

Programmation avec Xcas ou Python

G. Aldon - J. Germoni - J.-M. Mény

IREM de Lyon

Mars 2012

Éditeur Xcas

- Le texte d'un programme s'écrit dans l'éditeur de programmes de Xcas.

Éditeur Xcas

- Le texte d'un programme s'écrit dans l'éditeur de programmes de Xcas.
- Cet éditeur s'ouvre par le raccourci alt+P (ou en passant par les menus).

Éditeur Xcas

- Le texte d'un programme s'écrit dans l'éditeur de programmes de Xcas.
- Cet éditeur s'ouvre par le raccourci alt+P (ou en passant par les menus).
- Lorsque le texte du programme est tapé, on le valide/compile en cliquant sur la touche OK.

Éditeur Python

- Il existe plusieurs éditeurs de texte dédiés à Python.

Éditeur Python

- Il existe plusieurs éditeurs de texte dédiés à Python.
- Un bon choix sous windows : pythonxy avec l'éditeur spyder.
- Sous Linux, spyder.

Éditeur Python

- Il existe plusieurs éditeurs de texte dédiés à Python.
- Un bon choix sous windows : pythonxy avec l'éditeur spyder.
- Sous Linux, spyder.
- Ou pythonner avec SAGE.

Commenter ses programmes

Une bonne habitude : commenter abondamment ses programmes.

Commenter ses programmes

Une bonne habitude : commenter abondamment ses programmes.

- Avec XCAS :
 - ligne commençant par //
 - commentaires longs : ouverture par /* et fermeture par */.

Commenter ses programmes

Une bonne habitude : commenter abondamment ses programmes.

- Avec XCAS :
 - ligne commençant par //
 - commentaires longs : ouverture par /* et fermeture par */.
- Avec Python :
 - commencer la ligne par # :
 - commentaire long commençant par """ et terminé par """.

Structures POUR et SI avec XCAS

Entrer et tester le programme qui suit :



Xcas

```
saisir(a,b);  
d :=1;  
pour k de 2 jusque min(a,b) faire  
si irem(a,k)==0 et irem(b,k)==0 alors d :=k ;fsi;  
fpour;  
afficher(d);
```

Ifékoï ?

Structures POUR et SI avec Python

Entrer et tester le programme qui suit :


Python

```
a=input(" Entrer un entier positif a ")
b=input(" Entrer un entier positif b ")
d=1
for k in range(2,min(a,b)+1) :
    if a%k==0 and b%k==0 :
        d=k
print d
```

Ifékoï ?

Conception comme fonction, it's better – XCAS


Entrer :

```
 Xcas  
pgcd(a,b) := {  
  local d,k;  
  d := 1;  
  pour k de 2 jusque min(a,b) faire  
  si irem(a,k)==0 et irem(b,k)==0 alors d := k ;fsi ;  
  fpour ;  
  retourne d ;} ;;
```

- On l'utilise en entrant par exemple en ligne de commandes : `pgcd(12,8)`.

Conception comme fonction, it's better – XCAS


Entrer :

```
 Xcas  
pgcd(a,b) := {  
  local d,k;  
  d := 1;  
  pour k de 2 jusque min(a,b) faire  
  si irem(a,k)==0 et irem(b,k)==0 alors d := k ;fsi ;  
  fpour ;  
  retourne d ;} ;;
```

- On l'utilise en entrant par exemple en ligne de commandes : `pgcd(12,8)`.
- `local` : variables connues uniquement dans le "champ de calcul" de la fonction.

Conception comme fonction, it's better – XCAS

Entrer :

```
 Xcas  
pgcd(a,b) := {  
  local d,k;  
  d := 1;  
  pour k de 2 jusque min(a,b) faire  
  si irem(a,k)==0 et irem(b,k)==0 alors d := k ;fsi ;  
  fpour ;  
  retourne d ;} ;;
```

- On l'utilise en entrant par exemple en ligne de commandes : `pgcd(12,8)`.
- `local` : variables connues uniquement dans le "champ de calcul" de la fonction.
- Différence entre `retourne` et `afficher`.

Conception comme fonction, it's better – Python

Entrer :

 **Python**

```
def pgcd(a,b) :  
    d=1  
    for k in range(2,min(a,b)+1) :  
        if a%k==0 and b%k==0 :  
            d=k  
    return d
```

- On l'utilise par exemple en exécutant :
 - `print pgcd(132,8)`
 - `D = pgcd(132,8)` (intérêt : réutiliser le résultat !)

