

## IX] Approche de la récursivité



### Définition

La récurrence et la récursivité sont des notions proches, l'apprentissage de l'une aide à comprendre l'autre. Voici une suite définie par récurrence :  $u_0 = 7$  et pour  $n \in \mathbb{N}^*$ , on a :  $u_n = 3 \times u_{n-1} + 1$ .

Si on demande de calculer  $u_5$ , on peut avoir deux approches :

1.  $u_0 = 7$ , donc  $u_1 = 3 \times 7 + 1 = 22$ , donc  $u_2 = 3 \times 22 + 1 = 67$ , donc  $u_3 = 3 \times 67 + 1 = 202$ , donc  $u_4 = 3 \times 202 + 1 = 607$ , donc  $u_5 = 3 \times 607 + 1 = 1822$ .
2.  $u_5 = 3 \times u_4 + 1$ , donc je demande la valeur de  $u_4$ . Or  $u_4 = 3 \times u_3 + 1$ , donc je demande la valeur de  $u_3$ , etc. Lorsque  $u_0$  est demandé, on répond 7, donc on peut, en remontant, calculer  $u_5$ .

La récursivité fonctionne suivant la deuxième approche, c'est un outil très puissant, parfois indispensable et qui souvent simplifie un programme. En Python, on peut programmer cette suite ainsi :

```
def suiteU(n):  
    if n==0:  
        return 7  
    else:  
        return 3*suiteU(n-1)+1
```

```
# suite: utilisation  
nombre = suiteU(5)  
print("u(5) est égal à", nombre)
```



### Exemples

Un bon exemple :

```
def facto(n):  
    if n==0:  
        return 1  
    else:  
        return n*facto(n-1)
```

Calculer à la main `facto(5)`.

N'oubliez pas que l'ordinateur ne conserve pas en mémoire ce qui n'a pas été demandé ; ainsi `fibonacci(3)` sera appelé deux fois, une première fois pour `fibonacci(5)`, et encore une fois pour `fibonacci(4)`, en recommençant tous les calculs!!! Exercice : reprogrammer `fibonacci(n)` de manière plus efficace.

Un mauvais exemple :

```
def fibo(n):  
    if n<2:  
        return n  
    else:  
        return fibo(n-1)+fibo(n-2)
```

Calculer à la main `fibonacci(5)`.

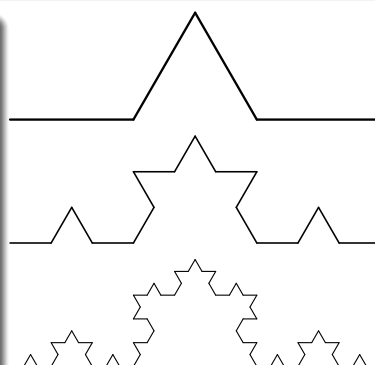


### Des fractales

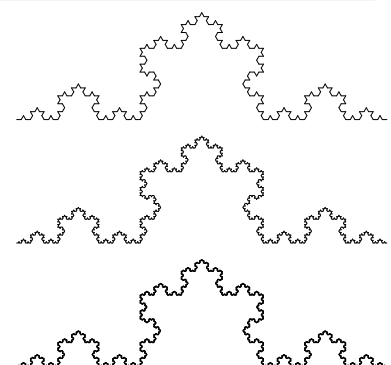
La courbe de VON KOCH

```
# avec la tortue  
def koch(x, n):  
    if n==0:  
        forward(x)  
    else:  
        koch(x/3, n-1)  
        left(60)  
        koch(x/3, n-1)  
        right(120)  
        koch(x/3, n-1)  
        left(60)  
        koch(x/3, n-1)  
koch(270,6) # Appel n°1
```

Tracer à la main `koch(9, 0)`, puis `koch(9, 1)`, puis `koch(9, 2)` et `koch(9, 3)`.



Étapes 1 à 6 de la courbe de VON KOCH.





