Faculté des Sciences
كلية العلوم
جامعة ابن طفيل Université Ibn Tofail

Département d'Informatique
Filière MIP - Semestre 2
2024-2025

Module : Informatique II (Algorithmique II / Python)

Chapitre 3

La récursivité

Pr. Khadija Louzaoui

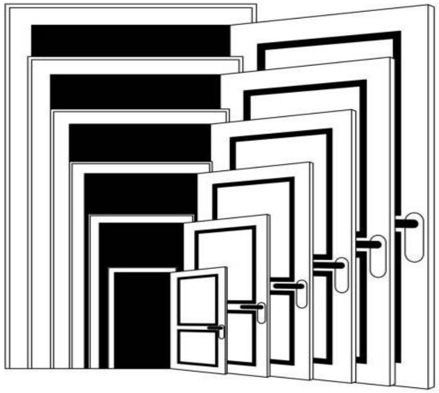
Introduction

La récursivité est un concept général qui peut être illustré dans (quasiment) tous les langages de programmation, et qui peut être utile dans de nombreuses situations.

La définition la plus simple d'une fonction récursive est la suivante :

C’est une fonction qui s'appelle elle-même.

Si dans le corps (le contenu) de la fonction, vous l'utilisez elle-même, alors elle est récursive.



Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

2- Donner la trace d'exécution des appels récursifs pour n=4

Version itérative:

```
fonction fact ( n: entier): entier
Variables i,f : entier
    Début
        f ← 1
        Pour i de 2 à n pas 1 faire
            f ← f* i
        FinPour
        Retourner (f)
    Fin
```

```python
def fac_iterative(n):
    f=1
    for i in range(2,n+1):
        f=f*i
    return f
```

```
>>> fac_iterative(10)
3628800
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

fact(0)= fact(1)=1

Version itérative:	**Version récursive:**
fonction fact (n: entier): entier **Variables i,f : entier** **Début** **f ← 1** **Pour i de 2 à n pas 1 faire** **f ← f* i** **FinPour** **Retourner (f)** **Fin**	

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

fact(0)= fact(1)=1

fact(4)=4*3*2*1

Version itérative:	Version récursive:
fonction fact (n: entier): entier **Variables i,f : entier** **Début** **f ← 1** **Pour i de 2 à n pas 1 faire** **f ← f* i** **FinPour** **Retourner (f)** **Fin**	

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

fact(0)= fact(1)=1

fact(4)=4*3*2*1

fact(5)=5*4*3*2*1=5*fact(4)

Version itérative:	Version récursive:
fonction fact (n: entier): entier **Variables i,f : entier** **Début** **f ← 1** **Pour i de 2 à n pas 1 faire** **f ← f* i** **FinPour** **Retourner (f)** **Fin**	

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:	Version récursive:
fonction fact (n: entier): entier Variables i,f : entier Début f ← 1 Pour i de 2 à n pas 1 faire f ← f* i FinPour Retourner (f) Fin	

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:

```
fonction  fact ( n: entier): entier
Variables  i,f : entier
     Début
         f ← 1
          Pour i de 2 à n pas 1 faire
              f ← f* i
          FinPour
          Retourner (f)
     Fin
```

Version récursive:

```
fonction  fact ( n: entier): entier
     Début



     Fin
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:

```
fonction  fact ( n: entier): entier
Variables  i,f : entier
     Début
          f ← 1
           Pour i de 2 à n pas 1 faire
                f ← f* i
           FinPour
           Retourner (f)
     Fin
```

Version récursive:

```
fonction  fact ( n: entier): entier
     Début
          si (n<=1) alors
            Retourner 1
          Sinon


          Finsi
     Fin
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d’un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d’un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Condition d’arrêt

Version itérative:

```
fonction  fact ( n: entier): entier
Variables  i,f : entier
      Début
          f ← 1
           Pour i de 2 à n pas 1 faire
               f ← f* i
           FinPour
           Retourner (f)
      Fin
```

Version récursive:

```
fonction  fact ( n: entier): entier
      Début
          si (n<=1) alors
            Retourner 1
          Sinon


          Finsi
      Fin
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

1- Ecrire une fonction récursive ***fact(n)*** permettant de calculer le factoriel d'un nombre entier naturel n.

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Version itérative:

```
fonction  fact ( n: entier): entier
Variables  i,f : entier
    Début
        f ← 1
        Pour i de 2 à n pas 1 faire
            f ← f* i
        FinPour
        Retourner (f)
    Fin
```

Version récursive:

```
fonction  fact ( n: entier): entier
    Début
        si (n<=1) alors
            Retourner 1
        Sinon
            Retourner n*fact(n-1)

        Finsi
    Fin
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```
fonction  fact ( n: entier): entier
      Début
            si (n<=1) alors
              Retourner 1
            Sinon
              Retourner n*fact(n-1)
            Finsi
      Fin
```

```
def fac_recursive(n):
   if (n<=1):
     return 1
   else:
     return n*fac_recursive(n-1)
```

```
def fac_recursive(n:int)->int :
   if (n<=1):
     return 1
   else:
     return n*fac_recursive(n-1)
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

Fact(4)

```
fonction  fact ( n: entier): entier
        Début
              si (n<=1) alors
                Retourner 1
             Sinon
               Retourner n*fact(n-1)
            Finsi
         Fin
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d’un nombre

2- Donner la trace d’exécution des appels récursifs pour n=4

```
fonction  fact ( n: entier): entier
        Début
             si (n<=1) alors
               Retourner 1
             Sinon
               Retourner n*fact(n-1)
             Finsi
        Fin
```

