编译原理与设计：编程语言

北京理工大学 计算机学院

内容

- 语言发展历史
- 语言分类
- 语言排名
- 语言实现
- Lab. 1

语言发展历史

- 程序设计语言的演化
  - 机器语言, 1940s

```
[  op  |  rs |  rt |  rd |shamt| funct]
    0     1     2     6     0     32      decimal
 000000 00001 00010 00110 00000 100000    binary
```

```
[  op  |  rs |  rt | address/immediate]
   35     3     8           68            decimal
 100011 00011 01000 00000 00001 000100    binary
```

程序设计语言是用于计算方法描述的表示和标识方法

语言发展历史

- 程序设计语言的演化
  - 汇编语言，1950s 早期

```
LDF   R2, id3
MULF  R2, R2, #60.0
LDF   R1, id2
ADDF  R1, R1, R2
STF   id1, R1
```

语言发展历史

- 程序设计语言的演化
  - 高级程序设计语言，1950s后期
    - FORTRAN 科学计算
    - COBOL 商业数据处理
    - Lisp 符号计算

```fortran
write (*,*) "Enter the points to average:"
read  (*,*) points

! Take the average by summing points and dividing by number_of_points
if (number_of_points > 0) average_points = sum(points) / number_of_points

! Now form average over positive and negative points only
if (count(points > 0.) > 0) then
   positive_average = sum(points, points > 0.) / count(points > 0.)
end if

if (count(points < 0.) > 0) then
   negative_average = sum(points, points < 0.) / count(points < 0.)
end if
```

```cobol
ADD 1 TO x
ADD 1, a, b TO x ROUNDED, y, z ROUNDED

ADD a, b TO c
    ON SIZE ERROR
        DISPLAY "Error"
END-ADD

ADD a TO b
    NOT SIZE ERROR
        DISPLAY "No error"
    ON SIZE ERROR
        DISPLAY "Error"
```

```lisp
(defun factorial (n)
  (if (= n 0) 1
      (* n (factorial (- n 1)))))
```

语言发展历史

- 程序设计语言的演化
  - First-generation languages
    - Machine languages
  - Second-generation languages
    - Assembly languages
  - Third-generation languages
    - Higher-level languages: Fortran, Cobol, Lisp, C, C++, C# and Java
  - Forth-generation languages
    - Designed for specific application: NOMAD, SQL
  - Fifth-generation languages
    - Prolog and OPS5

表示事实：

```
human(kate).
human(bill).
likes(kate,bill).
```

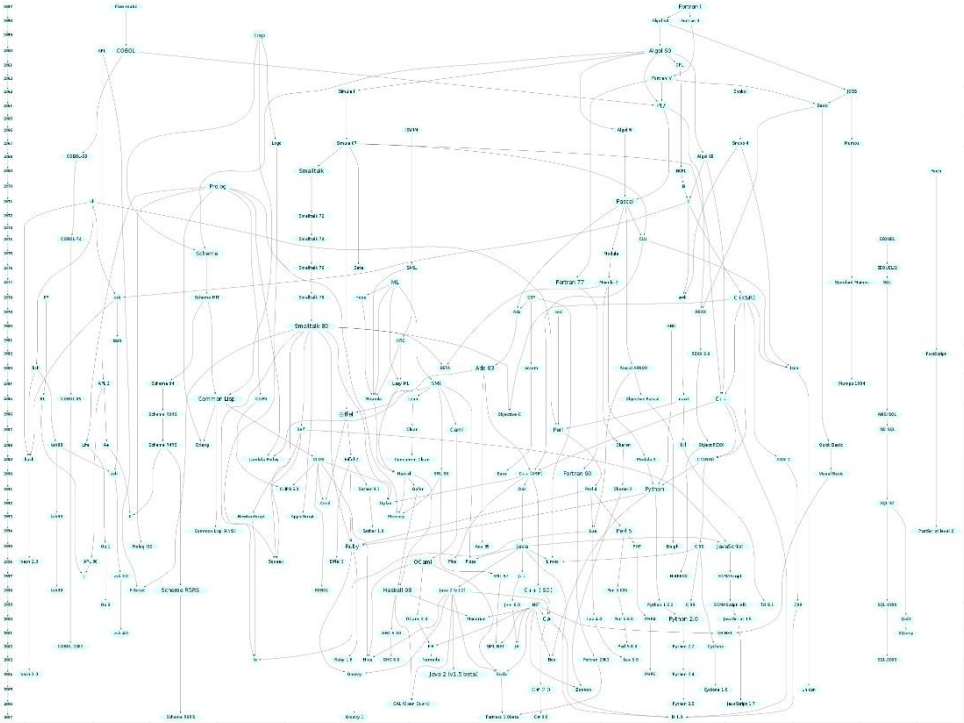
表示kate和bill是人（human），kate喜欢bill，而表示规则：

```
friend(X,Y):-likes(X,Y),likes(Y,X).
```

