Question. On considère le programme :

```
# a et b sont deux entiers >=0
while a>0:
    if a<b:
        (a,b) = (2*a,b-a)
    else:
        (a,b) = (a-b,2*b)
print(a,b)
```

(a) Pour quelles valeurs initiales de a et b (entiers positifs) la boucle *while* se termine-t-elle ?
(b) Dans le cas où la boucle *while* se termine, déterminer le nombre de fois où elle est répétée (en fonction de a et b).
(c) Dans le cas où la boucle *while* se termine, quel est le résultat affiché à la fin ?
(d) Établir que soit la boucle *while* se termine, soit le couple (a,b) reprend périodiquement les mêmes valeurs à partir d'un certain rang.
(e) Dans le cas où la boucle *while* ne se termine pas, quelle est la période ?

Réponse. **Valeurs particulières.** Il peut être intéressant d'étudier le comportement du programme sur certaines valeurs particulières de départ.

- Si $a = 0$, alors quel que soit b, la boucle *while* s'arrête immédiatement.
- Si $b = 0$, et $a > 0$, alors la boucle *while* remplace $(a,b) = (a,0)$ par $(a-b,2b) = (a,0)$. Le couple (a,b) ne change jamais, la boucle *while* ne s'arrête pas.

Première remarque fondamentale. Dans la boucle, (a,b) est transformé :

- Soit en $(a',b') = (2a, b-a)$ lorsque $b > a$, on a alors $a' \geqslant 0$ et $b' \geqslant 0$;
- Soit en $(a',b') = (a-b, 2b)$ lorsque $b \leqslant a$, on a de même $a' \geqslant 0$ et $b' \geqslant 0$.

Les quantités a et b restent donc toujours positives (et entières) au cours de l'exécution du programme.

Pour préciser les notations. Les variables utilisées dans le programme s'appellent a et b mais elles prennent différentes valeurs au cours de l'exécution de ce programme. Pour préciser les choses, on peut définir la suite $((a_n,b_n))_{n\in\mathbb{N}}$ des valeurs prises par le couple (a,b) au cours du programme par récurrence en posant :

$$\begin{aligned} (a_0,b_0) &= (a,b) \quad \text{(valeurs initiales)} \\ \forall n \in \mathbb{N},\ (a_{n+1},b_{n+1}) &= (2a_n, b_n - a_n) \quad \text{si} \quad a_n < b_n \\ &= (a_n - b_n, 2b_n) \quad \text{sinon} \end{aligned}$$

Ainsi, a_n et b_n sont les valeurs contenues dans les variables a et b après n exécutions de la boucle *while*. On peut noter que si $a_n = 0$, alors :

- Soit $b_n > 0$ et dans ce cas, $(a_{n+1},b_{n+1}) = (2a_n, b_n - a_n) = (0, b_n)$;
- Soit $b_n = 0$ et dans ce cas, $(a_{n+1},b_{n+1}) = (a_n - b_n, 2b_n) = (0,0)$.

Ainsi, à partir du moment où $a_n = 0$, la suite $((a_n,b_n))$ est constante. Comme ceci correspond au cas où la boucle *while* s'arrête, on aurait pu aussi choisir d'arêter là la construction de la suite. On a le résultat suivant : la boucle *while* s'arrête si, et seulement si, il existe un entier $n \in \mathbb{N}$ tel que $a_n = 0$.

Deuxième remarque fondamentale. On remarque que dans tous les cas, on a :

$$a_{n+1} + b_{n+1} = a_n + b_n$$

Autrement dit, la quantité $a+b$ reste constante tout au long de la boucle *while*. On pose $c = a+b$. Ceci permet de répondre à une question : si la boucle *while* s'arrête, alors au moment où l'instruction

`print` est effectuée on a $a = 0$ et donc $b = c$. Lorsque le programme affiche un résultat, il affiche $(0, c)$ où c est la somme des valeurs contenues au départ dans les variables a et b.
Autre conséquence. Comme a et b restent positifs et $a + b = c$, on a à tout moment $0 \leqslant a \leqslant c$ et $0 \leqslant b \leqslant c$. En particulier, a et b ne peuvent prendre qu'un nombre fini de valeurs.
Suite des valeurs prises par (a, b). La suite $((a_n, b_n))$ est à valeurs dans $E = [\![0, c]\!]^2$, cet ensemble est fini. L'ensemble des indices de la suite est infini donc il existe nécessairement des entiers $n, m \in \mathbb{N}$ tels que $n < m$ et $(a_n, b_n) = (a_m, b_m)$. Dit autrement, l'application

$$\begin{array}{rccc} f : & \mathbb{N} & \to & E \\ & n & \mapsto & (a_n, b_n) \end{array}$$

