Langages de programmation

Cours 1

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/lp-prog

Plan

- machines informatiques
- systèmes informatiques
- langages de programmation
- étude comparative sur plusieurs exemples
- un langage fil conducteur: Python 3

dès maintenant: télécharger Python 3 en `http://www.python.org`

Machines informatiques

- les premiers vrais ordinateurs en 1960: IBM 704, 7040, 360/370, …
- le langage des machines est exprimé en **binaire**: 0 ou 1 [pour des raison électriques]

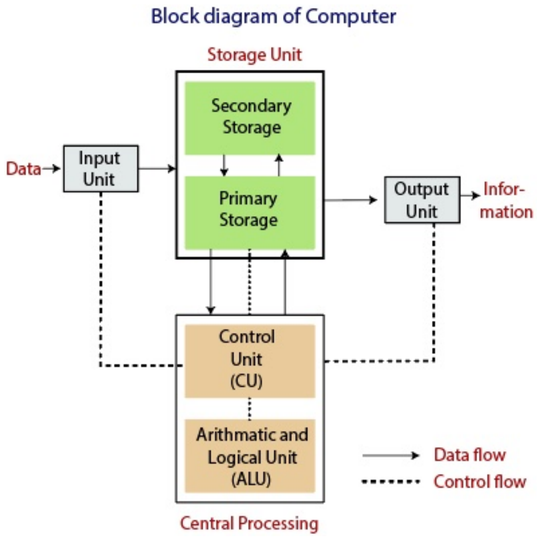
```
mac$ od -x a.out |head -4
0000000      0162    0006    8f30    2e89    6000    0000    0060    0000
0000020      0038    0007    010b    020a    4000    0000    2000    0000
0000040      0000    0000    0140    0040    0000    0040    0000    1000
0000060      2000    1000    fffe    f3ff    0000    0000    1013    0000
mac$ od -x a.out |tail -5
0116400      0000    0028    0216    0000    3030    0040    1046    0000
0116420      0000    0023    021c    0000    1580    1000    f0c5    ffff
0116440      0000    0023    0220    0000    12e0    1000    f341    ffff
0116460      0000    0023    0228    0000    12e4    1000    f341    ffff
```

```
mac$ cat t.c
#include <stdio.h>

char s[100] ="voici une chai^ne de caracte`res" ;

main()
{
    printf ("\%s\n", s);
}
```

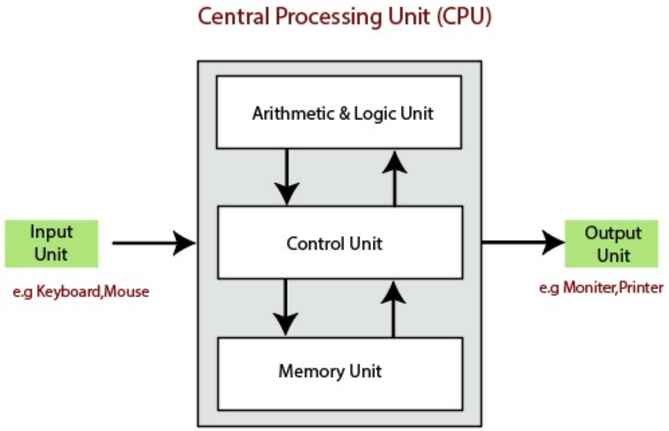
- ici 40256 octets = 132464 bits
- dans ce binaire, il y a des commandes [opérations à effectuer] et des données
- ce binaire est chargé dans la mémoire de l'ordinateur
- et l'ordinateur exécute les commandes du binaire
- chaque opération prend quelques micro-secondes

