Unité 3 : Débuter la programmation en Python | **Compétence 3 : Programmation et récursivité**

Dans cette troisième leçon de l'unité 3, vous allez utiliser les fonctions afin de réaliser une programmation récursive.

Objectifs :

- Appliquer une fonction.
- Découvrir et mettre en œuvre la programmation récursive.

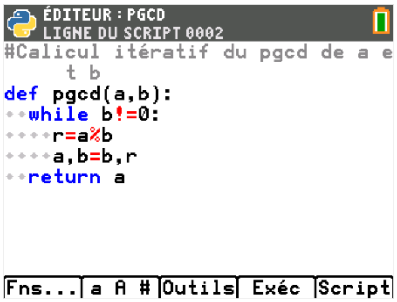
Calcul de PGCD (Méthode itérative)
Pour calculer le « plus grand commun diviseur de deux nombres » (PGCD), on utilise l'algorithme d'Euclide.
Remarque : $a > b$

On procède de la manière suivante :

- On effectue la division euclidienne de *a* par *b*. On note *r* le reste (on n'utilise pas le quotient).
- On remplace ensuite *a* par *b* et *b* par *r*.
- **Tant que** le reste est différent de 0, on **réitère** le procédé.

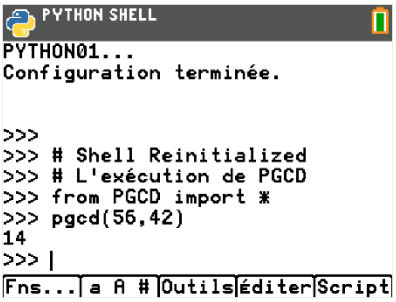
Après un certain nombre d'itérations, on obtient un reste égal à 0.

Le PGCD de ***a*** et de ***b*** est alors le reste précédent (c'est à dire **le dernier reste non nul**).



- Créer un nouveau script et le nommer **pgcd**.
- Créer une fonction pgcd(a,b) en tapant sur **F1** menu (**Fonc**).
- Le symbole ≠ s'écrit en langage Python à l'aide de ! = et se trouve dans le menu F2 (a A…) ou bien dans le menu tests de la calculatrice.
- Noter l'affectation a,b=b,r dans le script qui permet de gagner une ligne de code, mais qui correspond à la réalisation des affectations : a = b et b = r

- Exécuter le script pour différents nombres.



Un pas plus loin .

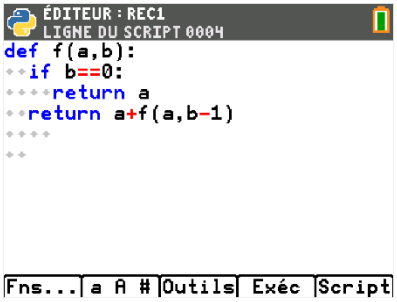
3 : Programmation récursive :

Un algorithme est dit récursif si, à un moment, il s'appelle lui-même.

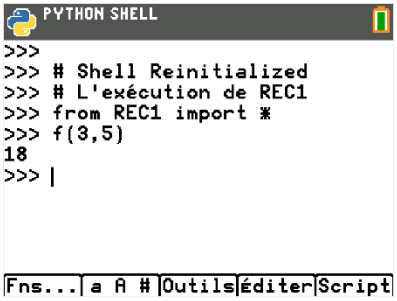
La récursivité peut posséder de nombreux avantages dans un algorithme. Premièrement, elle permet de résoudre des problèmes, d'habitude insolubles avec l'utilisation de simples boucles *pour* ou *tant que*. Elle peut aussi rendre un algorithme plus lisible et plus court, mais surtout, elle permet, dans certains cas, un gain colossal de temps comme c'est le cas dans les algorithmes de tri.

Un premier exemple de récursivité :

- Créer un nouveau script et le nommer **REC1**.
- Entrer le script ci-contre.
- Quel est le cas de base dans cette fonction récursive ?
- Qu'est ce qui garantit dans cette fonction récursive, que le script finira par s'arrêter ?
- Écrire le processus complet.



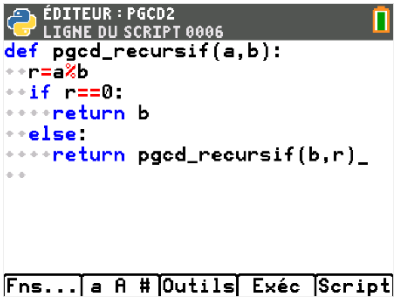
- Exécuter le script dans une console.
- Que retourne *f(a,b) a* et *b* étant des entiers naturels non nuls ?



3.1 : Un calcul de pgcd récursif :

Le calcul du PGCD de deux entiers positifs *a* et *b* utilise toujours l'algorithme d'Euclide. Soit *r* le reste de la division euclidienne de *a* par *b* :
$a = b*q + r$, $r < b$.

Tout diviseur commun de *a* et *b* divise aussi $r = a - b*q$
et réciproquement tout diviseur commun de *b* et *r* divise aussi $a = b*q + r$.
Donc le calcul du PGCD de *a* et *b* se ramène à celui du PGCD de *b* et *r* ; et on peut recommencer sans craindre une boucle sans fin, car les restes successifs sont strictement décroissants. Le dernier reste non nul obtenu est le PGCD cherché.

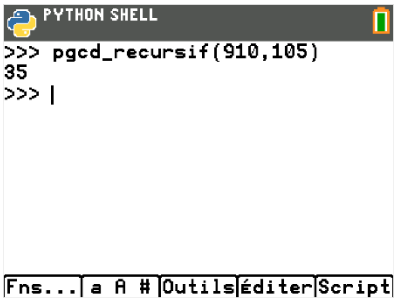


Par exemple si *a*=96 et *b*=81, les calculs sont les suivants :

et le PGCD est 3

a		*b*		*r*
96	= 1 *	81	+	15
81	= 5 *	15	+	6
15	= 2 *	6	+	3
6	= 2 *	3	+	0

- Exécuter le script en appuyant sur la touche **F4** (**Exec**) et le tester pour quelques valeurs.



Conseil à l'enseignant : Pour éviter les cercles vicieux, une fonction récursive doit toujours comporter un cas particulier où le résultat est calculé directement, c'est à dire sans appel récursif ; il faut aussi s'assurer que ce cas particulier finira toujours par se présenter.