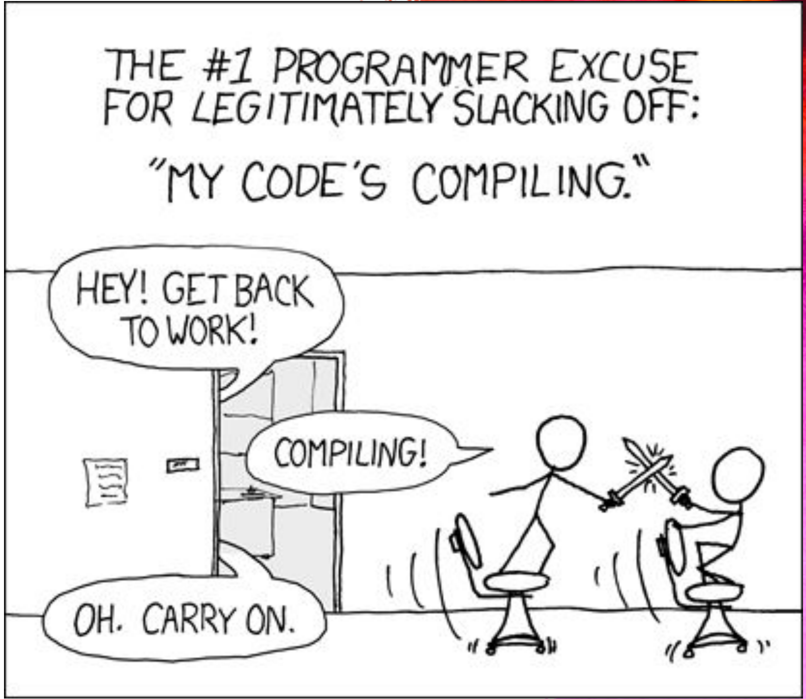
Compiled Version

- Write source code using a programming language
- Compile to object code
- Link into a binary executable
- Run the binary executable

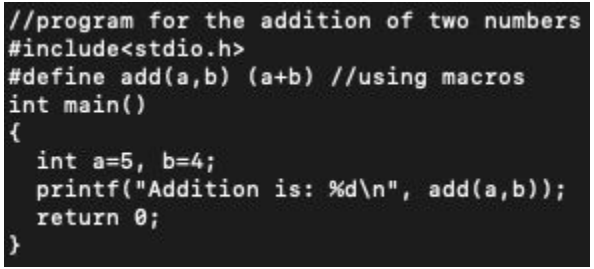
Making Software

Compiled Version

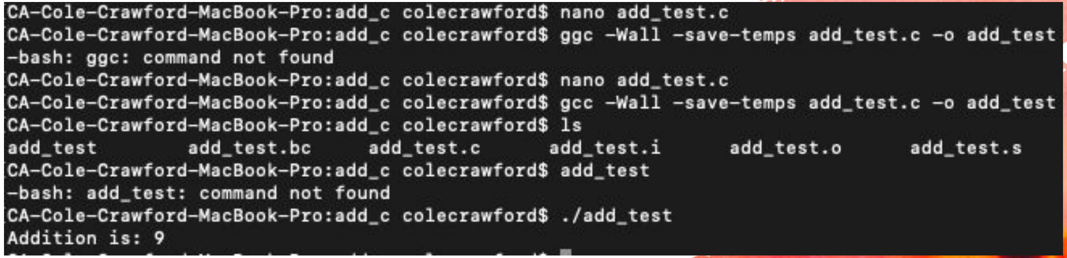
- Compilation: translates from one language into another
  - Source code -> object code
- Libraries
  - Static (included)
  - Dynamic (referenced)
- Examples
  - Assembly
  - C, C++, C#*
  - Java*



