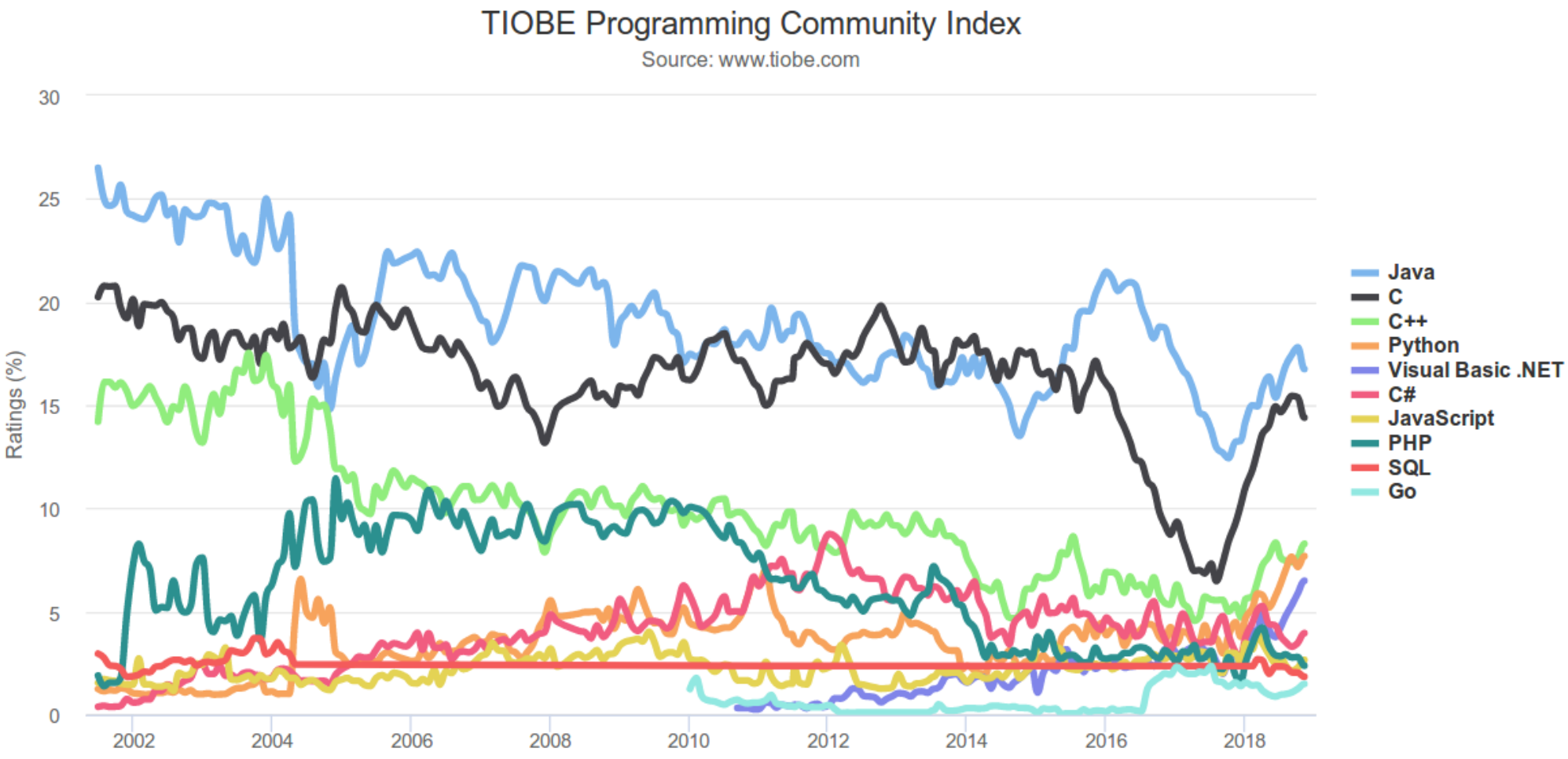
Présentation du langage Python

- 1989 : Guido van Rossum
- Licence libre
- Langage interprété, orienté objet
- Multiplateforme (Windows, Linux, Mac, etc.)
- Facile à apprendre
- Populaire dans le milieu éducatif et universitaire
- Usage professionnel

