

# FORMATION PYTHON



## UN PEU D'HISTOIRE

- Python est un langage de programmation, dont la première version est proposée par *Guido Van Rossum* programmeur néerlandais qui a travaillé pour Google pendant sept ans . Il a travaillé aussi chez Dropbox.
- Il fonctionne sur la plupart des plateformes informatiques (notamment Linux, Windows et MacOS).
- Appuyé par  
Python Software Foundation



## CARACTÉRISTIQUES

- Un langage de programmation interprété
- Simplicité

```
#include <stdio.h>
```

```
int main(void)
{
    printf("hello, world\n");
}
```

Langage C

```
print('Bonjour Tout le Monde')
```

Langage Python

- Les Communautés de Python sont vastes et diversifiées.

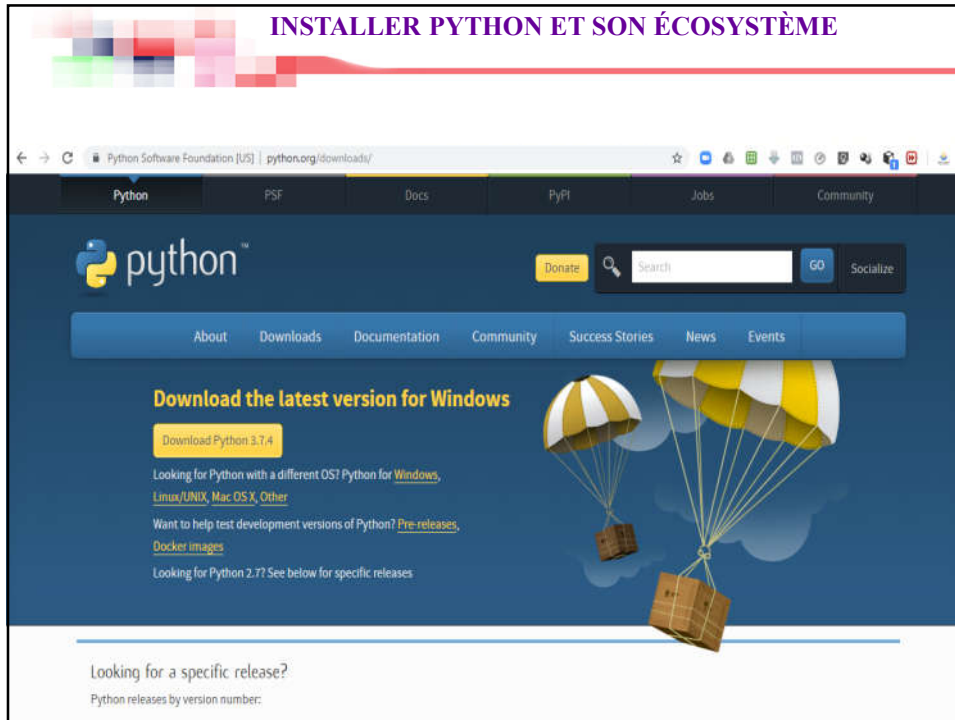
## CARACTÉRISTIQUES

Python possède des librairies pour à peu près tout ce que vous pouvez imaginer, et notamment:

- Numpy et Scipy pour les calculs matriciels
- Pandas pour gérer les données (les charger, appliquer des opérations d'algèbre relationnelle, etc.)
- Matplotlib et Seaborn pour la visualisation
- Scikit-learn pour les algorithmes IA
- Tensorflow et PyTorch pour le deep learning.



## INSTALLER PYTHON ET SON ÉCOSYSTÈME



The screenshot shows the Python Software Foundation website. The browser address bar displays "python.org/downloads/". The page features the Python logo, a search bar, and a navigation menu with links to About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area is titled "Download the latest version for Windows" and includes a button to "Download Python 3.7.4". Below this, there are links for other operating systems (Linux/UNIX, Mac OS X, Other), pre-releases, and Docker images. A graphic of two parachutes with boxes is also visible.

**Download the latest version for Windows**

[Download Python 3.7.4](#)

