Histoire des langages de programmations

Des premiers langages à ceux d'aujourd'hui.

DROSALYS

Terminologie

Quelques définitions pour mieux comprendre

Variable

Zone mémoire (RAM) où est stockée une valeur

Typage (des variables)

Fort / Faible

Un langage est dit

À typage forte si :

- Type de variable défini à l'avance
- Impossible de « convertir » une variable de type A en type B

À typage faible si :

- Pas de type de variable défini
- Conversion possible (caster)

Types de langages

Machine / Interprété / Compilé

Types de langages

Interprété

- Code source
- Interpréteur
- Lecture et exécution à la volée

Compilé

- Code source
- Compilateur
- Lecture et traduction en code machine
- Exécution du code machine

Types de langages

Interprété

- Exécution de code dynamique à la volée
- Même programme pour tous les environnements
- Code source disponible dans le programme

Compilé

- Plus rapide
- Un programme différent par environnement (plusieurs compilations)
- Code source par défaut non disponible dans le programme

Paradigmes / styles de programmation

Procédurale / Orienté Objet / Évènementielle / Fonctionnelle

Procédurale

- Évolution du séquentiel
- Exécution des instructions les unes après les autres
- Possibilité d'exécuter plusieurs fois les même lignes de code

Orienté Objet

- Permet des représentations de :
  - Concepts
  - Idées
  - « Objets » physiques
- Composition objet :
  - Une structure interne
  - Des comportements
- Modélisation visuel facilité : UML

Évènementielle

- Basé sur des évènements
- Changement d'état = Exécution d'une suite d'instructions

Fonctionnelle

- Abstraction des machines à états
- Pas de variables (faux pour quelques langages)
- Traitement en parallèle

Années 1950-1960

Les premiers langages

Grace Hopper en 1984
Source photo : https://fr.wikipedia.org

1951 - A-0 System

- Premier langage
- Premier compilateur

```
1.      C AREA OF A TRIANGLE - HERON'S FORMULA
2.      C INPUT - CARD READER UNIT 5, INTEGER INPUT, NO BLANK CARD FOR END
        OF DATA
3.      C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
4.      C INPUT ERROR DISPAYS ERROR MESSAGE ON OUTPUT
5.        501 FORMAT(3I5)
6.        601 FORMAT(" A= ",I5,"  B= ",I5,"  C= ",I5,"  AREA= ",F10.2,
7.           $"SQUARE UNITS")
8.        602 FORMAT("NORMAL END")
9.        603 FORMAT("INPUT ERROR OR ZERO VALUE ERROR")
10.           INTEGER A,B,C
11.        10 READ(5,501,END=50,ERR=90) A,B,C
12.           IF(A=0 .OR. B=0 .OR. C=0) GO TO 90
13.           S = (A + B + C) / 2.0
14.           AREA = SQRT( S * (S - A) * (S - B) * (S - C) )
15.           WRITE(6,601) A,B,C,AREA
16.           GO TO 10
17.        50 WRITE(6,602)
18.           STOP
19.        90 WRITE(6,603)
20.           STOP
21.           END
```

1954 - FORTRAN

- FORmula TRANslator
- Basé sur l' A-0 System
- Calcul scientifique
- Compilé

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. SALUTTOUS.
000300 DATE-WRITTEN. 21/05/05 19:04.
000400 AUTHOR UNKNOWN.
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 DEBUT.
100300 DISPLAY " " LINE 1 POSITION 1
ERASE EOS.
100400 DISPLAY "BONJOUR !" LINE 15
POSITION 10.
100500 STOP RUN.
```

1959 - Cobol

- Common Business Oriented Language
- Armée
- Banque
- Compilé

```
1.    INPUT "Quel est votre nom"; UserName$
2.    PRINT "Bonjour "; UserName$
3.    DO
4.       INPUT "Combien d'étoiles voulez-
      vous"; NumStars
5.       Stars$ = ""
6.       Stars$ = REPEAT$("*", NumStars) '<-
      ANSI BASIC
7.       'Stars$ = STRING$(NumStars, "*")
      '<-MS BASIC
8.       PRINT Stars$
9.       DO
10.         INPUT "Voulez-vous plus
      d'étoiles"; Answer$
11.      LOOP UNTIL Answer$ <> ""
12.   LOOP WHILE UCASE$(LEFT$(Answer$, 1))
      = "O"
13.   PRINT "Au revoir ";
14.   FOR A = 1 TO 200
15.      PRINT UserName$; " ";
16.   NEXT A
17.   PRINT
```

1964 - Basic

- Beginner's All-purpose Symbolic Instruction Code
- Inspiré du FORTRAN
- Populaire avec MS-DOS
- Interprété

Années 1970-1980

Langages modernes

