



Les \* signifient que la correction est disponible.

Pour les premiers pas, voici un cours miniature et brouillons : [Les chaînes de caractères](#)

**Exercice \*C1/** L'exercice consiste à créer une fonction **searchC** qui a pour paramètres *un mot* et *une chaîne de caractères* et qui renvoie False si l'élément n'est pas dans la chaîne de caractères, ou la position du mot dans le cas contraire. (ici False est un booléen, pas une chaîne de caractères)

[Correction](#)

-une solution fonctionnant sous Python 2.7.5 et Python 3.4 :

```
1 def searchC(i,C):
2     for k in range(len(C)):
3         if C[k]==i:
4             return(k)
5     return(False)
```

**L1.** Début de création de la fonction s'appelant **searchC**, et ayant comme paramètres *i* et *C*.

**Boucle for :** **L2.** Pour tout indice *k* allant de 0 -par défaut- jusqu'à *len(C)* exclu (*len(C)* est la longueur de *C*)

**L3.** Si l'élément d'indice *k* est *i*,

**L4.** Alors la fonction renvoie *k*.

Fin de boucle for

**L5.** Si on est là c'est que la condition *C[k]=i* n'a jamais été vérifiée, donc *i* n'est pas dans *C*.

On retourne alors le booléen **False**.

**Exercice \*C2/ ★** Voici une chaîne de caractères : "La v13 3st un myst3r3 qu'il faut v1vr3, et non un probl3m3 a r3soudr3.Gandh1". L'exercice consiste à afficher cette chaîne de caractères en remplaçant les 1 par des i et les 3 par des e. (Cette fois-ci je ne demande pas la définition d'une fonction.)

[Correction](#)

```
1 text="La v13 3st un myst3r3 qu'il faut v1vr3, et non un probl3m3 a r3soudr3.Gandh1"
2 new_text=""
3
4 for lettre in text:
5     if lettre=='1':
6         new_text+='i'
7     elif lettre=='3':
8         new_text+='e'
9     else:
10        new_text+=lettre
11
12 print(new_text)
```

**L1.** Création de la chaîne de caractères sous le nom de `text`.

**L2.** On crée une nouvelle chaîne nommée `new_text`, cette fois-ci vide. (elle sera le résultat final)

**Boucle for :** **L4.** Pour chaque élément de la chaîne de caractères initiale, on va ajouter l'élément voulu dans la nouvelle chaîne :

**L5.** à **L6.** Si l'élément est '1' alors on met 'i' dans la nouvelle chaîne,

**L7.** à **L8.** Si l'élément est '3' alors on met 'e' dans la nouvelle chaîne,

**L9.** à **L10.** Sinon on met l'élément lui-même dans la nouvelle chaîne.

**Fin de boucle for**

**L12.** On affiche la nouvelle chaîne de caractères à l'aide de la fonction `print`.

Ceci doit retourner: **La vie est un mystere qu'il faut vivre, et non un probleme a resoudre.Gandhi**

**ATTENTION!!** On ne peut pas faire `texte[3]='i'` car les chaînes de caractères ne sont pas mutables !

L'erreur renvoyée est la suivante : **TypeError:** 'str' object does not support item assignment

**REMARQUE!!** `new_text+='i'` signifie `new_text= new_text + 'i'`

**Exercice \*C3/** -L'exercice consiste à créer une fonction **compteur** qui a pour paramètres *une lettre* et *une chaîne de caractères*, et qui renvoie le nombre d'apparitions de la lettre dans la chaîne de caractères.

[Correction](#)

-une solution fonctionnant sous **Python 2.7.5** et **Python 3.4** :

```
1 def compteur(chaine,lettre):
2     j=0
3     for i in chaine:
4         if i==lettre:
5             j+=1
6     return(j)
```