add_test.c

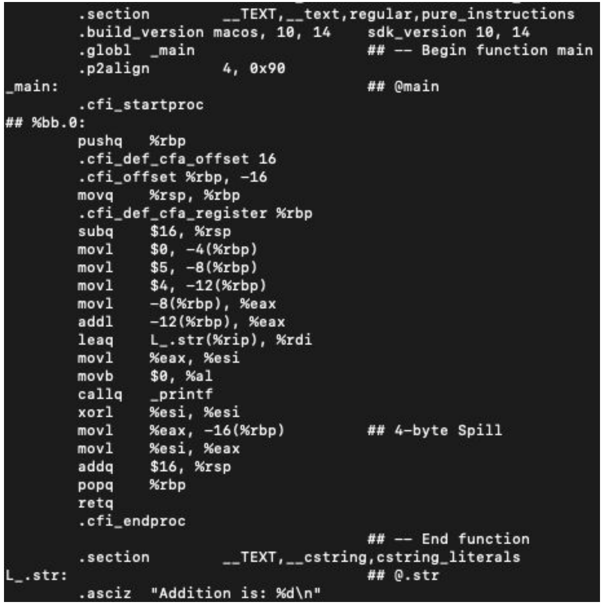


```
//program for the addition of two numbers
#include<stdio.h>
#define add(a,b) (a+b) //using macros
int main()
{
  int a=5, b=4;
  printf("Addition is: %d\n", add(a,b));
  return 0;
}
```



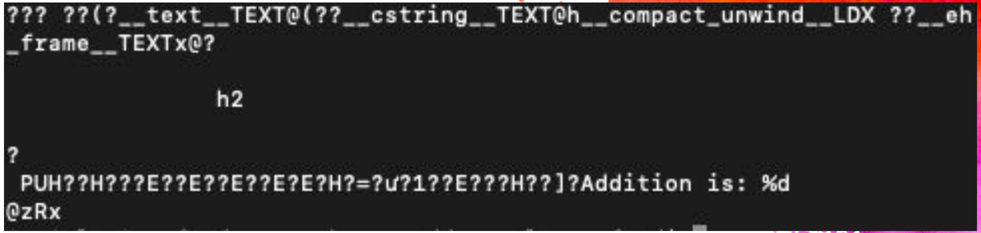
```
CA-Cole-Crawford-MacBook-Pro:add_c colecrawford$ nano add_test.c
CA-Cole-Crawford-MacBook-Pro:add_c colecrawford$ ggc -Wall -save-temps add_test.c -o add_test
-bash: ggc: command not found
CA-Cole-Crawford-MacBook-Pro:add_c colecrawford$ nano add_test.c
CA-Cole-Crawford-MacBook-Pro:add_c colecrawford$ gcc -Wall -save-temps add_test.c -o add_test
CA-Cole-Crawford-MacBook-Pro:add_c colecrawford$ ls
add_test        add_test.bc     add_test.c      add_test.i      add_test.o      add_test.s
CA-Cole-Crawford-MacBook-Pro:add_c colecrawford$ add_test
-bash: add_test: command not found
CA-Cole-Crawford-MacBook-Pro:add_c colecrawford$ ./add_test
Addition is: 9
```

add_test.s



```
        .section        __TEXT,__text,regular,pure_instructions
        .build_version macos, 10, 14    sdk_version 10, 14
        .globl  _main                   ## -- Begin function main
        .p2align        4, 0x90
_main:                                  ## @main
        .cfi_startproc
## %bb.0:
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset %rbp, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register %rbp
        subq    $16, %rsp
        movl    $0, -4(%rbp)
        movl    $5, -8(%rbp)
        movl    $4, -12(%rbp)
        movl    -8(%rbp), %eax
        addl    -12(%rbp), %eax
        leaq    L_.str(%rip), %rdi
        movl    %eax, %esi
        movb    $0, %al
        callq   _printf
        xorl    %esi, %esi
        movl    %eax, -16(%rbp)         ## 4-byte Spill
        movl    %esi, %eax
        addq    $16, %rsp
        popq    %rbp
        retq
        .cfi_endproc
                                        ## -- End function
        .section        __TEXT,__cstring,cstring_literals
L_.str:                                 ## @.str
        .asciz  "Addition is: %d\n"
```