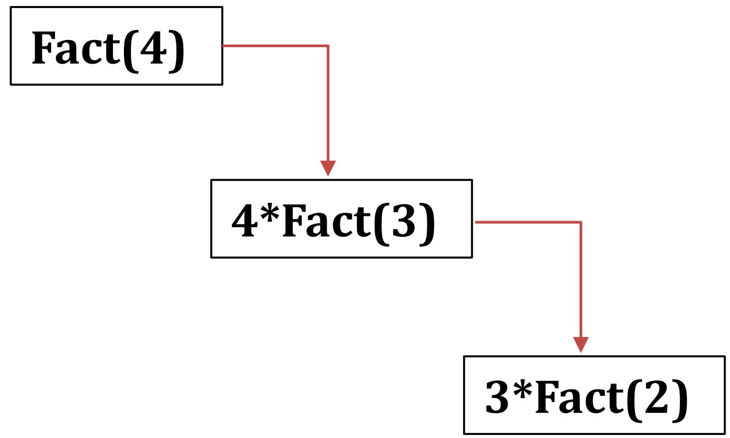
Fact(4)

4*Fact(3)

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d’un nombre

2- Donner la trace d’exécution des appels récursifs pour n=4



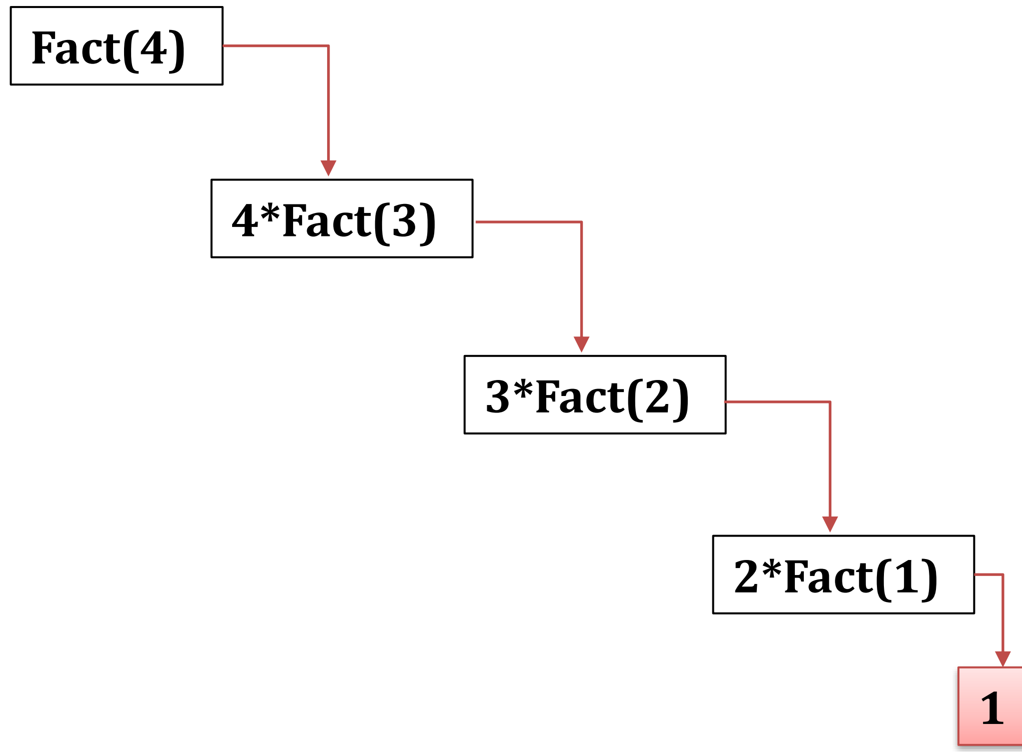

```
fonction  fact ( n: entier): entier
        Début
            si (n<=1) alors
              Retourner 1
            Sinon
              Retourner n*fact(n-1)
           Finsi
        Fin
```

Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```
fonction fact ( n: entier): entier
        Début
               si (n<=1) alors
                 Retourner 1
               Sinon
                 Retourner n*fact(n-1)
               Finsi
         Fin
```

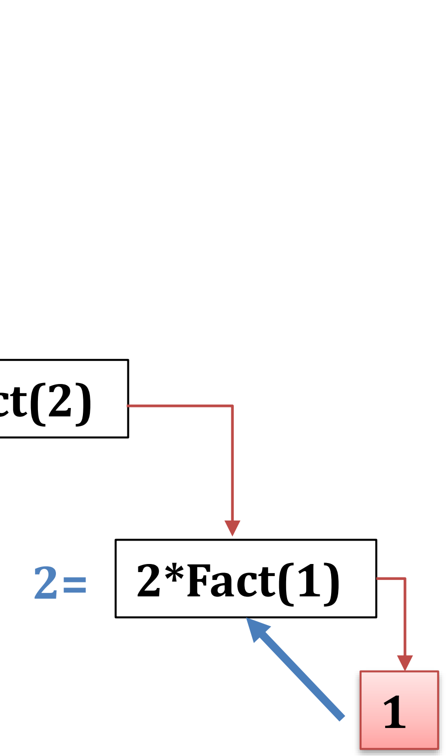


Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d'un nombre

2- Donner la trace d'exécution des appels récursifs pour n=4

