python

```
print("Hello, world!")
```

FORMATION PYTHON : LA RECURSIVITE

Ce chapitre a pour but d'initier à la programmation des fonctions récursives en Python.

said_anis@hotmail.com

http://cahier-de-prepa.fr/info-ipein

La récursivité

Définition :

- Un **algorithme récursif** est un algorithme qui résout un problème en calculant des solutions d'instances plus petites du même problème.
- Une fonction est dite récursive si elle s’appelle elle-même

said_anis@hotmail.com

La récursivité

L’idée générale de la récursivité c’est pour traiter un problème de taille n, on le décompose récursivement en sous-problèmes

Cela n’a d’intérêt que si le cumulé des appels récursifs est inférieur au traitement direct du problème

said_anis@hotmail.com

Analyse des programmes récursifs

Nous seront confrontés lors de l'écriture et de l'analyse des programmes récursifs à trois problèmes :

1. Le programme termine-t-il ?
2. Est-il conforme à sa spécification ?
3. Quelle est sa complexité (temps et mémoire) ?

said_anis@hotmail.com

La récursivité

On dit qu'une fonction f est récursive si son exécution peut provoquer un ou plusieurs appels de f elle-même.

On distinguera l'appel principal de f des appels récursifs

L'appel principal se fait lorsque f n'est pas encore en cours d'exécution

Les appels récursifs sont les appels provoqués par l'exécution de f

➔ Un langage récursif est un langage dans lequel on peut programmer des fonctions récursives

said_anis@hotmail.com

Analyse récursive

- Le cas de base (*n = 0*) :
  - C'est un cas particulier pour lequel la valeur à retourner ne nécessite pas d'appel à la fonction *f*.
  - Sans sa présence, la fonction ne peut pas se terminer. (garantie l'arrêt du programme).
  - Une fonction récursive peut avoir un ou plusieurs cas de bases
- *Le cas de propagation (cas général)* :
  - C'est lui qui contient l'appel récursif et dont l'un ou plusieurs des arguments qui lui sont transmises doivent converger et arriver à la condition d'arrêt (n=0).
  - le cas général doit tendre vers le cas de base

said_anis@hotmail.com

Application 1

- Ecrire une fonction récursive **puiss (x, n)**, qui permet de calculer x^n .

```
>>> def puiss (x,n):
        "Calcul x**n de facon recursive"

        if n==0:
                return(1)
        else:
                return(x*puiss(x,n-1))

>>> puiss(2,5)
32
```

Exercice 1


```
def f(a,b) :
    """ a et b sont deux entiers naturels non nuls. """
    if b==1 :
        return a
    return a+f(a,b-1)

print f(3,5)
```

1. Quel est le résultat affiché par ce programme?
2. Quel est le cas de base dans cette fonction récursive ?
3. Qu'est ce qui garantit dans les appels récursifs que le programme finira par s'arrêter ?
4. Que retourne *f* (*a*,*b*) (*a* et *b* étant des entiers naturels non nuls) ?

said_anis@hotmail.com

Remarques :

- A chaque nouvel appel récursif, des nouvelles variables sont crées qui portent les mêmes noms mais pas les mêmes valeurs
- L'ensemble des variables avec leurs valeurs de chaque appels récursifs composent le contexte de l'appel récursif principal