Conception comme fonction, it's better – Python

Entrer :

 **Python**

```
def pgcd(a,b) :  
    d=1  
    for k in range(2,min(a,b)+1) :  
        if a%k==0 and b%k==0 :  
            d=k  
    return d
```

- On l'utilise par exemple en exécutant :
 - `print pgcd(132,8)`
 - `D = pgcd(132,8)` (intérêt : réutiliser le résultat !)
- Localité : une variable affectée dans une fonction est *locale*.

Conception comme fonction, it's better – Python

Entrer :

 **Python**

```
def pgcd(a,b) :  
    d=1  
    for k in range(2,min(a,b)+1) :  
        if a%k==0 and b%k==0 :  
            d=k  
    return d
```

- On l'utilise par exemple en exécutant :
 - `print pgcd(132,8)`
 - `D = pgcd(132,8)` (intérêt : réutiliser le résultat !)
- Localité : une variable affectée dans une fonction est *locale*.
- Différence entre `return` et `print`.

Bogue, Structure "TANT QUE" – Xcas

Pour détecter un bug dans un programme, on affiche en général des valeurs intermédiaires des variables.

Exemple. Tester :

Xcas

```
ifekoi2(a,b) := {  
  tantque b != 0 faire  
  si a > b alors a := a - b ;  
  sinon b := b - a ; fsi ;  
  afficher(a,b) ;  
  ftantque ;  
  retourne a ; } ; ;
```

Bogue, Structure "TANT QUE" – Python

Pour détecter un bug dans un programme, on affiche en général des valeurs intermédiaires des variables.

Exemple. Tester :



Python

```
def ifekoi2(a,b) :  
    while b!=0 :  
        if a>b : a -= b  
        else : b -= a  
        print a,b  
    return a
```

Les listes avec Xcas.

Tester :

 **Xcas**

```
ifekoi3(a,b) :={  
local R,r;  
R :=[ ]; // R est une liste vide  
tantque b!=0 faire  
r :=irem(a,b);  
R :=append(R,r); // on ajoute r à la liste R  
a :=b;b :=r;  
ftantque;  
retourne R;} ::
```

Les listes avec Python.

Tester :



Python

```
def ifekoi3(a,b) :  
    R=[ ]  
    while b !=0 :  
        r=a%b  
        R.append(r)  
        a,b=b,r  
    return R
```

Et avec une liste de listes – Xcas

Écrire une fonction `Euclide(a,b)` (a et b entiers naturels non nuls) renvoyant une liste `T` dont les éléments sont des listes contenant les valeurs `[a, b, quotient, reste]` de chaque étape de l'algorithme.

Et avec une liste de listes – Xcas

Écrire une fonction `Euclide(a,b)` (a et b entiers naturels non nuls) renvoyant une liste `T` dont les éléments sont des listes contenant les valeurs `[a, b, quotient, reste]` de chaque étape de l'algorithme.



Xcas

```
euclide(a,b) := {  
  local q,r,T ;  
  T := [[" a", " b", " q", " r"]];  
  tantque b != 0 faire  
    (q,r) := iquorem(a,b);  
    T := append(T,[a,b,q,r]);  
    a := b; b := r;  
  ftantque;  
  retourne T ;} ;;
```


Et avec une liste de listes – Python

Écrire une fonction `Euclide(a,b)` (a et b entiers naturels non nuls) renvoyant une liste T dont les éléments sont des listes contenant les valeurs $[a, b, \text{quotient}, \text{reste}]$ de chaque étape de l'algorithme.

Et avec une liste de listes – Python

Écrire une fonction `Euclide(a,b)` (a et b entiers naturels non nuls) renvoyant une liste T dont les éléments sont des listes contenant les valeurs $[a, b, \text{quotient}, \text{reste}]$ de chaque étape de l'algorithme.