À l'étape  $n$ , chaque segment de l'étape 1 est remplacé par une réduction au tiers de l'étape  $n-1$ . C'est un appel récursif. Lorsqu'on appelle `koch(270,6)`, on souhaite une courbe qui a une base de 270 pixels de large, et une **profondeur de récursion** de 6 ; il y a 6 étapes.

Mais on peut aussi considérer que chaque segment de l'étape  $n$  se transforme en un chemin de l'étape 1 ; un triangle équilatéral est placé au tiers du segment.


Vous pouvez faire la construction de l'arbre de PYTHAGORE avec l'aide de la **feuille n°8**.

## X] Pour aller plus loin

### X-a) Différences entre Python 2 et 3

Nous avons choisi  Python 3 parce qu'il est très répandu et idéal pour débiter. Mais c'est la version 2 qui est aujourd'hui la plus répandue, et de loin. Pourquoi débiter avec  Python 3 ?


1. Guido VAN ROSSUM, le créateur de Python, le conseille.
2. L'encodage des fichiers est naturellement en UTF-8, il n'y a donc plus aucun problème d'accent.
3. `print( . )` est une fonction comme les autres, son fonctionnement est simplifié et amélioré.
4. La division est redéfinie, elle n'est plus source d'erreurs : `13/3 == 13.0/3` mais faux avec la version 2 !
5. Il n'y a pas de limite aux nombres entiers, seuls le temps de calcul et la capacité d'affichage sont un frein.
6. De manière générale le code a été épuré. . .


Ainsi la version d'avenir, pour un débutant est  Python 3.

Les détracteurs répondront que toutes les bibliothèques de la version 2 n'ont pas encore été portées à la version 3 ; les débutants s'en moquent, cela ne les concernerait que dans plusieurs années, et alors tout aura été porté.

### X-b) Des ouvrages de référence

En français :


Un cours complet facile sur  Python 3 : faire une recherche Google avec [apprendre python 3](#).


Et télécharger gratuitement le livre libre *Apprendre à programmer avec  Python 3*, de Gérard SWINNEN.

Pour les étudiants, une approche plus rapide, proposée par [Bob CORDEAU](#) sur l'afpy.

En anglais :


Un autre cours, très détaillé et facile : chercher [non-programmer tutorial python 3](#).

La [documentation officielle](#) de  Python 3 propose un tutoriel et toutes les références possibles.

Enfin, pour ceux qui connaissent déjà l'algorithmique et souhaite apprendre  Python 3 plus vite : [Plongez au cœur de Python 3](#).

## A Annexe : Feuilles d'exercices

Les pages suivantes sont des feuilles d'exercices de difficulté variée.

Pour comparer  Python 3 à une calculatrice : [feuille n° 1](#)

Pour faire son premier programme : [feuille n° 2](#)

Pour travailler sur la boucle `while` : [feuille n° 3](#)

Pour travailler sur la boucle `for` et les tableaux : [feuille n° 4](#)

Pour faire vos premiers dessins avec la tortue : [feuille n° 5](#), [feuille n° 6](#) et [feuille n° 7](#)

Pour construire un arbre de PYTHAGORE : [feuille n° 8](#)

3. Pour plusieurs valeurs de  $x$ , calculer `progA(x)` et `progB(x)`. Quelle conjecture faire ?
4. Démontrer cette conjecture. (Niveau 4<sup>e</sup>-3<sup>e</sup>)
5. Écrire `progC(x)` en Python une fonction d'un argument  $x$ , qui calcule  $C = x(3x + 8) - 15$ .  
*Attention, ici on ne veut qu'une seule opération mathématique par instruction. C'est la règle du jeu, certes stupide !*
6. Même chose avec  $D = (5x - 2)(2x + 8)$

**A2 : mordu ?****Exercice n° 1 : Tout ça pour ça ?**

On suppose qu'au départ  $A = 1$ ,  $B = 1$ ,  $C = 1$ .