bits = binary digit
octets = 8 bits = bytes

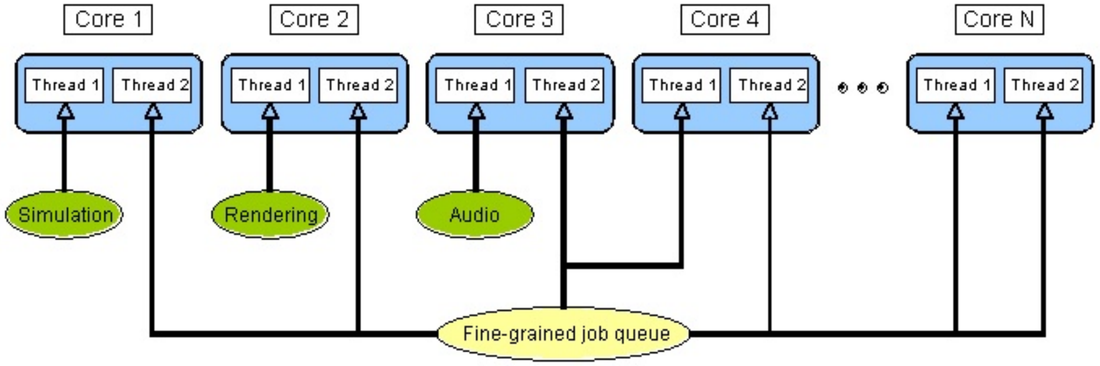
Machines informatiques



processeur



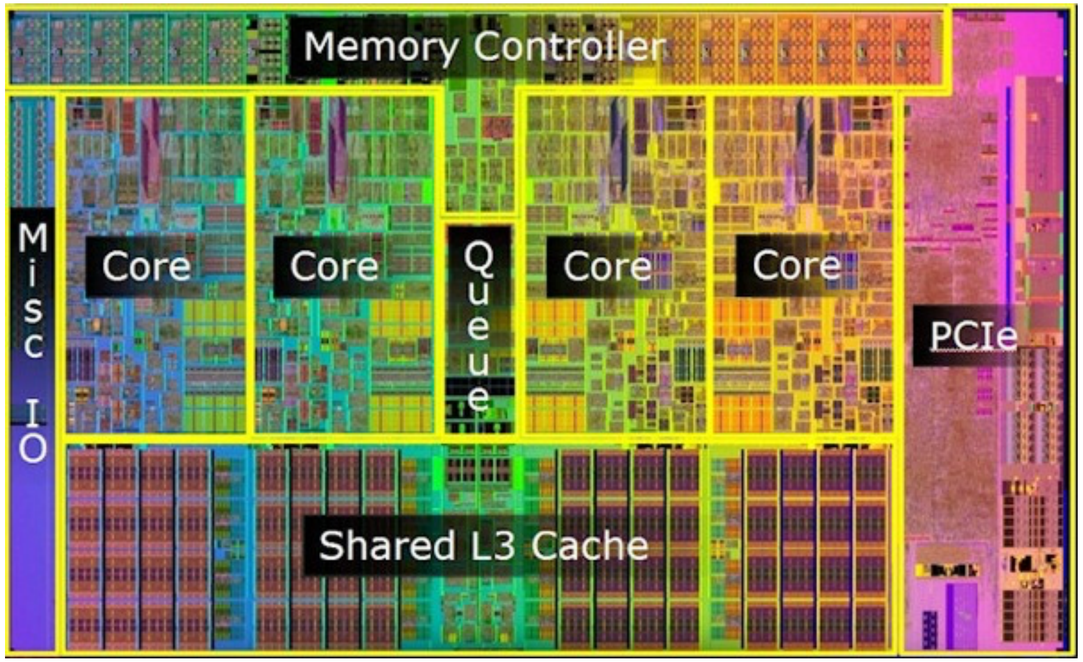
unité centrale



multicore

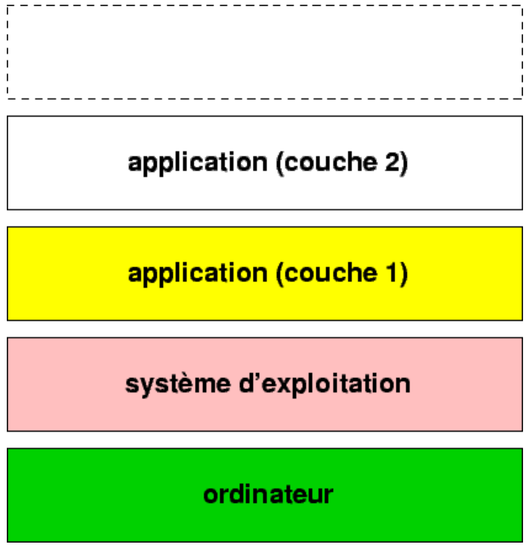
Machines informatiques

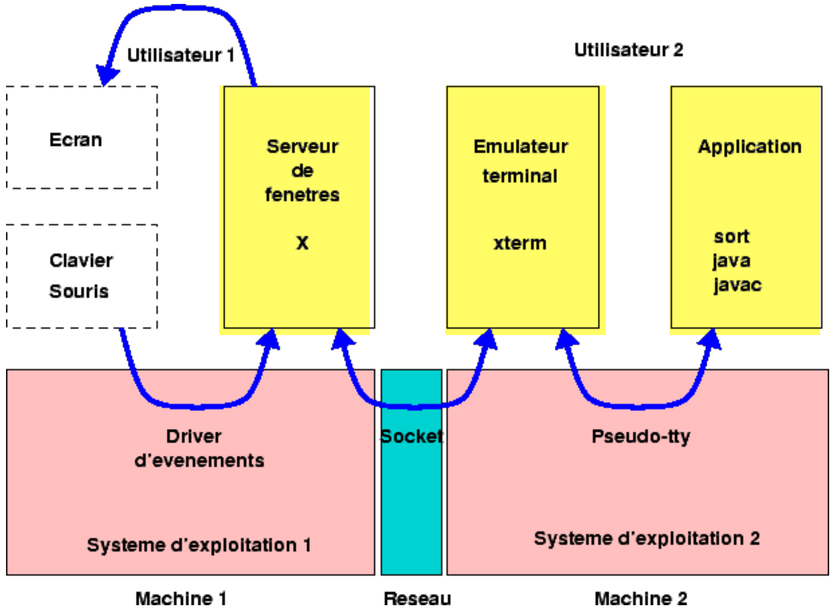
- Intel Core i5, 3.3GHz, 774m transistors, surface 296 mm2
- gravure 14 (en 2014) nm à 45nm



Systèmes informatiques

- Windows, MacOS, Unix, Linux, ios, android, GFS, …
- faire marcher l'ordinateur avec ses périphériques (écran, disques, clavier, souris, utilisateurs, réseau, …)
- les OS sont d'une complexité extrême (~100 millions de lignes de code)





OS = operating system
= système d'exploitation

- au 15ème siècle, on construisait des cathédrales
- au 20ème siècle, on a construit des systèmes informatiques

Langages de programmation et abstraction

- on écrit des programmes dans des langages plus ou moins éloignés du langage machine

