Correction des exercices du polycopié photocopiés

Exercice 118 : Il y a sept appels au total (en comptant le tout premier) :

```
rendu_monnaie([1,2], 3)
   rendu_monnaie([1 , 2] ,  2)
       rendu_monnaie( [1 , 2] ,  1)
            rendu_monnaie([1, 2 ] , 0)
        rendu_monnaie([1  , 2] , 0)
  rendu_monnaie( [1 , 2] , 1)
            rendu_monnaie([1 , 2] , 0)
```

On constate qu'on a fait deux fois l'appel pour s = 1 et trois fois l'appel pour s = 0.

Exercice 119

[0 , 1, 2, 3, 4, 5, 1, 2, 3, 4, 1, 2, 2]

La réponse est dans la dernière case : donc 2 dans ce cas.

Exercice 120 : A côté du tableau nb qui contient le nombre minimal de pièces nécessaires pour chaque entier, on ajoute un second tableau, sol, qui contient la solution, sous la forme d'une liste de pièces.

Exercice 121 : On se sert d'un tableau f dans lequel on calcule successivement toutes les valeurs $F_0, F_1 , \dots , F_n$.

```
  def   fib(n) :
        f  = [0]  *  (n+1)
        f[1]  = 1
        for  i  in  range(2 ,  n  +  1):
             f[i]    = f[i – 2]  + f[ i -1]
        return  f[n]
```

Exercice 122 :

		C	A	T
	0	-1	-2	-3
C	-1	1	0	-1
H	-2	0	0	-1
A	-3	-1	1	0
T	-4	-2	0	2

Le score maximal est donc 2.

Exercice 125

Comme dans le programme 47 du cours, on ajoute un second tableau, sol, à côté du tableau sc, l'idée étant que sol[i][j] contient une solution pour l'alignement des chaînes s1[0 :i] et s2[0 :j].
L'initialisation se fait avec des couples de chaînes vides :

```
sol = [[( '', '')] * (n2 + 1) for _in range(n1 + 1)]
```

Pour initialiser les bords du tableau, on écrit respectivement :

```
sol[i][0] = (s1[0 : i] , ''-'' * I )
```

et

```
sol [0][j] = ( '' –'' * j, s2[0 :j] )
```

ce qui correspond à des alignements avec le caractère - . En fin, il y a quatre cas de figure pour déterminer sol[i][j], selon la situation qui maximise le score. On peut réécrire la double boucle ainsi :

```
for i in range(1, n1 + 1) :
    for j in range(1, n2 + 1) :
        s = -1 + sc[i – 1][j]
        x , y = sol[I - 1][j]
        sol[i][j] = (x + s1[i - 1],[ y + ''-'')
        if s < -1 + sc[i][j – 1]:
            s = -1 + sc[i][j -1]
            x , y = sol[i][j-1]
            sol[i][j] = (x + ''- '' , y + s2[j-1])
        if s1[i – 1] == s2[j – 1] and s < 1 + sc[i – 1][j -1]:
            s = 1 + sc[i – 1][j -1]
            x , y = sol[i - 1][j -1]
```

sol [i][j] = (x + s1[i -1], y + s2[j – 1])

if s1[i-1] != s2[j – 1] and s < -1 + sc[i-1][j-1]:

Nous terminerons demain !!!