On lance l'algorithme suivant :

N°	Instructions	A	B	C
1	$A \leftarrow 2$	2	1	1
2	$B \leftarrow 10$			
3	$C \leftarrow A + B$			
4	$C \leftarrow B - C$			

1. Compléter le tableau.
2. Refaire l'exercice en remplaçant l'instruction 1 :  
 $A \leftarrow 2$  par  $A \leftarrow 5$ , puis par  $A \leftarrow (-3)$ .
3. Quelle conjecture faites-vous ?  
Démontrez-la avec  $A \leftarrow x$  à l'instruction 1.

**Exercice n° 2 : Compléter le tableau**

	X
Instructions	0
$A \leftarrow 1$	
$B \leftarrow 9$	
$C \leftarrow 4$	
$D \leftarrow 8$	
$X \leftarrow A$	
$X \leftarrow X * 10 + B$	
$X \leftarrow X * 10 + C$	
$X \leftarrow X * 10 + D$	

Conseil : écrire à droite, ligne par ligne, les valeurs de A, B, C et D.

**Exercice n° 3 : Échange de variables ?**

	A	B
Instructions	a	b
$A \leftarrow B$		
$B \leftarrow A$		

1. Que fait cet algorithme ?
2. En utilisant une troisième variable C, écrire un algorithme qui permute les valeurs des variables A et B.
3. [Bien plus dur] Et sans utiliser d'autre variable ?

**Exercice n° 4 : B = A ?**

	A	B
Instructions	a	b
$A \leftarrow A + B$		
$B \leftarrow A - B$		
$A \leftarrow A - B$		

Que fait cet algorithme ?

**Exercice n° 5 : Permutation circulaire**

Donner un algorithme qui réalise ceci :

Situation initiale :

A	B	C
a	b	c

Situation finale :

A	B	C
c	a	b

On utilisera une autre variable D.

**Exercice n° 6 : La fonction carré**

Avec Python (*shell*), définir la fonction carré.  
(/!\ Avec Python 2, accents interdits !)

```
>>> def carré(x):
    return x*x

>>> carré(11) # le carré du nombre 11
121
```

Calculer les carrés des entiers jusqu'à 13.  
Il est conseillé de connaître ces nombres.

**Exercice n° 7 : Premier programme**

Voici un programme de calcul :

- ↪ Choisir un nombre entier N
- ↪ Lui ajouter 4
- ↪ Multiplier la somme obtenue par N
- ↪ Ajouter 4 à ce produit.

1. Faire fonctionner « à la main » ce programme de calcul pour les entiers N compris entre 0 et 6.
2. Faire une conjecture.
3. Traduire ceci en un algorithme, qui contiendra exactement trois instructions d'affectation.
4. Programmer ceci en Python, vérifier pour  $N \in \llbracket 0; 6 \rrbracket$ .  
Tester d'autres valeurs pour conforter votre conjecture.
5. Prouver la conjecture. (Ça, c'est des maths)

### A3 : recherche **while** désespérément

#### Exercice n° 1 : S'arrêtera ?

```
N = 153
i = 121
while (N>i):
    N=2*N-5
    i=2*N+7
    print(N, i)
```

Ce programme fait

- $N \leftarrow 2 \times 153 - 5 = 301$
- $i \leftarrow 2 \times 121 + 7 = 249$
- Affichage de : 301 249
- $301 > 249$  donc on boucle à nouveau
- $N \leftarrow 2 \times 301 - 5 = 597...$

Écrire la suite.



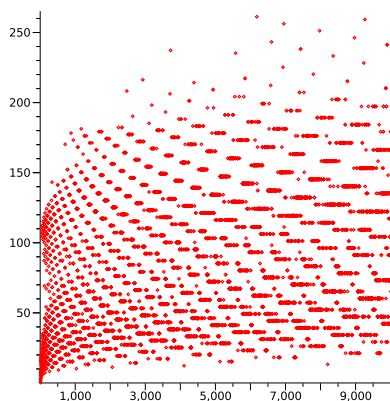
#### Exercice n° 2 : Liste des diviseurs

```
N = 154
i = 1
while (i**2<N):
    if (N%i==0):
        print(i, "*", N/i, "=", N)
    i=i+1
print(N, "possède ? diviseurs.")
```

1. Que fait ce programme ? Écrire toutes les étapes.  
Aide : % donne le reste d'une division euclidienne.
2. Modifier le pour qu'il affiche un nombre correct au lieu de « ? ».
3. Vérifier avec  $N = 7$ ,  $N = 8$  et  $N = 9$  après avoir fait le calcul à la main.