Langages de programmation et buts professionnels

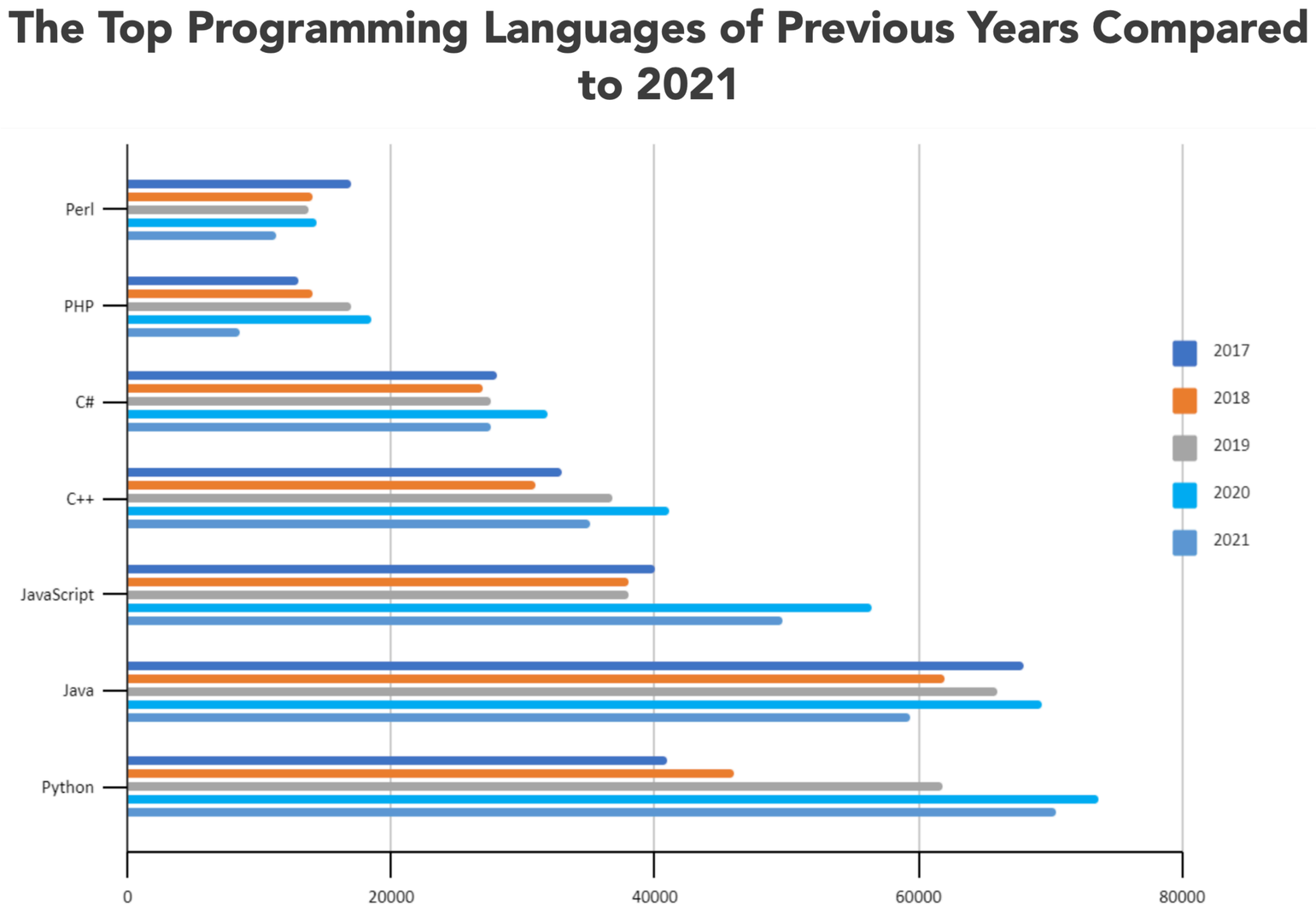
Front-end web development	Back-end web development	Mobile development
JavaScript	JavaScript	Swift
Elm	Scala	Java
TypeScript	Python	Objective C
	Go	JavaScript
	Ruby	

Game development	Desktop applications	Systems programming
Unity	Scala	Go
TypeScript	Go	Rust
	Python	

Langages de programmation et buts professionnels

- **Front-end web development:** JavaScript
- **Back-end web development:** JavaScript, Java, Python, PHP, Ruby
- **Mobile development:** Swift, Java, C#
- **Game development:** C++, C#
- **Desktop applications:** Java, C++, Python
- **Systems programming:** C, Rust