表示对于两个对象XY，如果X喜欢Y，且Y喜欢X，那么他们是朋友。

语言发展历史

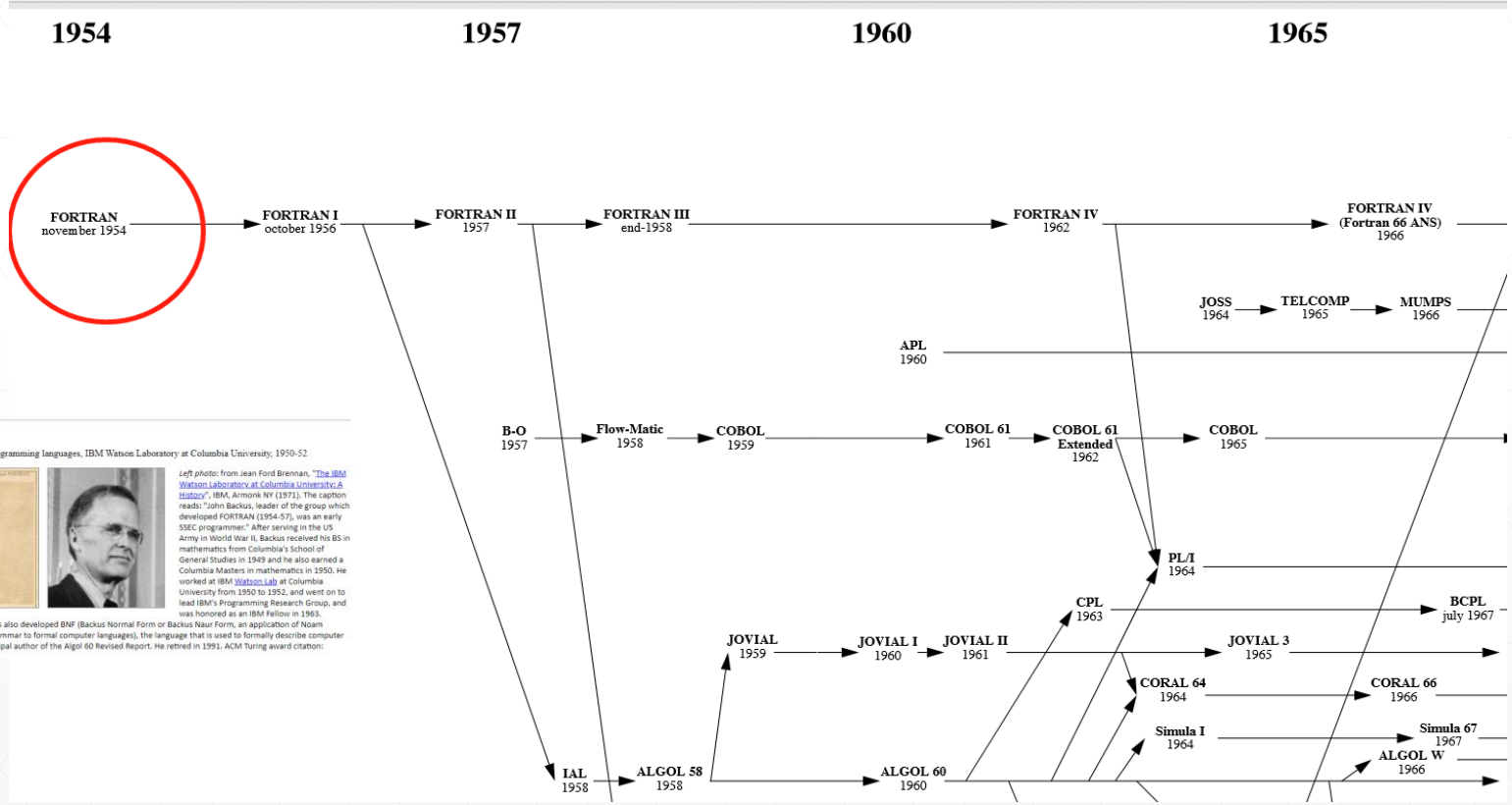
- Family Tree



https://github.com/stereobooster/programming-languages-genealogical-tree

语言发展历史

- 第一个程序设计语言



https://github.com/stereobooster/programming-languages-genealogical-tree

语言发展历史

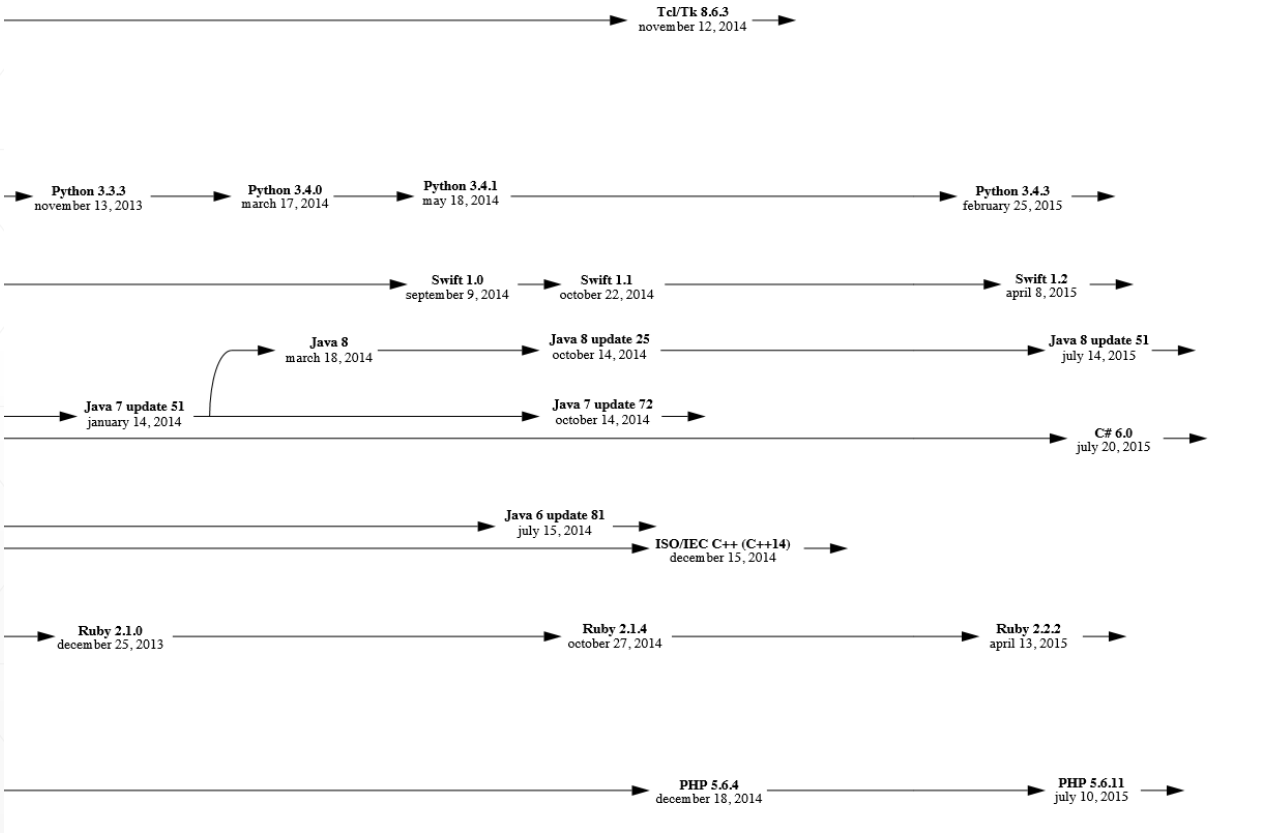
- 第一个程序设计语言

- 1951 – Regional Assembly Language
- 1952 – Autocode
- 1954 – IPL (forerunner to LISP)
- 1955 – FLOW-MATIC (led to COBOL)
- 1957 – FORTRAN (first compiler)
- 1957 – COMTRAN (precursor to COBOL)