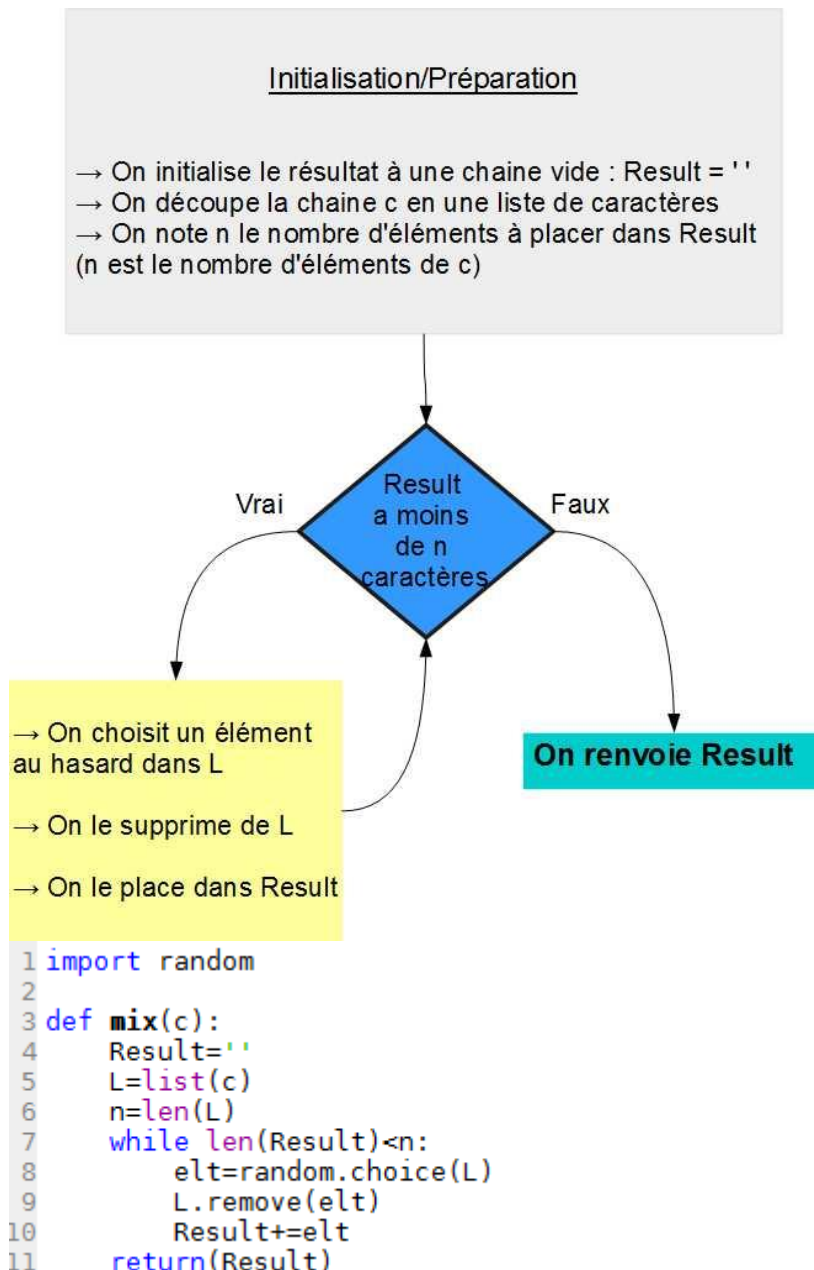
**L1.** Création de la fonction nommée **compteur**, avec deux paramètres : *chaine* et *lettre*.  
**L2.** On initialise le nombre d'apparitions `j` de "lettre" à 0.  
**Boucle for :**  
**L3.** On parcourt la chaîne de caractères `chaine` en prenant chacun de ses éléments `i`.  
**L4.** Si l'élément `i` en question correspond à la lettre recherchée (`lettre`), alors :  
**L5.** On incrémente `j`  
**Fin de boucle for.**  
**L6.** On renvoie le nombre d'apparitions `j` à l'aide de `return`.

**Exercice \*C4/ ★** L'exercice consiste à créer une fonction **mix** qui a pour paramètre *une chaîne de caractères* `c`, et qui renvoie une chaîne de caractères créée aléatoirement à partir de c.

[Correction](#)

-une solution fonctionnant sous **Python 2.7.5** et **Python 3.4** :

Code copiable : [\[Afficher\]](#)



**L1.** On `importe` random pour pouvoir utiliser une fonction qui choisira un élément au hasard.

**L3.** On crée la fonction mix qui a pour paramètre c.

**L4.** On initialise Result à une chaîne de caractères vide ''.

**L5.** On découpe la chaîne de caractères c en une liste de caractères à l'aide de la fonction `list()`.

**L6.** On note n la taille de la liste, c'est-à-dire le nombre de caractères constituant c.

**Boucle While :** **L7.** Tant que Result a moins de n caractères,

**L8.** On choisit un élément (noté elt) au hasard dans L grâce à la fonction `choice()` se trouvant dans random,

**L9.** On supprime de L l'élément choisi, pour qu'il ne puisse pas être choisi une deuxième fois, à l'aide de `remove()`.

