Corrige_DM2_ISN

September 30, 2019

0.1 DM numéro 2 ISN 2019/2020

0.2 Exercice 1

0.2.1 Question 1 : fonction retournant la somme des nombres contenus dans une liste

```
In [10]: def somme1(t):
             s = 0
             for k in t:
                 s = s + k
             return s

         def somme2(t):
             s = 0
             for k in range(len(t)):
                 s = s + k
             return s

         def somme3(t):
             s = 0
             for k in range(len(t)):
                 s = s + t[k]
             return s

         def somme4(t):
             s = 0
             for k in range(1,len(t)+1):
                 s = s + t[k]
             return s
```

Faisons un petit test

```
In [11]: for f in [somme1, somme2, somme3, somme4]:
             print(f([4,5,6]))
```

```
15
3
15
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-11-b2c2038a7405> in <module>
  1 for f in [somme1, somme2, somme3, somme4]:
----> 2     print(f([4,5,6]))


<ipython-input-10-f0f0a58f9faf> in somme4(t)
 20     s = 0
 21     for k in range(1,len(t)+1):
---> 22         s = s + t[k]
 23     return s


IndexError: list index out of range
```

Il faudrait le démontrer rigoureusement, mais on peut affirmer que les fonctions `somme1` et `somme3` retournent bien la somme des éléments de la liste de nombres qui leur est passée en argument.

0.2.2 Question 2

```
In [13]: def produit(t):
             """Retourne le produit des éléments d'une liste de nombres"""
             p = 1
             for e in t:
                 p = p * e
             return p
```

0.2.3 Question 3

```
In [14]: def memeTaille(t1, t2):
             """Retourne un booléen indiquant si les deux listes
             t1 et t2 sont de même taille"""
             return len(t1) == len(t2)
```

0.2.4 Question 4

```
In [16]: def somme2listes(t1, t2):
             """Retourne une liste t3 dont les éléments sont
             la somme des éléments de t1 et t2 de même index"""
             if memeTaille(t1, t2):
                 t3 = [0] * len(t1)
                 for k in range(len(t1)):
```

```
                t3[k] = t1[k] + t2[k]
            return t3
        return None
```

L'appel `somme2listes([4,5,6,7], [1,2,3])` retourne `None` car les deux listes `[1,2,3]` et `[4,5,6,7]` ne sont pas de même taille. La comparaison des tailles de `t1` et `t2` avec `memeTaille(t1, t2)` au début du bloc d'instruction de la fonction permet d'éviter la levée d'une exception en retournant `None` si les deux listes ne sont pas de même taille avec une liste `t1` plus longue que `t2`.

```
In [24]: def somme2listesV2(t1, t2):
             """Retourne une liste t3 dont les éléments sont
             la somme des éléments de t1 et t2 de même index"""
             t3 = [0] * len(t1)
             for k in range(len(t1)):
                 t3[k] = t1[k] + t2[k]
             return t3

In [25]: somme2listesV2([4,5,6,7], [1,2,3])


         ---------------------------------------------------------------------------

         IndexError                                Traceback (most recent call last)

         <ipython-input-25-f2d99b03c254> in <module>
         ----> 1 somme2listesV2([4,5,6,7], [1,2,3])


         <ipython-input-24-63172c10e710> in somme2listesV2(t1, t2)
               4     t3 = [0] * len(t1)
               5     for k in range(len(t1)):
         ----> 6         t3[k] = t1[k] + t2[k]
               7     return t3


         IndexError: list index out of range
```

0.3 Exercice 2 Récherche séquentielle

0.3.1 Question 1

```
In [28]: def rechercheMax(t):
             """Retourne le maximum d'une liste d'entiers"""
             if len(t) == 0:
                 return None
             maxi = t[0]
             for k in range(1, len(t)):
                 if t[k] > maxi:
```

```
                maxi = t[k]
        return maxi
```

0.3.2 Question 2

```
In [27]: def rechercheMiniMaxi(t):
             """Retourne le couple (minimum, maximum) d'une liste d'entiers"""
             if len(t) == 0:
                 return None
             maxi = t[0]
             mini = t[0]
             for k in range(1, len(t)):
                 if t[k] > maxi:
                     maxi = t[k]
                 elif t[k] < mini:
                     mini = t[k]
             return (mini, maxi)
```

