C1 Récursivité

Définition

C1 Récursivité

Définition

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

C1 Récursivité

Définition

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

C1 Récursivité

Définition

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

Remarques

C1 Récursivité

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).

C1 Récursivité

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.

C1 Récursivité

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 : 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(a,n)` qui retourne 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(a,n)` qui retourne 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(a,n)` qui retourne 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_recursif(a,n):
2      if n==0:
3          return 1
4      else:
5          return a*puissance_recursif(a,n-1)
```

Exemple : une fonction à analyser

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

Exemple : une fonction à analyser

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

- Que fait la fonction `mystere` ci-dessus ?

Exemple : une fonction à analyser

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

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

C1 Récursivité

Exemple : une fonction à analyser

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

- 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'occurence de `elt` dans `liste`

Exemple : une fonction à analyser

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

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

Recursivite

Remarques importantes

Recursivite

Remarques importantes

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

Recursivite

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.

Recursivite

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.