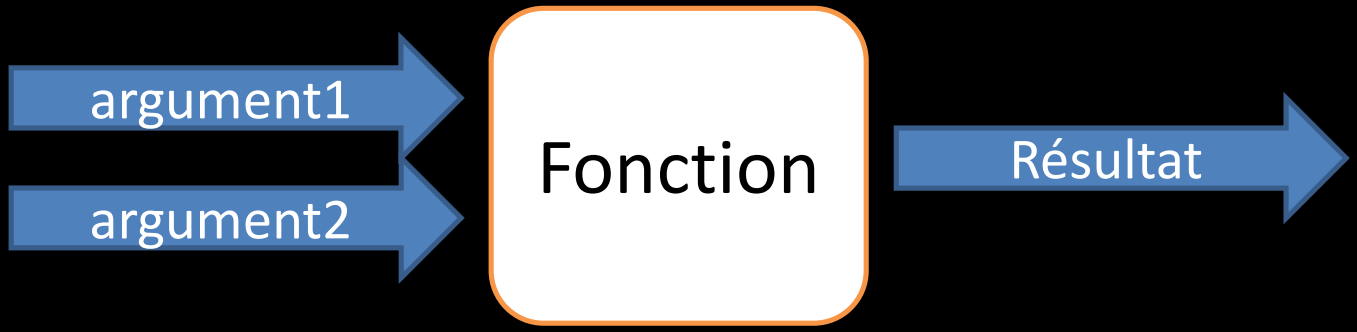
Les fonctions

Une fonction est une partie d'un programme - ou sous-programme - qui fonctionne indépendamment du reste du programme.
Elle reçoit une liste de paramètres et retourne un résultat



Déclaration

```
def fonction_nom (par_1, ..., par_n) :
    instruction_1
    ...
    instruction_n
    return res_1, ..., res_n
```

Remarquez l'indentation


```
>>> def carre(x):
... return x**2
...
>>> print(carre(2))
4
```

La valeur retournée est stockable dans une variable

```
>>> res = carre(2)
>>> print(res)
4
```


```
>>> def hello():
... print("bonjour")
...
>>> hello()
bonjour
```

Fonction sans valeur de sortie donc pas besoin de la stocker dans une variable (sortie ="none")

Python est un langage au *typage dynamique, c'est-à-dire qu'il* reconnaît pour vous le type des variables au moment de l'exécution

```
>>> def fois(x,y):
... return x*y
...
>>> fois(2,3)
6
>>> fois(3.1415,5.23)
16.430045000000003
>>> fois('to',2)
'toto'
```

Portée des variables

Une variable est dite locale lorsqu'elle est créée dans une fonction, car elle n'existera et ne sera visible que lors de l'exécution de la dite fonction.
Une variable est dite globale lorsqu'elle est créée dans le programme principal ; elle sera visible partout dans le programme.

Nous allons suivre un exemple simple illustrant L'exécution d'un programme principal qui met en jeux une variable globale contenant une fonction avec des variables locales

```
def carre(x):                    # définition d'une fonction carre()
        y = x**2                 # x et y sont locales
        return y

z = 5                            # déclaration d'une variable globale
resultat = carre(5)              # récupération de la fonction dans resultat

print(resultat)                  # une variable globale et l'afficher
```

Trace d'un programme avec fonction

http://www.pythontutor.com

line that just executed
next line to execute

Etape 1 : Python est prêt à lire la première ligne de code.

Python 3.6

```
1 def carre(x):
2     y = x**2
3     return y
4
5 # programme principal
6 z = 5
7 resultat = carre(5)
8 print(resultat)
```

Print output (drag lower right corner to resize)

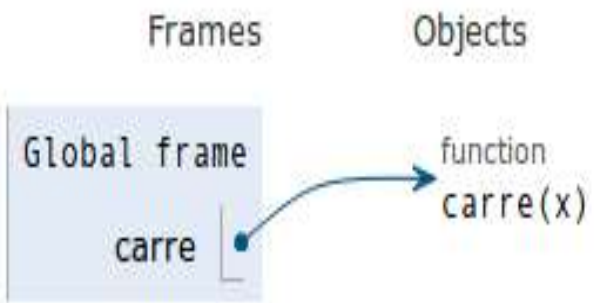
Frames Objects

Etape 2 : Python met en mémoire la fonction carre() (notez qu'il ne l'exécute pas !). La fonction est mise dans une case de la mémoire nommée ***global frame*** qui est l'espace mémoire du programme principal.
Python est maintenant prêt à exécuter le programme principal