- 1958 – LISP
- 1958 – ALGOL 58
- 1959 – FACT (forerunner to COBOL)
- 1959 – COBOL
- 1959 – RPG
- 1962 – APL
- 1962 – Simula
- 1962 – SNOBOL
- 1963 – CPL (forerunner to C)
- 1964 – Speakeasy
- 1964 – BASIC
- 1964 – PL/I
- 1966 – JOSS
- 1966 – MUMPS
- 1967 – BCPL (forerunner to C)

https://en.wikipedia.org/wiki/History_of_programming_languages

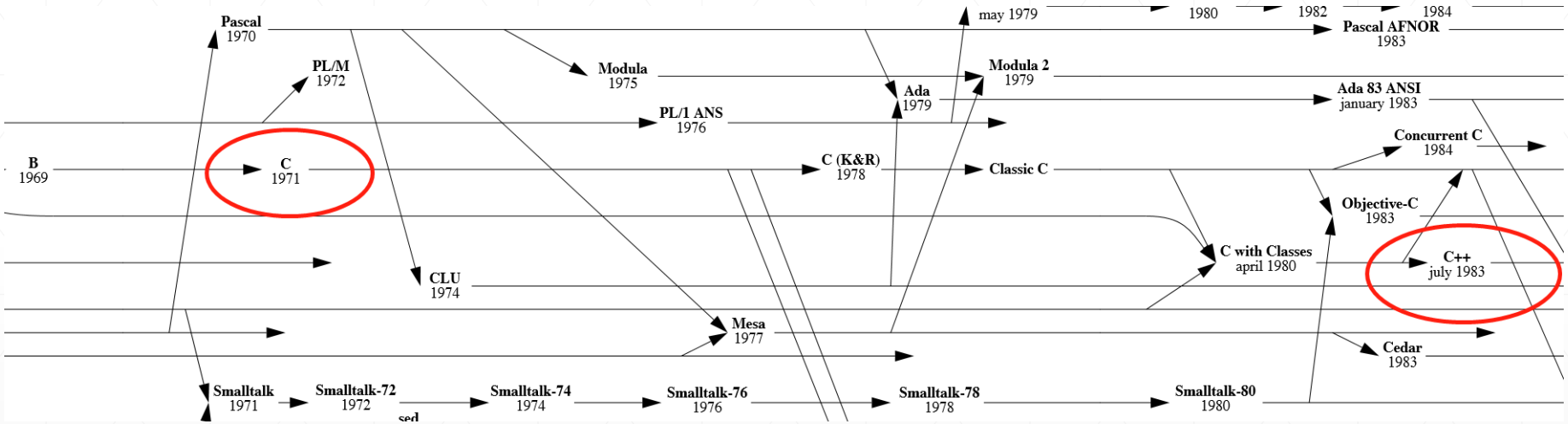
语言发展历史

- 新语言


Tcl/Tk 8.6.3
november 12, 2014
Python 3.3.3
november 13, 2013
Python 3.4.0
march 17, 2014
Python 3.4.1
may 18, 2014
Python 3.4.3
february 25, 2015
Swift 1.0
september 9, 2014
Swift 1.1
october 22, 2014
Swift 1.2
april 8, 2015
Java 8
march 18, 2014
Java 8 update 25
october 14, 2014
Java 8 update 51
july 14, 2015
Java 7 update 51
january 14, 2014
Java 7 update 72
october 14, 2014
C# 6.0
july 20, 2015
Java 6 update 81
july 15, 2014
ISO/IEC C++ (C++14)
december 15, 2014
Ruby 2.1.0
december 25, 2013
Ruby 2.1.4
october 27, 2014
Ruby 2.2.2
april 13, 2015
PHP 5.6.4
december 18, 2014
PHP 5.6.11
july 10, 2015

