Reformulation de programmes avec PluriAlgo

Introduction

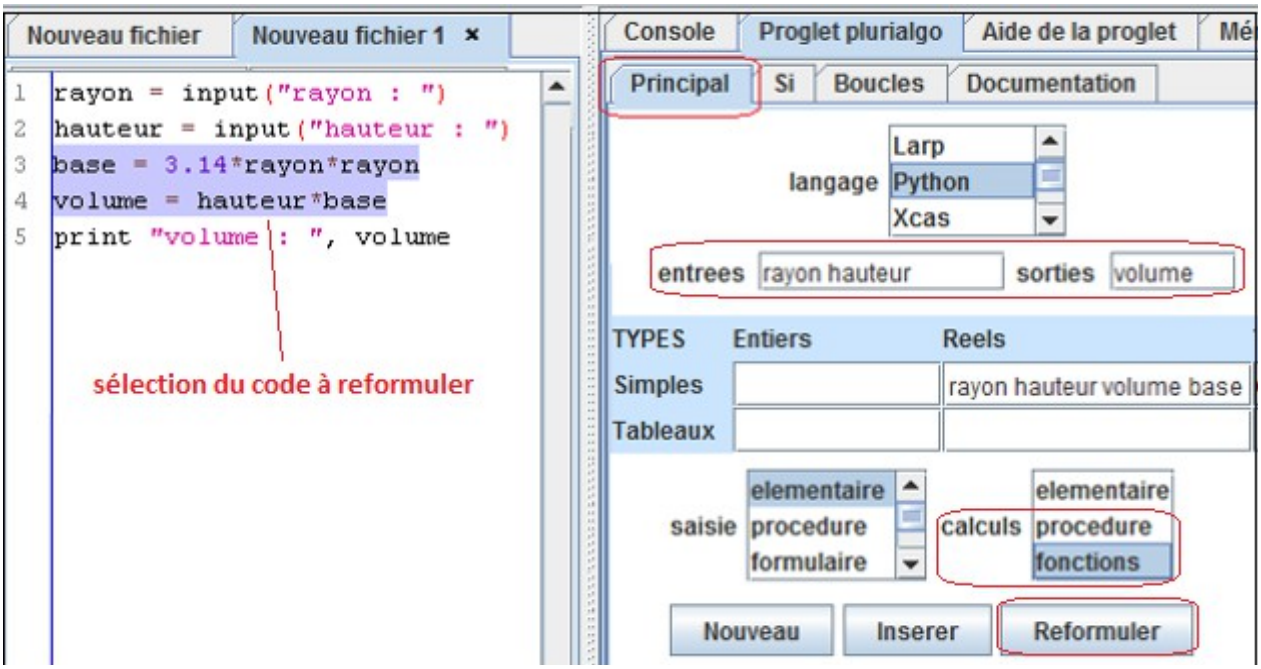
PluriAlgo propose deux mécanismes de reformulation :

- la reformulation avec sélection de code permet de réécrire un programme initial en introduisant des sous-programmes
- le reformulation sans sélection de code permet non seulement d'introduire des sous-programmes, mais aussi de créer de nouveaux types (enregistrements ou classes), des formulaires, de changer de langage...

