



Accueil > Cours >

Apprenez à

programmer en

Python > Faites vos

premiers pas avec

l'interpréteur de

commandes Python

# Apprenez à programmer en Python

40 heures



Difficile

Mis à jour le

23/06/2020



## Faites vos premiers pas avec l'interpréteur de commandes Python

Après les premières notions théoriques et l'installation de Python, il est temps de découvrir un peu l'interpréteur de commandes de ce langage. Même si ces petits tests

vous semblent  
anodins, vous  
découvrirez dans  
ce chapitre les  
premiers  
rudiments de la  
syntaxe du  
langage et je vous  
conseille  
fortement de me  
suivre pas à pas,  
surtout si vous êtes  
face à votre  
premier langage  
de  
programmation.

Comme tout  
langage de  
programmation,  
Python a une  
syntaxe claire : on  
ne peut pas lui  
envoyer n'importe  
quelle information  
dans n'importe  
quel ordre. Nous  
allons voir ici ce  
que Python  
mange... et ce qu'il  
ne mange pas.

## Où est-ce qu'on est, là ?

---



Pour commencer,  
je vais vous  
demander de  
retourner dans  
l'interpréteur de  
commandes  
Python (je vous ai

montré, à la fin du chapitre précédent, comment y accéder en fonction de votre système d'exploitation).


Je vous rappelle les informations qui figurent dans cette fenêtre, même si elles peuvent être différentes chez vous en fonction de votre version et de votre système d'exploitation.

```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:24:06) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

À sa façon, Python vous souhaite la bienvenue dans son interpréteur de commandes.

Attends, attends.  
C'est quoi cet interpréteur ?

Souvenez-vous, au chapitre précédent, je vous ai donné une brève explication sur la différence entre langages compilés et langages interprétés. Eh

bien, cet interpréteur de commandes va nous permettre de tester directement du code. Je saisis une ligne d'instructions, j'appuie sur la touche  de mon clavier, je regarde ce que me répond Python (s'il me dit quelque chose), puis j'en saisis une deuxième, une troisième... Cet interpréteur est particulièrement utile pour comprendre les bases de Python et réaliser nos premiers petits programmes. Le principal inconvénient, c'est que le code que vous saisissez n'est pas sauvegardé (sauf si vous l'enregistrez manuellement, mais chaque chose en son temps).

Dans la fenêtre que vous avez sous les yeux, l'information qui ne change pas d'un système d'exploitation à

l'autre est la série  
de trois chevrons  
qui se trouve en  
bas à gauche des  
informations :



. Ces trois signes  
signifient : « je suis  
prêt à recevoir tes  
instructions ».

Comme je l'ai dit,  
les langages de  
programmation  
respectent une  
syntaxe claire. Vous  
ne pouvez pas  
espérer que  
l'ordinateur  
comprenne si,  
dans cette fenêtre,  
vous commencez  
par lui demander :  
« j'aimerais que tu  
me codes un jeu  
vidéo génial ». Et  
autant que vous le  
sachiez tout de  
suite (bien qu'à  
mon avis, vous  
vous en doutiez),  
on est très loin  
d'obtenir des  
résultats aussi  
spectaculaires à  
notre niveau.

Tout cela pour dire  
que, si vous  
saisissez n'importe  
quoi dans cette  
fenêtre, la  
probabilité est  
grande que  
Python vous

indique,  
clairement et  
fermement, qu'il  
n'a rien compris.

Si, par exemple,  
vous saisissez «  
premier test avec  
Python », vous  
obtenez le résultat  
suivant :

pycon

```
1 >>> premier test
   avec Python
2 File "<stdin>",
   line 1
3 premier test avec
   Python
4 ^
5 SyntaxError:
   invalid syntax
6 >>>
```

Eh oui,  
l'interpréteur parle  
en anglais et les  
instructions que  
vous saisirez,  
comme pour  
l'écrasante  
majorité des  
langages de  
programmation,  
seront également  
en anglais. Mais  
pour l'instant, rien  
de bien compliqué  
: l'interpréteur vous  
indique qu'il a  
trouvé un  
problème dans  
votre ligne  
d'instruction. Il  
vous indique le  
numéro de la ligne  
(en l'occurrence la

première), qu'il  
vous répète  
obligamment  
(ceci est très utile  
quand on travaille  
sur un programme  
de plusieurs  
centaines de  
lignes). Puis il vous  
dit ce qui l'arrête,

SyntaxError:  
ici: invalid syntax

. Limpide n'est-ce  
pas ? Ce que vous  
avez saisi est  
incompréhensible  
pour Python. Enfin,  
la preuve qu'il n'est  
pas rancunier, c'est  
qu'il vous affiche à  
nouveau une série  
de trois chevrons,  
montrant bien qu'il  
est prêt à retenter  
l'aventure.

Bon, c'est bien joli  
de recevoir un  
message d'erreur  
au premier test  
mais je me doute  
que vous aimeriez  
bien voir des trucs  
qui fonctionnent,  
maintenant. C'est  
parti donc.

**Vos  
premières  
instructions  
: un peu de  
calcul**



# mental pour l'ordinateur

---

C'est assez trivial,  
quand on y pense,  
mais je trouve qu'il  
s'agit d'une  
excellente manière  
d'aborder pas à  
pas la syntaxe de  
Python. Nous  
allons donc essayer  
d'obtenir les  
résultats de calculs  
plus ou moins  
compliqués. Je  
vous rappelle  
encore une fois  
qu'exécuter les  
tests en même  
temps que moi sur  
votre machine est  
une très bonne  
façon de vous  
rendre compte de  
la syntaxe et  
surtout, de la  
retenir.

## Saisir un nombre

Vous avez pu voir  
sur notre premier  
(et à ce jour notre  
dernier) test que  
Python n'aimait  
pas  
particulièrement  
les suites de lettres  
qu'il ne comprend  
pas. Par contre,



l'interpréteur adore  
les nombres.  
D'ailleurs, il les  
accepte sans  
sourciller, sans une  
seule erreur :

pycon