```
fonction fact ( n: entier): entier
     Début
          si (n<=1) alors
             Retourner 1
          Sinon
             Retourner n*fact(n-1)
          Finsi
     Fin
```

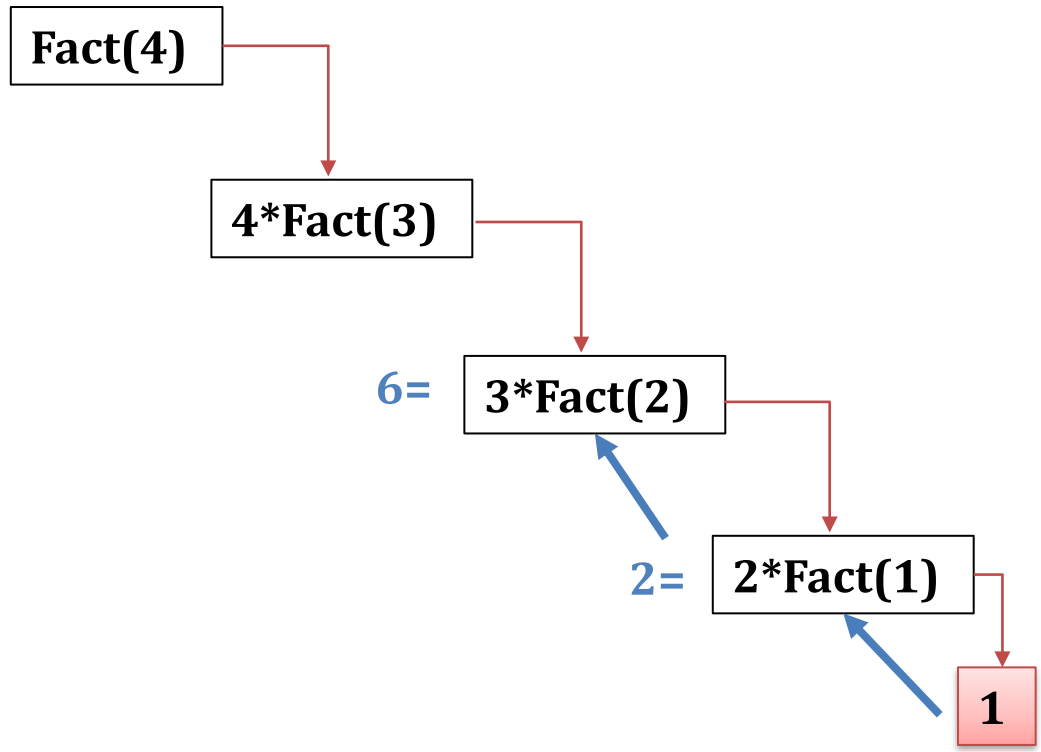


Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d’un nombre

2- Donner la trace d’exécution des appels récursifs pour n=4

```
fonction  fact ( n: entier): entier
        Début
              si (n<=1) alors
                Retourner 1
              Sinon
                Retourner n*fact(n-1)
              Finsi
          Fin
```

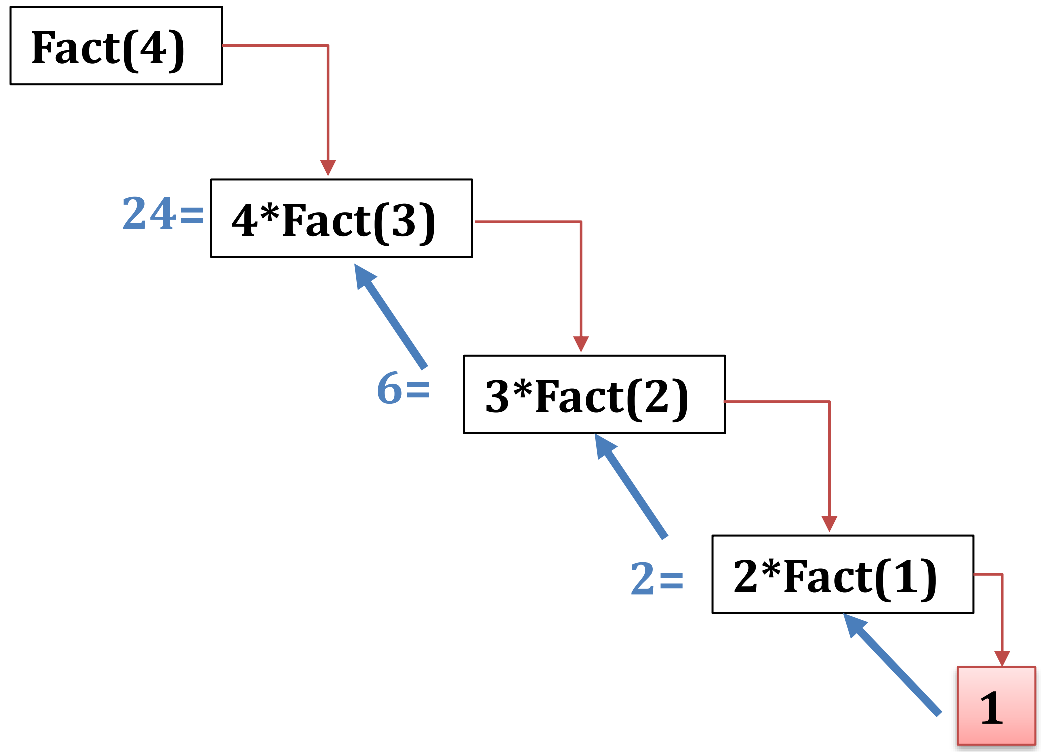


Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d’un nombre

2- Donner la trace d’exécution des appels récursifs pour n=4