**L10.** On place elt à la fin de Result par concaténation.

**Fin Boucle While :** On est ici lorsque Result a le même nombre de caractère que c. C'est-à-

dire que Result est un mélange des lettres de c.

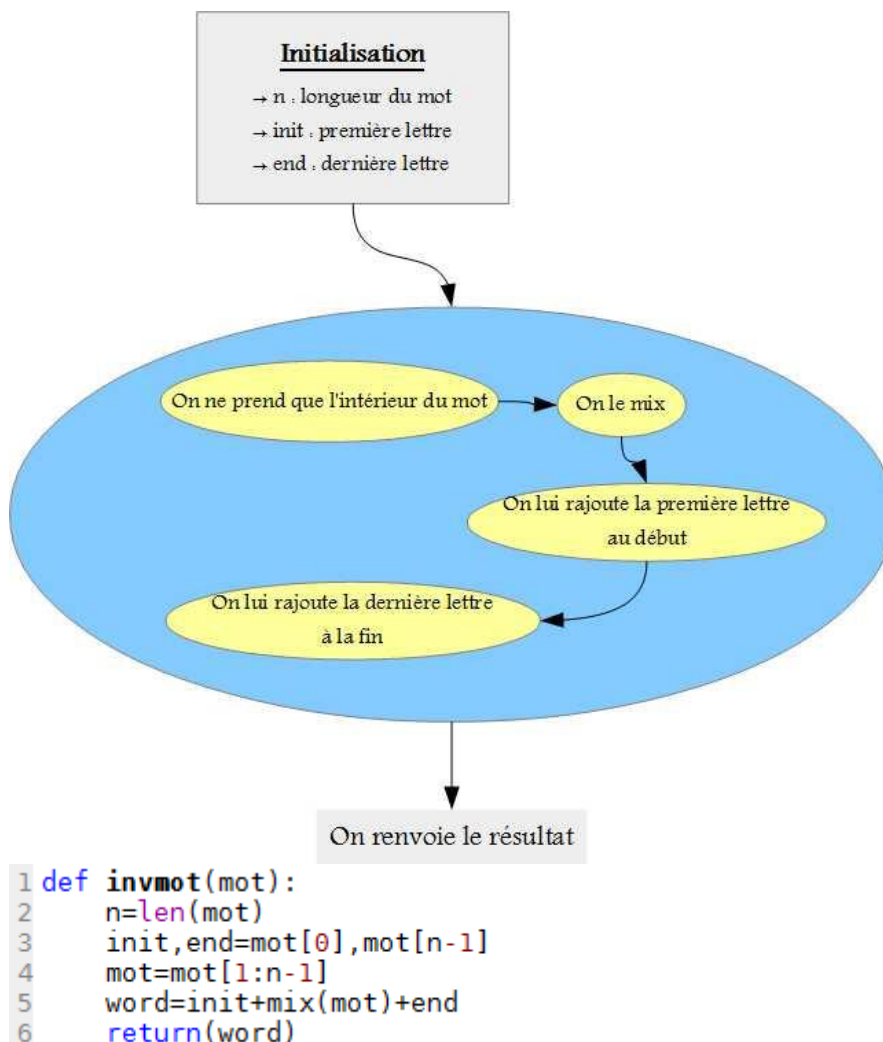
L11. On renvoie Result.

---

**Exercice \*C5/ ☆** Utiliser la fonction de l'exercice C4 ( mix ) pour créer une fonction **invmot** qui a pour paramètre un mot, *mot*, et qui renvoie un autre mot en mélangeant les lettres sauf la première et la dernière lettre.

[Correction](#)

- une solution fonctionnant sous Python 2.7.5 et Python 3.4 :  
Code copiable : [Afficher]



L1. On définit la fonction s'appelant **invmot** et ayant un paramètre nommé *mot*.

L2. On pose n, la longueur de mot, grâce à la fonction `len()`.

L3. On nomme init le premier caractère de mot (i.e. celui d'indice 0) et end le dernier caractère de mot (i.e. celui d'indice `len(mot)-1`).

**L4.** On ne garde que l'intérieur du mot : i.e. on ne prend que les lettres allant de la deuxième lettre (indice 1) à l'avant dernière (indice n-2).\*

**L5.** On crée un nouveau mot word composé de la première lettre init, suivit de mot mixé par la fonction mix de l'exercice précédent, suivit de la dernière lettre end.

