

### Définition

### Définition

En informatique, on dit qu'une fonction est **récursive**,

### Définition

En informatique, on dit qu'une fonction est **récursive**, lorsque cette fonction fait appel à elle-même.

### Définition

En informatique, on dit qu'une fonction est **récursive**, lorsque cette fonction fait appel à elle-même.

### Remarques

### Définition

En informatique, on dit qu'une fonction est **récursive**, lorsque cette fonction fait appel à elle-même.

### Remarques

- Une fonction récursive permet donc, *comme une boucle*, de répéter des instructions. Une même fonction peut donc souvent se programmer de façon **itérative** (avec des boucles) ou de façon **récursive** (en s'appelant elle-même).

### Définition

En informatique, on dit qu'une fonction est **récursive**, lorsque cette fonction fait appel à elle-même.

### Remarques

- Une fonction récursive permet donc, *comme une boucle*, de répéter des instructions. Une même fonction peut donc souvent se programmer de façon **itérative** (avec des boucles) ou de façon **récursive** (en s'appelant elle-même).
- Une fonction récursive doit toujours **contenir une condition d'arrêt**, dans le cas contraire elle s'appelle elle-même à l'infini et le programme ne se termine jamais.

### Définition

En informatique, on dit qu'une fonction est **récursive**, lorsque cette fonction fait appel à elle-même.

### Remarques

- Une fonction récursive permet donc, *comme une boucle*, de répéter des instructions. Une même fonction peut donc souvent se programmer de façon **itérative** (avec des boucles) ou de façon **récursive** (en s'appelant elle-même).
- Une fonction récursive doit toujours **contenir une condition d'arrêt**, dans le cas contraire elle s'appelle elle-même à l'infini et le programme ne se termine jamais.
- Les valeurs passées en paramètres lors des appels successifs doivent être différents, sinon la fonction s'exécute à l'identique à chaque appel et donc boucle à l'infini.

### Exemple : compte à rebours

- Que fait la fonction suivante :

```
1 def rebours(n):  
2     for i in range(n,0,-1):  
3         print(i)  
4     print("Partez !")
```



### Exemple : compte à rebours

- Que fait la fonction suivante :

```
1 def rebours(n):  
2     for i in range(n,0,-1):  
3         print(i)  
4     print("Partez !")
```

Elle prend en argument un entier  $n$ , puis affiche un compte à rebours de  $n$  à 1 et ensuite "Partez !".

### Exemple : compte à rebours

- Que fait la fonction suivante :

```
1 def rebours(n):  
2     for i in range(n,0,-1):  
3         print(i)  
4     print("Partez !")
```

Elle prend en argument un entier  $n$ , puis affiche un compte à rebours de  $n$  à 1 et ensuite "Partez !".

- Proposer une version récursive de cette fonction

### Exemple : compte à rebours

- Que fait la fonction suivante :

```
1 def rebours(n):  
2     for i in range(n,0,-1):  
3         print(i)  
4     print("Partez !")
```

Elle prend en argument un entier  $n$ , puis affiche un compte à rebours de  $n$  à 1 et ensuite "Partez !".

- Proposer une version récursive de cette fonction

```
1 def rebours_recuratif(n):  
2     if n==0:  
3         print("Partez !")  
4     else:  
5         print(n)  
6         rebours_recuratif(n-1)
```

### Exemple : les puissances positives

En mathématiques, pour un nombre quelconque  $a$  et un entier positif  $n$ , on définit  $a$  puissance  $n$  par :

$a^n = a \times a \times \cdots \times a$ , et on convient que  $a^0 = 1$

### Exemple : les puissances positives

En mathématiques, pour un nombre quelconque  $a$  et un entier positif  $n$ , on définit  $a$  puissance  $n$  par :

$a^n = a \times a \times \cdots \times a$ , et on convient que  $a^0 = 1$

- Définir une fonction Python `puissance` qui prend en argument `a` et `n` et renvoie  $a^n$  en effectuant ce calcul de façon itératif

### Exemple : les puissances positives

En mathématiques, pour un nombre quelconque  $a$  et un entier positif  $n$ , on définit  $a$  puissance  $n$  par :

$a^n = a \times a \times \cdots \times a$ , et on convient que  $a^0 = 1$

- Définir une fonction Python `puissance` qui prend en argument `a` et `n` et renvoie  $a^n$  en effectuant ce calcul de façon itératif
- Recopier et compléter :  $a^n = \cdots \times a^{\cdots}$

### Exemple : les puissances positives

En mathématiques, pour un nombre quelconque  $a$  et un entier positif  $n$ , on définit  $a$  puissance  $n$  par :

$a^n = a \times a \times \cdots \times a$ , et on convient que  $a^0 = 1$

- Définir une fonction Python `puissance` qui prend en argument `a` et `n` et renvoie  $a^n$  en effectuant ce calcul de façon itératif
- Recopier et compléter :  $a^n = \cdots \times a^{\cdots}$
- En déduire une version récursive de la fonction calculant les puissances

### Exemple : les puissances positives

- Puissance : version itérative

```
1  def puissance_iteratif(a,n):  
2      p=1  
3      for k in range(n):  
4          p=p*a  
5      return p
```