Python 3.6

```
1 def carre(x):
2     y = x**2
3     return y
4
5 # programme principal
6 z = 5
7 resultat = carre(5)
8 print(resultat)
```

Print output (drag lower right corner to resize)

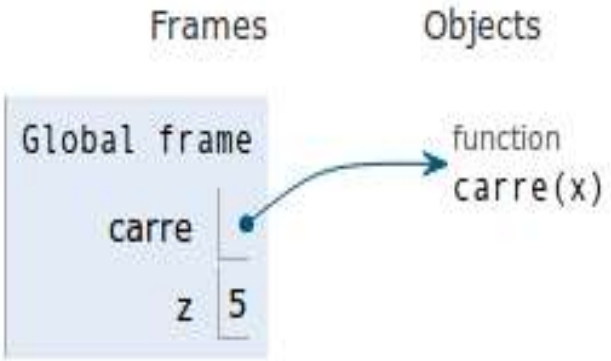


Etape 3 : Python lit et met en mémoire la variable z. Celle-ci étant créée dans le programme principal, il s'agira d'une variable globale. Ainsi, elle sera également stockée dans le global frame

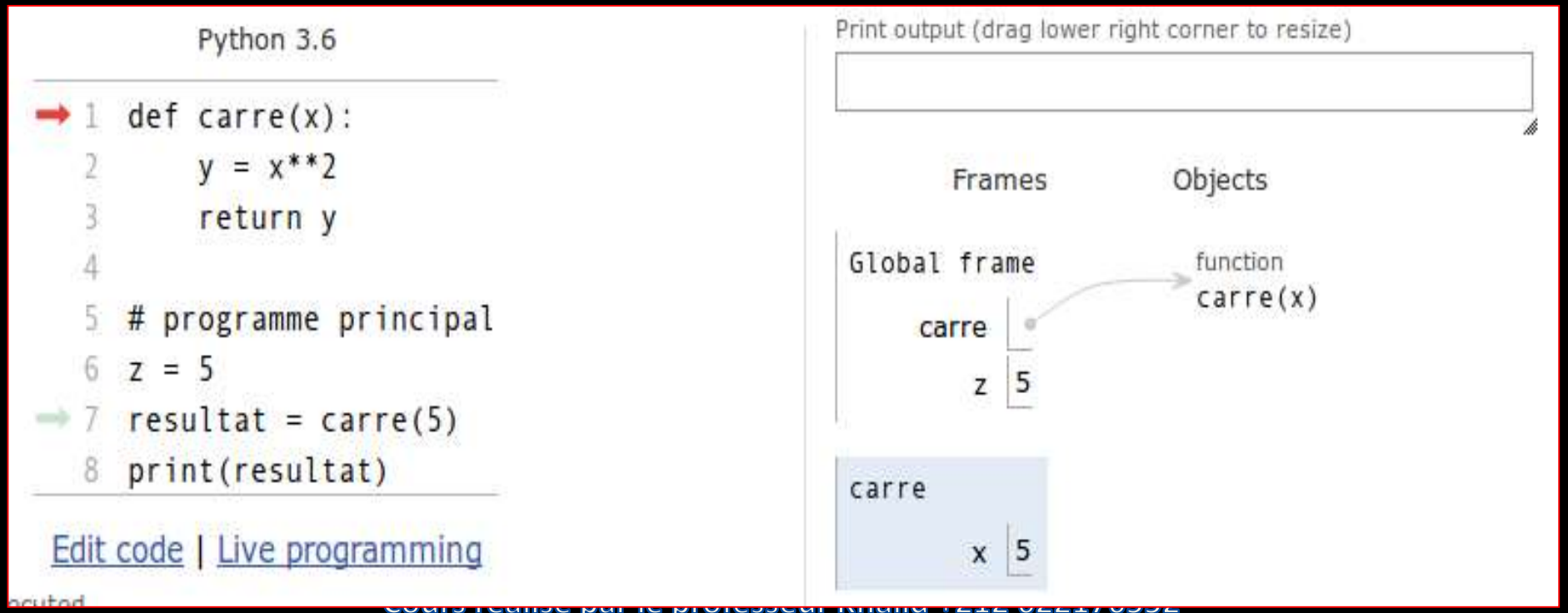
Python 3.6

```
1 def carre(x):
2     y = x**2
3     return y
4
5 # programme principal
6 z = 5
7 resultat = carre(5)
8 print(resultat)
```

