

☞ Éléments d'algorithmique et de programmation

1 Définitions



Définition

Une **variable** permet de nommer des informations afin de manipuler ces informations plus facilement.
C'est un emplacement de la mémoire de l'ordinateur.
Une variable est semblable à une inconnue dans une équation.
Affecter une variable, c'est lui donner du contenu.



Définition

Dans Python, il n'est pas nécessaire de déclarer les variables avant de pouvoir leur affecter une valeur.
La valeur que l'on affecte possède un **type** qui dépend de la nature des données :

- ⇒ le type **int** utilisé pour stocker un entier.
- ⇒ le type **float** utilisé pour stocker des nombres à virgule flottante.
- ⇒ le type **list** utilisé pour contenir une collection d'éléments.
- ⇒ le type **array** utilisé pour construire des tableaux qui contiennent des données du même type. Plus facile pour faire des opérations compliquées en plusieurs dimensions.
- ⇒ le type **bool** utilisé pour les booléens : un booléen peut prendre les valeurs True ou False
- ⇒ le type **str** utilisé pour les chaînes de caractères : c'est une suite de caractères délimitée par des guillemets et qu'on affiche avec **print**.
- ⇒ le type **complex** utilisé pour les nombres complexes.

Pour connaître le type d'une donnée ou le type de la valeur d'une variable, il suffit d'utiliser la fonction **type()**.



Définition

La syntaxe Python pour la définition d'une fonction est :

```
def nom_fonction(liste de paramètres) :  
    bloc d'instructions
```

La liste des paramètres constitue les arguments de la fonction.

Les valeurs renvoyées par les fonctions peuvent être de plusieurs types : chaîne de caractères avec **print**, des tableaux avec **array** ou encore des courbes avec la commande **plot**.

On peut utiliser la commande **return** pour renvoyer une valeur par une fonction : on pourra ainsi utiliser cette valeur dans des calculs.



Définition

Les instructions au sein d'une fonction sont elles-mêmes des fonctions :

- ☞ soit des fonctions que nous avons construites précédemment
- ☞ soit des fonctions "de base" : **If, else, elif, for** ou encore **while**

Les instructions **for** ou encore **while** sont des boucles : bornée pour la première (car on connaît le nombre de fois où l'instruction se répète) et non bornée pour la seconde (car on ne sait pas combien de fois on va devoir répéter l'instruction ni même si cette répétition va finir).

Exemples

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays the following Python code:

```
1 x=5
2 if x<8:
3     print("t'es un grand")
```

On the right, the IPython console shows the execution results:

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.26.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/PHILI/untitled0.py', wdir='C:/Users/PHILI')
t'es un grand

In [2]:
```

The screenshot shows the Spyder Python IDE interface. On the left, the code editor displays the following Python code:

```
1 x=5
2 if x<5:
3     print("t'es un grand")
4 else:
5     print("t'es pas un grand")
```

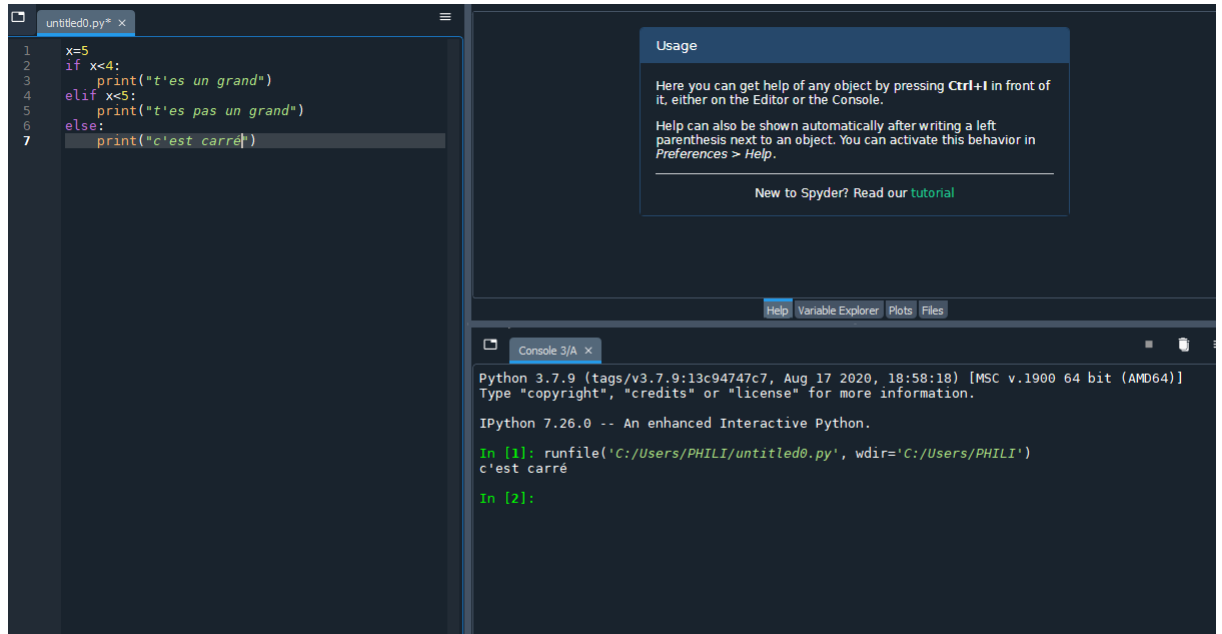
On the right, the IPython console shows the execution results:

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.26.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/PHILI/untitled0.py', wdir='C:/Users/PHILI')
t'es pas un grand

In [2]:
```