ne peut pas être injective puisque l'ensemble de départ est infini et l'ensemble d'arrivée est fini. À l'indice m, la suite est donc revenue dans le même état que celui où elle était à l'indice n et elle va donc reprendre les mêmes valeurs. La suite $((a_n, b_n))$ est donc périodique à partir de l'entier n. Ceci montre que soit la boucle *while* se termine (s'il existe n tel que $a_n = 0$), soit les variables a et b reprennent périodiquement les mêmes valeurs à partir d'un certain moment.
À partir de maintenant, on suppose que les valeurs de a et b au départ sont strictement positives.
Reformulation du programme. On note que $b = c - a$. La condition $a < b$ est donc équivalente à $a < c - a$ autrement dit $2a < c$. Dans ce cas, la nouvelle valeur de a est $2a$. Dans le cas contraire, la nouvelle valeur de a est $a - b = a - (c - a) = 2a - c$. On peut donc aussi bien écrire le code sous la forme :

```
# a et b sont deux entiers >=0
c = a+b
while a>0:
    if 2*a<c:
        a = 2*a
    else:
        a = 2*a-c
b = c-a
print(a,b)
```

Reformulation de la suite des valeurs prises. De même, il n'est pas utile de considérer la suite $((a_n, b_n))$ puisqu'il suffit de connaitre les valeurs de a_n. On peut donc considérer la suite (a_n) définie par récurrence en posant :

$$\begin{array}{rl} & a_0 = a \quad \text{(valeur de départ)} \\ \forall n \in \mathbb{N},\ a_{n+1} = & 2a_n \quad \text{si } 2a_n < c \\ & = 2a_n - c \quad \text{si } 2a_n \geqslant c \end{array}$$

Ou encore, en utilisant les notations PYTHON :

$$\begin{array}{rl} & a_0 = a \quad \text{(valeur de départ)} \\ \forall n \in \mathbb{N},\ a_{n+1} = & 2a_n \% c \end{array}$$

Condition de terminaison de la boucle. L'algorithme consiste donc à faire des multiplications par 2 successives à partir de la valeur de a initiale et à prendre le reste de la division euclidienne par c. Ainsi, on a $a_n = 2^n a \% c$ après n répétitions de la boucle, $n \geqslant 1$. La boucle se termine lorsqu'il existe un entier $n \geqslant 1$ tel que $a_n = 0$, autrement dit lorsque c est un diviseur de $2^n a$. La condition de terminaison est alors : il existe un entier $n \geqslant 1$ tel que $c \mid 2^n a$. Le nombre de fois où la boucle est répétée est alors n, plus petit entier naturel non nul tel que $c \mid 2^n a$.

Justification de $a_n = 2^n a\%c$ pour $n \geqslant 1$. On le démontre par récurrence. Pour $n = 1$, on a déjà noté que $a_1 = 2a\%c$ donc le résultat est vrai. Soit $n \geqslant 1$ et supposons que $a_n = 2^n a\%c$. On écrit cette division euclidienne :

$$2^n a = qc + a_n$$

avec $q \in \mathbb{N}$ et $a_n \in [\![0, c-1]\!]$. On multiplie par 2 :

$$2^{n+1} a = 2qc + 2a_n$$

On a alors deux cas :

- Soit $2a_n < c$, dans ce cas $a_{a+1} = 2a_n$ est bien le reste dans la division euclidienne de $2^{n+1}a$ par c ;
- Soit $2a_n \geqslant c$, on écrit alors :

$$2^{n+1} a = (2q+1)c + 2a_n - c$$

et $a_{n+1} = 2a_n - c$ est le reste dans la division euclidienne de $2^{n+1}a$ par c.

Dans tous les cas, $a_{n+1} = 2^{n+1} a\%c$.

Étude de la période. Supposons que la boucle *while* ne se termine pas, la suite (a_n) est alors périodique à partir d'un certain rang. Il existe alors un entier n et un entier p tel que $a_n = a_{n+p}$. On a alors :

$$2^n a\%c = 2^{n+p} a\%c$$

Autrement dit $2^n a(2^p - 1)\%c = 0$. L'entier p est donc tel que $c \mid 2^n a(2^p - 1)$.