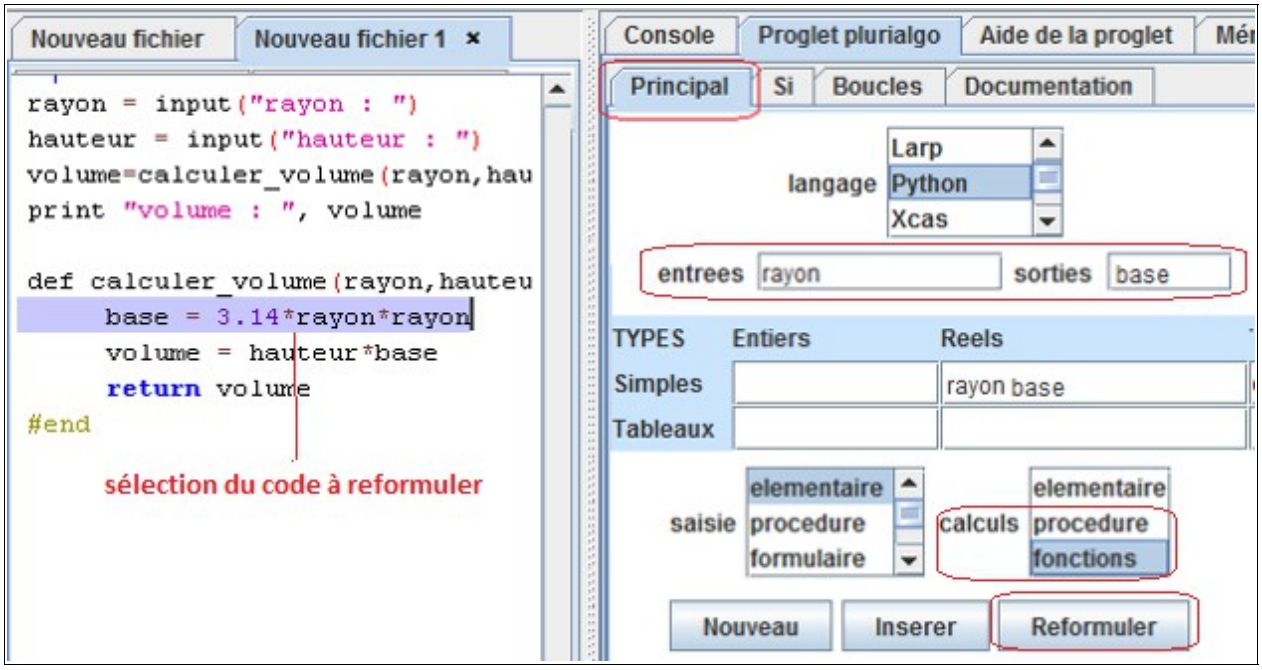
Si vous souhaitez uniquement introduire des sous programmes, il vaut mieux privilégier la reformulation avec sélection de code car elle offre une plus grande souplesse d'utilisation.

Reformulation avec sélection de code

Pour illustrer la souplesse d'utilisation de la reformulation de code avec sélection de code, nous détaillons un exemple (Python) calculant le volume d'un cylindre :



La fonction créée (nommée par défaut calculer_volume) peut à son tour être reformulée :



Cela permet de définir une fonction calculant la base du cylindre :

```
rayon = input("rayon : ")
hauteur = input("hauteur : ")
volume=calculer_volume(rayon,hauteur)
print "volume : ", volume

def calculer_volume(rayon,hauteur) :
     base = calculer_base(rayon)
     volume = hauteur*base
     return volume
#end

def calculer_base(rayon) :
     base = 3.14*rayon*rayon
     return base
#end
```

Reformulation sans sélection de code

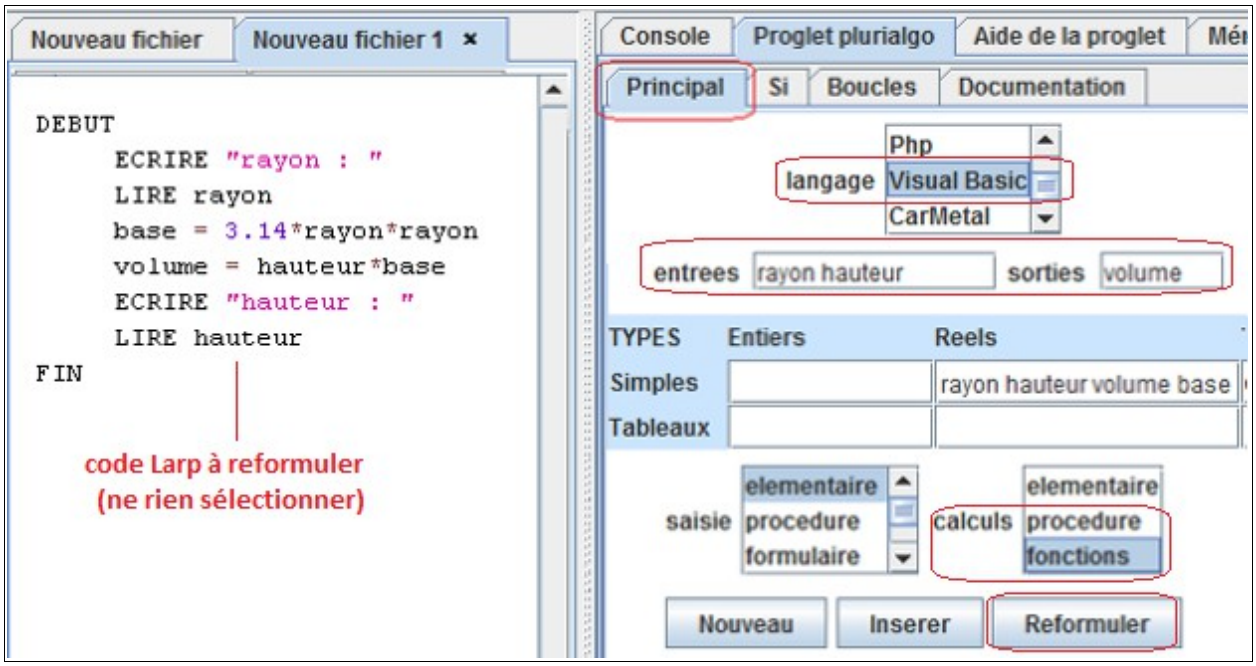
Le mécanisme de reformulation sans sélection de code est beaucoup plus contraignant : il nécessite en effet que PluriAlgo soit capable d'analyser le programme initial, ce qui exclut plusieurs langages (Java, Javascript...).