add_test.o



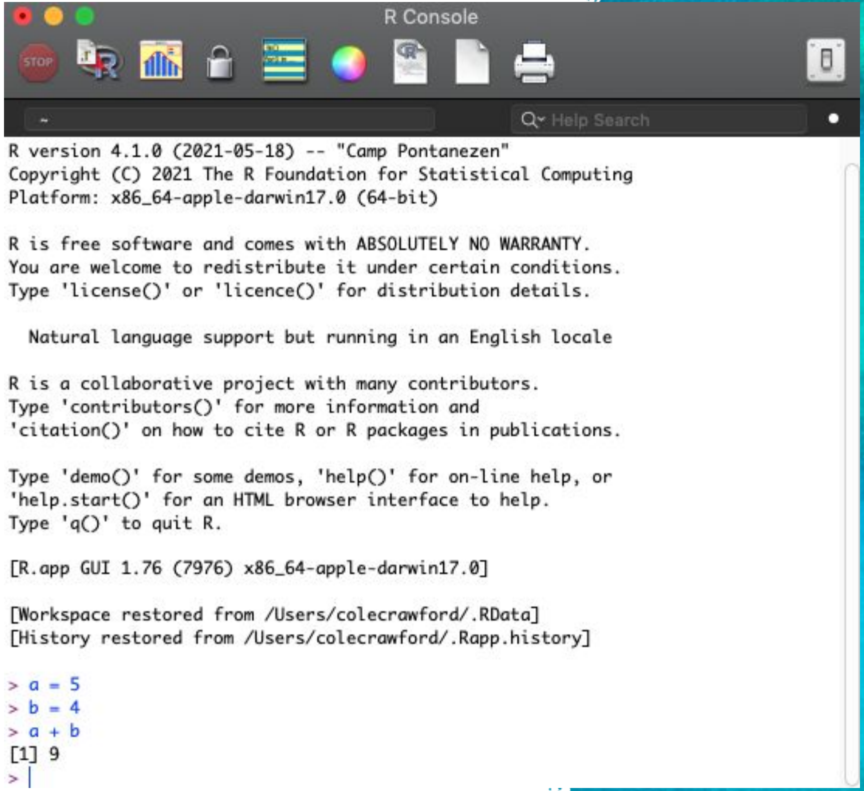
```
??? ??(?__text__TEXT@(??__cstring__TEXT@h__compact_unwind__LDX ??__eh
_frame__TEXTx@?

               h2

?
 PUH??H???E??E??E??E?E?H?=?u?1??E???H??]?Addition is: %d
@zRx
```

https://www.geeksforgeeks.org/compiling-a-c-program-behind-the-scenes/

Making Software

Interpreted Version

- Write source code using a programming language
- An interpreter interprets / runs the script
- “Slower” because of interpretation
- Can be run in “interactive” mode
- Examples
  - PHP
  - Javascript
  - Python
  - R



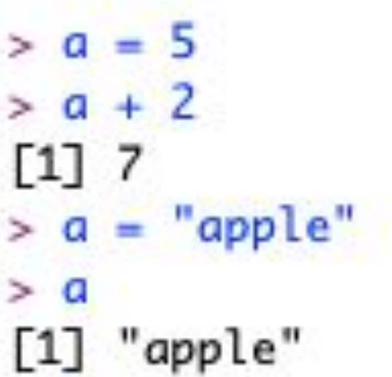
```
R version 4.1.0 (2021-05-18) -- "Camp Pontanezen"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.76 (7976) x86_64-apple-darwin17.0]

[Workspace restored from /Users/colecrawford/.RData]
[History restored from /Users/colecrawford/.Rapp.history]

> a = 5
> b = 4
> a + b
[1] 9
> 
```