#### Exercice n° 3 : La suite de Syracuse

1. Comment en Python peut-on savoir si un entier est pair ou impair ?  
(Penser à % : reste de division euclidienne)
2. Voici une suite : 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.  
Règle : on part de  $N = 14$ , qui est pair, donc on le divise par 2.  
On obtient 7 qui est impair, dans ce cas on calcule  $3 \times 7 + 1 = 22$ .  
Et on recommence : pair ou impair, tant que le résultat n'est pas égal à 1.  
Personne ne sait si cette suite s'arrête pour tout entier  $N$  au départ, on conjecture que oui.  
Écrire en Python cette boucle avec  $N = 14$  au départ.
3. Essayer avec d'autres valeurs de départ, comme  $N = 27$  par exemple.
4. Modifier votre programme pour afficher le nombre de boucles (qu'on appelle *durée de vol*), et la valeur maximale atteinte (qu'on appelle *altitude maximale*).  
Vérifier que pour  $N = 14$ , la durée de vol est 17, et l'altitude maximale est 52.
5. Quel est le plus petit entier  $N$  qui donne une durée de vol supérieure ou égale à 100 ?
6. Quel est le plus petit entier  $N$  qui donne une altitude maximale supérieure ou égale à 9232 ?



7. Ce graphique est en lien avec l'exercice.  
Trouver lui une légende et un titre.

## **A4 : Tableaux et boucle `for`**

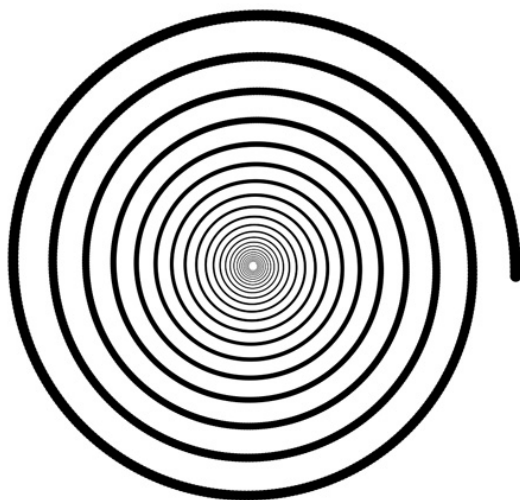
à suivre

## **A5 : Premières figures**

à suivre

**A6 : Tournicoti**

Objectif : construire une spirale  
avec le module `turtle`

**Exercice n° 1 : Première tentative**

1. Faire le programme suivant :
  - Choisir un rayon  $R$  de 200 pixels
  - faire un quart de cercle avec ce rayon
  - diminuer le rayon de 3 pixels
  - recommencer en boucle 50 fois.
2. Que se passe-t-il si on fait 100 boucles ?
3. Au lieu de faire une réduction de 3 pixels par quart de cercle, faire une réduction de 2 % par quart de cercle.
4. Modifier le programme pour avoir un trait d'épaisseur 5 pixels au départ, et 1 pixel à la fin de la boucle.

*Aide : vous aurez surement besoin d'arrondir un nombre.*

`from math import *` # Ça sert toujours  
`ceil(2.568)` renvoie 3, l'entier arrondi par excès.

Cette spirale n'était qu'une approximation d'une spirale d'ARCHIMÈDE.  
Essayons maintenant l'approximation d'une spirale de FIBONACCI.