```c
#include <stdlib.h>

struct int_list {
    struct int_list *next;
    int value;
};

struct int list *insert next(struct int_list *node, int value) {
    struct int list *const new_next = malloc(sizeof *new_next);

    if (new_next) {
        new_next->next = node->next;
        node->next = new_next;
        new_next->value = value;
    }

    return new_next;
}
```

1972 - C

- Procédurale
- Typage fort
- Compilé
- Réécriture d'UNIX
- Les plus utilisés

```
1.  DECLARE N INTEGER;
2.  SET N = 1;
3.  FOR C
4.  AS C_USR_MISE_A_JOUR
5.      CURSOR FOR
6.          SELECT USR_ID,
    USR_NOM
7.          FROM
    T_UTILISATEUR_USR
8.          ORDER  BY USR_ID
9.      FOR UPDATE OF USR_NOM
10. DO
11.     IF MOD(N, 2) = 0
12.     THEN
13.         UPDATE
    T_UTILISATEUR_USR
```

1975 - SQL

- Structured Query Language
- Base de données relationnelles
- Interprété

```
1.  #include<iostream>
2.
3.  int main()
4.  {
5.      std::cout << "Hello, new world!\n";
6.  }
```

1983 - C++

- Basé sur le C
- Typage fort
- Orienté Objet
- Compilé

```
1. @interface Personne : NSObject
2. {
3.       // variables d'instance
4.       NSString *surname;
5. }
6. // methodes
7. @property (copy) NSString
   *surname;
8. @end
```

1983 - Objectif-C

- Basé sur le C
- Orienté objet
- Typage fort/faible
- Compilé

```
1. # let square x = x * x;;
2. val square : int -> int = <fun>
3. # square 3;;
4. - : int = 9
5. # let rec fact x =
6.       if x <= 1 then 1 else x *
   fact (x - 1);;
7. val fact : int -> int = <fun>
8. # fact 5;;
9. - : int = 120
10.# square 120;;
11.- : int = 14400
```

1985 - Caml

- Categorical Abstract Machine Language
- INRIA
- Recherche
- Fonctionnelle
- Orienté Objet

```
1. #!/bin/bash
2.
3. # Deps
4. apt-get update
5. apt-get install -y build-essential
   libpcre3
6. # Compile & Install
7. mkdir -p nginx/build
8. cd nginx/build
9. wget
   http://nginx.org/download/nginx-
   1.12.0.tar.gz
10.tar xzf nginx-1.12.0.tar.gz
11.cd nginx-1.12.0
12./configure --with-http_ssl_module
13.make -j4
14.make install
15.
16.# Start Daemon
17./usr/local/nginx/sbin/nginx
```

1989 - Bash

- Bourne Again Shell
- Interprété
- Procédurale
- Typage faible
- Shell UNIX

Années 1990 à aujourd'hui

```
1. def factorielle(n):
2.     if n < 2:
3.         return 1
4.     else:
5.         return n * factorielle(n
   - 1)
```

1991 - Python

- Orienté objet
- Fonctionnelle
- Typage fort dynamique
- Interprété
- Très présent sur linux

```
1. Sub Main()
2.     MsgBox("Hello World")
3. End Sub
4.
5. Private Function AireDuCercle(Rayon As
   Double) As Double
6.     Const PI = 3.14159265358979
7.     AireDuCercle = PI * (Rayon ^ 2)
8. End Function
```

1991 - Visual Basic

- Basé sur le Basic
- Évènementielle
- Interprété

1993 - Brainfuck

```
1.  ++++++++++[>+++++++>++++++++++>+++>+<<<<-
    ]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------
    .>+.>.
```

- Urban Müller
- 8 instructions
- <>[]+-.,[]
- Possibilité de créer n'importe quel programme