**L6.** On renvoie word.

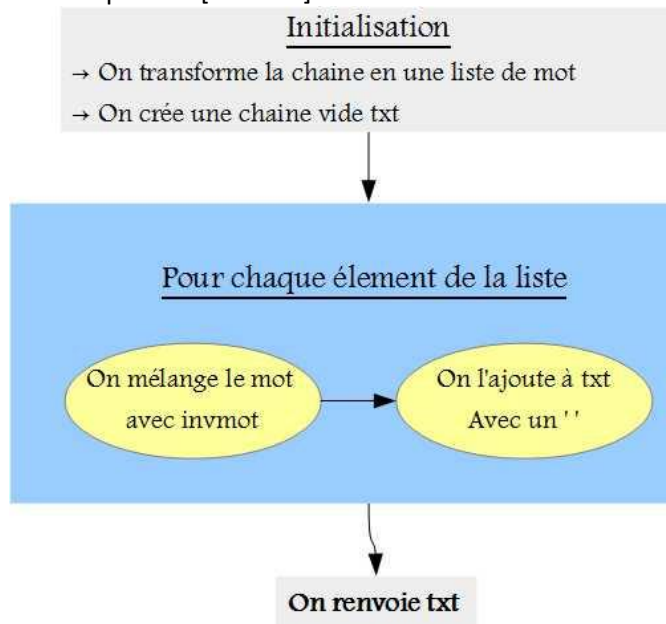
\* **ATTENTION !!!** truc[a,b] inclu l'élément a mais exclu l'élément b.

**Exercice \*C6/ ★** Utiliser la fonction de l'exercice C5 (invmot) pour créer une fonction **melange** qui a pour paramètre une chaîne de caractères cet qui en renvoie une autre telle que les lettres de chaque mot soient mélangées excepté la première et la dernière lettre.

[Correction](#)

-une solution fonctionnant sous Python 2.7.5 et Python 3.4:

Code copiable : [Afficher]



```
1 def melange(c):
2     L=c.split()
3     txt=''
4     for i in range(len(L)):
5         L[i]=invmot(L[i])
6
7         txt +=L[i]+ ' '
8     return(txt)
```

**L1.** On définit la fonction **melange** de paramètre c.

**L2.** On découpe c en une liste de mots à l'aide de la méthode **split()** (découpe à chaque espace ' ').

**L3.** On initialise **txt** à une chaîne de caractères vide ''.

**Boucle for : L4.** Pour tous mots, i.e. pour tous éléments de L,

**L5.** On mélange le mot à l'aide de la fonction **invmot**.

**L7.** On l'ajoute à la fin de **txt**, sans oublier l'espace ' ' pour séparer le mot du prochain.

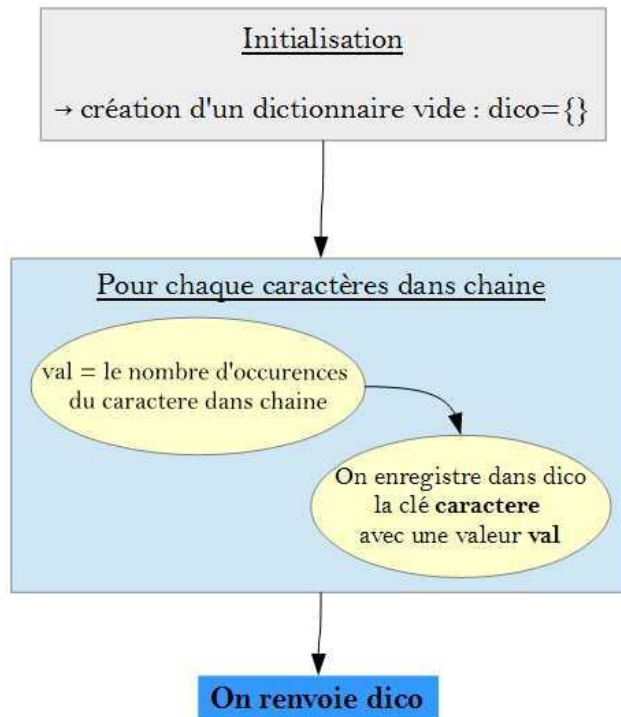
**Fin Boucle for**

**L8.** On renvoie **txt**.

**Exercice \*C7/ ★** L'exercice consiste à créer une fonction **nbr\_occur** qui a pour paramètre une chaîne de caractères *chaîne* et qui renvoie un dictionnaire dont les clés sont les caractères utilisés dans la chaîne, et dont les valeurs sont le nombre d'occurrences de la clé dans la chaîne.