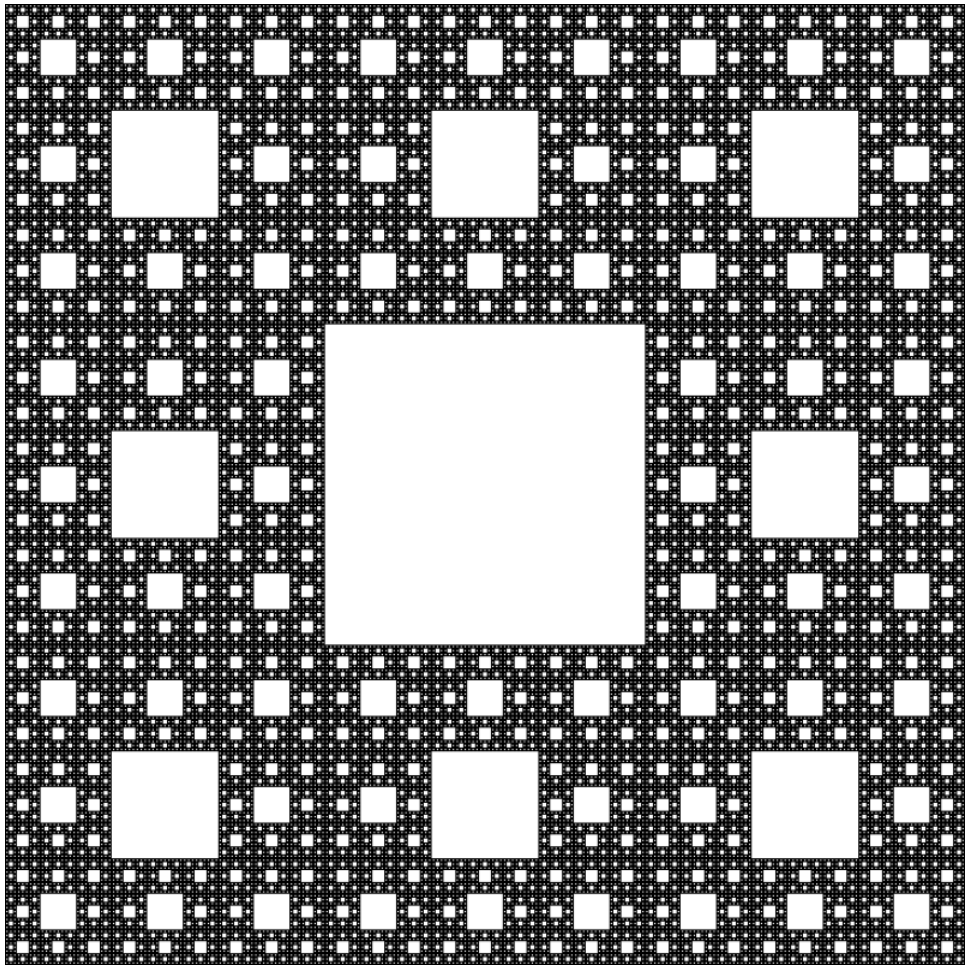
said_anis@hotmail.com

La récursivité : Exemples

- **Exemple :**

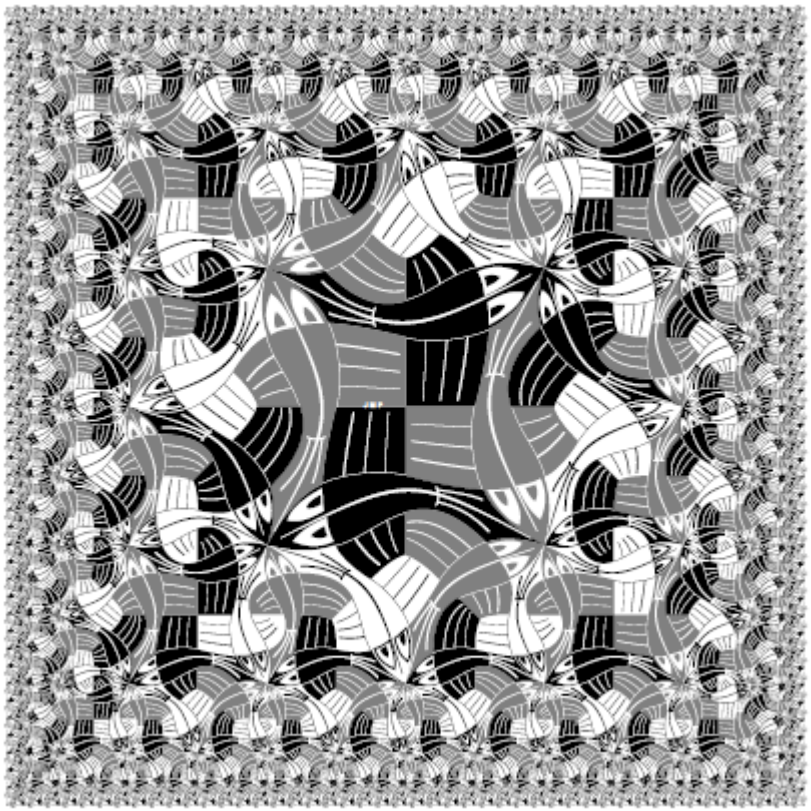
Le *tapis de Sierpiński* est obtenu à partir d'un carré. Le tapis se fabrique en découpant le carré en neuf carrés égaux avec une grille de trois par trois, et en supprimant la pièce centrale,

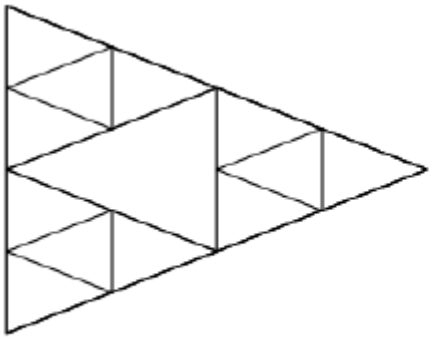
et en appliquant cette procédure récursivement aux huit carrés restants.