```
1. class Application1
2. {
3.     static void main(String[ ] args)
4.     {
5.      char [ ] Tablecar
   ={'a','b','c','d','e','f'} ;
6.      int i, j ;
7.      System.out.println("tableau avant :
   " +
8.           String.valueOf(Tablecar));
9.      for ( i = 0 , j = 5 ;  i<j ; i++ ,
   j-- )
10.     { char car ;
11.        car = Tablecar[i];
12.        Tablecar[i ]= Tablecar[j];
13.        Tablecar[j] = car;
14.     }
15.
16.     System.out.println("tableau après :
   " + String.valueOf(Tablecar));
17.   }
18.}
```

1995 - Java

- Inspiration C++
- Sun microsystem
- Orienté objet
- Compilé
- Typage fort
- Portabilité
- Machine virtuel

```php
<?php
use Symfony\Component\HttpFoundation\Request;

require __DIR__.'/../vendor/autoload.php';
if (PHP_VERSION_ID < 70000) {
    include_once __DIR__.'/../var/bootstrap.php.cache';
}

$kernel = new AppKernel('prod', false);
if (PHP_VERSION_ID < 70000) {
    $kernel->loadClassCache();
}

$request = Request::createFromGlobals();
$response = $kernel->handle($request);
$response->send();
$kernel->terminate($request, $response);
```

1995 - PHP

- Basé sur le C et Bash
- Zend
- Procédurale
- Orienté objet
- Typage fort dynamique
- Interprété

```
1. function initDatePicker($section) {
2.     $section.find('input[data-
   datepicker]').each(function (index, elem) {
3.         var $elem = $(elem);
4.         var options =
   JSON.parse($elem.attr('data-datepicker'));
5.
6.         $elem.datePicker(options);
7.     });
8. }
```

1995 - Javascript

- Rien à voir avec Java
- ECMAScript (ESx)
- Typage faible
- Interprété
- « Orienté objet »

```
1. using System;
2.
3. class HelloWorld
4. {
5.     public static void Main()
6.     {
7.         Console.WriteLine("hello world!");
8.     }
9. }
```

2000 - C#

- C Sharp
- Inspiré de C++ & Java
- Microsoft
- Typage fort
- Compilé

```
1. extension String: SupportsToString {
2.     func toString() -> String {
3.         return self
4.     }
5. }