0.3.3 Question 3

```
In [29]: def negatif1(t):
             for e in t:
                 if e < 0:
                     return True
                 else:
                     return False


         def negatif2(t):
             for e in t:
                 if e < 0:
                     return True
             return False

         def negatif3(t):
             for k in range(len(t)):
                 if t[k] < 0:
                     return True
             return False

         def negatif4(t):
             rep = False
             for k in range(len(t)):
                 if t[k] < 0:
                     rep = True
             return rep
```

Parmi les quatre fonctions précédentes, seule la fonction `negatif2` ne retournent pas `True` si l'un au moins des nombres contenus dans la liste `t` est négatif. En effet la fonction `negatif2`

retourne `True` ou `False` dès le premier tour de boucle donc si la liste comporte deux nombres tels que le premier est strictement positif et le second négatif, elle retourne `False` ce qui correspond uniquement au signe du premier nombre.

0.3.4 Question 4

```
In [30]: def listesEgales(t1, t2):
             """Retourne True si les listes sont de même taille avec les mêmes
             éléments dans l'ordre et False sinon"""
             if len(t1) != len(t2):
                 return False
             for k in range(len(t1)):
                 if t1[k] != t2[k]:
                     return False
             return True
```

0.4 Exercice 3

0.4.1 Question 1

```
In [46]: def mystere(n, debug = False):
             t = []
             if debug:
                     print("Test n >= 10 : ", n >= 10, " et t = ", t, "et n = ", n)
             while n >= 10:
                 t.append(n % 10)
                 n = n // 10
                 if debug:
                     print("Test n >= 10 : ", n >= 10, " et t = ", t, "et n = ", n)
             t.append(n % 10)
             if debug:
                 print("Test n >= 10 : ", n >= 10, "et t = ", t, " et n = ", n)
             return t
```

```
In [47]: mystere(842, debug = True)
```

```
Test n >= 10 :  True  et t =  [] et n =  842
Test n >= 10 :  True  et t =  [2] et n =  84
Test n >= 10 :  False  et t =  [2, 4] et n =  8
Test n >= 10 :  False et t =  [2, 4, 8]  et n =  8
```

```
Out[47]: [2, 4, 8]
```

0.4.2 Question 2

La liste retournée par `mystere(n)` représente la liste des chiffres de l'entier `n` dans l'ordre inverse de l'ordre habituel : les chiffres de poids faible d'abord.

0.4.3 Question 3

```
In [98]: def est_palindrome(n):
             """teste si nombre est palindrome"""
             chiffres = mystere(n)
             return listesEgales(chiffres, chiffres[::-1])
```

```
In [50]: (est_palindrome(353), est_palindrome(354))
```

```
Out[50]: (True, False)
```

```
In [115]: def chiffres2nombre(t):
              """Retourne le nombre à partir de la liste de ses
              chiffres dans l'ordre inverse"""
              n = 0
              puissance = 1
              for c in t:
                  n = n + c * puissance
                  puissance = puissance * 10
              return n
```

```
In [52]: chiffres2nombre(mystere(842))
```

```
Out[52]: 842
```

```
In [79]: chiffres2nombre([8,4,2])
```

```
Out[79]: 248
```

```
In [116]: def est_lychrel(n):
              """Retourne True si n est un pseudo-nombre de Lychrel
              c'est-à-dire qu'on n'aboutit pas à un palindrome en
              moins de 50 itérations"""
              k = 1
              #au moins une itération
              chiffres = mystere(n)
              n = n + chiffres2nombre(chiffres[::-1])
              while k <= 50 and not est_palindrome(n):
                  chiffres = mystere(n)
                  n = n + chiffres2nombre(chiffres[::-1])
                  k = k + 1
              return  k == 51
```

```
In [117]: est_lychrel(9999)
```

```
Out[117]: True
```

```
In [118]: lychrel = []
          for n in range(1, 10000):
              if est_lychrel(n):
                  lychrel.append(n)
          print(lychrel)
```

```
[196, 295, 394, 493, 592, 689, 691, 788, 790, 879, 887, 978, 986, 1495, 1497, 1585, 1587, 1675
```

```
In [119]: print(len(lychrel))
```

```
249
```