Langages de programmation et histoire



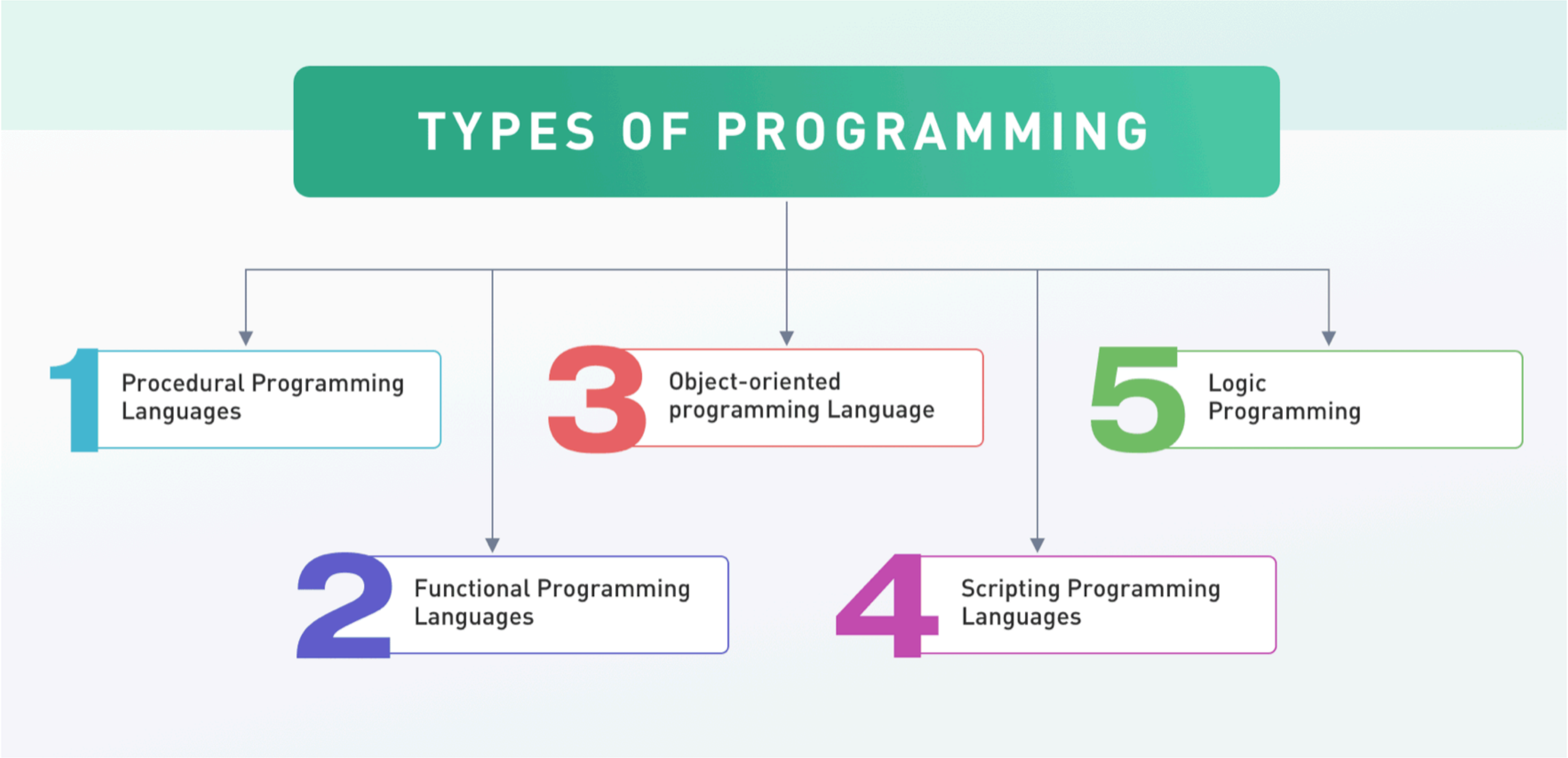
TIOBE Index for February 2021

Feb 2021	Feb 2020	Change	Programming Language	Ratings	Change
1	2	↑	C	16.34%	-0.43%
2	1	↓	Java	11.29%	-6.07%
3	3		Python	10.86%	+1.52%
4	4		C++	6.88%	+0.71%
5	5		C#	4.44%	-1.48%
6	6		Visual Basic	4.33%	-1.53%
7	7		JavaScript	2.27%	+0.21%
8	8		PHP	1.75%	-0.27%
9	9		SQL	1.72%	+0.20%
10	12	↑	Assembly language	1.65%	+0.54%
11	13	↑	R	1.56%	+0.55%
12	26	⇈	Groovy	1.50%	+1.08%
13	11	↓	Go	1.28%	+0.15%
14	15	↑	Ruby	1.23%	+0.39%
15	10	⇊	Swift	1.13%	-0.33%
16	16		MATLAB	1.06%	+0.27%
17	18	↑	Delphi/Object Pascal	1.02%	+0.27%
18	22	⇈	Classic Visual Basic	1.01%	+0.40%
19	19		Perl	0.93%	+0.23%
20	20		Objective-C	0.89%	+0.20%
21			Scratch	0.82%	
22			SAS	0.66%	
23			Fortran	0.63%	
24			D	0.59%	
25			COBOL	0.58%	
26			Transact-SQL	0.55%	
27			Prolog	0.54%	
28			PL/SQL	0.52%	
29			Julia	0.52%	
30			Rust	0.49%	
31			Ada	0.47%	
32			Dart	0.42%	
33			(Visual) FoxPro	0.42%	
34			ABAP	0.38%	
35			Lisp	0.34%	
36			Scala	0.34%	
37			Lua	0.34%	
38			Logo	0.33%	
39			Kotlin	0.32%	
40			TypeScript	0.29%	
41			VHDL	0.26%	
42			Bash	0.25%	
43			LabVIEW	0.24%	
44			Haskell	0.24%	
45			VBScript	0.24%	
46			Ladder Logic	0.23%	
47			Apex	0.23%	
48			Elixir	0.22%	
49			Alice	0.22%	
50			PowerShell	0.21%	