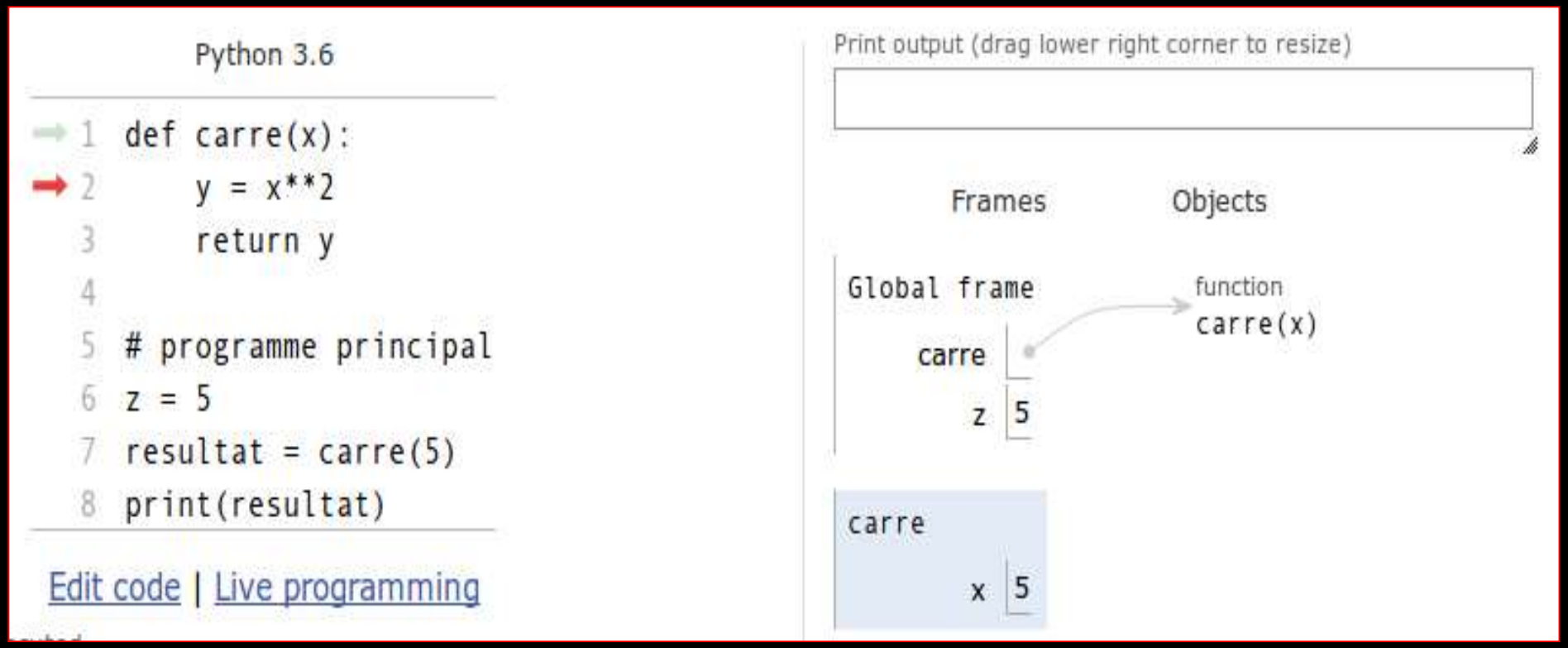
Print output (drag lower right corner to resize)



Etape 4 : La fonction carre() est appelée et on lui passe en argument l'entier 5. La fonction rentre alors en exécution et un nouveau cadre bleu est créé dans lequel python tutor va nous indiquer toutes les variables locales à la fonction. Notez bien que la variable passée en argument, qui s'appelle x dans la fonction, est créée en tant que variable locale.



Etape 5 : Python est maintenant prêt à exécuter chaque ligne de code de la fonction.



Etape 6 : La variable y est créée dans la fonction. Celle-ci est donc stockée en tant que variable locale à la fonction.

