Calcul rapide de puissances

```
def puiss_rapide(A,N):
    """
    Calcul rapide de A^N
    A : int ou float
    N entier >=0
    """
    x=A
    n=N
    y=1
    # Point 1
    while n>0:
        # Point 2
        if n%2==1:
            y=y*x
        x=x**2
        n=n/2
        # Point 3
    # Point 4
    print x
    return y
print puiss_rapide(2,21)
```

```
4294967296 2097152
```

◇ Les valeurs successives de x, y et n pour $A = 2$ et $N = 21$:

x	~~2~~	~~4~~	~~16~~	~~256~~	~~65536~~	4294967296	
y	~~1~~	~~2~~	~~32~~	2097152			
n	~~21~~	~~10~~	~~5~~	~~2~~	~~1~~	0	

Ceci correspond au calcul « à la main » :

$$
\begin{aligned}
2^{21} &= 2^{21} \times 1 \\
&= 2^{2\times 10+1} = (2^2)^{10} \times 2 = 4^{10} \times 2 \\
&= 4^{2\times 5} \times 2 = (4^2)^5 \times 2 = 16^5 \times 2 \\
&= 16^{2\times 2+1} \times 2 = (16^2)^2 \times 16 \times 2 = 256^2 \times 32 \\
&= 65536^1 \times 32 \\
&= (65536^2)^0 \times 65536 \times 32 = 4294967296^0 \times 2097152 \\
&= 2097152
\end{aligned}
$$

◇ La quantité n est entière, positive et strictement décroissante à chaque tour de boucle. Par conséquent, la fonction *termine*.

◇ La propriété $y \times x^n = A^N$ est un invariant de boucle. En effet :

- Cette propriété est vraie au point 1 car $A^N = x^n = y \times x^n$ puisque $x = A$, $n = N$ et $y = 1$;
- Si on suppose cette propriété vraie au point 2, alors il faut distinguer deux cas :
  - Si n est pair, notons $n = 2n'$, alors n' est la nouvelle valeur de n au point 3, $x' = x^2$

est la nouvelle valeur de x en 3 et $y' = y$. Ainsi :

$$y' \times x'^{n'} = y \times (x^2)^{n'} = y \times x^{2n'} = y \times x^n = A^N$$

– Si n est impair, notons $n = 2n' + 1$, alors n' est la nouvelle valeur de n au point 3, $x' = x^2$ est la nouvelle valeur de x en 3 et $y' = xy$. Ainsi :

$$y' \times x'^{n'} = xy \times (x^2)^{n'} = xy \times x^{2n'} = y \times x^{2n'+1} = y \times x^n = A^N$$

Dans tous les cas, la propriété reste vraie au point 3.
Nécessairement, cette propriété est vraie à la sortie de la boucle. Or, quand la boucle est terminée, on a $n = 0$ donc $x^n = 1$ et $A^N = y \times x^n = y$. Le résultat retourné est donc bien égal à A^N. La fonction est *correcte*.

◇ Calcul de la *complexité* (en fonction de N). Soit $p \in \mathbb{N}$ tel que $2^p \leqslant N < 2^{p+1}$, la boucle *while* est repétée $p+1$ fois (on divise à chaque fois par 2 jusqu'à obtenir 0). Or en appliquant la fonction ln :

$$p\ln(2) \leqslant \ln(N) \leqslant (p+1)\ln(2)$$

Donc la boucle est effectuée au plus $\dfrac{\ln(N)}{\ln(2)} + 1$ fois et ainsi :

$$\mathrm{T}_{\texttt{puiss\_rapide}}(N) = \mathrm{O}(\ln(N))$$

Remarque. Maintenant que le fonctionnement du programme est compris, on peut l'écrire de manière plus concise : on utilise directement A et N (qui peuvent être considérées comme des variables locales de la fonction) à la place de x et n :

```
def puiss_rapide(A,N):
    """
    Calcul rapide de A^N
    A de type int ou float
    N entier >=0
    """
    y=1
    while N>0:
        if N%2==1:
            y=y*A
        A=A**2
        N=N/2
    return y
print puiss_rapide(2,21)
```

2097152 □