TIOBE Index for February 2021

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Java	2	1	1	1	3	28	-	-
Python	3	5	6	7	24	15	-	-
C++	4	3	3	3	2	2	2	8
C#	5	4	5	6	9	-	-	-
JavaScript	6	7	10	9	6	30	-	-
PHP	7	6	4	4	19	-	-	-
R	8	14	39	-	-	-	-	-
SQL	9	-	-	-	-	-	-	-
Go	10	57	16	-	-	-	-	-
Perl	14	9	7	5	4	3	-	-
Lisp	29	24	13	13	17	5	3	2
Ada	33	23	21	15	15	6	9	3

Types de langages de programmation



Les langages du cours

- Python [Guido van Rossum, 1995]
- C [Dennis Ritchie, 1978]
- Ocaml [Xavier Leroy, 1985]
- Haskell [Simon Peyton-Jones, 1990]
- Java [James Gosling, 1995]
- C++ [Bjarne Stroustrup, 1984]
- on télécharge Python 3 en `http://www.python.org`
- C et C++ sont en natifs sur pratiquement tous les systèmes
- on télécharge Ocaml en `http://www.ocaml.org`, et Haskell en `http://www.haskell.org`
- on télécharge Java en `http://www.java.com`

Les langages du cours

- Python (fil conducteur)
- C

} langages impératifs

- Ocaml
- Haskell

} langages fonctionnels

- Java
- C++

} langages orienté-objets

et plus brièvement

- langages de script (php, perl)
- programmation logique (prolog, sql)
- programmation concurrente (rust, modula3, ada)

Exemples de programmes

- Python (fil conducteur)
- C

} langages impératifs

- Ocaml
- Haskell

} langages fonctionnels

- Java
- C++

} langages orienté-objets

- copie de tableaux
- tri des éléments d'un tableau

} programmation impérative

- arbre binaire de recherche
- file d'attente

} structures de données dynamiques

- interface graphique
- modularité

} réutilisabilité

Ecrire des programmes

- utiliser un système intégré (Visual Studio ou autre)
- ou utiliser une simple fenêtre terminal [le plus rapide pour démarrer]
- avec un éditeur de texte (Emacs, VI, TextEdit, ..)

exemple avec Python

- sur la fenêtre terminal, on tape:

```
mac$ python
Python 3.8.8 (default, Feb 20 2021, 17:46:49)
[Clang 12.0.0 (clang-1200.0.32.27)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
```

- et on peut fonctionner en mode calculette:

```
>>> 23 + 42
65
>>> 438 * 234
102492
>>> (438 * 234) + 35
102527
>>> ((438 * 234) + 35 / 3)
102503.66666666667
```

Débuter en Python

- mode calculette - *toplevel* interactif

```
>>> x = 45
>>> 3 * x + 2
137
```

- avec des chaînes de caractères

```
>>> print ("Bonjour les amis !")
Bonjour les amis !

>>> y = "Bonjour les amis"
>>> y + " et la famille"
'Bonjour les amis et la famille’

>>> z = " !"
>>> y + " et la famille" + z
'Bonjour les amis et la famille !'
```

```
>>> s = "Bonjour les amis !"
>>> s[0]
'B'
>>> s[1]
'o'
>>> s[2]
'n'
>>> s[9]
'e'
>>> s[-1]
'!'
>>> s[-2]
' '
>>> s[-3]
's'
```

Débuter en Python

- définir des fonctions

```
>>> def double (x) :
        return (2 * x)

>>> double(3)
6


>>> def concatene (s1, s2) :
       return (s1 + " " + s2)

>>> concatene ("Hello", "World !")
'Hello World !'
```

Premiers programmes

Le calcul de la date de Pâques

Premier dimanche après la première lune
qui suit l'équinoxe de printemps.

Soit Y l'année, dont on cherche la date de Pâques.