6. var someSortOfPrintableObject:
   SupportsToString

7. print(someSortOfPrintableObject.toString())
```

2014 - Swift

- Apple
- Compilé
- Multi paradigme

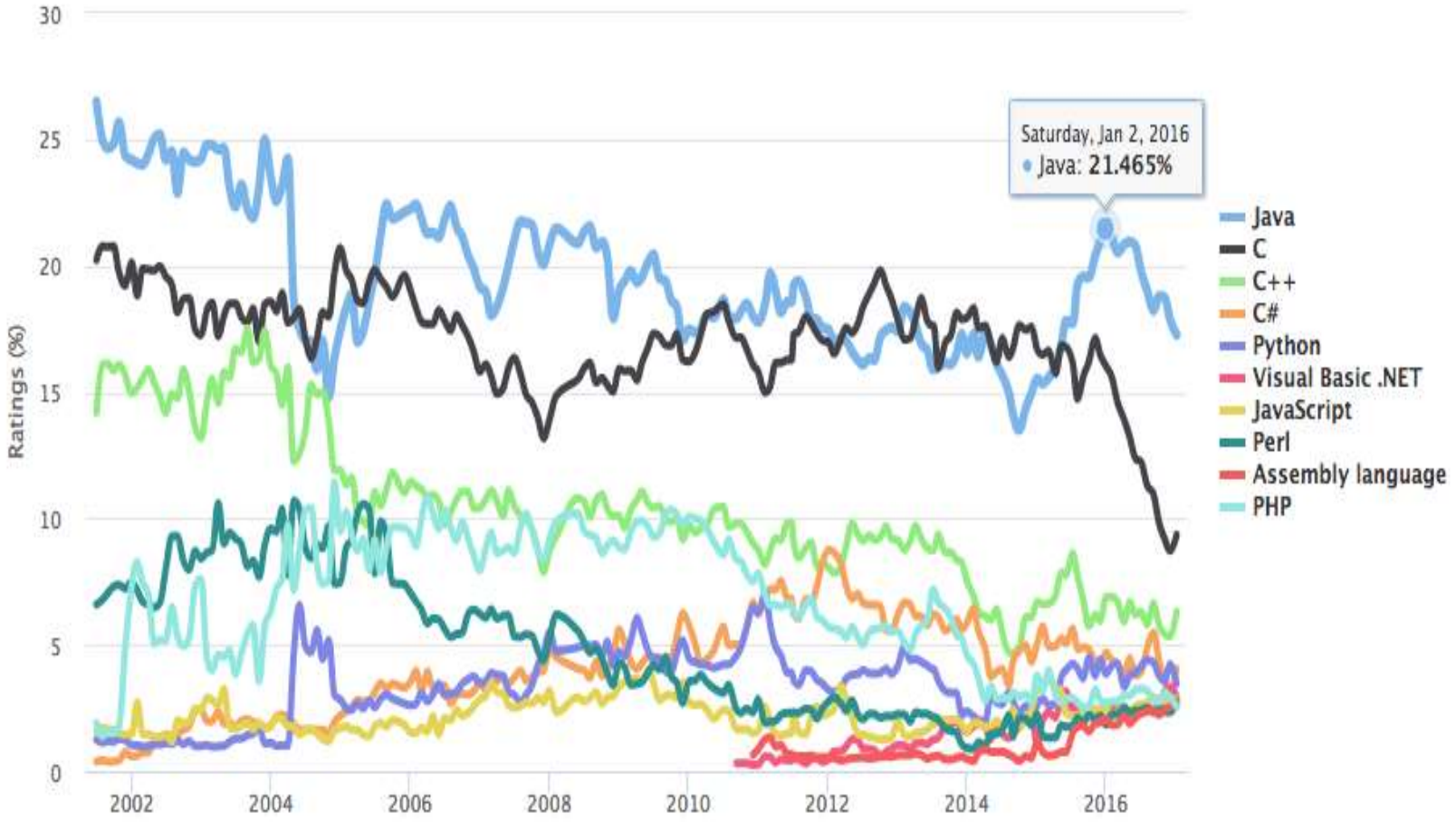
Popularité des langages

Classements

Classement titobe

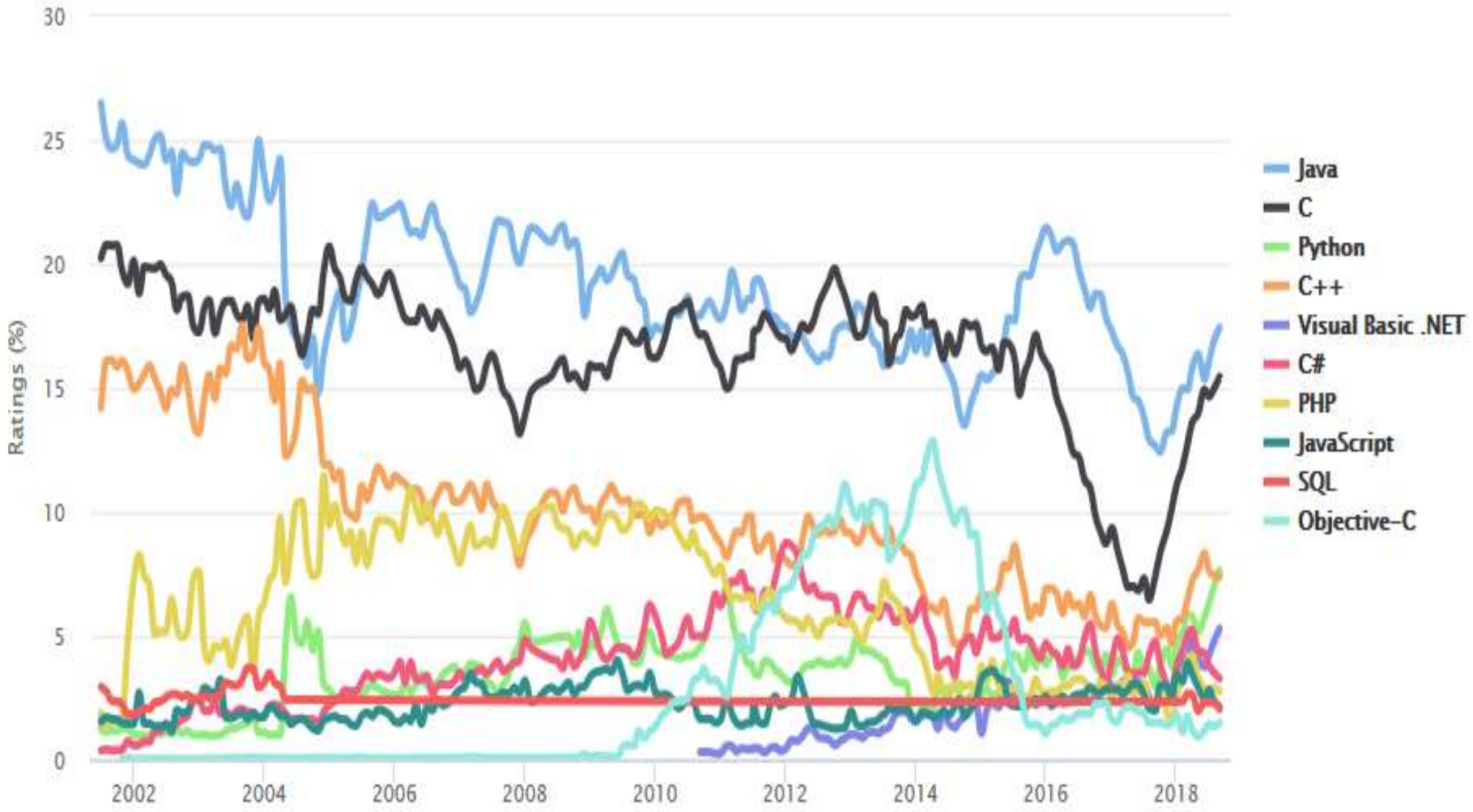
- Classement général
- Popularité des termes sur les moteurs de recherche

2017


30
25
20
15
10
5
0
Ratings (%)
2002
2004
2006
2008
2010
2012
2014
2016
Saturday, Jan 2, 2016
Java: 21.465%
Java
C
C++
C#
Python
Visual Basic .NET
JavaScript
Perl
Assembly language
PHP

2018

Drosalys



SPECOPS

Keyword Search	Global 'Google' Volume	Global 'Youtube' Volume	Google and Youtube Total
Learn Python	182,000	53,000	235,000
Learn Java	64,000	20,000	84,000
Learn C++	48,000	8,400	56,400