```
1 >>> 7
2 7
3 >>>
```

D'accord, ce n'est  
pas extraordinaire.  
On saisit un  
nombre et  
l'interpréteur le  
renvoie. Mais dans  
bien des cas, ce  
simple retour  
indique que  
l'interpréteur a  
bien compris et  
que votre saisie est  
en accord avec sa  
syntaxe. De même,  
vous pouvez saisir  
des nombres à  
virgule.

pycon

```
1 >>> 9.5
2 9.5
3 >>>
```

Attention : on  
utilise ici la  
notation anglo-  
saxonne, c'est-à-  
dire que le point  
remplace la  
virgule. La virgule a  
un tout autre sens  
pour Python,  
prenez donc cette

habitude dès  
maintenant.

Il va de soi que l'on  
peut tout aussi  
bien saisir des  
nombres négatifs  
(vous pouvez  
d'ailleurs faire  
l'essai).

## **Opérations courantes**

Bon, il est temps  
d'apprendre à  
utiliser les  
principaux  
opérateurs de  
Python, qui vont  
vous servir pour la  
grande majorité de  
vos programmes.

### **Addition, soustraction, multiplication, division**

Pour effectuer ces  
opérations, on  
utilise  
respectivement les  
symboles +, -, \* et /.

pycon

```
1 >>> 3 + 4
2 7
3 >>> -2 + 93
4 91
5 >>> 9.5 + 2
6 11.5
7 >>> 3.11 + 2.08
8 5.1899999999999995
9 >>>
```

Pourquoi ce  
dernier résultat  
approximatif ?

Python n'y est pas  
pour grand chose.  
En fait, le  
problème vient en  
grande partie de la  
façon dont les  
nombres à virgule  
sont écrits dans la  
mémoire de  
votre ordinateur.  
C'est pourquoi, en  
programmation,  
on préfère  
travailler autant  
que possible avec  
des nombres  
entiers.  
Cependant, vous  
remarquerez que  
l'erreur  
est infime et  
qu'elle n'aura pas  
de réel impact sur  
les calculs. Les  
applications qui  
ont besoin d'une  
précision  
mathématique à  
toute épreuve  
essayent de pallier  
ces défauts par  
d'autres moyens  
mais ici, ce ne sera  
pas nécessaire.

Faites également  
des tests pour la  
soustraction, la  
multiplication et la

division : il n'y a rien de difficile.

## Division entière et modulo

Si vous avez pris le temps de tester la division, vous vous êtes rendu compte que le résultat est donné avec une virgule flottante.

pycon

```
1 >>> 10 / 5
2 2.0
3 >>> 10 / 3
4 3.3333333333333335
5 >>>
```

Il existe deux autres opérateurs qui permettent de connaître le résultat d'une division entière et le reste de cette division.

Le premier opérateur utilise le symbole « // ». Il permet d'obtenir la partie entière d'une division.

pycon

```
1 >>> 10 // 3
2 3
3 >>>
```

L'opérateur « % », que l'on appelle le « modulo », permet de connaître le reste de la division.

pycon

```
1 >>> 10%3
2 1
3 >>>
```

Ces notions de *partie entière* et de *reste de division* ne sont pas bien difficiles à comprendre et vous serviront très probablement par la suite.

Si vous avez du mal à en saisir le sens, sachez donc que :

- La partie entière de la division de 10 par 3 est le résultat de cette division, sans tenir compte des chiffres au-delà de la virgule (en l'occurrence, 3).
- Pour obtenir le modulo d'une division, on « récupère » son reste. Dans notre exemple,  $10/3 = 3$  et il reste 1. Une fois que l'on a compris cela, ce n'est pas bien compliqué.

Souvenez-vous bien de ces deux opérateurs, et surtout du modulo « % », dont vous aurez besoin dans

vos programmes  
futurs.

## En résumé

- L'interpréteur de commandes Python permet de tester du code au fur et à mesure qu'on l'écrit.
- L'interpréteur Python accepte des nombres et est capable d'effectuer des calculs.
- Un nombre décimal s'écrit avec un point et non une virgule.
- Les calculs impliquant des nombres décimaux donnent parfois des résultats approximatifs, c'est pourquoi on préférera, dans la mesure du possible, travailler avec des nombres entiers.

INDIQUER QUE CE  
CHAPITRE N'EST  
PAS TERMINÉ



DÉCOUVREZ  
PYTHON

ENTREZ DANS LE  
MONDE  
MERVEILLEUX  
DES VARIABLES



## Le professeur

Vincent Le Goff

Découvrez  
aussi ce cours  
en...



Livre



PDF

---

OPENCLASSROOMS



---

ENTREPRISES



---

CONTACT



---

EN PLUS



Français



Télécharger dans  
l'App Store