```
fonction fact ( n: entier): entier
    Début
        si (n<=1) alors
            Retourner 1
        Sinon
            Retourner n*fact(n-1)
        Finsi
    Fin
```

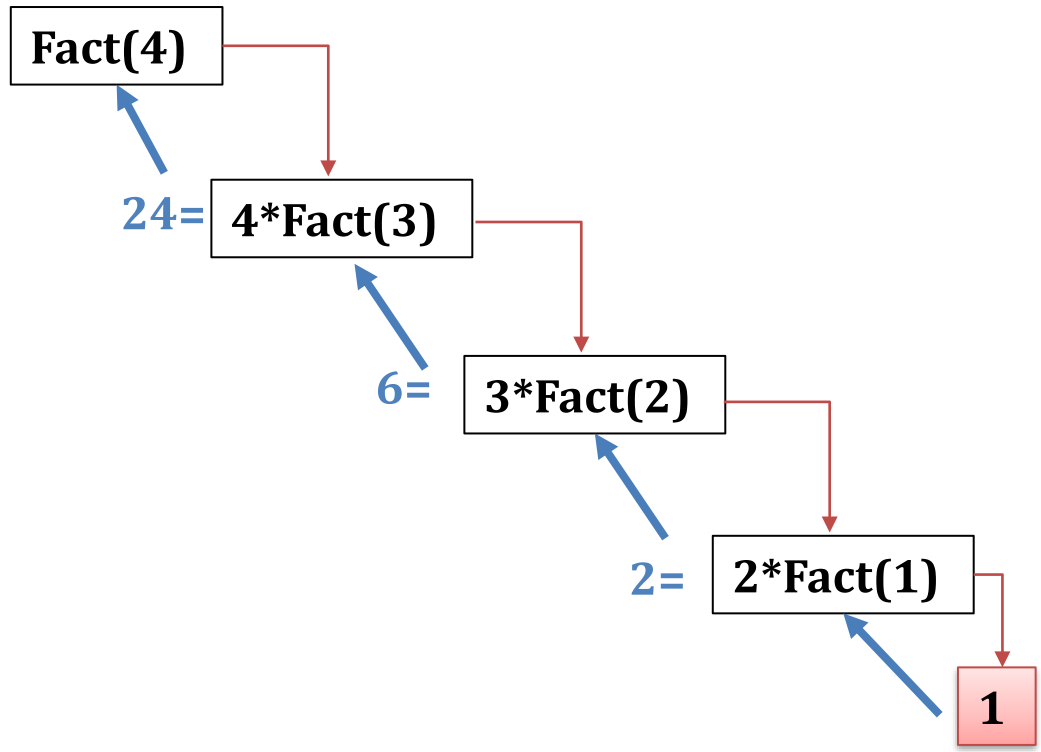


Algorithmes Récursifs : Exemple

Exemple 1 : Factoriel d’un nombre

2- Donner la trace d’exécution des appels récursifs pour n=4

```
fonction fact ( n: entier): entier
        Début
            si (n<=1) alors
              Retourner 1
            Sinon
              Retourner n*fact(n-1)
            Finsi
        Fin
```



Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

1- Ecrire une fonction récursive ***fibo(n)*** permettant de calculer le nième nombre de la suite Fibonacci F_{n} définie comme suit:

$$\begin{cases} F_{\mathrm{n}} = 1 \text{ si n} = 0 \text{ ou n} = 1 \\ F_{\mathrm{n}} = F_{\mathrm{n}-2} + F_{\mathrm{n}-1} \;\; si\ n > 1 \end{cases}$$

2- Donner la trace d'exécution des appels récursifs pour n=5

Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

1- Ecrire une fonction récursive ***fibo(n)*** permettant de calculer le nième nombre de la suite Fibonacci F_n définie comme suit:

$$\begin{cases} F_n = 1 \text{ si } n = 0 \text{ ou } n = 1 \\ F_n = F_{n-2} + F_{n-1} \quad si\ n > 1 \end{cases}$$

Expression récursive

Cas de base

fibo(0)= fino(1)=1
fibo(n)= fino(n-2)+fibo(n-1)

```
fonction  fibo ( n: entier): entier
Début
   si ((n=1) ou (n=0)) alors
         retourne 1
  sinon
        retourne fibo(n-2) + fibo(n-1)
  Finsi
Fin
```