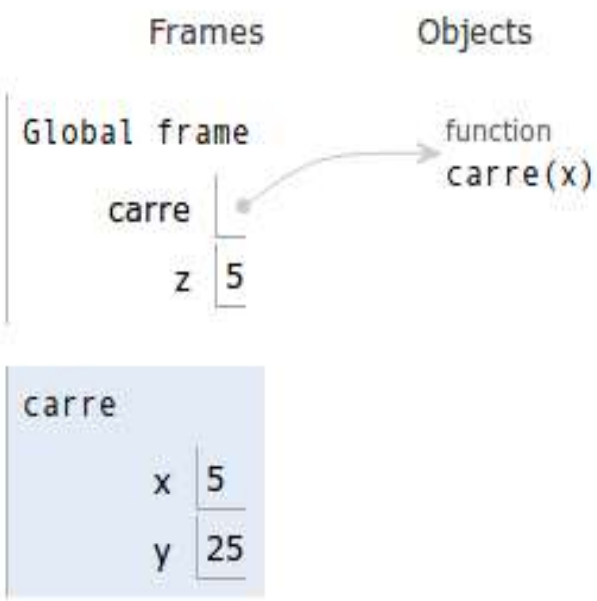
Python 3.6

```
1  def carre(x):
2      y = x**2
3      return y
4
5  # programme principal
6  z = 5
7  resultat = carre(5)
8  print(resultat)
```

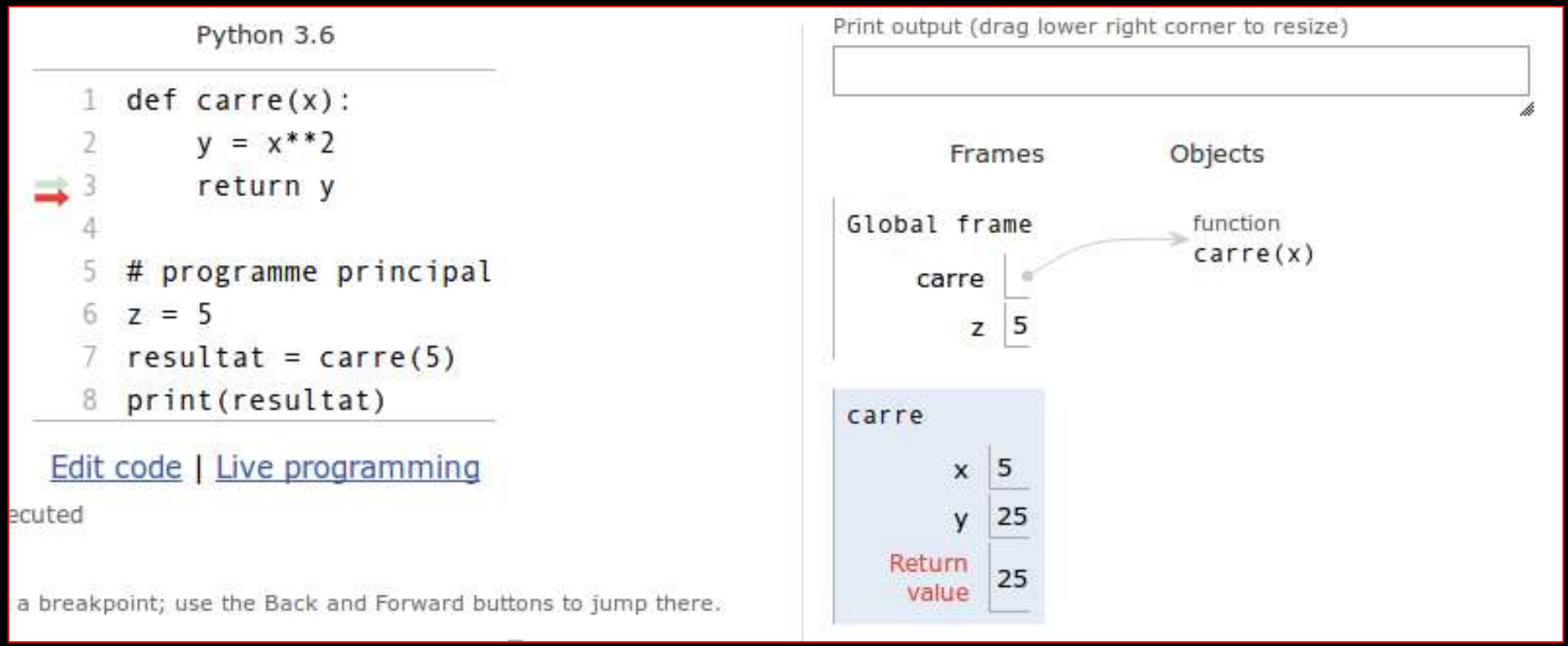
Edit code | Live programming

ecuted

Print output (drag lower right corner to resize)



Etape 7 : Python s'apprête à retourner la variable locale y au programme principal python tutor nous indique le contenu de la valeur retournée).

