Diviseurs d'un entier

TI-82 Advanced
Edition Python

TI-83 Premium CE
Edition Python

Diviseur d'un entier non nul

Définition : Soit a et b deux entiers non nuls. On dit que b divise a dès qu'il existe $k \in \mathbb{Z}$ tel que $a = kb$.

Remarque : b divise a signifie que lorsqu'on effectue la division euclidienne de a par b alors le reste est nul.

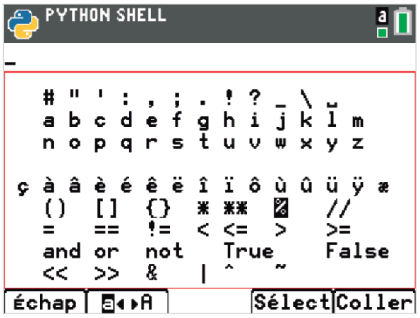
En Python pour obtenir le reste de la division euclidienne de a par b on écrit **a%b**.

Accédons à la programmation en Python en appuyant sur [prgm] puis **Python App**. Commençons en exécutant quelques commandes dans le Shell (la console) en sélectionnant [Shell] :

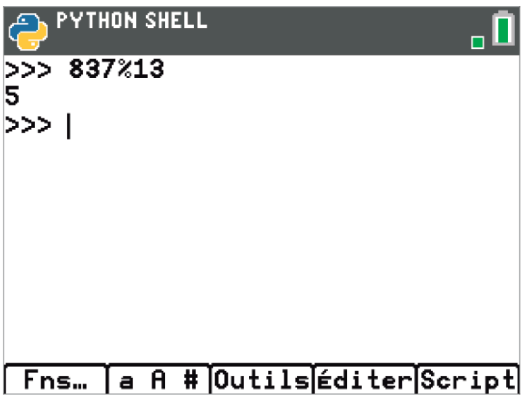
Exemple 1 : Déterminer si 13 divise 837, en calculant le reste de la division euclidienne de 837 par 13.

Pour cela on va écrire : **837%13**.

Pour obtenir le symbole **%**, dans [a A #], à l'aide des flèches de direction, sélectionner le symbole **%** puis appuyer sur **Sélect** et **Coller** afin de l'insérer dans le Shell.



Le reste de la division euclidienne de 837 par 13 est 5, donc 13 ne divise pas 837.



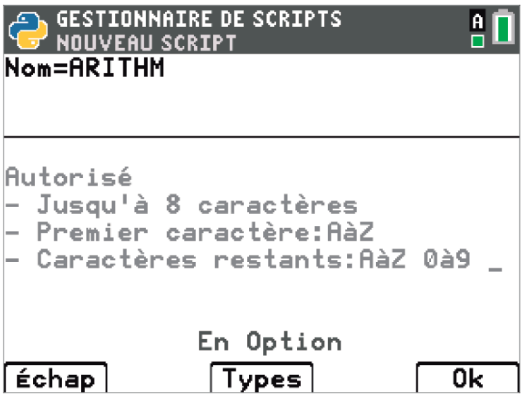
Exemple 2 : Déterminer si 17 divise 493.

On entre dans la console 493%17 est on obtient 0. Ce qui prouve que 17 divise 493.

Ecrivons maintenant une fonction **divise** qui prend comme arguments deux entiers non nuls **a** et **b** et qui renvoie **True** si **a** divise **b** et **False** sinon.

Application : Utiliser cette fonction avec les deux exemples précédents.

Créer un nouveau script en appuyant sur [Script] (pour entrer dans le gestionnaire de scripts) puis [Nouv]. Ecrire le nom du script (par exemple **ARITHM**) et valider en appuyant sur [Ok].



Diviseurs d'un entier

def et **return** sont accessibles dans [Fns…].

Le test conditionnel **if** … **else** est accessible dans [Fns…] onglet **Ctl**.

True et **False** peuvent être saisies au clavier ou accessibles dans [Fns…] onglet **Ops** (tout en bas de la liste).

Exécuter le script en appuyant sur [Exéc] , on arrive alors dans le Shell. Pour obtenir rapidement le nom de notre fonction on appuie sur [var]

On retrouve bien les résultat précédents (voir copie d'écran ci-contre).

ÉDITEUR : ARITHM
LIGNE DU SCRIPT 0005

```
def divise(a,b):
  if b%a==0:
    return True
  else:
    return False
```

PYTHON SHELL

```
>>> # Shell Reinitialized
>>> # L'exécution de ARITHM
>>> from ARITHM import *
>>> divise(13,837)
False
>>> divise(17,493)
True
```

Nombres parfaits

Définition : Soit a un entier non nul. On dit que a est un nombre parfait si la somme de ses diviseurs stricts est égale à a.

Remarque : On ne prend que les diviseurs positifs de a. Un diviseur strict de a est un diviseur de a qui est différent de a.

Exemple 3 : 6 est-il parfait ? 10 est-il parfait ?

Les diviseurs stricts de 6 sont 1 ; 2 ; 3. De plus $1 + 2 + 3 = 6$. Donc 6 est un nombre parfait.

Les diviseurs stricts de 10 sont 1 ; 2 ; 5. De plus $1 + 2 + 5 = 8$. Donc 10 n'est pas un nombre parfait.

Ecrivons une fonction Python **parfait** qui prend comme argument un entier non nul **a** et qui renvoie **True** si **a** est parfait et **False** sinon.

Pour cela on va écrire une boucle avec un entier **b** qui va varier de **1** à **a-1** (on s'arrête à **a-1** car on ne cherche que des diviseurs stricts de **a**).

De plus on va utiliser une variable **s** qui va représenter la somme des diviseurs stricts de **a**. Ainsi dans la boucle, on va tester si **b** divise **a**, et si c'est le cas on ajoutera ce diviseur à **s**.

On obtient le script ci-contre :

Application : Parmi les nombres suivants, déterminer ceux qui sont parfaits : 28 ;136 ;496.

A l'aide de notre fonction **parfait** on peut conclure :

Conclusion : 28 est parfait, 136 n'est pas parfait et 496 est parfait.

ÉDITEUR : ARITHM
LIGNE DU SCRIPT 0016

```
def parfait(a):
  s=0
  for b in range(1,a):
    if divise(b,a):
      s=s+b
  if s==a:
    return True
  else:
    return False
```

[Fns…] [a A #] [Outils] [Exéc] [Script]

PYTHON SHELL

```
>>> # Shell Reinitialized
>>> # L'exécution de ARITHM
>>> from ARITHM import *
>>> parfait(28)
True
>>> parfait(136)
False
>>> parfait(496)
True
>>> |
```

[Fns…] [a A #] [Outils] [éditer] [Script]