Lycée Marcelin BERTHELOT
BCPST 1C – Mme MOREL

Fonctions récursives

1 Définition et syntaxe

Définition 1 *Une* **fonction récursive** *est une fonction qui s'appelle elle-même.*
Python accepte de faire appel à une fonction `f` *au sein même de la définition de* `f`*. On parle d'*__appel récursif__ *à* `f`*.*

Exemple 1 (u_n) est la suite donnée par $u_0 = 4$ et $\forall n \geqslant 1$, $u_n = n + u_{n-1}^2$.

```
def suite(n):
        if n ==0:
                return 4
        else:
                return n+suite(n-1)**2
```

Syntaxe :

```
def fonction(args):
        if condition d'arrêt :
                return valeur
        appel récursif
```

Remarque 1 Lors de l'appel récursif, attention au **type** de données qu'il renvoie : il doit être cohérent avec le type renvoyé par la condition d'arrêt !

Exemple 2 : Écrire une fonction qui renvoie la valeur de la somme : $S_n = \sum_{k=0}^{n} 2^k$**.**

Remarque 2 Attention, l'appel récursif à `f` est autorisé, mais c'est à vous de vous assurer que cet appel permet à Python de redescendre in fine à la condition d'arrêt.
Que se passe-t-il par exemple dans la situation suivante ?

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

```
RecursionError :  maximum recursion depth exceeded in comparison
```

2 `for` vs récursivité

2.1 Retour sur les suites récurrentes

Exemple 3 Écrire deux fonctions Python, l'une récursive, l'autre utilisant une boucle `for`, prenant en argument $n \in \mathbb{N}$ et renvoyant u_n où (u_n) est la suite définie par $u_1 = 3$ et : $\forall n \in \mathbb{N}^*,\ u_{n+1} = \dfrac{u_n}{n^2+2}$.

Exemple 4 Écrire deux fonctions Python, l'une récursive, l'autre utilisant une boucle `for`, prenant en argument $n \in \mathbb{N}$ et renvoyant u_n où (u_n) est la suite donnée par $u_0 = 4,\ u_1 = 4$ et $\forall n \geqslant 2$,
$u_n = n + u_{n-1}^2 + u_{n-2}^3$.

2.2 Retour sur les sommes / produits

Parfois, la relation de récurrence n'est pas explicitement donnée, c'est alors à vous de la trouver.

Exemple 5 Écrire deux fonctions Python, l'une récursive, l'autre utilisant une boucle `for`, prenant en argument $n \in \mathbb{N}$ et renvoyant $n!$

Exemple 6 Écrire deux fonctions Python, l'une récursive, l'autre utilisant une boucle `for`, prenant en argument $n \in \mathbb{N}$ et renvoyant la valeur de $u_n = \displaystyle\sum_{k=0}^{n} \frac{1}{k+n}$

3 Entraînement

3.1 Lecture d'algorithme

Exercice 1 On définit la fonction suivante, qui sera appelée avec `a` et `b` des entiers naturels non nuls :

```
def f(a,b):
    if b==1 :
        return a
    else :
        return a + f(a,b-1)
```

1. Que renvoie `f(3,4)` ? De manière générale, que renvoie `f(a,b)` ?
2. Quelle est la condition d'arrêt de cette fonction récursive ? Pourquoi n'y a-t-il pas de boucle infinie ?

Exercice 2 Vrai / Faux ?
Le programme suivant permet d'afficher des entiers dans l'ordre décroissant :

```
def prog(n):
        if n!=0 :
                print("valeur",n)
                prog(n-1)
```

3.2 Suites récurrentes

Exercice 3 :
(u_n) définie par : $u_0 = 0$ et $\forall n \geqslant 0,\ u_{n+1} = u_n^2 - 3$.
Écrire une fonction récursive prenant en argument $n \in \mathbb{N}$ et renvoyant u_n.

Exercice 4 : suite de Fibonacci.
(u_n) définie par : $u_0 = 0,\ u_1 = 1$ et $\forall n \geqslant 0,\ u_{n+2} = u_{n+1} + u_n$.
Écrire une fonction récursive prenant en argument $n \in \mathbb{N}$ et renvoyant u_n.

Exercice 5 Soit la suite (u_n) définie par :

$$u_0 = 0 \ \text{ et } \ \forall n \in \mathbb{N},\, u_{n+1} = \begin{cases} u_n + 2n + 1 & \text{si } n+1 \text{ est impair} \\ 4\, u_{\frac{n+1}{2}} & \text{si } n+1 \text{ est pair} \end{cases}$$

Écrire une fonction récursive d'argument n et renvoyant la valeur de u_n pour tout entier n.

3.3 Exponentiation rapide

Si `f` est une fonction définie récursivement, calculer `f(n)` ne se fait pas toujours à l'aide de l'appel récursif de `f(n-1)`. On peut faire un appel récursif à n'importe quelle valeur `f(k)` tant que cet appel nous ramène à la condition d'arrêt.
Par exemple, une situation classique consiste à faire appel à `f(n/2)` si `n` est pair et à `f((n-1)/2)` si `n` est impair. On parle d'**exponentiation rapide**.

Exercice 6 :

1. Écrire une fonction récursive `puissance` prenant en argument un réel x et un entier naturel n et renvoyant x^n.
2. (a) Soient $x \in \mathbb{R}$ et $n \in \mathbb{N}$. Si n est pair, comment calculer x^n en fonction de $x^{n/2}$? Si n est impair, comment calculer x^n en fonction de $x^{(n-1)/2}$?
   (b) Écrire alors une autre fonction récursive `exporapide` prenant en argument un réel x et un entier naturel n et renvoyant x^n.
3. Combien de multiplications sont nécessaires au calcul de x^{17} dans `puissance(x,17)` et dans `exporapide(x,17)` ?

3.4 Triangle de Pascal

Exercice 7 :

1. Rappeler la formule de Pascal.
2. En déduire une fonction récursive prenant en arguments deux entiers et renvoyant le coefficient binomial correspondant.
3. Écrire un programme permettant d'écrire les N premières lignes du triangle de Pascal.