Language Typing

Static

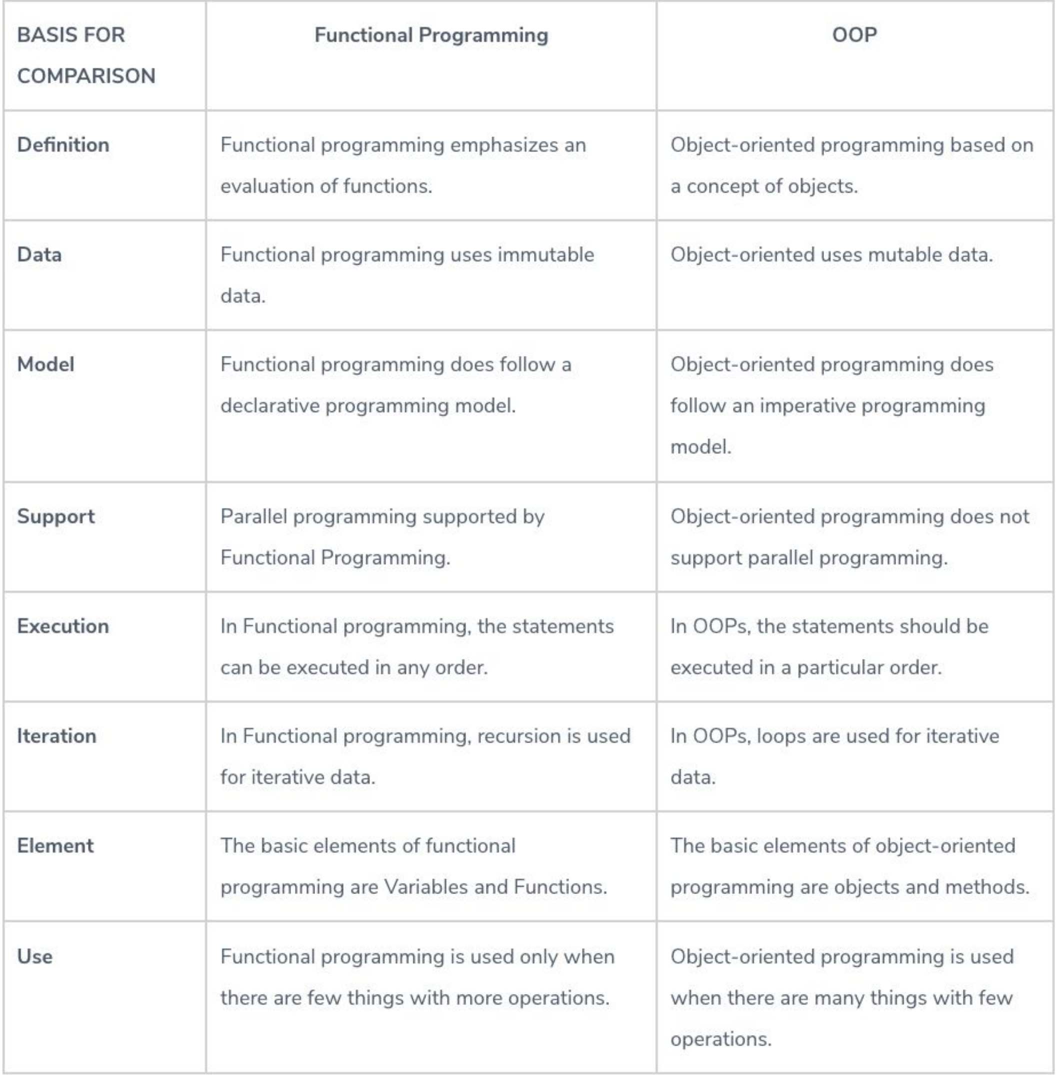
```
int a = 5;
a + 2            // returns 7
a = “apple”      // error

public static int addTwo(int a, int b){
      int sum = a + b;
      return sum;
}
```