The screenshot shows the Spyder IDE interface. On the left, the editor displays a Python script named `untitled0.py` with the following code:

```
1 x=5
2 if x<4:
3     print("t'es un grand")
4 elif x<5:
5     print("t'es pas un grand")
6 else:
7     print("c'est carré")
```

On the right, the IPython console shows the execution of the script. The output is:

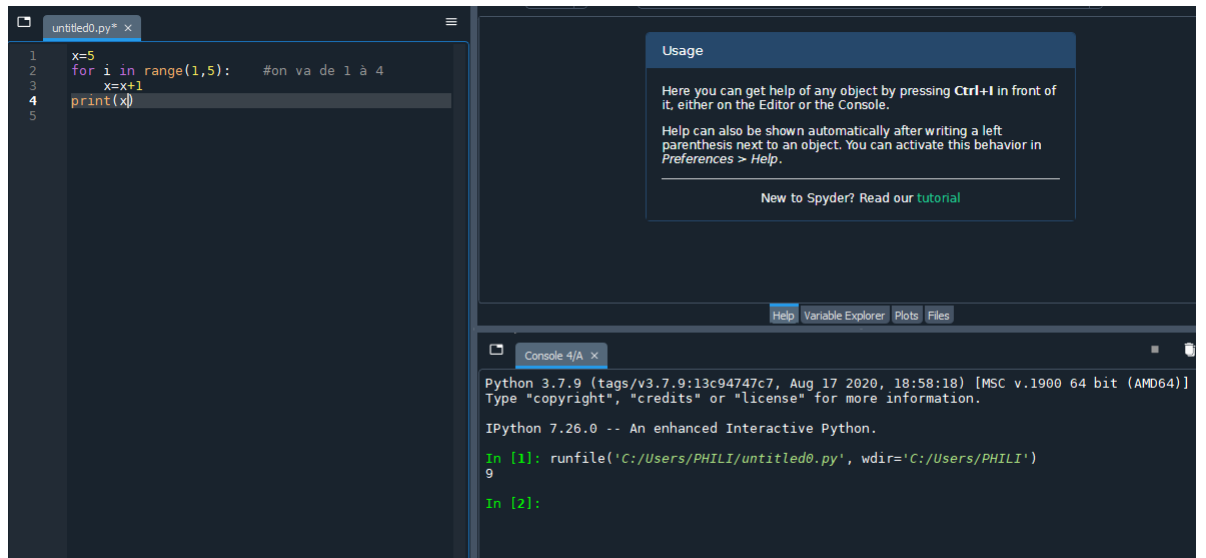
```
Python 3.7.9 (tags/v3.7.9:13c9474c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.26.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/PHILI/untitled0.py', wdir='C:/Users/PHILI')
c'est carré

In [2]:
```

Below the console, there are tabs for `Help`, `Variable Explorer`, `Plots`, and `Files`. The `Help` tab is active, displaying a "Usage" section with instructions on how to get help for any object by pressing `Ctrl+I`.



