

## *Première puissance d'un nombre positif supérieure ou inférieure à une valeur donnée*

Auteur : Yves Coudert

**Niveaux scolaires :**  
Seconde

**Mots clés :**  
Algorithme/Programmation

### **ENONCE**

Programmer, en langage Python, une fonction qui permet d'obtenir la première puissance d'un nombre positif, supérieure ou inférieure à une valeur donnée. La fonction aura trois paramètres : un nombre positif  $q$ , une valeur donnée  $a$  et  $s$  qui prendra la valeur 1 pour supérieure ou -1 pour inférieure.

Utiliser la fonction pour déterminer la première puissance pour les cas suivants :

a)  $2^n \geq 100$     b)  $(0.5)^n \leq 0.01$

- Algorithme de la fonction :

Fonction première puissance d'un nombre positif supérieure ou inférieure à un nombre donné

Arguments de la fonction  $q$ ,  $a$  et  $s$

Si  $q \geq 0$  alors

    Si  $s = 1$  alors

$n \leftarrow 0$

        Tant que  $q^n < a$

$n \leftarrow n + 1$

        Retourner  $n$

    Sinon

$n \leftarrow 0$

        Tant que  $q^n > a$

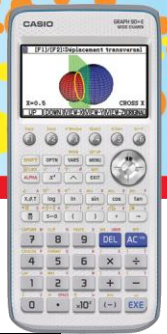
$n \leftarrow n + 1$

        Retourner  $n$

    Sinon

        Afficher "erreur  $q$  n'est pas positif"

# Graph 90+E



- Programmation de la fonction :

Nous allons créer une fonction "puis(q,a,s)".  
Cette fonction prend comme arguments :

- un nombre positif q,
- un nombre donné auquel on cherche la première puissance de q supérieure à a si le nombre s vaut 1, ou inférieure à a sinon.

Nous effectuons, dans un premier temps, un test "if else" afin de savoir si q est bien positif.

Les commandes "if else" sont disponibles dans le catalogue (**SHIFT** **4**).

Si le test est "vrai", on effectuera un deuxième test "if else" pour déterminer si s est égal à 1.

Si ce deuxième test est aussi "vrai", on initialise une variable n à 0 et on fait alors une boucle while "tant que" avec la condition "q^n est strictement inférieure à a".

On incrémente n jusqu'à sortir de la boucle.

A la sortie de la boucle, on retourne la valeur de n qui est la première puissance de q supérieure à a.

Si le deuxième test est "faux", on initialise une variable n à 0 et on fait alors une boucle while "tant que" avec la condition "q^n est strictement supérieur à a".

On incrémente n jusqu'à sortir de la boucle. A la sortie de la boucle, on retourne la valeur de n qui est la première puissance de q inférieure à a.

```
puis.py 001/005
def puis(q, a, s):
    if q >= 0:

else:
```

```
puis.py 005/006
def puis(q, a, s):
    if q >= 0:
        if s == 1:

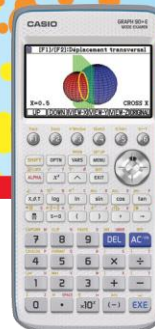
else:
```

```
puis.py 001/014
def puis(q, a, s):
    if q >= 0:
        if s == 1:
            n = 0
            while q**n < a:
                n = n + 1
            return n

else:
```

```
puis.py 006/014
        n = n + 1
        return n
else:
    n = 0
    while q**n > a:
        n = n + 1
    return n
```

# Graph 90+E



Si le premier test est "faux", la dernière condition else concerne le cas où q est négatif et il faut alors indiquer une erreur : "erreur q n'est pas positif".

Remarque : pour comparer les valeurs de deux variables, on utilise **SHIFT** **□** **{=}** pour faire la double égalité `==` et **F2** **{OPERAT}** **F3** **{>}** ou **F4** **{<}** ou **F1** **{=}** pour obtenir `>=` ou `>` ou `<`.

L'utilisation dans la fonction de la commande `return` plutôt que `print` permet de réutiliser la variable n dans un autre programme ou dans le SHELL.

- Utilisation de la fonction

Le cas a) correspond à  $q=2$ ,  $a=100$  et  $s=1$ .

Le cas b) correspond à  $q=0.5$ ,  $a=0.01$  et  $s=-1$ .

Pour utiliser la fonction, on va saisir, dans le SHELL, le nom de la fonction et les paramètres :

```
puis(2, 100, 1)
puis(0.5, 0.01, 1)
```

Dans les deux cas, on trouve 7.

En effet :

$$2^7 = 128 \geq 100 \text{ et } 0.5^7 = 0.0078125 \leq 0.01$$

On peut également vérifier que si on utilise un réel q négatif, par exemple `puis(-2, 100, 1)`, la fonction indique le message d'erreur.

```
puis.py 014/014
else:
    n=0
    while q**n>a:
        n=n+1
    return n
else:
    return "erreur q
```

```
CASIO COMPUTER CO.,
>>>from puis import *
>>>puis(2,100,1)
7
>>>puis(0.5,0.01,-1)
7
>>>
RUN A↔a CHAR
```

```
>>>puis(0.5,0.01,-1)
7
>>>2**7
128
>>>0.5**7
0.0078125
>>>
RUN A↔a CHAR
```

```
>>>2**7
128
>>>0.5**7
0.0078125
>>>puis(-2,100,1)
"erreur q n'est pas p
>>>
RUN A↔a CHAR
```