Learn SQL	38,000	7,000	45,000
Learn PHP	25,000	6,400	31,400
Learn R	12,000	2,000	14,000
Learn Swift	6,400	600	7,000
Learn Kotlin	5,600	600	6,200
Learn MATLAB	4,000	600	4,600
Learn C language	4,200	1,600	5,800
Learn Ruby on Rails	2,200	250	2,450
Learn Java Script	1,500	400	1,900
Learn Rust	1,500	650	2,150

The top 10 most searched for programming languages around the world

Source photo : https://specopssoft.com

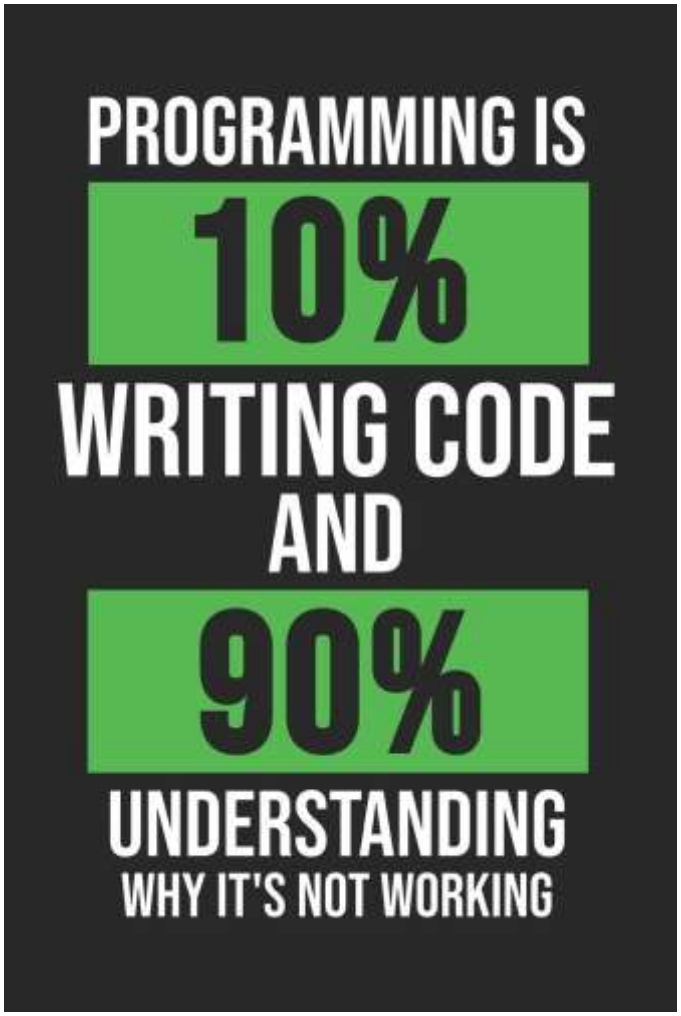
2020

According to our 2021 CodinGame annual survey of HR professionals and developers, the top 10 in-demand programming languages are currently:

1. ***JavaScript*** *(62% of respondents are on the hunt for candidates with this skill)*
2. ***Java*** *(59%)*
3. ***Python*** *(48%)*
4. ***C#*** *(40%)*
5. ***PHP*** *(32%)*