```python
def Fibo(n):
    if n<=1:
       return 1
   else:
       return Fibo(n-2) + Fibo(n-1)
```

Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

1- Ecrire une fonction récursive ***fibo(n)*** permettant de calculer le nième nombre de la suite Fibonacci F_n définie comme suit:

$$\begin{cases} F_n = 1 \text{ si } n = 0 \text{ ou } n = 1 \\ F_n = F_{n-2} + F_{n-1} \quad si\ n > 1 \end{cases}$$

Expression récursive

Cas de base

fibo(0)= fino(1)=1
fibo(n)= fino(n-2)+fibo(n-1)

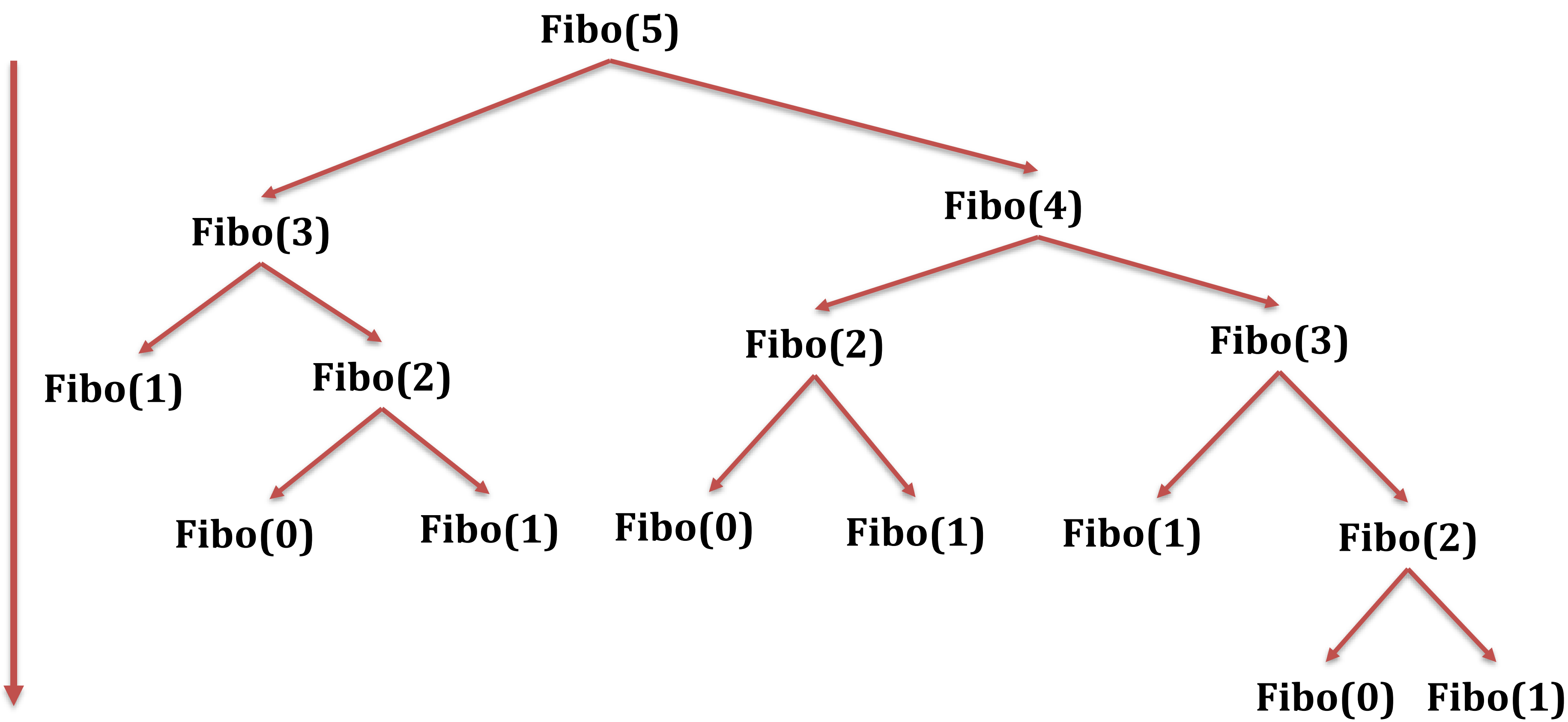
```
fonction  fibo ( n: entier): entier
Début
   si ((n=1) ou (n=0)) alors
          retourne 1
   sinon
         retourne fibo(n-2) + fibo(n-1)
   Finsi
Fin
```