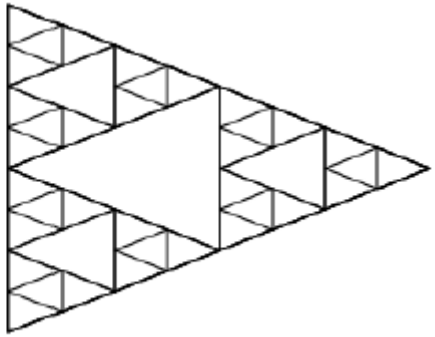


said_anis@hotmail.com

La récursivité : Exemples







said_anis@hotmail.com

Les types des fonctions récursives :

1. La récursivité simple
2. La récursivité multiple
3. La récursivité mutuelle (croisée)
4. La récursivité imbriquée

said_anis@hotmail.com

1. Fonction récursive simple

Définition :

Une fonction récursive simple, contient un seul appel récursif dans son corps.

Exemple: le calcul de la somme des éléments d'une liste.

```
>>> def somme_liste(l):
        if l== []:
                return(0)
        return(l[0]+somme_liste(l[1:]))

>>> somme_liste([1,2,3,4,5])
15
```

Exercice n° 2

- Écrire une fonction récursive qui calcule la somme des *n* premiers entiers.

```
>>> def g(x):
        if (x==0): #cas de base
                return 0
        else:
                return x + g(x-1)


>>> g(3)
6
>>> g(10)
55
```

said_anis@hotmail.com

2. Fonction récursive multiple

Définition :

Une fonction récursive multiple est une fonction qui contient plus qu'un appel récursif dans son corps.

Exemple: Le calcul du nombre de combinaisons

$$C_n^p = \begin{cases} 1 & si\ p = 0\ ou\ p = 1 \\ C_{n-1}^{p} + C_{n-1}^{p-1} & sinon \end{cases}$$

```
>>> def combinaisons (n,p):
        if p==0 or p==n:
                return(1)
        else:
                return(combinaisons (n-1,p)+combinaisons (n-1,p-1))
```

Exercice n° 3

- **Suite de Fibonacci.** On considère la suite de Fibonacci définie par :

$$F = \begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-2} + F_{n-1} \quad pour\ n \geq 2 \end{cases}$$

Écrire une fonction récursive basée sur ces relations qui prend *n* en argument et renvoie F_n.

Combien d'appels à la fonction *f* sont-ils faits pour calculer *f* (5) ?

said_anis@hotmail.com

Exercice n° 3 : Solution

- **Suite de Fibonacci : Solution + calcul de nombre d'appels récursifs**

```
c=0
def fibo (n) :
    global c
    c+=1
    if n==0 or n==1:
        return 1
    else :
        return (fibo(n-1) + fibo(n-2))

print(fibo(5))
print(c)
```

said_anis@hotmail.com

3. Fonctions mutuellement récursives

Définition :

Deux fonctions sont dites mutuellement récursives si elles dépendent les unes des autres

Exemple :

f et g sont deux fonctions *mutuellement récursives* si la définition de la fonction **f** appelle la fonction **g**, et la définition de **g** appelle elle-même **f**.

```
f():
    g()
```

```
g():
    f()
```

said_anis@hotmail.com

3. Fonctions mutuellement récursives

- **Exemple :** La fonction **pair** qui permet de déterminer si un entier *n* est pair par récurrence mutuelle avec une fonction **impair** qui, elle, détermine si un entier *n* est impair.

```
>>> def pair(n):
        return (n == 0) or impair(n-1)

>>> def impair(n):
        return (n != 0) and pair(n-1)

>>> pair(3)
False
>>> pair(30)
True
```

said_anis@hotmail.com

4. Fonctions récursives imbriquées

Définition :

Une fonction récursive est dite imbriquée si l'appel récursif contient lui aussi un appel récursif.