6. ***C++*** *(27%)*
7. ***Typescript*** *(24%)*
8. ***C*** *(15%)*
9. ***Kotlin*** *(15%)*
10. ***Swift*** *(14%)*

Top 10: Most In-Demand Programming Languages 2021

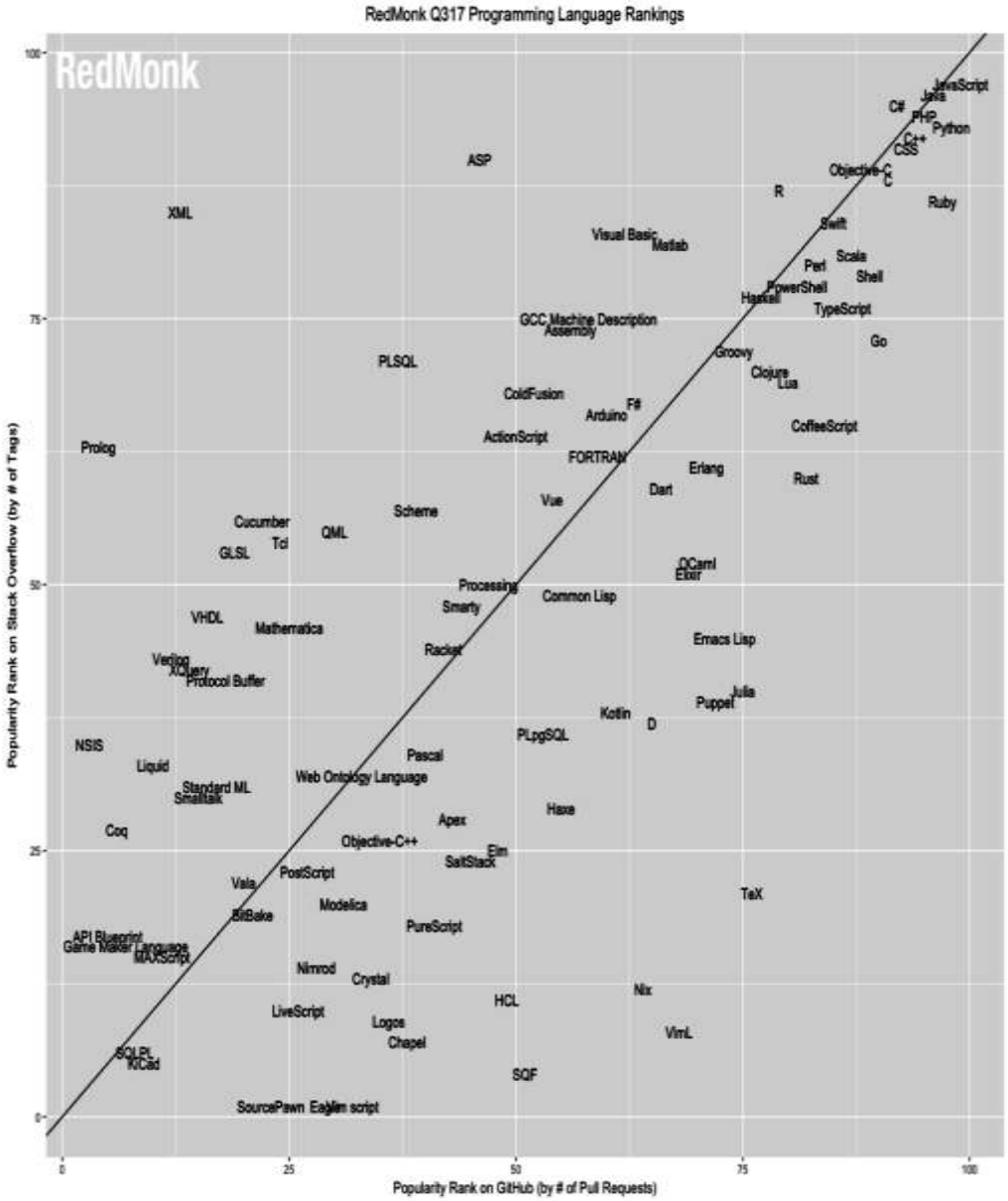
Source photo : https://www.codingame.com



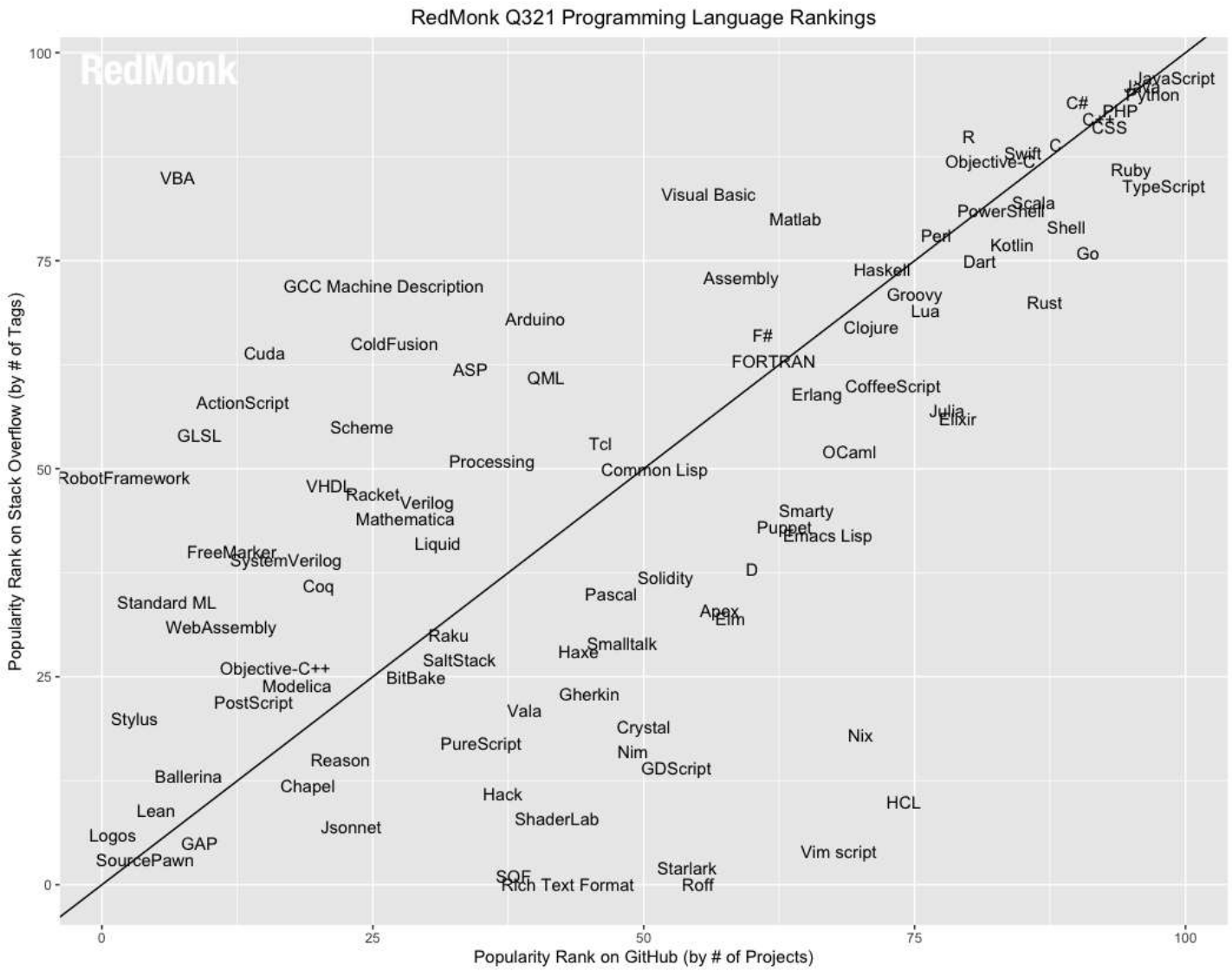
RedMonk

- Échanges sur les plateformes de dev
- Github
- Stackoverflow

2017


RedMonk Q317 Programming Language Rankings
RedMonk
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Pull Requests)
0
25
50
75
100
JavaScript
Java
C#
PHP
Python
C++
CSS
Objective-C
C
R
Ruby
Swift
Visual Basic
Matlab
Scala
Perl
Shell
PowerShell
Haskell
TypeScript
Go
GCC Machine Description
Assembly
Groovy
Clojure
Lua
PLSQL
ColdFusion
F#
Arduino
CoffeeScript
ActionScript
FORTRAN
Erlang
