seance_2020-12-01_com

December 7, 2020

1 Manipulations de listes et de la boucle for

Définitions de quelques variables pour faciliter les manipulations ultérieures

[1]:
```
toto = [3, 7, 9, 1, 0, 8, 8, 8]
titi = [9, False, 8.34, True, 22, 1e-3, 'a', -1]
tutu = [[0,1], ['a', 'b', 'c'], 78.5]
vecteur = [0]*100
lv = []
```

Exercice: calculer la somme des entiers de la liste toto en utilisant une boucle.

[2]:
```
# Ce calcul est possible directement avec la fonction prédéfinie sum()
print(sum(toto))
```

```
44
```

[6]:
```
somme = 0
for elt in toto:
    somme = somme + elt
print("La somme vaut %d" % somme)
```

```
La somme vaut 44
```

Variante sous la forme d'une fonction. `c` correspond ici à la somme

[3]:
```
def exo1():
  c = 0
  for i in toto:
    c += i
  print(c)
exo1()
```

```
44
```

Parenthèse: Formatage des chaînes de caractères. On illustre ci-dessous différentes manières d'imprimer une chaîne de caractères en recourant à un formatage.

[5]:
```
print("5","+","6","=",5+6)
print("%d + %d = %d" % (5,6,5+6))
```

```
print("{} + {} = {}".format(5,6,5+6))
```

```
5 + 6 = 11
5 + 6 = 11
5 + 6 = 11
```

NB Les deux méthodes de formatage (avec %, méthode héritée du langage C et encore très vivante; ou avec .format(), méthode plus récente et plus puissante) sont utilisables indépendamment de la fonction print: dès qu'il s'agit de mettre en forme une chaîne de caractères.

[6]:
```
x = "%d + %d = %d" % (5,6,5+6)
```

Exercice : Faire une fonction qui prend en entrée une liste d'entiers et affiche la valeur minimale de la liste. Ici encore il existe une fonction pré-définie en python qui réalise ce calcul, mais dans cet exercice on demande de faire le calcul de manière explicite.

[7]:
```
# Pour contrôle: fonction min()
print(min(toto))
```

```
0
```

Version la plus simple: le minimum de départ est le premier élément de la liste, et cet élément est ensuite comparé successivement à tous les autres éléments de la liste.

[10]:
```
min = toto[0]
for elt in toto[1:]:
    if elt < min:
        min = elt
print("Valeur min: %d" % min)
```

```
Valeur min: 0
```

Même algorithme, mais avec utilisation de l'indice dans la liste pour manipuler les éléments (et donc une boucle while)

[23]:
```
min = toto[0]
i = 1
while i < len(toto):
    if toto[i] < min:
        min = toto[i]
    i += 1
print("Valeur min: %d" % min)
```

```
Valeur min: 0
```

Même algorithme, mais réalisé sous la forme d'une fonction. D'abord sans aucune paramètre, ce qui manque clairement de généralité.

[11]:
```
def min_liste():
    min = toto[0]
```

```
    for elt in toto[1:]:
        if elt < min:
            min = elt
    print("Valeur min: %d" % min)
```

[12]:
```
min_liste()
```

```
Valeur min: 0
```

… puis sous forme d'une fonction avec un paramètre, en l'occurrence la liste dont on cherche le minimum, ce qui paraît plus pertinent.

[14]:
```
def min_liste_p(l):
    min = l[0]
    for elt in l[1:]:
        if elt < min:
            min = elt
    print("Valeur min: %d" % min)
min_liste_p(toto)
```

```
Valeur min: 0
```

Exercice : Remplacer tous les éléments de la liste vecteur par l'entier correspondant à leur indice.

Illustration de ce dont il s'agit:

[ ]:
```
vecteur[1] = 1
vecteur[2] = 2
print(vecteur[:10])
```

Première proposition, avec une boucle while et une gestion explicite des indices.

[36]:
```
i = 0
while i < len(vecteur):
    vecteur[i] = i
    i += 1
print(vecteur)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
```

Deuxième proposition: avec une boucle `for` + range() qui permet de manipuler aussi des indices. (On ré-initialise la variable `vecteur` avant de réaliser cette variante)

[16]:
```
vecteur = [0]*100
for i in range(len(vecteur)):
    vecteur[i] = i
```

```
print(vecteur)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
```

Parenthèse : quelques exemples sur l'utilisation de range().

[17]:
```
x = range(2,18,2)
print(x)
for e in x:
    print(e, end=' ; ')
print(list(x))
```

```
range(2, 18, 2)
2 ; 4 ; 6 ; 8 ; 10 ; 12 ; 14 ; 16 ; [2, 4, 6, 8, 10, 12, 14, 16]
```

Exercice : afficher les éléments de la liste titi qui ne sont pas de type "entier" (int).

Première proposition qui utilise le mot-clé `int` et le compare à la valeur de la fonction `type()`.

[19]:
```
titi = [9, False, 8.34, True, 22, 1e-3, 'a', -1]
def exo4(liste):
  for x in liste:
    if type(x) != int:
      print(x)
exo4(titi)
```

```
False
8.34
True
0.001
a
```

Deuxième proposition où on compare directement le type des éléments de la liste avec le type d'une constante connue (ici 3).

[20]:
```
for e in titi:
    if type(e) != type(3):
        print(e)
```

```
False
8.34
True
0.001
a
```

Exercice : dans une chaîne de caractères remplacer tous les 'a' par 'e'.

Parenthèse: Si `s` est une variable de type *string*, on peut la manipuler comme une liste pour l'afficher:

[27]:
```
s = "Bonjour"
for l in s:
    print(l, end='|')
print()
for l in list(s):
    print(l, end='|')
```

```
B|o|n|j|o|u|r|
B|o|n|j|o|u|r|
```

... mais on ne peut pas modifier une chaîne directement par affectation:

[28]:
```
s[3] = 'g'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-f5be08e1b8d4> in <module>
----> 1 s[3] = 'g'

TypeError: 'str' object does not support item assignment
```

... alors qu'on peut (évidemment) modifier la liste obtenue par conversion d'une chaîne de caractères:

[29]:
```
ls = list(s)
ls[3] = 'g'
print(ls)
```

```
['B', 'o', 'n', 'g', 'o', 'u', 'r']
```

Solution de l'exercice:

[30]:
```
s = "Sois sage ô ma douleur"
ls = list(s)
for i in range(len(ls)):
    if ls[i] == 'a':
        ls[i] = 'e'
print(ls)
```

```
['S', 'o', 'i', 's', ' ', 's', 'e', 'g', 'e', ' ', 'ô', ' ', 'm', 'e', ' ', 'd',
'o', 'u', 'l', 'e', 'u', 'r']
```