Classement TIOBE



Python est 4ème derrière Java, C et C++

Versions

- La version majeure en cours est la version 3 mais la version 2 est encore largement utilisée

- Pas de compatibilité ascendante entre la version 2 et la version 3 :(

Installation

- Téléchargement et procédure d'installation :

  Python 3.7.1 (octobre 2018)

  Python 2.7.15 (mai 2018)

  https://www.python.org/downloads/

Modules (librairies)

- Le richesse de Python réside dans le nombre impressionnant de modules disponibles (120 000 actuellement)

- PyPI - the Python Package Index

  https://pypi.python.org/pypi

Quelques modules célèbres

- calcul scientifique (modules *NumPy* et *SciPy*)
- graphiques (module *matplotlib*)
- traitement d'images (module *PIL*)
- vision artificielle par caméra (framework *SimpleCV*)
- bio-informatique (module *Biopython*)
- interface graphique GUI (modules *Tkinter*, *PyQt*, *wxPython*, *PyGTK*)
- applications multi-touch (framework *kivy* pour tablette et smartphone à écran tactile)
- applications Web (serveur Web *Zope* ; frameworks Web *Flask*, *Django*)

Quelques modules célèbres

- systèmes de gestion de base de données (module *SQLalchemy*)
- analyses big data (*pandas*)
- applications réseau (framework *twisted*)
- communication avec des ports série RS232 (module *PySerial*), en Bluetooth (module *pybluez*)
- multitâches (module *threading*)

- les modules *py2exe* (sous Windows) et *cx_Freeze* permettent de rendre vos scripts Python exécutables :)

Modules standards et modules externes

- Les modules standards sont présents par défaut dans la distribution Python
  - *math*, *time*, *sys*, …
- Les modules externes doivent être ajoutés (la procédure d'installation est variable suivant l'OS)
  - Exemple : pour installer le module *matplotlib* sur une Raspberry Pi (distribution linux raspbian), il suffit de taper la commande système suivante :

    sudo apt-get install python-matplotlib (python 2)

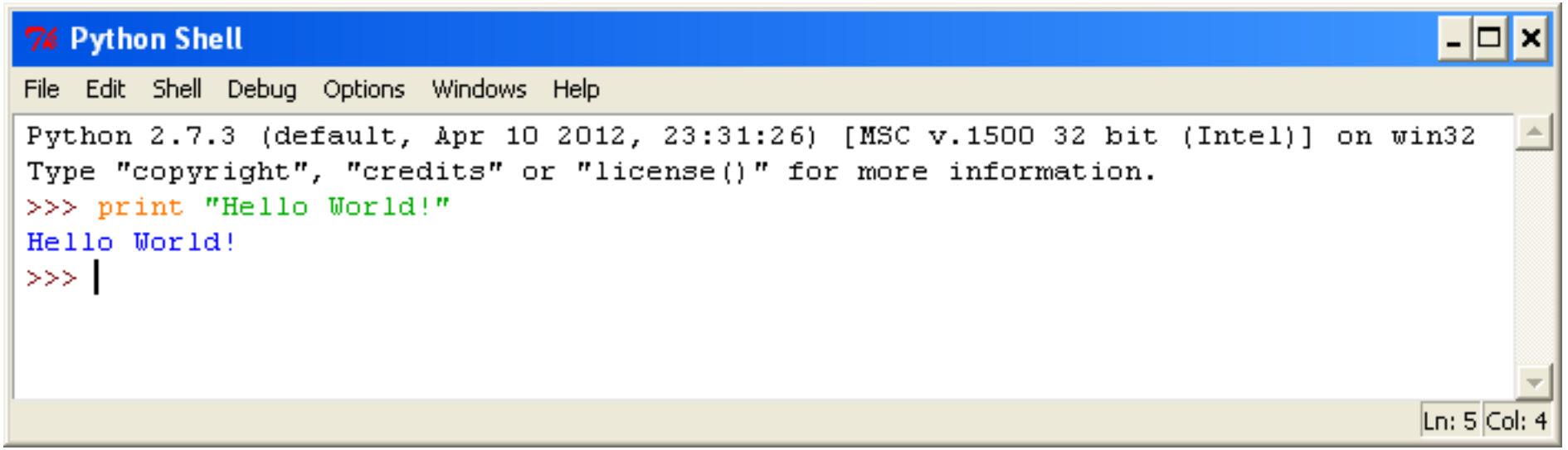
    sudo apt-get install python3-matplotlib (python 3)