The screenshot shows the Spyder IDE interface. On the left, the editor displays a Python script named `untitled0.py` with the following code:

```
1 x=5
2 for i in range(1,5): #on va de 1 à 4
3     x=x+1
4     print(x)
5
```

On the right, the IPython console shows the execution of the script. The output is:

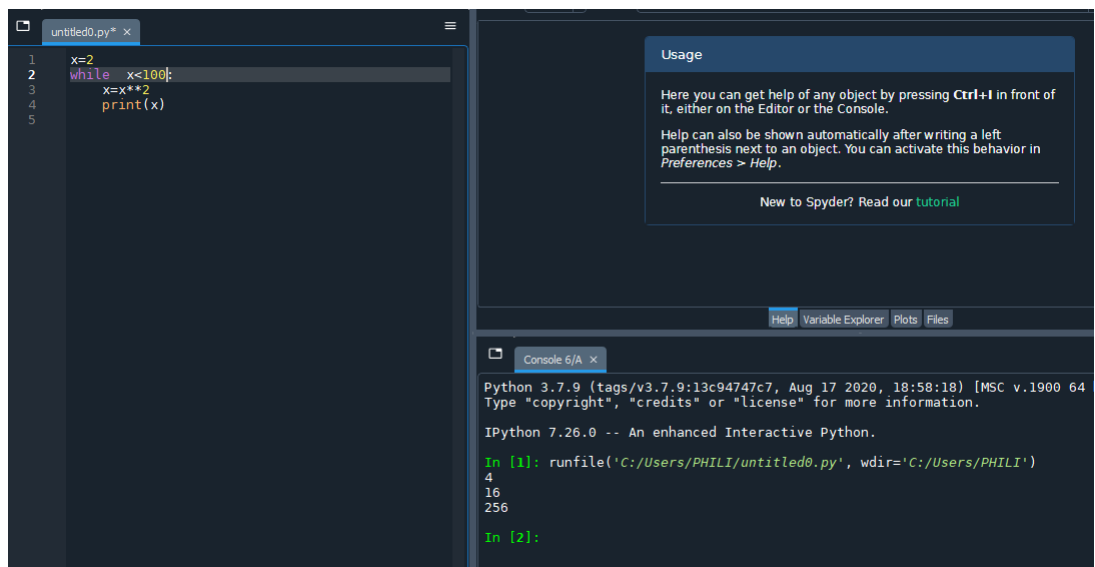
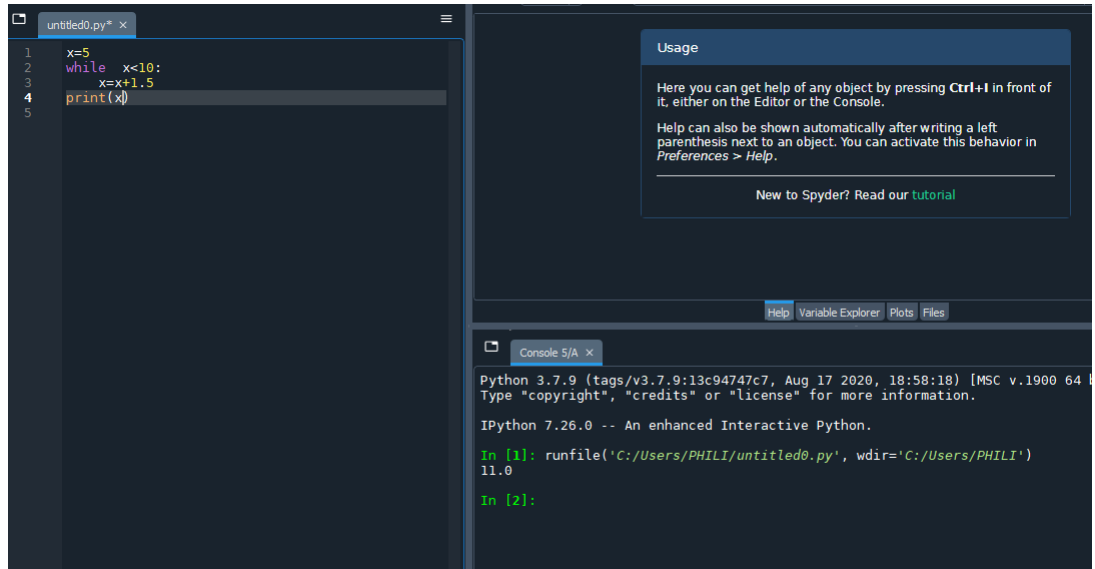
```
Python 3.7.9 (tags/v3.7.9:13c9474c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

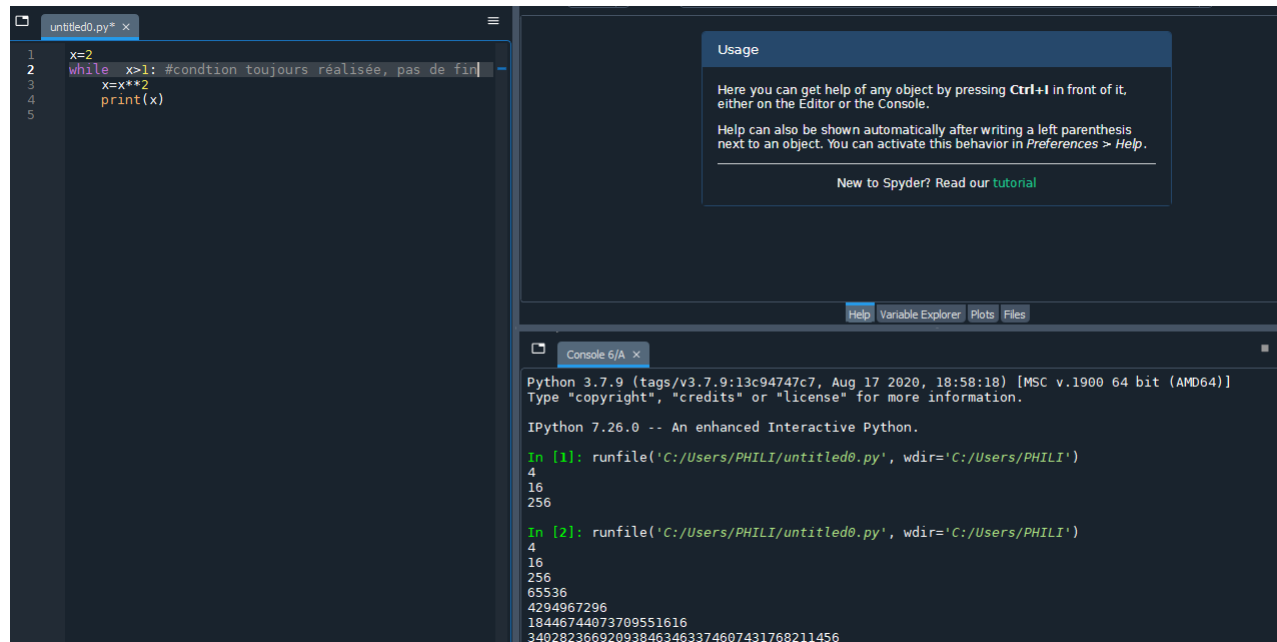
IPython 7.26.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/PHILI/untitled0.py', wdir='C:/Users/PHILI')
9

In [2]:
```