Python

```
def euclide(a,b) :  
    T=[[" a", " b", " q", " r"]]  
    while b !=0 :  
        q,r=a//b,a%b  
        T.append([a,b,q,r])  
        a,b=b,r  
    return T
```

Récursivité – Xcas

Une fonction récursive est une fonction qui fait un appel à elle-même.
Exemple – Tester (a et b entiers positifs non nuls) :

Xcas

```
⌈ polezzi(a,b,n) := {  
⌈ si n<=0 alors retourne 0 ;fsi ;  
⌈ retourne polezzi(a,b,n-1)+floor(n*b/a) ;} ;;  
⌈ ifekoi(a,b) := {retourne polezzi(a,b,a-1)*2+a+b-a*b ; } ;;
```

Récursivité – Xcas

Une fonction récursive est une fonction qui fait un appel à elle-même.
Exemple – Tester (a et b entiers positifs non nuls) :

Xcas

```

} polezzi(a,b,n) :={
} si n<=0 alors retourne 0 ;fsi ;
} retourne polezzi(a,b,n-1)+floor(n*b/a) ;} ;;
} ifekoi(a,b) :={retourne polezzi(a,b,a-1)*2+a+b-a*b ; } ;;

```

Formule de Marcelo Polezzi

- $\text{polezzi}(a,b,a-1)$ calcule $\lfloor \frac{b}{a} \rfloor + \lfloor 2\frac{b}{a} \rfloor + \lfloor 3\frac{b}{a} \rfloor + \dots + \lfloor (a-1)\frac{b}{a} \rfloor$.
- $\text{ifekoi}(a,b)$ renvoie le pgcd des entiers naturels a et b .

Récursivité – Python

Une fonction récursive est une fonction qui fait un appel à elle-même.
Exemple – Tester (a et b entiers naturels) :



Python

```
def polezzi(a,b,n) :  
    if n<=0 : return 0  
    return polezzi(a,b,n-1) + n*b//a  
def ifekoi(a,b) :  
    return polezzi(a,b,a-1)*2+a+b-a*b
```

Récursivité – Python

Une fonction récursive est une fonction qui fait un appel à elle-même.
Exemple – Tester (a et b entiers naturels) :



Python

```
def polezzi(a,b,n) :
    if n<=0 : return 0
    return polezzi(a,b,n-1) + n*b//a
def ifekoi(a,b) :
    return polezzi(a,b,a-1)*2+a+b-a*b
```

Formule de Marcelo Polezzi

- $\text{polezzi}(a,b,a-1)$ calcule $\lfloor \frac{b}{a} \rfloor + \lfloor 2\frac{b}{a} \rfloor + \lfloor 3\frac{b}{a} \rfloor + \dots + \lfloor (a-1)\frac{b}{a} \rfloor$.
- $\text{ifekoi}(a,b)$ renvoie le pgcd des entiers naturels a et b .

Récursivité

Exercice. Donner une version récursive du calcul du pgcd par l'algorithme d'Euclide.

Récursivité

Exercice. Donner une version récursive du calcul du pgcd par l'algorithme d'Euclide.



Xcas

```
pgcd(a,b) := {  
  si b==0 alors retourne a ;fsi ;  
  retourne pgcd(b,irem(a,b)) ; } ;;
```



Python

```
def PGCD(a,b) :  
    if b==0 : return a  
    return PGCD(b,a%b)
```


Récursivité

Exercice Donner une version récursive du calcul du pgcd d'une liste d'entiers positifs non nuls (présentant au moins deux éléments).

Récursivité

Exercice Donner une version récursive du calcul du pgcd d'une liste d'entiers positifs non nuls (présentant au moins deux éléments).

Xcas

```
pgc(L) := {  
  si dim(L)==2 alors  
    si L[1]==0 alors retourne L[0] ;  
    sinon retourne pgc([L[1],irem(L[0],L[1])]) ;fsi ;  
  sinon retourne pgc([L[0],pgc(tail(L))]) ;  
  fsi ;} ;;
```

Exemple d'utilisation : `pgc ([120,3,9]) .`

Récursivité

Exercice Donner une version récursive du calcul du pgcd d'une liste d'entiers positifs non nuls (présentant au moins deux éléments).

Python

```
def pgc(L) :  
    if len(L)==2 :  
        if L[1]==0 : return L[0]  
        else : return pgc([L[1],L[0]%L[1]])  
    else : return pgc([L[0],pgc(L[1 :])])
```

Exemple d'utilisation : `print pgc ([120,3,6]) .`

Récursivité

Et avec les coefficients de Bachet-Bézout :



Xcas

```

bezout(a,b) := {
local d,x,y;
si b==0 alors retourne [a,1,0];fsi;
[d,x,y] :=bezout(b,irem(a,b));
[x,y] :=[y,x-iquo(a,b)*y];
retourne [d,x,y];} ;;
  
```



Python

```

def bezout(a,b) :
    if b==0 : return (a,1,0)
    (d,x,y)=bezout(b,a%b)
    x,y=y,x-a//b*y
    return (d,x,y)
  
```