Rust
Dart
Vue
Scheme
Cucumber
QML
Tcl
GLSL
OCaml
Elixir
Processing
Common Lisp
Smarty
VHDL
Mathematica
Emacs Lisp
Racket
Verilog
XQuery
Protocol Buffer
Julia
Puppet
Kotlin
D
PLpgSQL
NSIS
Pascal
Liquid
Web Ontology Language
Standard ML
Smalltalk
Haxe
Apex
Coq
Objective-C++
Elm
SaltStack
PostScript
Vala
TeX
Modelica
BitBake
PureScript
API Blueprint
Game Maker Language
MAXScript
Nimrod
Crystal
Nix
HCL
LiveScript
Logos
VimL
Chapel
SQL PL
KiCad
SQF
SourcePawn
Eagle
Prolog
XML
ASP

2021


RedMonk Q321 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
C#
PHP
C++
CSS
R
Swift
C
Objective-C
Ruby
TypeScript
VBA
Visual Basic
Scala
PowerShell
Matlab
Shell
Perl
Kotlin
Go
Dart
Haskell
Assembly
GCC Machine Description
Groovy
Lua
Rust
Arduino
Clojure
F#
ColdFusion
Cuda
FORTRAN
ASP
QML
Erlang
CoffeeScript
Julia
Elixir
ActionScript
Scheme
GLSL
Tcl
OCaml
Processing
Common Lisp
RobotFramework
VHDL
Racket
Verilog
Mathematica
Smarty
Puppet
Emacs Lisp
Liquid
FreeMarker
SystemVerilog
Coq
D
Solidity
Pascal
Standard ML
Apex
Elm
WebAssembly
Raku
Smalltalk
Haxe
SaltStack
Objective-C++
BitBake
Modelica
Gherkin
PostScript
Vala
Stylus
Crystal
Nix
PureScript
Nim
Reason
GDScript
Ballerina
Chapel
Hack
HCL
Lean
ShaderLab
Jsonnet
Logos
GAP
Vim script
SourcePawn
Starlark
SQF
Rich Text Format
Roff

Des questions ?

Sources

- https://fr.wikipedia.org/wiki/Chronologie_des_langages_de_programmation
- https://www.tiobe.com/tiobe-index/
- http://redmonk.com/sogrady/2017/06/08/language-rankings-6-17/
- https://redmonk.com/sogrady/2018/08/10/language-rankings-6-18/