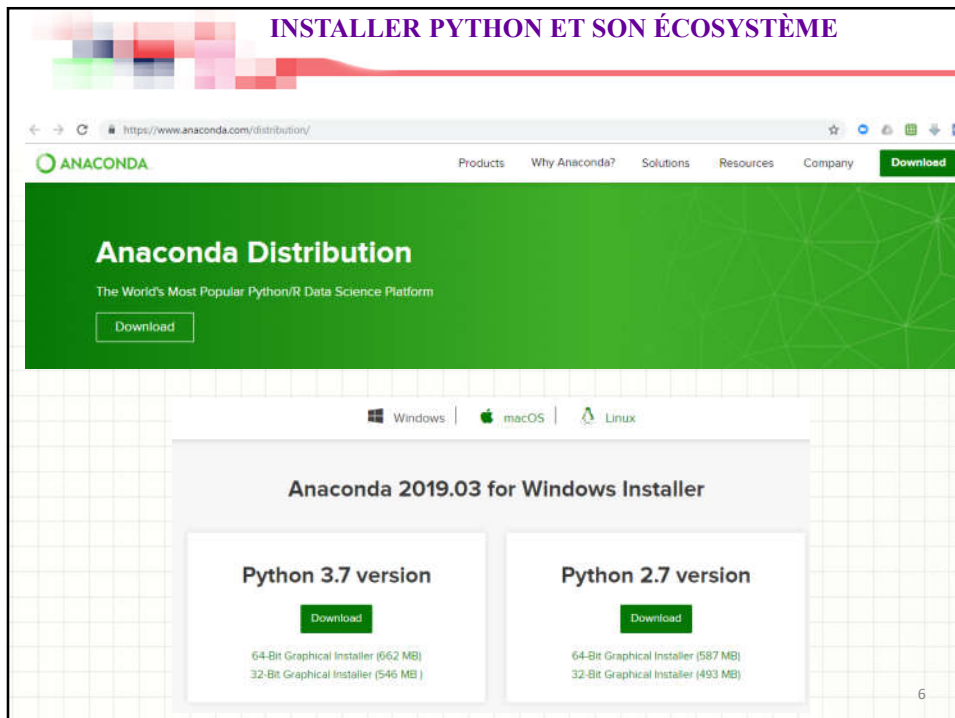
Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Pre-releases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

Looking for a specific release?  
Python releases by version number:

## INSTALLER PYTHON ET SON ÉCOSYSTÈME



The screenshot shows the Anaconda Distribution website. The browser address bar displays "https://www.anaconda.com/distribution/". The page features the Anaconda logo, a navigation menu with links to Products, Why Anaconda?, Solutions, Resources, Company, and a Download button. The main content area is titled "Anaconda Distribution" and includes the subtitle "The World's Most Popular Python/R Data Science Platform". Below this, there is a "Download" button. The page also features a section for "Anaconda 2019.03 for Windows Installer" with two options: "Python 3.7 version" and "Python 2.7 version". Each option has a "Download" button and links to 64-bit and 32-bit graphical installers.

**Anaconda Distribution**

The World's Most Popular Python/R Data Science Platform

[Download](#)

Windows | macOS | Linux

**Anaconda 2019.03 for Windows Installer**

**Python 3.7 version**

[Download](#)

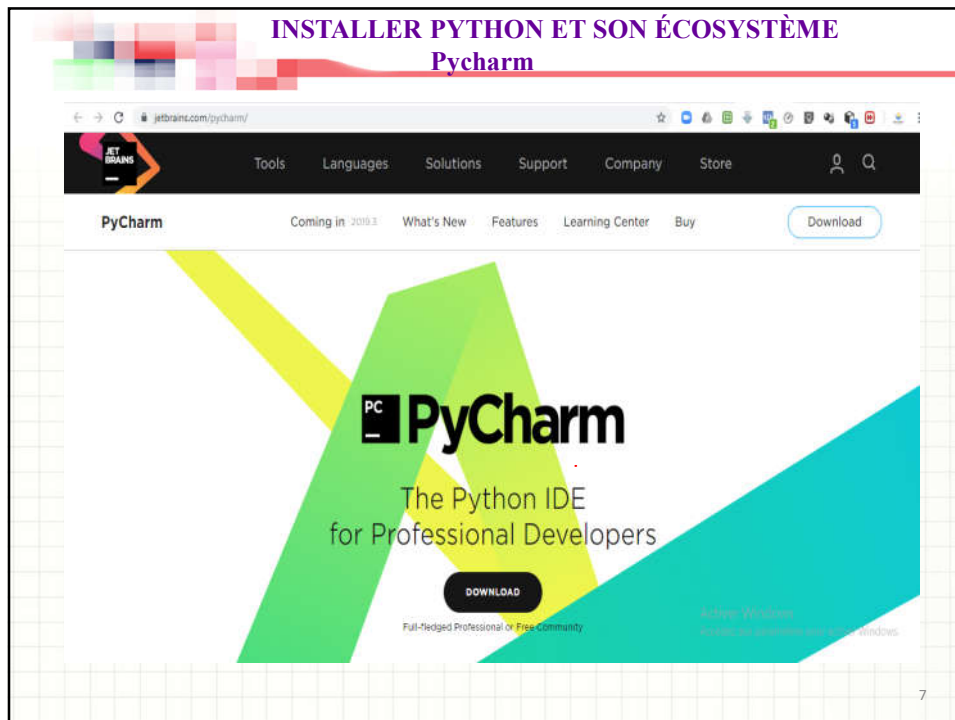
64-Bit Graphical Installer (662 MB)  
32-Bit Graphical Installer (546 MB)