Et pour les autres langages (Javascool, Python, Larp...), l'analyseur peut ne pas fonctionner : si vous souhaitez uniquement introduire des sous programmes, il vaut donc mieux privilégier la reformulation avec sélection de code, d'autant plus qu'elle permet un découpage plus fin en sous-programmes.

Mais la reformulation sans sélection a d'autres atouts...

Exemple 1 : changement de langage

Le langage du programme final (à fixer dans l'onglet Principal) peut différer de celui du programme initial (« deviné » par PluriAlgo). Dans mes enseignements, j'utilise parfois cette possibilité pour gagner du temps lorsque je passe d'une initiation à l'algorithmique effectuée en Larp à l'étude d'un langage professionnel (Visual Basic, Java...) :



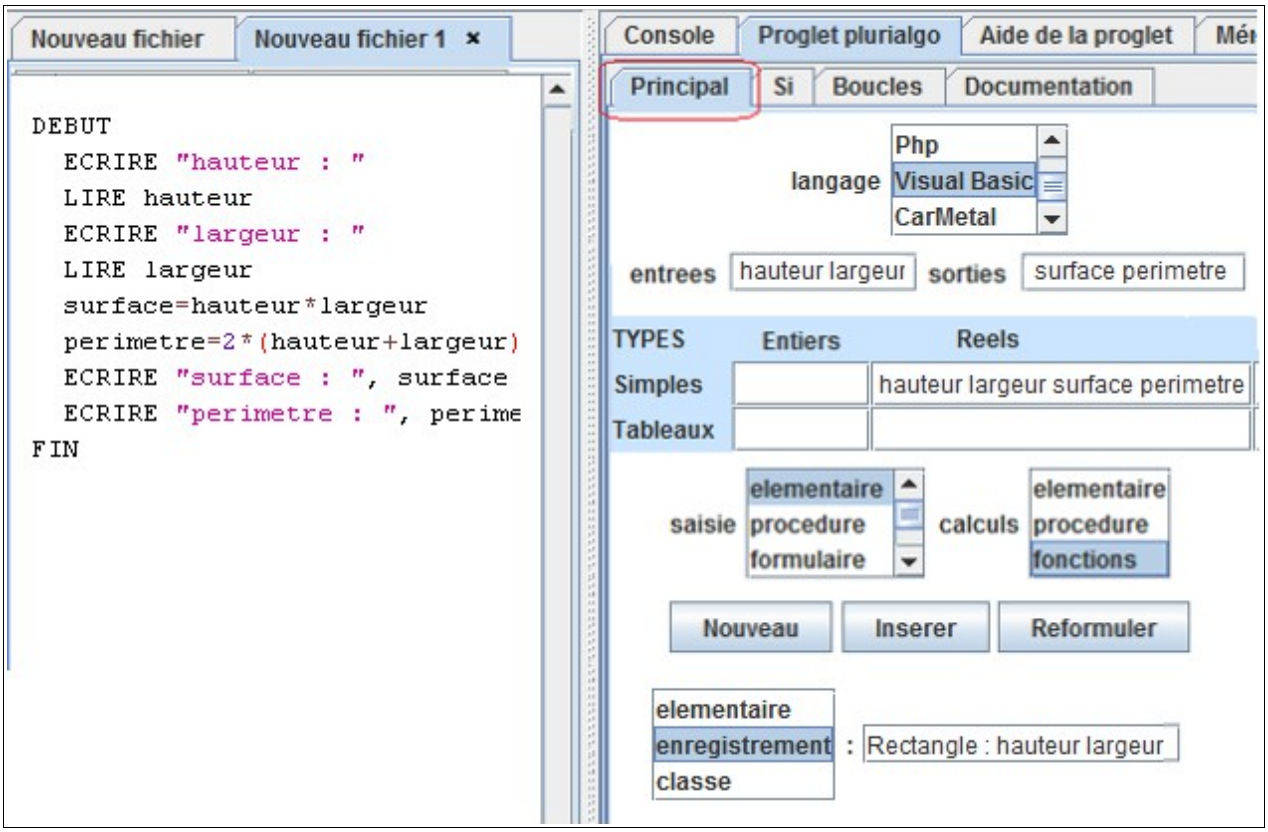
Le résultat obtenu, ici en Visual Basic, est le suivant :

```
function calculer_volume(ByVal rayon as double, ByVal hauteur as double) as double
   Dim base as double
   Dim volume as double
   base = 3.14*rayon*rayon
   volume = hauteur*base
   calculer_volume = volume
end function
```

```
sub main()
   Dim rayon as double
   Dim hauteur as double
   Dim volume as double
   rayon = InputBox( "rayon : " )
   hauteur = InputBox( "hauteur : " )
   volume = calculer_volume(rayon, hauteur)
   MsgBox( "volume : " & volume )
end sub
```