Comment tester ce code avec PYTHON ? Pour pouvoir tester avec différentes valeurs de a et b, on préfère définir une fonction :

```
def test(a,b):
    while a>0:
        if a<b:
            (a,b) = (2*a,b-a)
        else:
            (a,b) = (a-b,2*b)
    print(a,b)
```

Le problème c'est que pour certaines valeurs de a et b, la fonction `test(a,b)` ne s'arrêtera pas. Une première méthode est d'ajouter à la boucle *while* un compteur pour limiter arbitrairement le nombre d'itérations effectuées.

```
def test(a,b):
    n = 0
    a0 = a
    b0 = b
    while a>0 and n<1000:
        if a<b:
            (a,b) = (2*a,b-a)
        else:
            (a,b) = (a-b,2*b)
        n = n+1
    if a==0:
        print(f"Si au début a={a0}, b={b0} alors la boucle s'arrête et à la fi
    else:
        print(f"Si au début a={a0}, b={b0} alors la boucle semble ne pas s'arr

for a in range(1,5): # Le cas a=0 est évident
    for b in range(0,5):
        test(a,b)
```

```
Si au début a=1, b=0 alors la boucle semble ne pas s'arrêter
Si au début a=1, b=1 alors la boucle s'arrête et à la fin a=0 et b=2
Si au début a=1, b=2 alors la boucle semble ne pas s'arrêter
Si au début a=1, b=3 alors la boucle s'arrête et à la fin a=0 et b=4
Si au début a=1, b=4 alors la boucle semble ne pas s'arrêter
Si au début a=2, b=0 alors la boucle semble ne pas s'arrêter
Si au début a=2, b=1 alors la boucle semble ne pas s'arrêter
Si au début a=2, b=2 alors la boucle s'arrête et à la fin a=0 et b=4
Si au début a=2, b=3 alors la boucle semble ne pas s'arrêter
Si au début a=2, b=4 alors la boucle semble ne pas s'arrêter
Si au début a=3, b=0 alors la boucle semble ne pas s'arrêter
Si au début a=3, b=1 alors la boucle s'arrête et à la fin a=0 et b=4
Si au début a=3, b=2 alors la boucle semble ne pas s'arrêter
Si au début a=3, b=3 alors la boucle s'arrête et à la fin a=0 et b=6
Si au début a=3, b=4 alors la boucle semble ne pas s'arrêter
Si au début a=4, b=0 alors la boucle semble ne pas s'arrêter
Si au début a=4, b=1 alors la boucle semble ne pas s'arrêter
Si au début a=4, b=2 alors la boucle semble ne pas s'arrêter
Si au début a=4, b=3 alors la boucle semble ne pas s'arrêter
Si au début a=4, b=4 alors la boucle s'arrête et à la fin a=0 et b=8
```

On peut procéder rigoureusement une fois qu'il est admis qu'il n'y a que deux possibilités : soit la boucle *while* s'arrête, soit le couple (a,b) reprendra deux fois la même valeur. Il suffit alors de retenir la liste L des valeurs prises par (a,b) et si la nouvelle valeur calculée est déjà dans L c'est que la boucle ne se termine pas.

```python
def test(a,b):
    L = []
    a0 = a
    b0 = b
    while a>0 and (a,b) not in L:
        L.append((a,b))
        if a<b:
            (a,b) = (2*a,b-a)
        else:
            (a,b) = (a-b,2*b)
    if a==0:
        print(f"Si au début a={a0}, b={b0} alors la boucle s'arrête et à la fi
    else:
        print(f"Si au début a={a0}, b={b0} alors la boucle ne s'arrête pas")

for a in range(1,5): # Le cas a=0 est évident
    for b in range(0,5):
        test(a,b)
```

```
Si au début a=1, b=0 alors la boucle ne s’arrête pas
Si au début a=1, b=1 alors la boucle s’arrête et à la fin a=0 et b=2
Si au début a=1, b=2 alors la boucle ne s’arrête pas
Si au début a=1, b=3 alors la boucle s’arrête et à la fin a=0 et b=4
Si au début a=1, b=4 alors la boucle ne s’arrête pas
Si au début a=2, b=0 alors la boucle ne s’arrête pas
Si au début a=2, b=1 alors la boucle ne s’arrête pas
Si au début a=2, b=2 alors la boucle s’arrête et à la fin a=0 et b=4
Si au début a=2, b=3 alors la boucle ne s’arrête pas
Si au début a=2, b=4 alors la boucle ne s’arrête pas
Si au début a=3, b=0 alors la boucle ne s’arrête pas
Si au début a=3, b=1 alors la boucle s’arrête et à la fin a=0 et b=4
Si au début a=3, b=2 alors la boucle ne s’arrête pas
Si au début a=3, b=3 alors la boucle s’arrête et à la fin a=0 et b=6
Si au début a=3, b=4 alors la boucle ne s’arrête pas
Si au début a=4, b=0 alors la boucle ne s’arrête pas
Si au début a=4, b=1 alors la boucle ne s’arrête pas
Si au début a=4, b=2 alors la boucle ne s’arrête pas
Si au début a=4, b=3 alors la boucle ne s’arrête pas
Si au début a=4, b=4 alors la boucle s’arrête et à la fin a=0 et b=8
```