**Python 2.7 version**

[Download](#)

64-Bit Graphical Installer (587 MB)  
32-Bit Graphical Installer (493 MB)

6



## LES BASES SUR LES VARIABLES

- Pas besoin de déclarer ou typer explicitement une variable avant de lui affecter une valeur. Il suffit de faire :

`x=120`

`x = 'Bonjour'`

- Le nom d'une variable peut commencer par n'importe lettre minuscule ou majuscule ou un '\_', puis des lettres, des chiffres ou des '\_'.
- Les noms de variables sont sensibles à la casse (age, Age et AGE sont trois variables différentes)
- Une variable sans valeur est définie par : `x = None`  
(None est l'équivalent de null dans d'autres langages).

9

## LES BASES SUR LES VARIABLES

### Types primitifs

#### Liste des types

|                |   |
|----------------|---|
| <b>int</b>     | <b>Nombre entier optimisé</b>             |
| <b>long</b>    | <b>Nombre entier de taille arbitraire</b> |
| <b>float</b>   | <b>Nombre à virgule flottante</b>         |
| <b>complex</b> | <b>Nombre complexe</b>                    |
| <b>str</b>     | <b>Chaîne de caractère</b>                |
| <b>list</b>    | <b>Liste de longueur variable</b>         |
| <b>file</b>    | <b>Fichier</b>                            |
| <b>bool</b>    | <b>Booléen</b>                            |
| <b>module</b>  | <b>module</b>                             |

10

## LES BASES SUR LES VARIABLES

### Visibilité des variables

#### Exemple

```
x = str("Bonjour")  
x = int(20)  
x = float(20.5)
```

```
print(type(x))
```

- Une variable définie en dehors de toutes les fonctions est globale.
- par défaut, une variable est toujours locale, donc disponible seulement dans la fonction dans laquelle elle est définie.

11

## LES BASES SUR LES VARIABLES

### conversion des types

Il existe plusieurs fonctions qui permettent de forcer le type d'une variable en un autre type.

- `int()` : permet de modifier une variable en entier. Provoque une erreur si cela n'est pas possible.
- `long()` : transforme une valeur en long.
- `str()` : permet de transformer la plupart des variables d'un autre type en chaînes de caractère.
- `float()` : permet la transformation en flottant.
- `split()` : sépare une chaîne en liste.
- `round(a,3)` : arrondir un nombre flottant

12

**LES ENTRÉES/ SORTIES**  
**La fonction print ()**

La fonction print() sert à afficher des données sur la sortie standard, qui est l'écran.

**Affichage d'une chaîne de caractères**  
`print('Hello World!')` → **Hello World!**

**Affichage d'un nombre**  
`print(-1.5)` → **-1.5**

**Affichage du Calcul**  
`print(32+7*2)` → **46**

**Affichage d'une Variable**  
`Var="Laila"`  
`print(Var)` → **Laila**

**calcul de 2 variables**  
`a=3`  
`b=7`  
`print(a+b)` → **10**

13

**LES ENTRÉES/ SORTIES**  
**La fonction print ()**

**Affichage d'une variable de type string**  
`Nom="Walid"`  
`Age=25`  
`print("Bonjour ",Nom, "votre age :",Age,"ans.")`

**Bonjour Walid votre age : 25 ans.**

**Imprimer un espace vide**  
`print("\n\n\n\n\n")`  
**Ou**  
`print(5* "\n")`

**Imprimer sans retour à la ligne**  
`print("Bonjour",end="")`  
`print("Python")`

**BonjourPython**

14



## LES ENTRÉES/ SORTIES

### La fonction input ( )

La fonction input ( ) demande à l'utilisateur d'entrer une donnée qui va servir de valeur affectée à une variable.

```
a = input("Quel est votre nom ? ")
Quel est votre nom ? Reda
Print(a)
'Reda'
```

Attention, par défaut, cette valeur est de type "string" ou "chaîne de caractère". si vous voulez un autre type de variable, il faudra la convertir.

```
a = int(input("« donner? "))
Quel est votre nom ? Reda
Print(a)
'Reda'
```

15

## Atelier 1: Premier contact avec Python



**Travaillons ensemble- Atelier**



## COMMENTAIRES ET DOCUMENTATION

- Le commentaire sur une seule ligne est précédé d'un caractère dièse #.
- Le commentaire sur plusieurs lignes doit être entouré de triples guillemets.

```
print(1) # voici un commentaire
# print(2)
print(3) # un autre commentaire @!#!@$
```

```
""" Ceci est un commentaire ou documentation
   que j'écris sur plusieurs
   lignes """
```

17

## OPÉRATEURS DE COMPARAISONS

- == : égalité (pour des nombres ou des chaînes).
- != : inégalité (pour des nombres ou des chaînes).
- > >= < <= : comparaison
- is, is not permettent de comparer l'identité des objets.