[Correction](#)

Code copiable : [Afficher]



```

1 def nbr_occurences(chaine):
2     dico={}
3     for caractere in chaine:
4         val=chaine.count(caractere)
5         dico[caractere]=val
6     return(dico)
7

```

**L1.** On commence la définition de la fonction en écrivant le nom de la fonction (**`nbr_occurences`**) et le nom de son paramètre (*chaine*).

**L2.** On initialise `dico` à un dictionnaire vide `{}`.

**Boucle for : L3.** Pour tous caractères **dans** *chaine*,

**L4.** On enregistre son nombre d'occurrences dans `val`, à l'aide de la méthode **`count`**.


**L5.** On crée la clé caractère de valeur `val` dans `dico`.

**Fin Boucle for**

**L6.** On **renvoie** `dico`.

**Exercice c8/ ★★** Vous êtes professeur d'anglais et vous avez demandé à votre classe d'écrire quelques lignes en utilisant le verbe "must". Sachant que vous ne vous intéressez qu'à l'utilisation de ce verbe, vous décidez de créer un outil python pour vérifier qu'ils n'ont pas écrit "must to".

Le but de l'exercice est de créer une fonction **`must`** qui a pour paramètre *une chaine de caractères text* et qui renvoie **`False`** s'il y a au moins une fois le mot "must to", et qui renvoie **`True`** sinon. Tester votre fonction avec les 3 chaînes de caractères du fichier suivant :

:  [vos\\_copies.txt](#)

**Exercice c9/ ★** Cet exercice reprend le contexte de l'exercice précédent. Vous décidez de créer un outil python pour corriger les copies.

Le but de l'exercice est de créer une fonction **`correct`** qui a pour paramètre *une chaine de*

caractères *text* et qui renvoie cette chaine de caractères en ayant supprimé les "to" après les "must".

## Exercice \*C10/ ★

1) En utilisant la fonction `ord` de python, créez une fonction qui prend comme paramètre *un mot composé de lettres sans accent*, et qui renvoie la liste des numéros de place de chaque lettre dans l'alphabet.

### Correction

Code copiable : [Afficher]

```
1 def formatage(mot):
2     L = list(mot)
3     n = len(L)
4     for i in range(n):
5         L[i] = (ord(L[i]) - 97)%256
6     return(L)
7
8 def lecture(L):
9     s = ''
10    n = len(L)
11    print(n)
12    for i in range(n):
13        s += chr( (L[i]+97)%256 )
14    return(s)
```

1) Formatage

**L2.** On commence par initialiser la liste à remplir : elle correspond initialement à la liste des lettres du mot.

**Boucle For : L3.** Pour chaque lettre :

**L4.** On la remplace par son numéro (`ord` de la lettre) moins 97 (car a correspond à 97 et non à 0 comme voulu) modulo 256 pour éviter les erreurs (car 256 caractères sont codés).

Ici `ord(L[i])` renvoie le numéro du caractère `L[i]` mais il faut savoir que l'alphabet commence à 97.

**Fin boucle for : L5.** On renvoie la liste.

2) Lecture

**L8.** On initialise la chaine de caractères à renvoyer à une chaine de caractères vide.

**Boucle For : L9.** Pour chaque nombre de la liste L :

**L10.** On rajoute dans s la lettre correspondant au numéro `L[i]` + 97 modulo 256 (modifications inverses à celles faites dans la question 1).

**Fin boucle for : L11.** On renvoie le mot s.

**Exercice \*C11/ ★★** Créez une fonction **rot13** qui a comme paramètre *un mot composé de lettres sans accent* et qui renvoie ce mot codé en rot13.

### Correction

Code copiable : [Afficher]

```
14 def rot13(mot):
15     L = formatage(mot)
16     for i in range(len(L)):
17         L[i] = (L[i]+13)%26
18     return(lecture(L))
```

Ici, j'ai utilisé les deux fonctions de l'exercice C10.

**L15.** On récupère la liste des nombres associés aux lettres du mot.

**Boucle for : L16.** Pour chaque nombre :

**L17.** On lui ajoute le nombre 13 comme l'algorithme le préconise, puis on prend le modulo 26 pour toujours retomber sur une lettre de l'alphabet.

**Fin boucle for : L18.** On renvoie le mot après avoir retranscrit les nombres en lettres à l'aide de la fonction lecture.