Exemple: La fonction d'Ackermann

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1,A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0. \end{cases}$$

said_anis@hotmail.com

Exercice n° 5 : Séquences de Collatz

Une séquence de Collatz s'obtient à partir d'un entier n de départ auquel on applique de manière itérative la fonction

$$C(n) = \begin{cases} n/2 & \text{si n est pair} \\ 3n+1 & \text{sinon} \end{cases}$$

Par exemple, si l'on part de la valeur n = 7, on obtiendra la suite : 7 ; 22 ; 11 ; 34 ; 17 ; 52 ; 26 ; 13 ; 40 ; 20 ; 10 ; 5 ; 16 ; 8 ; 4 ; 2 ; 1 …Cette suite boucle indéfiniment sur le cycle {1 ; 4 ; 2}. Et cela semble se produire quelque soit la valeur choisie initialement, bien que le nombre d'étapes soit parfois étonnement élevé avant d'y parvenir.

said_anis@hotmail.com

Exercice n° 5 : Séquences de Collatz

- La conjecture de Syracuse est que toute itération de cette fonction à partir d'un entier naturel non-nul aboutit à la valeur 1.

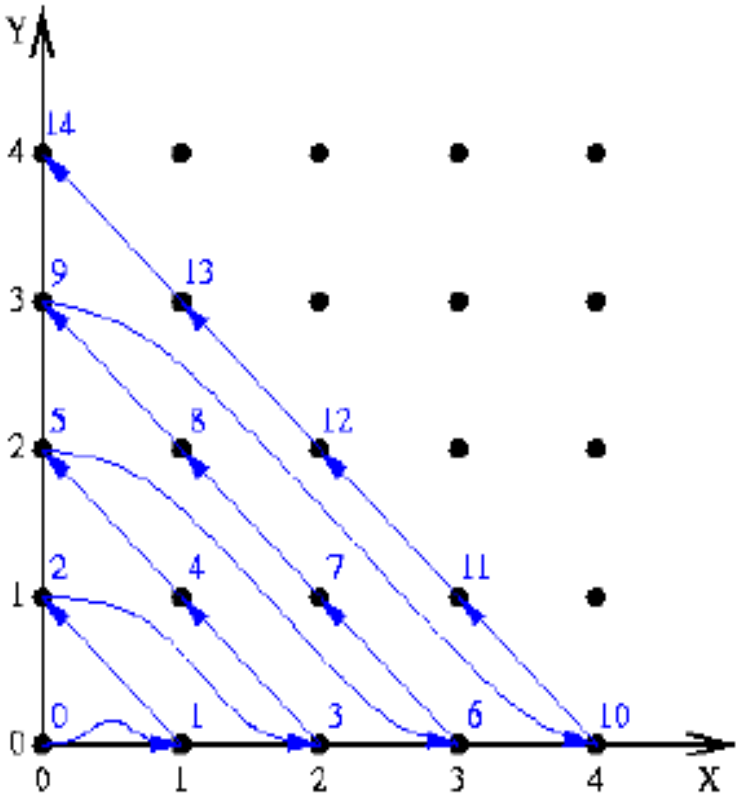
$$C(n) = \begin{cases} n/2 & \text{si n est pair} \\ 3n+1 & \text{sinon} \end{cases}$$

- Ecrire une fonction récursive qui calcule et affiche les termes de la suite de Syracuse.

said_anis@hotmail.com

Exercice n° 6

On numérote chaque point du plan de coordonnées (*x*; *y*) (où *x* et *y* sont des entiers naturels) par le procédé suggéré sur cette figure.



Écrire une fonction **numero (x, y)** définie de façon récursive, qui retourne le numéro du point de coordonnées (*x*; *y*).

Exemple : numéro (1,1) retourne 4

said_anis@hotmail.com

Exercice n° 6 : Solution


```
def numero (x,y):
    if (x==0 and y==0):
        return 0
    elif y==0:
        return 1 + numero(0,x-1)
    else:
        return 1+ numero(x+1,y-1)

print(numero(1,1))
print(numero(0,4))
```

said_anis@hotmail.com

Fin ,
Merci

Autres cours :

http://cahier-de-prepa.fr/info-ipein

Créer par : Anis SAIED

said_anis@hotmail.com