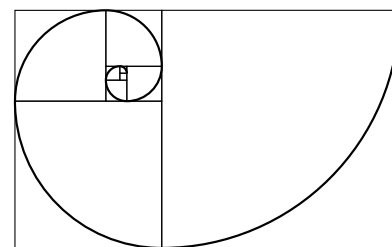
**Exercice n° 2 : La suite de FIBONACCI**

Voici la suite de FIBONACCI : 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

1. Vérifier que chaque terme est la somme des deux précédents (s'ils existent).
2. Calculer de tête les huit termes suivants (et les écrire).
3. Faire un programme Python qui stocke dans un tableau  $N$  termes de la suite à partir de [1, 2, ...]. ( $0$  n'est pas un bon candidat pour un côté.)

*Aide : Créer un tableau `Tablo=[1, 2]` avec les deux premières valeurs.*

`Tablo.append(3)` ajoute la valeur 3 à la suite de votre tableau.

**Spirale d'Or**

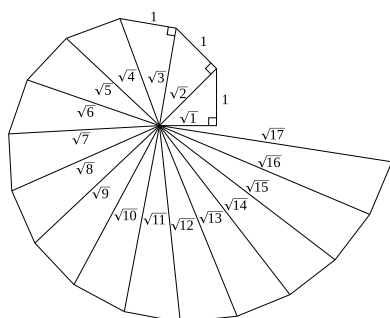
Les côtés des carrés sont des termes de la suite de FIBONACCI.

Pour faire une approximation de la spirale d'Or, on peut faire des quarts de cercle dans chaque carré.

**Exercice n° 3 : Votre spirale d'Or**

On choisit  $N=15$ .

1. Avec l'exercice précédent, créer un tableau de  $N$  valeurs successives de la suite de FIBONACCI, en commençant par [1, 2, ...].
2. En appliquant un coefficient de proportionnalité, construire un tableau dont le plus grand élément est égal à la moitié de la hauteur de votre écran en pixel.
3. Tracer des quarts de cercle de rayons décroissants issus de votre tableau.

**Exercice n° 4 : Et THÉODORE DE CYRÈNE alors ?**

On choisit  $a = 100$  pixels comme côté commun et  $N = 17$  triangles.

1. Construire un triangle rectangle isocèle de petit côté  $a$  pixels, les coordonnées de ses sommets seront  $(0;0)$ ,  $(a;0)$ ,  $(a;a)$ .  
*Aide : (Aller-retour) ici = `pos()` et `setpos(ici)` sont vos amis.*
2. Placée en  $(a,a)$ , l'instruction `setheading(towards(0,0))` dirige la tortue vers l'origine. La tourner de  $90^\circ$  et construire le triangle suivant.
3. Construire cette spirale (dite de THÉODORE DE CYRÈNE).
4. Prendre  $a = 10$  et  $N = 500$  et ne plus tracer l'hypoténuse.



Coupe d'un nautilus.



## B Sur ce document

### B1 : Licence option 1

DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE  
Version 2, December 2004

Copyright © 2004 Sam Hocevar

14 rue de Plaisance, 75014 Paris, France

*Everyone is permitted to copy and distribute verbatim or modified copies of this license document, and changing it is allowed as long as the name is changed.*

DO WHAT THE FUCK YOU WANT TO PUBLIC LICENSE  
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

o. You just DO WHAT THE FUCK YOU WANT TO.



### B2 : Licence option 2



Cet article est publié sous la licence Creative Commons-BY-NC-SA :

BY [Paternité] Vous devez citer le nom de l'auteur original

NC [Pas d'Utilisation Commerciale] Vous n'avez pas le droit d'utiliser cette création à des fins commerciales.

SA [Partage des Conditions Initiales à l'Identique] Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.

### B3 : Licence option 3

Copyright © 2011 Franck CHAMBON

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Lesser General Public License (see the file LICENSE in the top-level source directory).

### B4 : Auteur

Franck CHAMBON : <[Franck.Chambon@ac-grenoble.fr](mailto:Franck.Chambon@ac-grenoble.fr)>

Remerciements à IsaT, fidèle lectrice.