**EXPLICATION rot13 :** Il s'agit d'un algorithme de chiffrement de texte correspondant à un décalage de 13 lettres dans l'alphabet, appliqué à chaque lettre du mot. Par exemple, "a" devient "n" et "i" devient "v".

**Aide :** Vous pouvez vous servir des fonctions réalisées dans l'exercice C10.



**Exercice \*C12/** - Après avoir créé la chaîne de caractères correspondant au texte suivant :

texte = "\n<< Être seul dans un monde qui ne change jamais.>>\nDans le cerveau des autistes, Temple Grandin"

- Affichez le texte,
- Affichez le caractère d'indice 22 et le dernier caractère,
- Affichez la chaîne de caractères composée des caractères allant de l'indice 0 à l'indice 50,
- Ajoutez la chaîne de caractères " et Richard PaneK" au texte, puis affichez le résultat,
- Zut, le dernier caractère voulu n'est pas "K" mais "k"... Repartez de la variable texte modifiée pour corriger cette erreur.

**Attention :** on ne peut pas modifier un caractère d'une chaîne en faisant `texte[-1] = "k"`.

On dit que cet objet est immuable.

[Correction](#)

Voici une solution en **Python 3.4** :

```
1 texte = "\n<< Être seul dans un monde qui ne change jamais.>>\nDans le cerveau des autistes, Templ
2
3
4 print(texte)
5 print("\nle caractère d'indice 22 est ", texte[22], "\nle dernier caractère est ", texte[-1])
6 print(texte[0:51])
7
8 texte += " et Richard PaneK"
9 print(texte)
10
11 texte = texte[0:-1] + "k"
12 print(texte)
```

Code copiable : [\[Afficher\]](#)

**L1.** Création de la variable texte.

**Exercice \*C13/ ☆** Créez une fonction **inverse** qui prend *une chaîne de caractères* en paramètre et qui renvoie son inverse.

[Correction](#)

```
1 def inverse(mot):
2     inverse = ""
3     size = len(mot)
4     for i in range(size):
5         inverse += mot[size-1-i]
6     return(inverse)
7
8 print(inverse("mot"))
```

**L1.** On définit la fonction inverse et on nomme son paramètre mot.

**L2.** On initialise la chaîne de caractères inverse à une chaîne vide.

**L3.** On récupère la taille de la chaîne de caractères à l'aide de `len`.

**Boucle for :** **L4.** On parcourt la chaîne de caractères :

**L5.** On la lit en commençant par la fin : on place à la fin de inverse le caractère lu dans mot. L'indice du caractère récupéré dans mot va donc de `size-1` à 0.

**Fin boucle for :** **L6.** Une fois le mot lu en entier, on renvoie inverse.

**Exemple :** `inverse("mot")` doit renvoyer "tom".

**Exercice \*C14/ ☆** Écrivez un programme qui demande à l'utilisateur le prix unitaire des articles qu'il a achetés et le nombre d'articles achetés.

Le programme doit ensuite afficher le prix total hors taxe et le prix total TTC avec une TVA



de 20%. [Correction](#)

Code copiable : [Afficher]

```
1 prix = float(input("A quel prix achetez-vous vos articles ? "))
2 nbr = float(input("Combien d'articles achetez-vous ?"))
3
4 HT = prix*nbr
5 TTC = HT + HT*0.2
6
7 affichage = "nombre d'articles : {0} \nprix total Hors taxe : {1}€ \nprix total TTC : {2}€".format(nbr, HT, TTC)
8
9 print(affichage)
```

**L1.** On demande le prix avec la fonction `input` qui affiche ce qu'elle a en paramètre et attend de recevoir une chaîne de caractères. Ici il ne faut pas oublier de convertir en `float`.

**L2.** On fait de même pour récupérer le nombre d'articles.

**L4.** On calcule le coût hors taxe en multipliant le prix unitaire avec le nombre d'articles.

**L5.** On calcule le coût TTC en ajoutant au prix hors taxe 20% de lui-même.

**L7.** On affiche tout ça avec la fonction `format` qui place ses paramètres dans la chaîne de caractères là où les `{i}` se trouvent ( `{0}` sera remplacé par le premier paramètre, `{1}` par le second ...)

**L9.** On affiche le texte obtenu à l'aide de `print`.

**Exemple :** Le programme peut par exemple afficher les lignes suivantes pour un prix de 100 et un nombre d'articles de 5 :

nombre d'articles : 5.0

prix total Hors taxe : 500.0€

prix total TTC : 600.0€