### Opérateurs booléens

- and, or, not
- xor en python : bool(a) != bool(b)

18

## OPÉRATIONS

### Opérations diverses

- Quotient entier :  $7 / 2$  donne 3.
- Modulo (reste) :  $7 \% 2$  donne 1.
- Conversion en entier : `int(7.6)` donne 7 (attention `int(-7.6)` donne -7).
- `pow(2, 5)` ou  $2 ** 5$  : 2 puissance 5.
- Arrondissement : `round(3.14159)` : arrondissement à l'entier le plus proche.
- Arrondissement : `round(3.14159, 2)` : arrondissement à 2 chiffres après la virgule.

19

## OPÉRATIONS

`print(round(46.14559, 1))` → 46.1  
`print(round(46.14559, 2))` → 46.15  
`print(round(46.14559, -1))` → 50.0  
`print(round(56.14559, -2))` → 100.0

20

## Atelier 1: Premier contact avec Python



Travaillons ensemble- Atelier

## LES STRUCTURES CONDITIONNELLES

```
conditions :  
if x < 0:  
    print('negative')  
elif x == 0:  
    print('zero')  
else:  
    print('positive')
```

```
x=0  
if x < 0:  
    print('negative')  
elif x == 0:  
    print('zero')  
else:  
    print('positive')
```

zero

22

## LES BOUCLES

boucle while :  
while x < 10:  
    print(x)

```
x=5
while x < 10:
    print(x)
    x+=1
```

5  
6  
7  
8  
9

23

## LES BOUCLES

### La boucle FOR

```
for i in range (1,11):
    if i % 2 == 0 :
        print ("Le nombre ", i, "est pair")
    else :
        print ("Le nombre ", i, "est impair")
```

for i in range (1,11):

pour i dans l'intervalle de 1(inclus) à 11(exclus)  
donc i =1 jusqu'à i = 10

la variable i est déclarée dans la boucle for et  
prend toutes les valeurs à chaque tour de  
boucle.

Exécution :

```
>>>
Le nombre 1 est impair
Le nombre 2 est pair
Le nombre 3 est impair
Le nombre 4 est pair
Le nombre 5 est impair
Le nombre 6 est pair
Le nombre 7 est impair
Le nombre 8 est pair
Le nombre 9 est impair
Le nombre 10 est pair
>>>
```

24

## LES BOUCLES

### La boucle FOR

```
chaine = " Bonjour Python "
for x in chaine :
    print (x)
```

B  
o  
n  
j  
o  
u  
r  
  
P  
y  
t  
h  
o  
n

25

## LES MOTS-CLÉS BREAK ET CONTINUE

### Le mot-clé break

Le mot-clé break permet tout simplement d'interrompre une boucle.

```
while 1: # 1 est toujours vrai -> boucle infinie
    lettre = input (" Tapez 'Q' pour quitter : ")
    if lettre == "Q":
        print (" Fin de la boucle ")
        break
```

```
Tapez 'Q' pour quitter : 1
Tapez 'Q' pour quitter : 3
Tapez 'Q' pour quitter : Q
Fin de la boucle
```

### Le mot-clé continue

Le mot-clé continue permet de... continuer une boucle, en repartant directement à la ligne du while ou for.

```
for letter in 'Python': # First Example
    if letter == 'h':
        continue
    print 'Current Letter :', letter
```

26

## STRUCTURATION ET NOTION DE BLOC

Python utilise l'indentation pour définir les constructions de contrôle et de boucle. Cela contribue à la lisibilité de Python, cependant, le programmeur doit faire très attention à l'utilisation des espaces.

```
if a > b:      # If block starts here
    print(a)   # This is part of the if block
else:         # else must be at the same level as if
    print(b)   # This line is part of the else block
```

27

## COMMENTAIRES ET DOCUMENTATION

### Règles syntaxiques, blocs

- En Python, chaque instruction s'écrit sur une ligne sans mettre d'espace:

```
a = 10
b = 3
print(a, b)
```

- Les instructions simples peuvent cependant être mises sur la même ligne en les séparant par des points virgules ;,

```
a = 10; b = 3;
print(a, b)
```

le contenu ou 'bloc' d'instructions correspondant se fait par indentation des lignes.

```
instruction-1
en-tête-1:
instruction-2
instruction-2
en-tête-2:
instruction-3
instruction-3
en-tête-2:
instruction-3
instruction-3
instruction-2
instruction-2
instruction-1
instruction-1
```

28