1. Golden number G = (Y mod 19)+1
2. Century C=⌊Y/100⌋+1
3. Corrections X = ⌊3C/4⌋ − 12, Z = ⌊(8C + 5)/25⌋ − 5
4. Find Sunday D = ⌊5Y/4⌋−X −10
5. Epact E = (11G+20+Z −X) mod 30.
   Si E=25 et G>11, ou si E=24, alors E←E+1
6. Find full moon N=44−E. SiN<21,alorsN←N+30
7. Advance to Sunday N←N+7−((D+N)mod7)
8. Getmonth Si N>31, la date est le (N−31) AVRIL
   Sinon, la date est le N MARS.

```
>>> def paques (y) :
        g = (y % 19) + 1
        c = y // 100 + 1
        x = 3 * c // 4 - 12
        z = (8 * c + 5) // 25 - 5
        d = 5 * y // 4 - x - 10
        e = (11 * g + 20 + z - x) % 30
        if  e == 25 and g > 11 or e == 24 :
            e = e + 1
        n = 44 - e
        if n < 21 :
            n = n + 30
        j = n + 7 - ((d + n) % 7)
        if j > 31 :
            print (str(j - 31) + " avril")
        else :
            print (str(j) + " mars")
```

% modulo

// division entière

== test d'égalité

+ concaténation

```
>>> paques (2021)
4 avril
>>> paques (2020)
12 avril
>>> paques (2024)
31 mars
```

http://fr.wikipedia.org/wiki/Calcul_de_la_date_de_Pâques

Premiers programmes

- impression

```
>>> def concat_print0 (s1, s2):
        print (s1 + " " + s2)

>>> concat_print0 ("Hello", "World !")
Hello World !
```

+ concaténation

```
>>> def concat_print1 (s, n, x):
        print ("%s vaut %d ou %.2f" %(s, n, x))

>>> concat_print1 ("Le resultat", 32, 28.5)
Le resultat vaut 32 ou 28.50
```

`%s, %d, %f` formats d'impression (comme en C)

```
>>> def concat_print2 (s, n, x):
        print ("{} vaut {} ou {}".format (s, n, x))

>>> concat_print2 ("Le resultat", 32, 28.5)
Le resultat vaut 32 ou 28.5
```

`{}` `.format` autre style pour formats d'impression

Premiers programmes

- constantes litérales: nombres (entiers `int`, flottants `float`, complexes `complex`)

```
>>> a = 5
>>> print(a, "is of type", type(a))
5 is of type <class 'int'>

>>> a = 2.0
>>> print(a, "is of type", type(a))
2.0 is of type <class 'float'>

>>> a = 1+2j
>>> print(a, "is complex number?", isinstance(1+2j,complex))
(1+2j) is complex number? True
```

- constantes litérales: chaînes de caractères `string`

```
>>> s = "This is a string"
>>> print(s)
This is a string

>>> s = '''A multiline
    string'''
>>> print(s)
A multiline
string
```

- constantes litérales: booléens `bool`

```
>>> True
True
>>> type (True)
<class 'bool'>
>>> type (False)
<class 'bool'>
```

Premiers programmes

- quelques programmes sur des données scalaires

```
>>> import math
>>>
>>> def surface (r) :
        print ("%.2f" %(4 * math.pi * r ** 2))

>>> surface (3)
113.10
>>> surface (1.3)
21.24
```

- calculer la surface d’un carré ou triangle

- calculer le volume d’un cube ou sphère

Premiers programmes

- opérations de comparaison

```
>>> 3 == 4
False
>>> 3 != 4
True
>>> 3 < 4
True
>>> 3 <= 4
True
>>> True and False
False
>>> True or False
True
```

- expressions booléennes composites

```
>>> a = 4
>>> b = 5
>>> a > 8 and b < 16
False
>>> not a > 8
True
```

- itération while

```
>>> i = 0
>>> while i < 10 :
        i = i + 1
        print (i)

1
2
3
4
5
6
7
8
9
10
```

Ecrire des programmes

- utiliser un système intégré (Visual Studio ou autre)
- ou utiliser une simple fenêtre terminal [le plus rapide pour démarrer]
- avec un éditeur de texte (Emacs, VI, TextEdit, ..)

exemple avec C

- dans un éditeur de texte, on crée le fichier bonjour.c

```
#include <stdio.h>

char s[100] = "bonjour les amis! » ;

main()
{
    printf ("\%s\n", s);
}
```

- sur la fenêtre terminal, on tape:

```
mac$ cc bonjour.c
mac$ a.out
bonjour les amis!
```

Prochain cours

- un bon tutorial Python: `http://www.programiz.com/python-programming`
- les langages procéduraux