Python et C/C++ sont complémentaires

- Intégration dans Python de modules écrits en C/C++
- Intégration dans C/C++ de modules écrits en Python

Environnement de développement (IDE)

- IDE par défaut : IDLE
  - Éditeur de code
  - Interpréteur
  - Débogueur



Environnement de développement (IDE)

- Autres IDE
  - Eclipse (avec le plugin pydev)
  - Eric
  - Spyder
  - ipython (en mode console)
  - Geany
  - Notepad++

Python 2 – Les variables

- Typage dynamique (déclaration d'une variable sans préciser explicitement son type)
- Principaux types :
  - Entier (int)
  - Nombre « flottant » (float)
  - Chaîne de caractères (str)
  - Booléen (bool)
  - Liste (list)
  - Dictionnaire (dict)

Python 2 – Type int

```
>>> a = 5
>>> print a
5
>>> print type(a)
<type 'int'>
>>> b, c = 8, 3
>>> res = b / c   # attention : retourne un entier
>>> print res
2
```

Python 2 – Type float

```
>>> a = 12.0
>>> b = -8.23e3
>>> c = 81.5385
>>> D = b**2 - 4*a*c
>>> print D
67728986.152
```

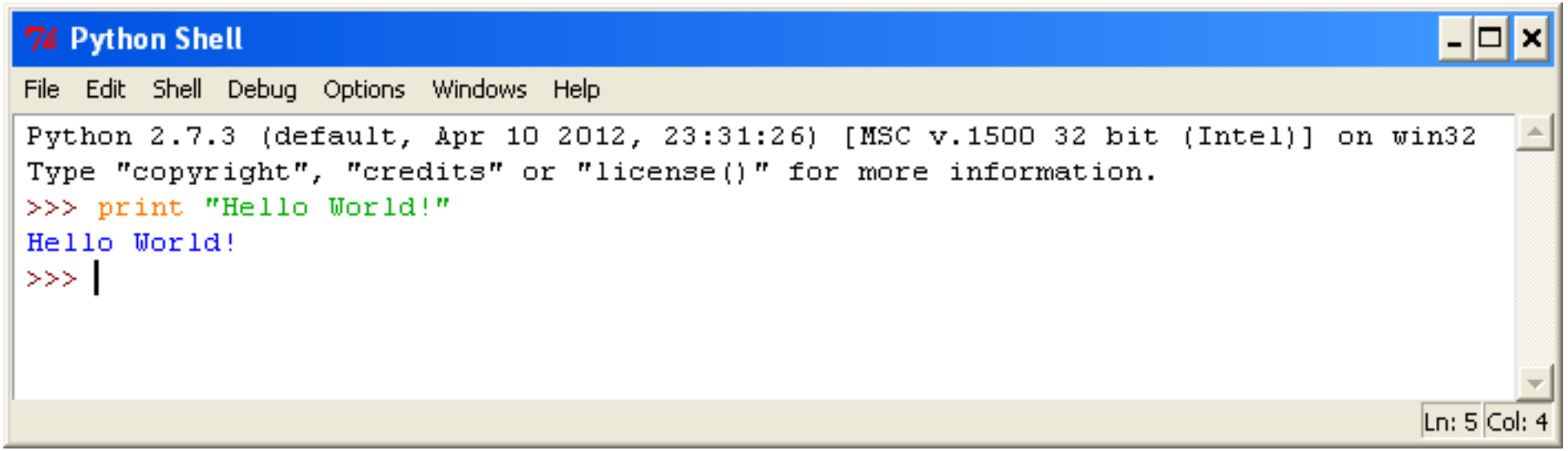
Python 2 – Le module standard math

```
>>> import math
>>> a = math.sqrt(5)
>>> print a
2.23606797749979
>>> b = math.sin(math.pi/3)
>>> print b
0.8660254037844386
```

Python 2 – L'instruction help

```
>>> help(math)   # aide sur le module math
>>> help(math.sin)
Help on built-in function sin in module math:
sin(...)
    sin(x)
    Return the sine of x (measured in radians).
```

Python 2 – Type str (chaîne de caractères)



Python 2 – Type str (chaîne de caractères)

```
>>> nom = "Dupont"
>>> print nom
Dupont
>>> prenom = "Pierre"
```