语言发展历史

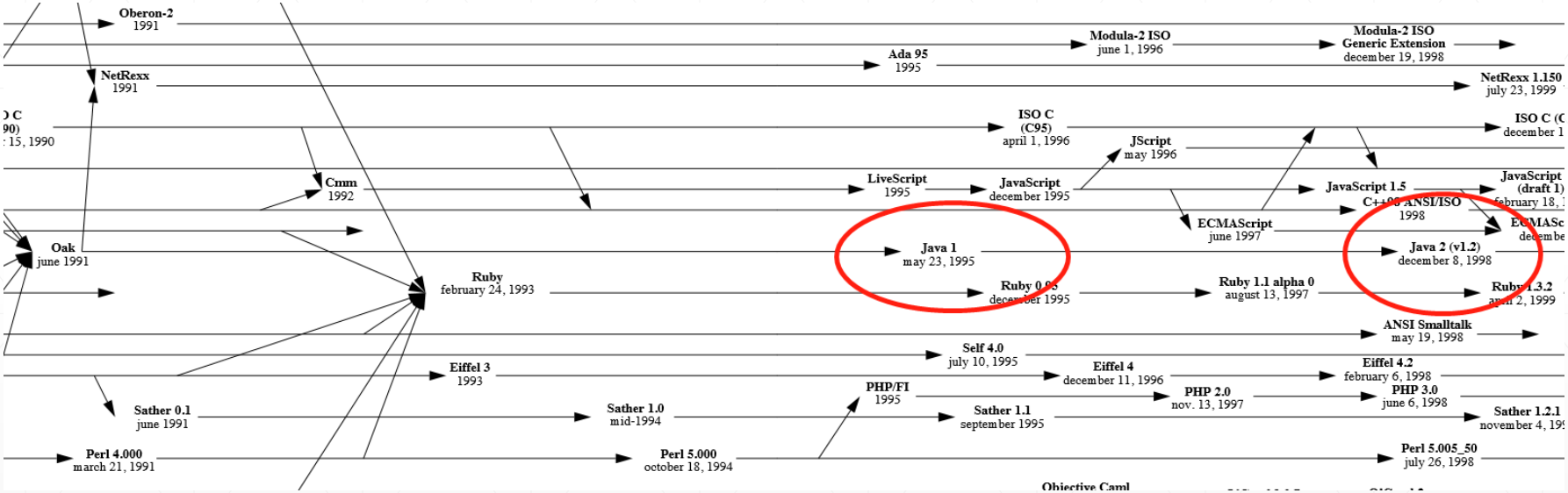
- C/C++



https://github.com/stereobooster/programming-languages-genealogical-tree

语言发展历史

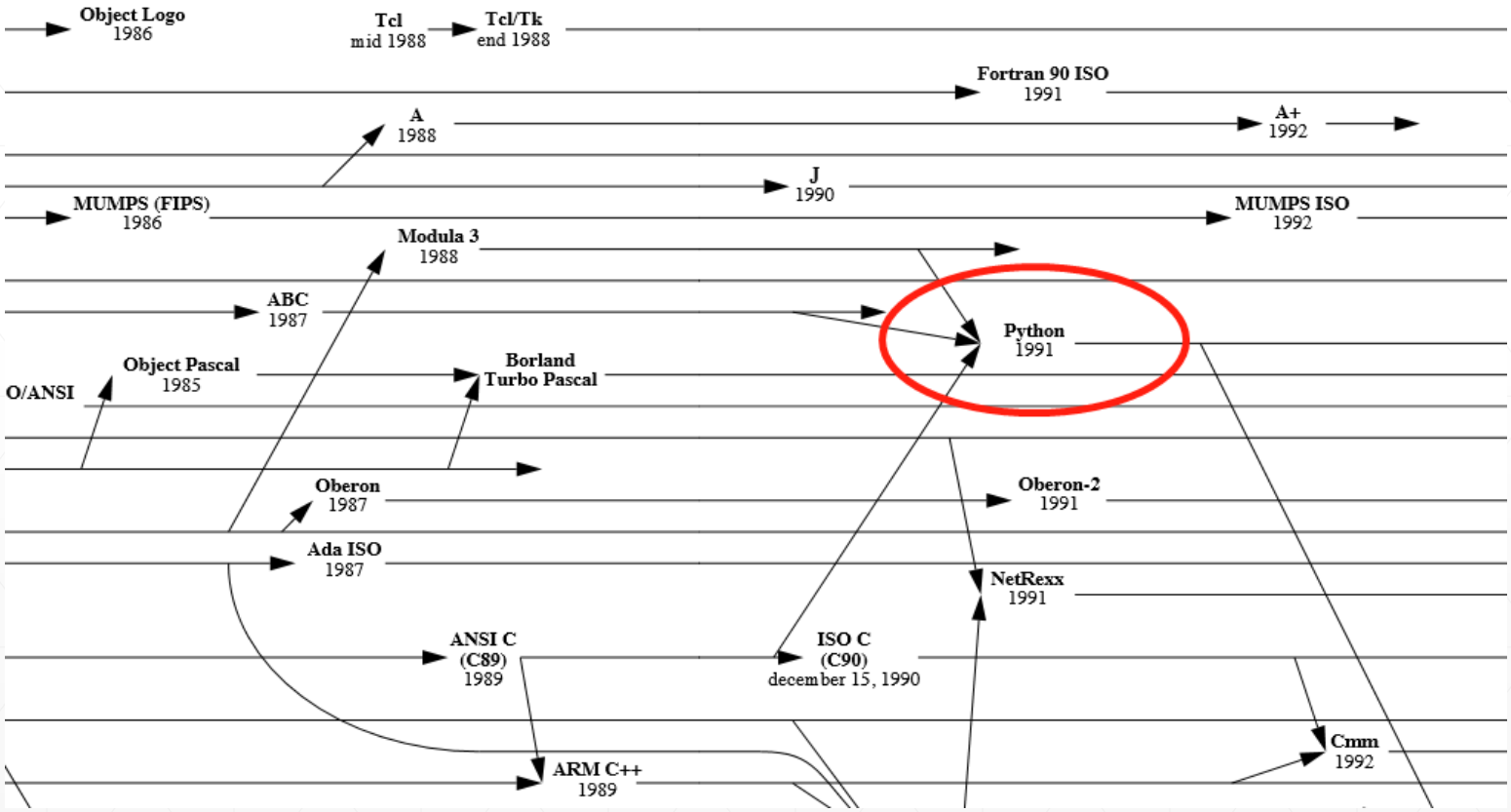
- Java



https://github.com/stereobooster/programming-languages-genealogical-tree