```
def Fibo(n:int)-> int :
    if n<=1:
       return 1
   else:
       return Fibo(n-2) + Fibo(n-1)
```

Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

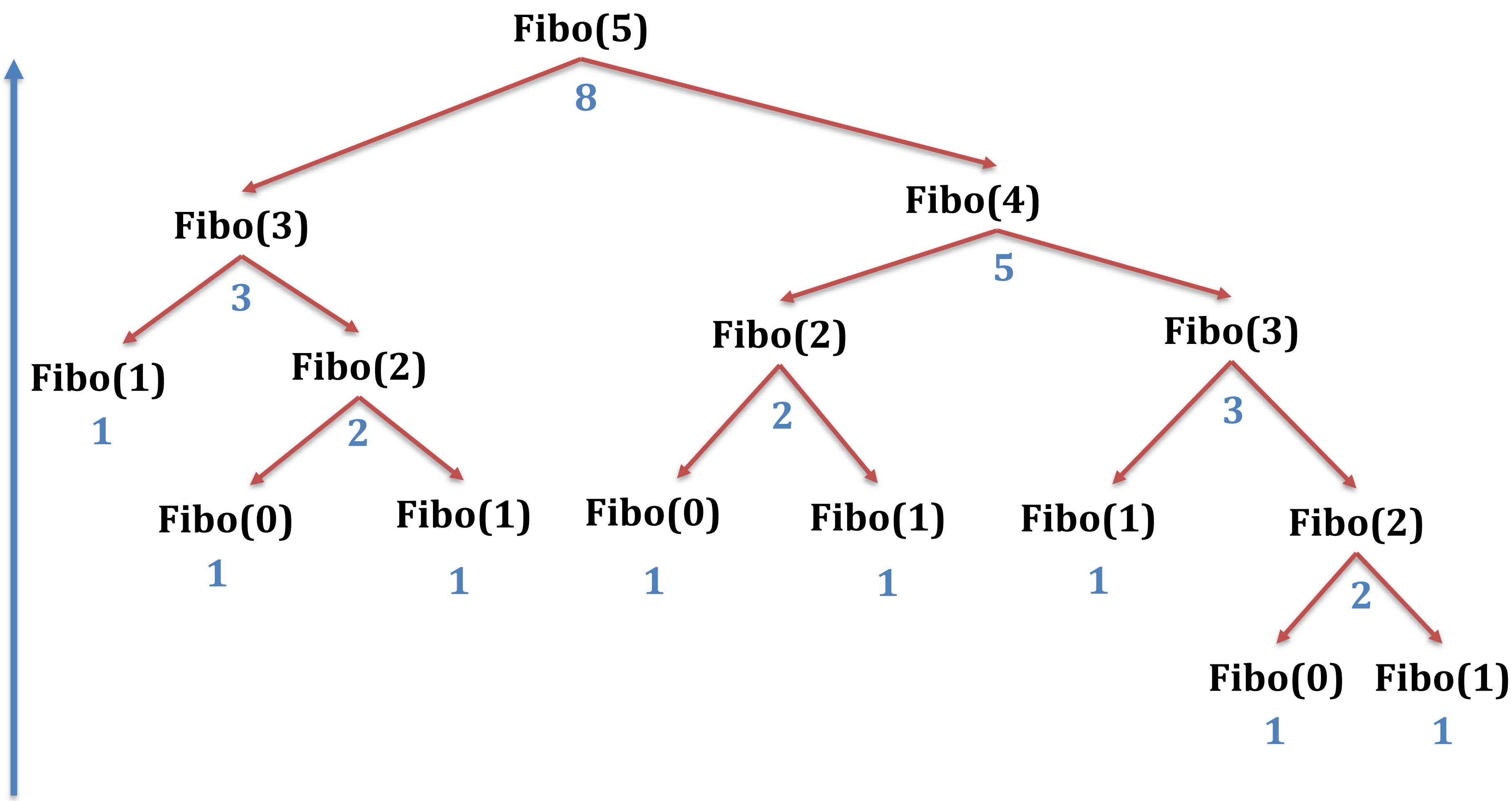
2- Donner la trace d'exécution des appels récursifs pour n=5



Algorithmes Récursifs : Exemple

Exemple 2 : Suite de Fibonacci

2- Donner la trace d'exécution des appels récursifs pour n=5



Définition

- La **récursivité** est une méthode de description d’algorithmes qui permet à une fonction (ou procédure) de s’appeler elle-même directement ou indirectement.

- La formulation d'une solution récursive décrit plusieurs éléments.
  - Les **cas** de **base** : ces cas sont les **conditions d'arrêt** de la **chaîne** des **appels récursifs**.
  - Les **appels récursifs** eux-mêmes.

- La façon de formuler ces appels a un impact sur la **convergence** de la solution récursive.
  - Il faut que tout appel initial nous amène éventuellement à un des **cas de base.**
  - Pour ce faire, il faut que chaque appel nous rapproche des cas de base sans pour autant les dépasser.

Définition

- Généralement, la **condition d'arrêt** se présente sous la forme d'une instruction « **Si… Alors…Sinon** » qui permet de stopper la récurrence si la condition d'arrêt est satisfaite.
- Dans le cas contraire, la fonction ou la procédure continue à exécuter les appels récursifs.
- le paramètre de l'appel récursif doit converger toujours vers la condition d'arrêt.
- Un processus récursif remplace en quelque sorte une boucle, ainsi tout processus récursif peut être également formulé en tant qu'un processus itératif.

Critères de récursivité

- Expression récursive du problème :
  C'est l' « équation » de la récursivité.

- Condition d'arrêt :
  Quand est-ce qu'on arrête les appels récursifs ?

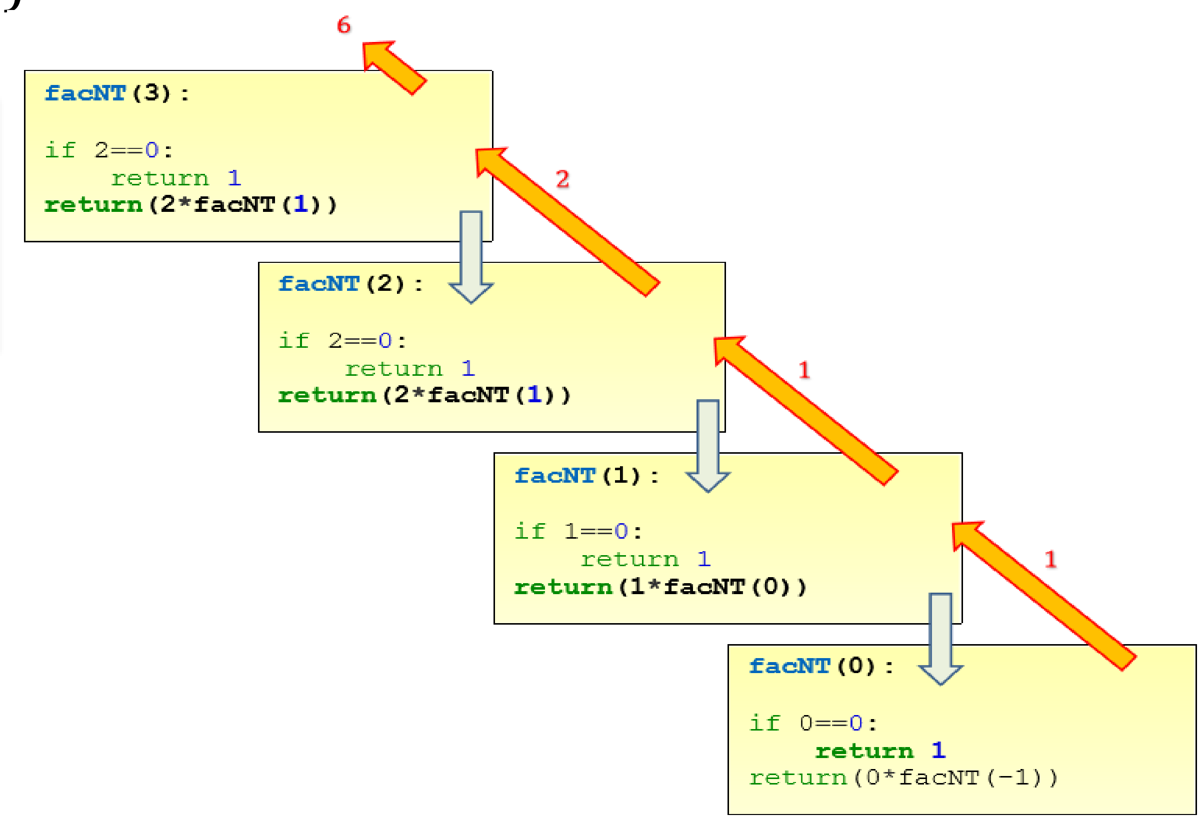
- Convergence (vers la condition d'arrêt):
  Une petite « preuve» qui nous assure qu'un jour on va atteindre la condition d'arrêt.

Types de récursivité

Récursivité non terminale

Une fonction récursive est dite non terminale **si le résultat de l'appel récursif est utilisé pour réaliser un traitement** (en plus du retour d'une valeur).