Exemple 2 : création de nouveaux types (enregistrements ou classes)

Ce nouvel exemple (calcul de la surface et du périmètre d'un rectangle) montre comment créer un nouveau type (Rectangle). Il permet également d'expliquer ce qui se passe lorsqu'on reformule un programme ayant plusieurs sorties.



Un type Rectangle regroupant la hauteur et la largeur est créé (ici en Visual Basic) quand on clique sur le bouton **Reformuler** :

```
Type Rectangle
     hauteur as double
     largeur as double
End Type
```

Deux fonctions (une par sortie) sont également créées puisque l'option de calcul « fonctions » a été choisie :

```
function calculer_surface(objet as Rectangle ) as double
     Dim perimetre as double
     Dim surface as double
     surface = objet.hauteur*objet.largeur
     perimetre = 2*(objet.hauteur+objet.largeur)
     calculer_surface = surface
end function

function calculer_perimetre(objet as Rectangle ) as double
     Dim surface as double
     Dim perimetre as double
     surface = objet.hauteur*objet.largeur
     perimetre = 2*(objet.hauteur+objet.largeur)
     calculer_perimetre = perimetre
end function
```

A supprimer

Ces deux fonctions sont paramétrées par une variable de type Rectangle, alors qu'elles auraient été paramétrées par les deux entrées (hauteur et largeur) s'il n'y avait pas eu regroupement.

On retrouve dans chaque fonction les deux instructions (hors entrées-sorties) du programme initial, sous une forme modifiée faisant intervenir la variable de type Rectangle. Pour chaque fonction, il faut supprimer une des deux instructions : le calcul du périmètre dans la fonction calculer_surface et le calcul de la surface dans la fonction calculer_perimetre.

En ce qui concerne le programme principal, il n'y a aucune adaptation à faire :

```
sub main()
     Dim objet as Rectangle
     Dim surface as double
     Dim perimetre as double
     objet.hauteur = InputBox( "objet.hauteur : " )
     objet.largeur = InputBox( "objet.largeur : " )
     surface = calculer_surface(objet)
     perimetre = calculer_perimetre(objet)
     MsgBox( "surface : " & surface )
     MsgBox( "perimetre : " & perimetre )
end sub
```