语言发展历史

- Python



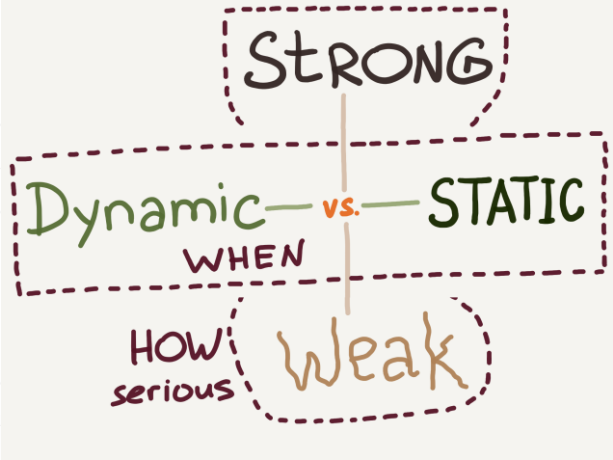
https://github.com/stereobooster/programming-languages-genealogical-tree

语言分类

- 分类标准:描述方式
  - 命令式：编程告诉计算机如何完成工作
    - 面向过程语言
    - 面向对象语言
  - 声明式：编程告诉计算机要完成哪些工作
    - 函数式语言
    - 逻辑编程语言