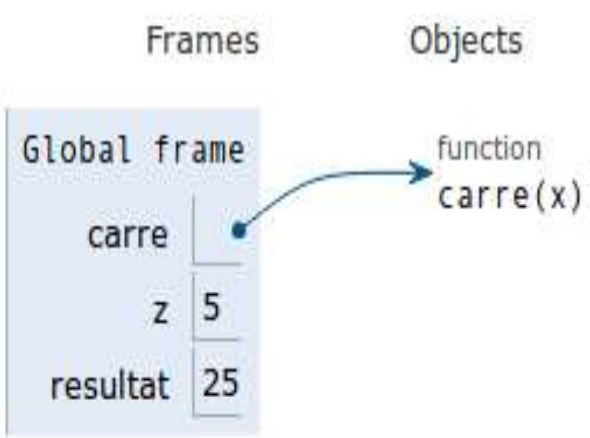


Etape 8 : Python quitte la fonction et la valeur retournée par celle-ci est affectée à la variable globale **resultat**. Notez bien que lorsque Python quitte la fonction, l'espace des variables allouées à la fonction est détruit. *Ainsi toutes les variables créées dans la fonction n'existent plus.* On comprend pourquoi elles portent le nom de locales ***puisqu'elles n'existent que lorsque la fonction est en exécution.***

Python 3.6

```
1  def carre(x):
2      y = x**2
3      return y
4
5  # programme principal
6  z = 5
7  resultat = carre(5)
8  print(resultat)
```

Print output (drag lower right corner to resize)



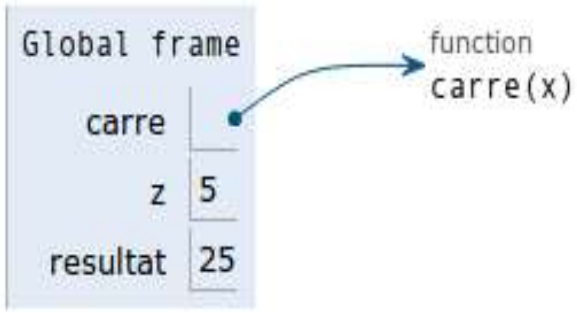
Etape 9 : Python affiche le contenu de la variable resultat et l'exécution est terminée.

Python 3.6

```
1  # definition d'une fonction
2  def carre(x):
3      y = x**2
4      return y
5
6  # programme principal
7  z = 5
8  resultat = carre(5)
9  print(resultat)
```

Print output (drag lower right corner to resize)

```
25
```



Autres exemples

```
>>> def mafonction():
... print(x)
...
>>> x = 3
>>> mafonction()
3
>>> print(x)
3
```

lorsqu'une variable déclarée à la racine du module (c'est comme cela que l'on appelle un programme Python), elle est visible dans tout le module. On parle de variable **globale**

```
>>> def mafonction():
... x = x + 1
...
>>> x=1
>>> mafonction()
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "<stdin>", line 2, in fct
UnboundLocalError: local variable 'x' referenced before
assignment
```

Python ne permet pas la modification d'une variable globale dans une fonction. Python pense que x est une variable locale qui n'a pas été encore assignée. Solution = il faut utiliser le mot-clé global :

```
>>> def mafonction():
... global x
... x = x + 1
...
>>> x=1
>>> mafonction()
>>> x
2
```

l'utilisation de variables globales est à bannir définitivement de votre pratique de la programmation.
Python est orienté objet et cela permet *"d'encapsuler" des variables dans des objets* et de s'affranchir définitivement des variables globales

Questionnaire

1- Donnez la syntaxe générale de déclaration d'une fonction
2- Définissez une variable locale et globale

Travaux pratiques *(solution page suivante)*

1. Proposez une fonction capable de générer le montant TTC à partir du HT avec une taxe qui prend deux valeurs selon le montant HT
   a. 20% si HT<1000
   b. 25% si HT>=1000

Arrondir le HT à deux chiffres après la virgule


```
def TTC():
    x=input("votre montant HT ")
    x=float(x)
    x=round(x,2)
    if x<1000:
        print ('votre TTC est '+str(x+x*20/100)+ ' DH')
    else:
        print ('votre TTC est '+str(x+x*25/100)+ ' DH')
TTC()
```