Dynamic



```
> a = 5
> a + 2
[1] 7
> a = "apple"
> a
[1] "apple"
```

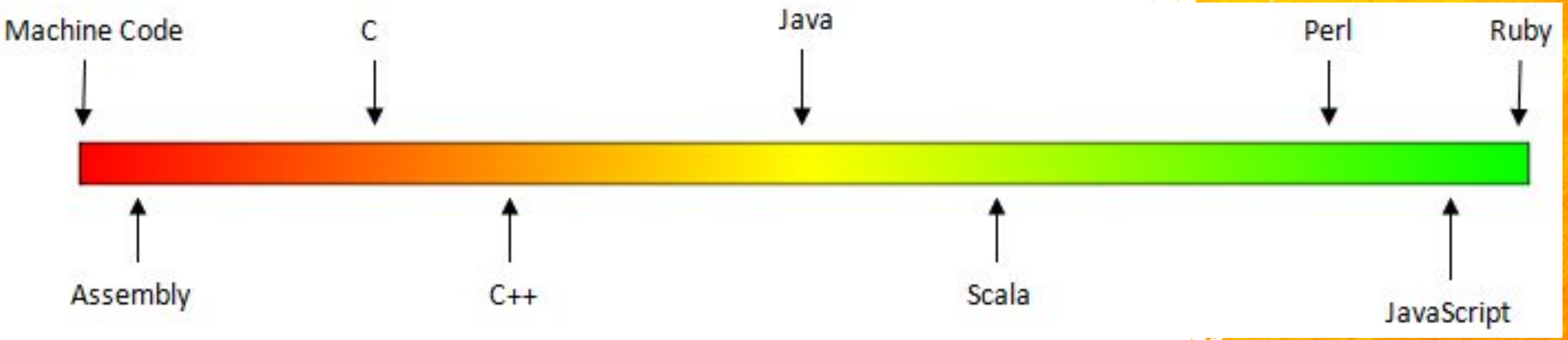
Functional vs Object-Oriented



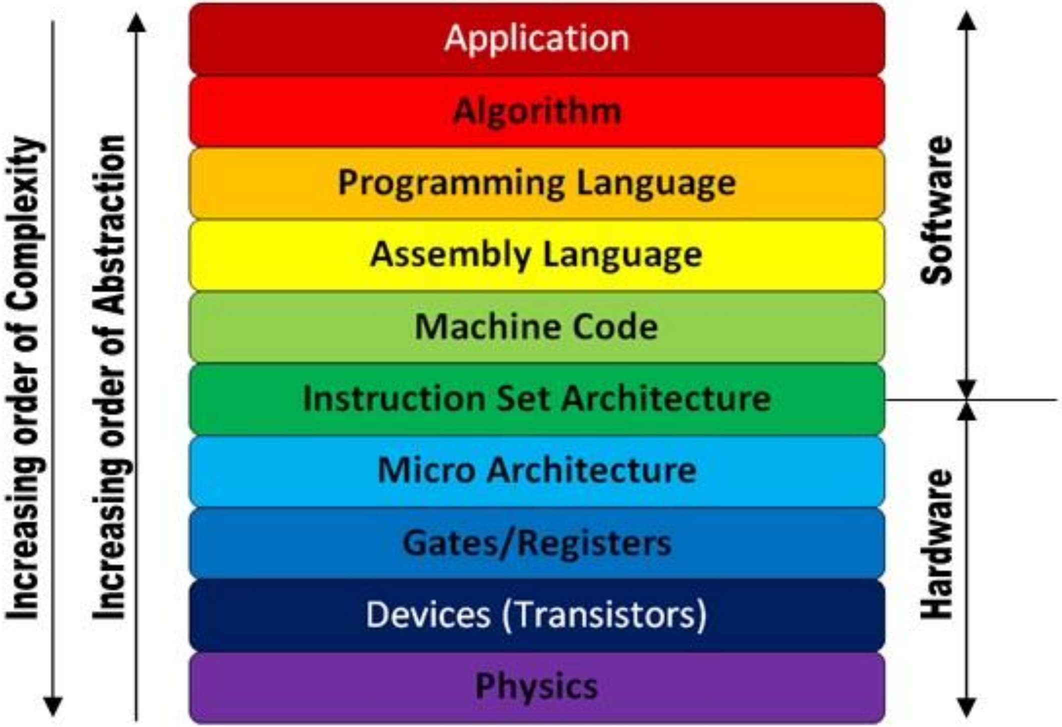
BASIS FOR COMPARISON	Functional Programming	OOP
Definition	Functional programming emphasizes an evaluation of functions.	Object-oriented programming based on a concept of objects.
Data	Functional programming uses immutable data.	Object-oriented uses mutable data.
Model	Functional programming does follow a declarative programming model.	Object-oriented programming does follow an imperative programming model.
Support	Parallel programming supported by Functional Programming.	Object-oriented programming does not support parallel programming.
Execution	In Functional programming, the statements can be executed in any order.	In OOPs, the statements should be executed in a particular order.
Iteration	In Functional programming, recursion is used for iterative data.	In OOPs, loops are used for iterative data.
Element	The basic elements of functional programming are Variables and Functions.	The basic elements of object-oriented programming are objects and methods.
Use	Functional programming is used only when there are few things with more operations.	Object-oriented programming is used when there are many things with few operations.