语言分类

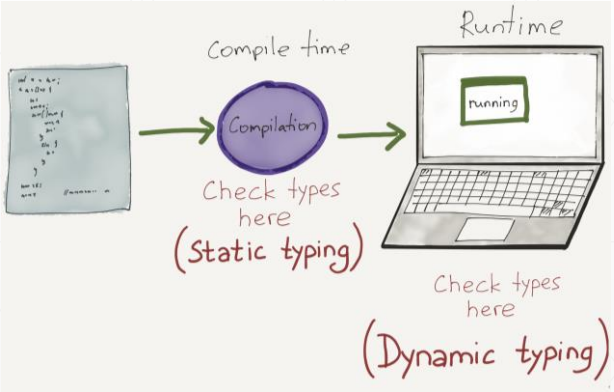
- 分类标准:Language by typing
  - 静态类型
    - 运行前类型检查： C, C++, and Java
  - 动态类型
    - 运行时检查类型：Python, Ruby等
  - 强类型
    - 不允许隐式数据类型转换
  - 弱类型
    - 允许隐式数据类型转换



https://softwareengineering.stackexchange.com/questions/17976/how-many-types-of-programming-languages-are-there

语言分类

- 分类标准：实现方式
  - 编译型
  - 解释型
- 分类标准：按照应用领域
  - 科学计算(FORTRAN)
  - 商业应用(SQL)
  - 系统编程(C/C++)



语言排名

Feb 2015	Feb 2014	Change	Programming Language	Ratings	Change
1	1		C	16.488%	-1.85%
2	2		Java	15.345%	-1.97%
3	4	^	C++	6.612%	-0.28%
4	3	v	Objective-C	6.024%	-5.32%
5	5		C#	5.738%	-0.71%
6	9	^	JavaScript	3.514%	+1.58%
7	6	v	PHP	3.170%	-1.05%
8	8		Python	2.882%	+0.72%
9	10	^	Visual Basic .NET	2.026%	+0.23%
10	-	^^	Visual Basic	1.718%	+1.72%
11	20	^^	Delphi/Object Pascal	1.574%	+1.05%
12	13	^	Perl	1.390%	+0.50%
13	15	^	PL/SQL	1.263%	+0.66%
14	16	^	F#	1.179%	+0.59%
15	11	vv	Transact-SQL	1.124%	-0.54%
16	30	^^	ABAP	1.048%	+0.69%
17	14	v	MATLAB	1.033%	+0.39%