- Concaténation

```
>>> chaine = prenom + "  " + nom
>>> print chaine
Pierre  Dupont
```

Python 2 – Type str (chaîne de caractères)

```
>>> chaine = " " "Première ligne
Deuxième ligne
Troisième ligne" " "
>>> print chaine
Première ligne
Deuxième ligne
Troisième ligne
```

Python 2 – La fonction raw_input()

- En mode console
- Retourne un type str

```
>>> nom = raw_input("Entrer votre nom : ")
Entrer votre nom : Dupont
>>> print nom
Dupont
```

Python 2 – Conversion de types

- Fonctions int(), float() et str()

```
>>> nombre = float(raw_input("Entrer un nombre : "))
Entrer un nombre : 4
>>> print nombre**2
16.0
```

Python 2 – Formatage des données

- Fonction format()

```
>>> nom, age, masse = 'Dupont', 18, 72.4
>>> print "Mon nom est {}, âge {} ans et masse {} kg".format(nom, age, masse)
Mon nom est Dupont, âge 18 ans et masse 72.4 kg
>>> print "Mon nom est {}, âge {} ans et masse {:.3f} kg".format(nom, age, masse)
Mon nom est Dupont, âge 18 ans et masse 72.400 kg
```

Python 2 – Le type list

- Une liste est une structure de données.
- Le premier élément d'une liste possède l'indice (l'index) 0.
- Dans une liste, on peut avoir des éléments de plusieurs types.

```
>>> infoperso = ['Pierre', 'Dupont', 18, 1.75]
>>> print infoperso[2]        # le troisième élément
18
```

Python2 – Le type list

```
>>> infoperso = ['Pierre', 'Dupont', 18, 1.75]
```

- Ajout d'un élément avec la méthode append()

```
>>> infoperso.append(72.4)
>>> print infoperso
['Pierre', 'Dupont', 18, 1.75, 72.4]
```

Python 2 – La fonction range()

- La fonction range() crée une liste d'entiers

```
>>> maliste = range(1, 11) # fin non comprise
>>> print maliste
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Python 2 – Le type dict

- Un dictionnaire stocke des données sous la forme clé ⇒ valeur
- Une clé est unique et n'est pas nécessairement un entier (comme c'est le cas de l'indice d'une liste).

```
>>> moyennes = {'math': 12.5, 'anglais': 15.8}  # entre accolades
>>> print moyennes['anglais']      # entre crochets
15.8
>>> moyennes['anglais'] = 14.3        # nouvelle affectation
>>> moyennes['sport'] = 11.0       # nouvelle entrée
>>> print moyennes
{'sport': 11.0, 'anglais': 14.3, 'math': 12.5}
```

Python 2 – Le type bool (booléen)

- Deux valeurs sont possibles : True et False
- Opérateurs de comparaison :

  <

  <=

  >

  >=

  ==

  !=

Python 2 – Le type bool (booléen)

```
>>> b = 10
>>> print b > 8
True
>>> print b == 5
False
>>> print 0 <= b <= 10
True
```

Python 2 – Le type bool (booléen)

- Les opérateurs logiques : and, or, not

```
>>> note = 13.5
>>> horslimite = not(0.0 <= note <= 20.0)
>>> print horslimite
False
```

Python 2 – L'opérateur in

- L'opérateur in s'utilise avec des chaînes (type str) ou des listes (type list) :

```
>>> chaine = 'Bonsoir'
>>> print 'soir' in chaine
True
>>> maliste = [4, 8, 15]
>>> print 9 in maliste
False
```

Python 2 – Les conditions

- Syntaxe

```
if expression :
        bloc d'instructions      # indentation
else :              # else est au même niveau que if
        bloc d'instructions      # indentation
# suite du programme
```

Python 2 – Les conditions

```
temperature = float(raw_input("Température en °C : "))
if temperature >= 19.0:
        print "Il fait bon"
else:
        print "Il fait froid"
print "Suite du programme"
```

```
>>> Température en °C : 20
Il fait bon
Suite du programme
```

Python 2 – Les conditions

- Instruction elif (sinon si)