```python
def facNT(n):
    if n==0:
        return 1
    return(n*facNT(n-1))
```



Types de récursivité

Récursivité terminale

Une fonction récursive est dite terminale **si aucun traitement n'est effectué à la remontée** d'un appel récursif (sauf le retour d'une valeur). Concrètement il n'y a pas de calcul entre l'appel récursif et l'instruction return.

```
def facT(n,acc=1):
    if n==0:
       return acc
    return facT(n-1,n*acc)
```

Cette fois-ci, les calculs se font à la descente, comme c'est illustré ci-dessous, pour facT(3):

facT(3,1) → facT(2,3) → facT(1,6) → facT(0,6)

Types de récursivité

Récursivité simple

La **récursivité simple** où l'algorithme fait un seul appel récursif dans son corps.

Exemple : la fonction factorielle

```
fonction  fact ( n: entier): entier
        Début
              si (n<=1) alors
                Retourner 1
              Sinon
                Retourner n*fact(n-1)
              Finsi
         Fin
```

Types de récursivité

Récursivité multiple

La **récursivité multiple** où l'algorithme fait plusieurs appels récursifs dans son corps.

Exemple : la suite Fibonacci

```
fonction  fibo ( n: entier): entier
Début
   si ((n=1) ou (n=0)) alors
          retourne 1
  sinon
        retourne fibo(n-2) + fibo(n-1)
  Finsi
Fin
```

Types de récursivité

Récursivité imbriquée

La **récursivité imbriquée** consiste à faire un appel récursif à l'intérieur d'un autre appel récursif.

Exemple : la suite d'Ackerman

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

```
fonction ackerman(n,m: entier ): entier
début
    si (m = 0) alors
            retourne n+1
    sinon
            si ((m>0) et (n=0)) alors
            retourne ackerman(m-1,1)
            sinon
                retourne ackerman(m-1,ackerman(m,n-1))
            finsi
    finsi
fin
```

Types de récursivité

Récursivité imbriquée

La **récursivité imbriquée** consiste à faire un appel récursif à l'intérieur d'un autre appel récursif.

$$A(m,n) = \begin{cases} A(0,n) = n+1 \text{ ; si } m = 0 \\ A(m,0) = A(m-1,1) \text{ ; si } n = 0 \\ A(m,n) = A\big(m-1, A(m,n-1)\big) \text{ ; si } m \neq 0 \text{ et } n \neq 0 \end{cases}$$

Exemple : la suite d'Ackerman

```
def Ackermann(m,n:int)->int :
    if m==0:
        return n+1
    elif n==0:
        return Ackermann(m-1,1)
    else:
        return Ackermann(m-1, Ackermann(m,n-1))
```

Question : Que vaut Ackermann(2,2) ?

Réponse : 7

Types de récursivité

Récursivité croisée ou mutuelle

La **récursivité croisée** consiste à écrire des fonctions qui s'appellent l'une l'autre (c.-à-d. un module P appelle un autre module Q qui fait à son tour un autre appel au module P).

Exemple : la définition de la parité d'un entier peut être écrite de la manière suivante :

$$Pair(n) = \begin{cases} vrai & si\ n = 0 \\ Impair\ (n-1) & sinon \end{cases}$$

et

$$Impair(n) = \begin{cases} faux & si\ n = 0 \\ Pair\ (n-1) & sinon \end{cases}$$

Types de récursivité

Récursivité croisée ou mutuelle

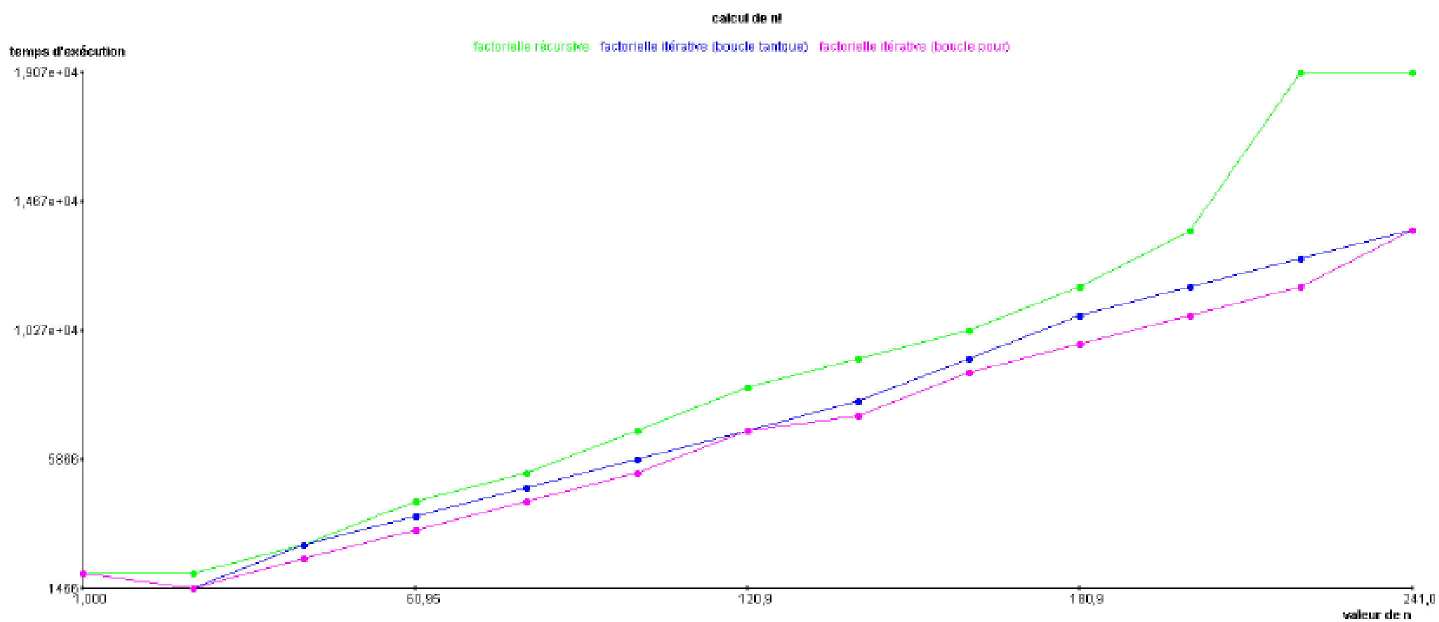
Exemple :