Below the console, there are tabs for `Help`, `Variable Explorer`, `Plots`, and `Files`. The `Help` tab is active, displaying a "Usage" section with instructions on how to get help for any object by pressing `Ctrl+I`.





2 Exemples de modules utiles en BTS ELT

- ☐ Afin de rédiger des programmes Python d'une bonne manière et pouvoir les enregistrer, on doit utiliser un éditeur de texte, préparer votre code dans un fichier et exécuter Python avec ce fichier en paramètre. Cela s'appelle créer un script. Dans le cas de longs programmes, on peut séparer le code dans plusieurs fichiers. Ainsi, il vous est facile de réutiliser des fonctions écrites pour un programme dans un autre sans avoir à les copier.
- ☐ Pour gérer cela, Python permet de placer des définitions dans un fichier et de les utiliser dans un script ou une session interactive. Un tel fichier est appelé un module et les définitions d'un module peuvent être importées dans un autre module ou dans le module principal.
- ☐ Un module est un fichier contenant des définitions et des instructions. Son nom de fichier est le nom du module suffixé de .py
Certains modules sont préexistants et ne sont pas chargés automatiquement par Python.
Dans la console et l'éditeur, on peut par exemple les appeler par la commande "import nommodule.py" et les fonctions contenues dans ce module s'utiliseront en tapant l'instruction "nommodule.fonction".
 1. le module "**cmath**" pour avoir accès à π (**cmath.pi**) , aux nombres complexes (i correspond à **1j**) et à l'argument (**cmath.phase**). Ce module convient tant qu'on agit pas sur un tableau de valeurs.
 2. le module "**numpy**" pour avoir accès aux calculs sur des tableaux . On pourra en construire grâce à la commande "**numpy.array**"; on calculera l'argument des éléments d'un tableau avec **numpy.angle** et le module avec **numpy.abs**; on résoudra des équations linéaire avec **numpy.linalg.solve**
 3. le module "**matplotlib.pyplot**" pour accéder aux graphiques grâce à **matplotlib.pyplot.plot** pour les construire puis **matplotlib.pyplot.show** pour les tracer et les faire apparaître dans la console.