```
temperature = float(raw_input("Température en °C : "))
if temperature >= 36.0:
        print "Canicule !"
elif temperature >= 19.0 :
        print "Il fait bon"
else:
        print "Il fait froid"
print "Suite du programme"
```

Python 2 – Les conditions

- Remarque : pas de switch / case / break comme en langage C

```
cas = 2
if cas == 1:
        print "Cas 1"
elif cas == 2:
        print "Cas 2"
elif cas == 3:
        print "Cas 3"
else :
        print "Default"
```

Python 2 – Les boucles while

- Syntaxe

```
while expression :
        bloc d'instructions          # indentation
# suite du programme
```

Python 2 – Les boucles while

```
i = 1
while i < 5:
        print i
        i += 1    # incrémentation
# suite du programme
```

```
>>> 1
2
3
4
```

Python 2 – Les boucles while

- L'instruction break

```
i = 1
while True:
        print i
        i += 1    # incrémentation
        if i > 4 :
                break
# suite du programme
```

Python 2 – Les boucles while

- Remarque : pas de do / while comme en langage C

```
while True:
        bloc d'instructions
        if expression:
                break
# suite du programme
```

Python 2 – Les boucles for

- Syntaxe

```
for element in sequence :
        bloc d'instructions     # indentation
# suite du programme
```

- Les éléments de la séquence sont issus d'une chaîne de caractères ou bien d'une liste.

Python 2 – Les boucles for

```
for i in range(1, 5):
        print i
# suite du programme


>>>
1
2
3
4
```

Python 2 – Les fonctions

```
def nomdelafonction(parametre1, parametre2, etc.):
        " " " Documentation
qu'on peut écrire
sur plusieurs lignes " " "          # docstring
        bloc d'instructions         # indentation
        return resultat
```

Python 2 – Les fonctions

```
def fahrenheit(degrecelsius):
    " " " Conversion degré Celsius en degré Fahrenheit " " "
    resultat = degrecelsius*9.0/5.0 + 32.0
    return resultat


>>> print fahrenheit(100)
212.0
>>> help(fahrenheit)
Help on function fahrenheit :
fahrenheit(degrecelsius)
   Conversion degré Celsius en degré Fahrenheit
```

Python 2 – Variables globales et locales

```
a = 10      # variable globale
b = 5
def mafonction():
        global a     # la variable est maintenant globale
        a = 20
        b = 100     # variable locale
        return [a, b]     # retourne une liste
>>> print mafonction()
[20, 100]
>>> print a, b
20    5
```

Python 2 – L'instruction pass

```
def unefonction() :
    # L'instruction pass ne fait rien !
    # Cela peut servir à définir une fonction
    # qui pour l'instant ne contient rien
    pass
```

Python 2 – Gestion des exceptions

```
valeur = 0
inverse = 1 / valeur
print inverse
print "Suite du programme"

>>>
Traceback (most recent call last):
  File "prog1.py", line 2, in <module>
    inverse = 1 / valeur
ZeroDivisionError: integer division or modulo by zero
```

Python 2 – Gestion des exceptions

```
valeur = 0
try :
        inverse = 1 / valeur
        print inverse
except :
        print "Oups ! Division par zéro !"
print "Suite du programme"
```

```
>>> Oups ! Division par zéro !
Suite du programme
```

Python 2 – Sortir d'un programme

```
try :
        import matplotlib
except :
        print "Impossible d'importer le module matplotlib"
        exit(1)   # on quitte le programme avec le code d'erreur 1
# on continue ici si l'importation a réussi
```

Python 2 – Programmation orientée objet

- Comme en C++, on retrouve les notions :
  - de classe
  - d'instances de classe
  - les méthodes et propriétés
  - les méthodes spéciales : constructeur, destructeur
  - les surcharges d'opérateurs
  - l'héritage de classes
  - etc.

Python 2 versus Python 3

- print 'bonjour'
- a = 5/2

  donne un entier (2)
- raw_input()
- import Tkinter

- print('bonjour')
- a = 5/2

  donne un float (2.5)
- input()
- import tkinter

Copyright

Document réalisé par Fabrice Sincère

Fabrice.sincere@wanadoo.fr

- Téléchargement :

  http://fsincere.free.fr/isn/python/download/diaporama/diaporama_presentation_python.pdf
- Bibliographie :

  http://fsincere.free.fr/isn/python/cours_python.php?version=2
- Version du document : 1.3 (novembre 2018)
-