TP 2 – Récursivité –Corrigé

Exercice 1 : Somme des n premiers entiers naturels

On remarque que $S_0 = 1$ et $\forall n \in \mathbb{N}^*, S_n = n + S_{n-1}$. On peut donc procéder comme suit :

```
def somme(n):
    if n==0:
        return 0
    else :
        return n+somme(n-1)
```

Exercice 2 : Récursion mutuelle

```
def u(n):
    if n==0:
        return 1                #u_0=1
    else:
        return u(n-1)+v(n-1)    #u_n=u_(n-1) + v_(n-1)

def v(n):
    if n==0:
        return 0                #v_0=0
    else:
        return 2*u(n-1)+v(n-1) #v_n=2u_(n-1) + v_(n-1)
```

Exercice 3 : Coefficients binomiaux

```
def coefficient(n,p):
    #cas de base, 0 parmi n et n parmi n valent 1
    if p==0 or n==p:
        return 1
    else:
        #sinon, on applique la formule de Pascal
        return coefficient(n-1,p-1)+coefficient(n-1,p)
```

Exercice 4 : Recherche de PGCD

On suit simplement les relations proposées. On obtient

```
def pgcd_sous(m,n):
    if n==0:
        return m
    else :
        return pgcd_sous(n,abs(m-n))
```

De même, pour l'algorithme d'Euclide :

```
def pgcd_eucl(m,n):
    if n==0:
        return m
    else :
        return pgcd_eucl(n,m%n)
```

Exercice 5 : Palindromes

```
def palindrome(s):

    #un mot de 0 ou 1 lettre est un palindrome
    if len(s)<=1:
        return True
    else:
        #on vérifie que les 2 lettres extérieures sont identiques
        if s[0]==s[len(s)-1]:
            #si oui, on regarde si le mot intérieur est un palindrome
            return palindrome(s[1:len(s)-1])
        else:
            #si non, le mot n'est pas un palindrome
            return False
```

Exercice 6 : Diagonale de Cantor

On initialise avec le numéro de $(0, 0)$, qui est égal à 0.

Pour l'"hérédité", si (x, y) n'est pas sur l'axe des abscisses, le point précédent dans l'ordre de numérotation a pour coordonnées $(x + 1, y - 1)$.

S'il est sur l'axe des abscisses (à la position $(x, 0)$), le point précédent est sur l'axe des ordonnées (à la position $(0, x - 1)$).

On rajoute à chaque fois 1 au numéro du point précédent.

```
def numero(x,y):
    if x==0 and y==0: #cas de base : origine du repere
        return 0
    else:
        if y==0: #le point est sur l'axe des abscisses
            return 1+numero(0,x-1)
        else: #le point n'est pas sur l'axe des abscisses
            return 1+numero(x+1,y-1)
```

Exercice 7 : Recherche par dichotomie du zéro d'une fonction

```
def zero(f,a,b,e):
    c=(a+b)/2
    if b-a<e: #cas de base, la longueur de [a,b] est <=e
        return c #on renvoie (a+b)/2
    else:
        if f(a)*f(c)<=0: #f(a) et f(c) de signes opposes
            return zero(f,a,c,e) #on cherche dans l'intervalle [a,c]
        else: #f(a) et f(c) de meme signe
            return zero(f,c,b,e) #on cherche dans l'intervalle [c,b]
```

Exercice 8 : Figures récursives

Remarque : Pour que les figures soient visibles, il faut exécuter l'instruction `plt.axis('scaled')` avant `plt.show()`.

```
def CerclesRec2(x,y,r,pos):
    cercle(x,y,r)
    if r>1:
        if pos=='haut':
            CerclesRec2(x+3*r/2,y,r/2,'droite')
            CerclesRec2(x,y+3*r/2,r/2,'haut')
            CerclesRec2(x-3*r/2,y,r/2,'gauche')
```

```
        if pos=='bas':
            CerclesRec2(x+3*r/2,y,r/2,'droite')
            CerclesRec2(x,y-3*r/2,r/2,'bas')
            CerclesRec2(x-3*r/2,y,r/2,'gauche')
        if pos=='gauche':
            CerclesRec2(x,y+3*r/2,r/2,'haut')
            CerclesRec2(x,y-3*r/2,r/2,'bas')
            CerclesRec2(x-3*r/2,y,r/2,'gauche')
        if pos=='droite':
            CerclesRec2(x,y+3*r/2,r/2,'haut')
            CerclesRec2(x,y-3*r/2,r/2,'bas')
            CerclesRec2(x+3*r/2,y,r/2,'droite')
        if pos=='centre':
            CerclesRec2(x,y+3*r/2,r/2,'haut')
            CerclesRec2(x,y-3*r/2,r/2,'bas')
            CerclesRec2(x+3*r/2,y,r/2,'droite')
            CerclesRec2(x-3*r/2,y,r/2,'gauche')
```

On obtient la figure en exécutant `CerclesRec2(0,0,8,'centre')`

```
def carre(x,y,r):
    rec=plt.Rectangle((x,y),r,r,fill=False)
    F.add_patch(rec)
def RectanglesRec(x,y,r,pos):
    carre(x,y,r)
    if r>1:
        if pos=='haut':
            RectanglesRec(x+r,y+r/2,r/2,'droite')
            RectanglesRec(x,y+r,r/2,'haut')
            RectanglesRec(x-r/2,y,r/2,'gauche')
        if pos=='bas':
            RectanglesRec(x+r,y+r/2,r/2,'droite')
            RectanglesRec(x+r/2,y-r/2,r/2,'bas')
            RectanglesRec(x-r/2,y,r/2,'gauche')
        if pos=='gauche':
            RectanglesRec(x,y+r,r/2,'haut')
            RectanglesRec(x+r/2,y-r/2,r/2,'bas')
            RectanglesRec(x-r/2,y,r/2,'gauche')
        if pos=='droite':
            RectanglesRec(x,y+r,r/2,'haut')
            RectanglesRec(x+r/2,y-r/2,r/2,'bas')
            RectanglesRec(x+r,y+r/2,r/2,'droite')
        if pos=='centre':
            RectanglesRec(x,y+r,r/2,'haut')
            RectanglesRec(x+r/2,y-r/2,r/2,'bas')
            RectanglesRec(x+r,y+r/2,r/2,'droite')
            RectanglesRec(x-r/2,y,r/2,'gauche')
```

On obtient la figure en exécutant `RectanglesRec(0,0,8,'centre')`