https://www.educba.com/functional-programming-vs-oop/

Abstraction



Abstraction



Application

Algorithm

Programming Language

Assembly Language

Machine Code

Instruction Set Architecture

Micro Architecture

Gates/Registers

Devices (Transistors)

Physics

Increasing order of Complexity

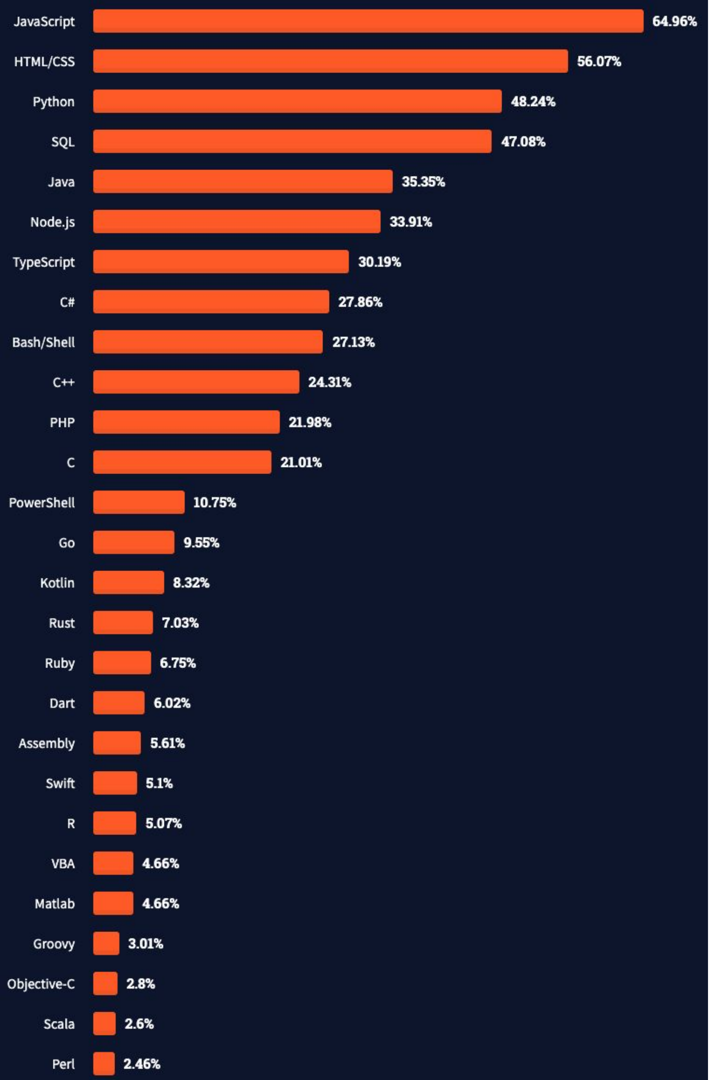
Increasing order of Abstraction

Software

Hardware

Layers of Abstraction

Popularity



https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language

Typical Use

- C: operating systems and high-performance
- C++: game development
- C#: web development, desktop applications
- Objective C and Swift: Apple / iOS apps
- Ruby (on Rails): website backends
- Python: website backends, data science, ML
- PHP: website backends
- R: data analysis, statistics, data visualization
- SQL: database interactions
- Javascript: website frontends, sometimes backends now
- HTML (markup language): web structure
- XML (markup language): structured data format
- CSS: styling HTML

Strengths

Java

× General purpose
× Large scale systems
× Speed
× Development time
× Stability

Weaknesses

- Verbose
- Not great at statistical modeling

Python

× General purpose
× Web development
× Transferable and easy to learn
× Machine learning: scikitlearn, TensorFlow, OpenCV
× Text Analysis: NLTK, spaCy, GenSim
× Data cleaning: Pandas, Numpy

Weaknesses

- Speed
- Requirements and environments

R

× Statistics and analysis
× Data cleaning: tidyverse
× Text analysis: Quanteda
× Data visualization and charts

Weaknesses

- Speed
- Not as general purpose