Boucle non bornée - Corrigé

NUMWORKS

1 Exercice

Écrire une fonction `max_cube(n)` qui prend un entier naturel n en argument et qui renvoie le plus grand entier dont le cube est inférieur ou égal à n.

1.1 Analyse de l'énoncé

Prenons pour exemple le cas $n = 30$.
On cherche le plus grand entier naturel (0, 1, 2, ...) dont le cube est inférieur à 30.
Naturellement nous testons les entiers un par un :

— Si entier = 0

$$0^3 = 0 < 30$$

— Si entier = 1

$$1^3 = 1 < 30$$

— Si entier = 2

$$2^3 = 8 < 30$$

— Si entier = 3

$$3^3 = 27 < 30$$

— Si entier = 4

$$4^3 = 64 > 30$$

Le cube de 4 étant trop grand, l'entier recherché est 3 : c'est en effet le plus grand nombre dont le cube est inférieur à 30.
Grâce à cet exemple, on a pu identifier la structure de la fonction que l'on va écrire :

1. On initialise une variable `entier` à 0
2. On teste si "`entier` au cube" est inférieur ou égal à n
3. Si c'est le cas, on ajoute 1 à entier
4. Si ce n'est pas le cas, la réponse est "`entier - 1`" (car `entier` au cube dépasse n)
5. On recommence à l'étape 2

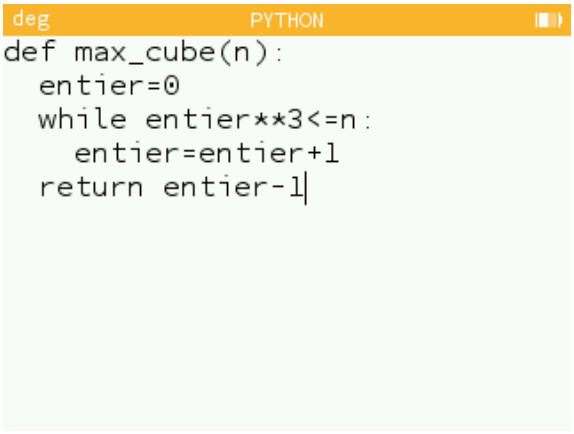

FIGURE 1 – image 1

Vous reconnaissez une structure de boucle **`while`** :

```
while condition:
        instruction
```

Ici la condition est “**`entier`** au cube est inférieur ou égal à n” :

```
entier**3 <= n
```

Et l’instruction est “ajouter 1 à **`entier`**” :

```
entier = entier + 1
```

1.2 Résolution

1.3 Remarque pour finir

Pour incrémenter un nombre d’une même valeur à chaque fois, vous pouvez écrire

```
entier += 1
```

à la place de

```
entier = entier + 1
```

2 Un autre exercice à traiter

Écrire une fonction **`facteurs(n)`** qui prend un nombre entier en argument et renvoie la liste des diviseurs de sa decomposition en facteurs premiers.

2.1 Corrigé

2.2 Quelques remarques sur le corrigé

Structure du code

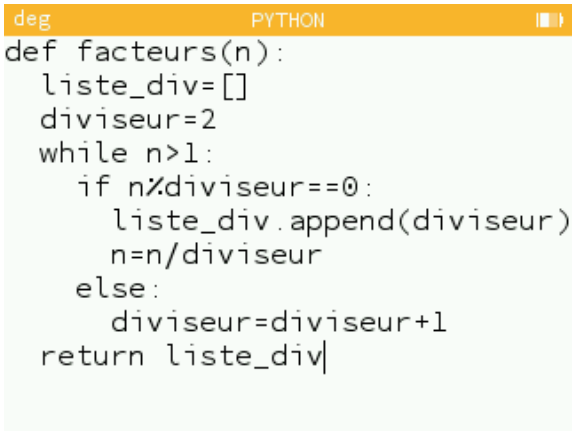

FIGURE 2 – image 2

On crée une liste des diviseurs premiers de n que l'on va remplir à chaque fois que l'on en trouve un : `liste_div`.

On commence par tester si **`n`** est divisible par 2.

Si c'est le cas, on stocke 2 dans `liste_div` et on remplace **`n`** par **`n/2`**.

Puis on recommence par tester si 2 divise le nouveau **`n`**.

Si ce n'est pas le cas, on ajoute 1 à 2 et on teste donc si le nouveau **`n`** est divisible par 3.

Ainsi de suite jusqu'à ce qu'à force de diviser **`n`** par ses diviseurs premiers, on arrive à n = 1.

Pour ajouter un élément à une liste, on utilise la fonction **`append`**. Pour ajouter l'élément 5 à une liste `L`, on écrira :

```
>>> L=[1,2]
>>> L.append(5)
>>> L
[1,2,5]
```