### Exemple : les puissances positives

- Puissance : version itérative

```
1  def puissance_iteratif(a,n):  
2      p=1  
3      for k in range(n):  
4          p=p*a  
5      return p
```

- $a^n = a \times a^{n-1}$

### Exemple : les puissances positives

- Puissance : version itérative

```
1 def puissance_iteratif(a,n):  
2     p=1  
3     for k in range(n):  
4         p=p*a  
5     return p
```

- $a^n = a \times a^{n-1}$

- Puissance : version récursive

```
1 def puissance_recuratif(a,n):  
2     if n==0:  
3         return 1  
4     return a * puissance_recuratif(a,n-1)
```

### Exemple : maximum des éléments d'une liste non vide

- Ecrire une fonction itérative qui renvoie le maximum des éléments d'une liste

### Exemple : maximum des éléments d'une liste non vide

- Ecrire une fonction itérative qui renvoie le maximum des éléments d'une liste

```
1 def maximum(entiers):  
2     max = entiers[0]  
3     for i in range(len(entiers)):  
4         if entiers[i] > max:  
5             max = entiers[i]  
6     return max
```

### Exemple : maximum des éléments d'une liste non vide

- Ecrire une fonction itérative qui renvoie le maximum des éléments d'une liste

```
1  def maximum(entiers):  
2      max = entiers[0]  
3      for i in range(len(entiers)):  
4          if entiers[i] > max:  
5              max = entiers[i]  
6      return max
```

- Proposer une version récursive de cette fonction

### Exemple : maximum des éléments d'une liste non vide

- Ecrire une fonction itérative qui renvoie le maximum des éléments d'une liste

```
1 def maximum(entiers):
2     max = entiers[0]
3     for i in range(len(entiers)):
4         if entiers[i]>max:
5             max = entiers[i]
6     return max
```

- Proposer une version récursive de cette fonction

```
1 def maximum(entiers):
2     if len(entiers)==1:
3         return entiers[0]
4     maxi_reste = maximum(entiers[1:])
5     if entiers[0]>maxi_reste:
6         return entiers[0]
7     else:
8         return maxi_reste
```

### Une fonction à analyser

```
1 def mystere(elt,liste):
2     if liste==[]:
3         return 0
4     if elt==liste[0]:
5         return 1+mystere(elt,liste[1:])
6     else:
7         return mystere(elt,liste[1:])
8
```

### Une fonction à analyser

```
1 def mystere(elt,liste):
2     if liste==[]:
3         return 0
4     if elt==liste[0]:
5         return 1+mystere(elt,liste[1:])
6     else:
7         return mystere(elt,liste[1:])
8
```

- Que fait la fonction mystere ci-dessus ?



### Une fonction à analyser

```
1 def mystere(elt,liste):
2     if liste==[]:
3         return 0
4     if elt==liste[0]:
5         return 1+mystere(elt,liste[1:])
6     else:
7         return mystere(elt,liste[1:])
8
```

- Que fait la fonction mystere ci-dessus ?
- Cette fonction est-elle programmée de façon itérative ? récursive ? Justifier.

### Une fonction à analyser

```
1 def mystere(elt,liste):
2     if liste==[]:
3         return 0
4     if elt==liste[0]:
5         return 1+mystere(elt,liste[1:])
6     else:
7         return mystere(elt,liste[1:])
8
```

- Que fait la fonction mystere ci-dessus ?
- Cette fonction est-elle programmée de façon itérative ? récursive ? Justifier.
- Proposer une version de cette fonction qui ne s'appelle pas elle-même.

Exemple : une fonction à analyser

### Exemple : une fonction à analyser

- Cette fonction compte le nombre d'occurrence de `elt` dans `liste`

### Exemple : une fonction à analyser

- Cette fonction compte le nombre d'occurrence de `elt` dans `liste`
- Elle ne contient pas de boucle, elle n'est donc pas programmée de façon itérative. Par contre c'est une fonction récursive car elle fait appel à elle même.

### Exemple : une fonction à analyser

- Cette fonction compte le nombre d'occurrence de `elt` dans `liste`
- Elle ne contient pas de boucle, elle n'est donc pas programmée de façon itérative. Par contre c'est une fonction récursive car elle fait appel à elle-même.
- Version itérative

```
1  def occurrence(elt,liste):  
2      occ=0  
3      for x in liste:  
4          if x==elt:  
5              occ=occ+1  
6      return occ
```

### Remarques importantes

### Remarques importantes

- On peut toujours transformer une fonction itérative en son équivalent récursif.



### Remarques importantes

- On peut toujours transformer une fonction itérative en son équivalent récursif.
- Certains problèmes (que nous verrons en exercice) ont une solution récursive très lisible et rapide à programmer. La formulation récursive est donc parfois « plus adaptée » à un problème.

### Remarques importantes

- On peut toujours transformer une fonction itérative en son équivalent récursif.
- Certains problèmes (que nous verrons en exercice) ont une solution récursive très lisible et rapide à programmer. La formulation récursive est donc parfois « plus adaptée » à un problème.
- La programmation récursive est parfois gourmande en ressource car les appels récursifs successifs doivent parfois être conservés dans une **pile** dont la taille est limitée.