```
fonction estPair(entier n):booléen
début
     si (m = 0) alors
       retourne VRAI
     sinon
       retourne estImpair(n-1)
     finsi
fin
```

```
fonction estImpair(entier n): booléen
début
       si (m = 0) alors
         retourne FAUX
       sinon
         retourne estPair(n-1)
       finsi
fin
```

Récursif versus itératif

- L'exécution d'une version récursive d'un algorithme est généralement un peu moins rapide que celle de la version itérative, même si le nombre d'instructions est le même (à cause de la gestion des appels de fonction).



- Il est souvent possible d'écrire un même algorithme en itératif et en récursif.

Récursif versus itératif

PROPRIÉTÉ	RÉCURSION	ITÉRATION
Définition	La fonction **s'appelle elle-même.**	Un ensemble d'instructions **exécutées** de manière **répétitive**.
Application	Pour les **fonctions**.	Pour les **boucles**.
Terminaison	Par le **cas de base**, où il n'y aura pas d'appel de fonction.	Lorsque la **condition** de sortie de l'itérateur **cesse d'être remplie**.
Utilisation	Utilisé lorsque **la taille** du code doit être **petite** et que la **complexité du temps** ne pose pas de problème.	Utilisé lorsque la **complexité temporelle** doit être compensée par une **taille** de code plus **importante**.
Taille du code	**Taille** du code plus **petite**	**Taille** du code **plus grande**.
Complexité temporelle	**Très grande** (généralement exponentielle) complexité temporelle.	Complexité temporelle relativement plus **faible** (généralement polynomiale et logarithmique).

Exercices

1- Ecrire une fonction **récursive** et une autre **itérative** pour la calcul de la somme:

$$S_{(n,m)} = 1 + 2^m + 3^m + \cdots + n^m$$

Exercices

$$S_{(n,m)} = 1 + 2^m + 3^m + \cdots + n^m$$

$$S_n = \sum_{i=0}^{n} i^m$$

$$S_{(n,m)} = f\left(S_{(n-1,m)}\right) \ ???$$

$$S_{(n,m)} = S_{(n-1,m)} + n^m$$

```
fonction  somiter ( n,m: entier): entier
Variable i,som : entier
Début
    som ← 0
    Pour i de 1 à n faire
       som ← som+i^m
    FinPour
    Retourner (som)
Fin
```

```
fonction  somrec ( n,m: entier): entier

Début
    Si (n<=0) alors
          Retourne 0
    Sinon
          Retourner (n^m+somrec(n-1,m))
    FinSi
Fin
```

Exercices

2- Ecrire une fonction **récursive** et une autre **itérative** qui calcul n$^{\text{ième}}$ terme de la suite(U_n) définie comme suit :

$$\begin{cases} U_0 = 2 \\ U_1 = 3 \\ U_n = \frac{2}{3}U_{n-1} - \frac{1}{4}U_{n-2} \end{cases}$$

Exercices

$$U_n = \begin{cases} U_0 = 2 \\ U_1 = 3 \\ U_n = \frac{2}{3} U_{n-1} - \frac{1}{4} U_{n-2} \end{cases}$$

Cas de base

suiterec(0)= 2 et suiterec(1)=3

Version itérative:	Version récursive:
fonction suiteiter (n: entier): réel Variables u,up,upp: réel Début upp ← 2 up ← 3 Pour i de 2 à n pas 1 faire u ← (2/3)*up-(1/4)*upp upp ← up up ← u finpour retourner u Fin	fonction suiterec (n: entier): réel Début Si (n==0) alors retourner 2 Sinon Si (n==1) alors retourner 3 Sinon retourner ((2/3)*suiterec(n-1)-(1/4)*suiterec(n-2)) Finsi Finsi Fin