http://www.tiobe.com/

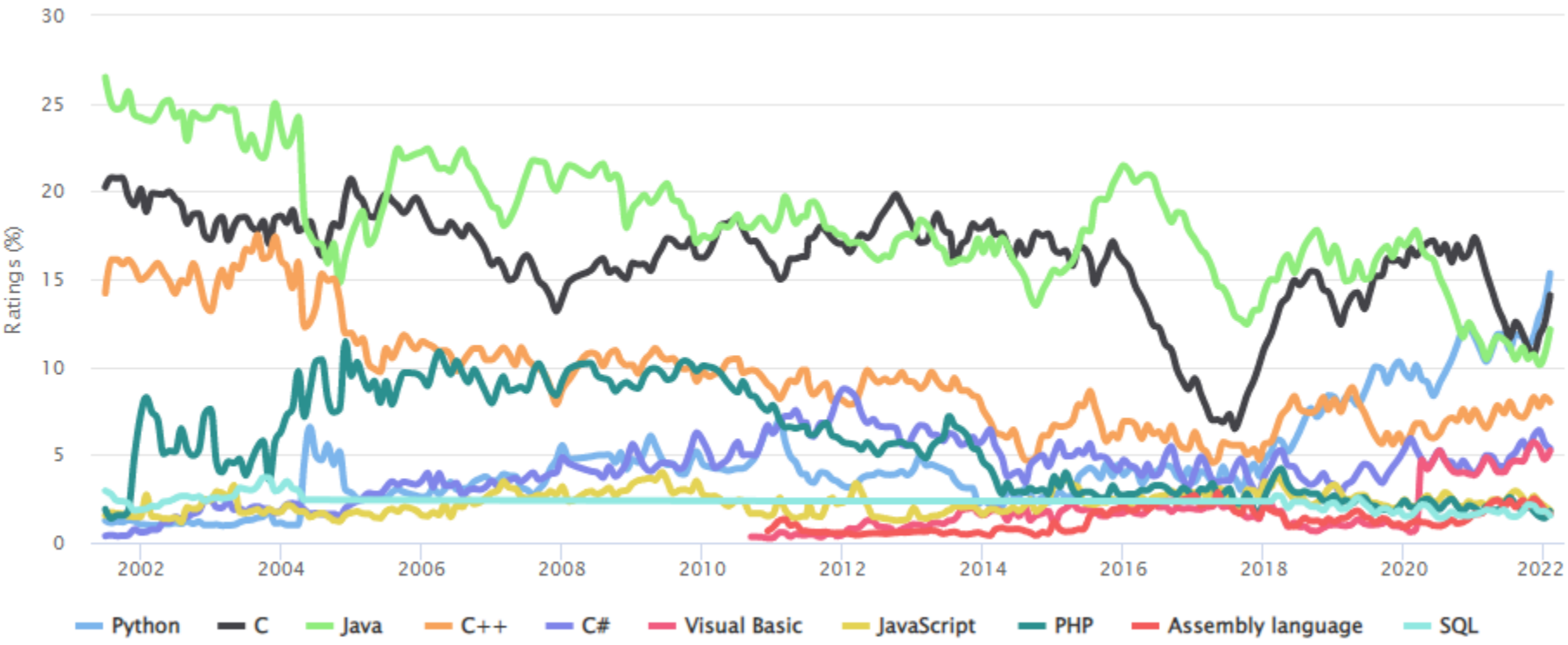
语言排名

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Feb 2022	Feb 2021	Change	Programming Language	Ratings	Change
1	3	^	Python	15.33%	+4.47%
2	1	v	C	14.08%	-2.26%
3	2	v	Java	12.13%	+0.84%
4	**4**		**C++**	**8.01%**	**+1.13%**
5	5		C#	5.37%	+0.93%
6	6		Visual Basic	5.23%	+0.90%
7	7		JavaScript	1.83%	-0.45%
8	8		PHP	1.79%	+0.04%
9	10	^	Assembly language	1.60%	-0.06%
10	9	v	SQL	1.55%	-0.18%
11	13	^	Go	1.23%	-0.05%

http://www.tiobe.com/

语言排名



Source: http://www.tiobe.com/

语言排名

Programming Language	2022	2017	2012	2007	2002	1997	1992	1987
C	1	2	2	2	2	1	1	1
Python	2	5	8	7	12	28	-	-
Java	3	1	1	1	1	16	-	-
C++	4	3	3	3	3	2	2	4
C#	5	4	4	8	14	-	-	-
Visual Basic	6	14	-	-	-	-	-	-
JavaScript	7	7	10	9	9	21	-	-
Assembly language	8	10	-	-	-	-	-	-
PHP	9	6	5	5	8	-	-	-
SQL	10	-	-	-	34	-	-	-
Prolog	24	33	43	27	28	19	13	3
Lisp	32	30	13	14	11	11	11	2
Pascal	271	97	15	20	18	6	3	6

http://www.tiobe.com/

语言排名

- 为什么会有这么多语言出现?
  - 语言演化
    - 不断发现有更好的方式做事情
  - 特殊目的
    - 为了特定的应用各领域和问题设计新的语言
    - 大数据、人工智能
  - 个人喜好

语言排名

- 什么样的语言会更成功?
  - 表达方式强大：在某一方面抽象程度高
  - 新手容易学习
  - 容易实现:小机器上也可以实现、容易移植
  - 开源且社区活跃
  - 强大的编译器支持
  - …

语言实现

- Python
  - Cpython、Jython、IronPython、PyPy
- Ruby
  - CRuby、Jruby、IronRuby

语言实现

- 用数组实现Stack
  - 数组存储压入的元素
  - 使用一个整数标识当前的栈顶
- 用链表实现Stack
  - 使用指针和动态分配
- 其它方式

语言实现

- 基于数组的实现

```
char Store[MAX];
int top = 0;
void push(char x)
{
    if (top < MAX)
        Store[top++] = x;
    else
        printf("full\n");
}
```

```
char pop()
{
    if (top > 0)
        return Store[--top];
    else
        printf("empty\n");
}
...
```

Lab. 1 语言认知实验

- 见乐学平台上相关材料