Boucles *while*

◊ Une boucle *while* répète une liste d'instructions tant qu'une certaine condition est satisfaite. Une telle boucle est décrite par 2 éléments : une *condition c* et une *suite d'instructions* (le corps de la boucle).

◊ En arrivant sur l'instruction *while*, on teste si la condition *c* est vraie. Si c'est le cas les instructions du corps de la boucle sont exécutées puis on recommence (on revient sur la condition *c*, on regarde à nouveau si elle est vraie, si c'est le cas on effectue les instructions du corps de la boucle, etc.). On continue ainsi jusqu'à ce que la condition *c* devienne fausse et, lorsque c'est le cas, on passe aux instructions qui suivent la boucle.

```
tant que c
  Instruction 1
  :
  Instruction n
fin tant que
Instructions suivantes
```

△ La condition *c* est la condition pour *continuer* à exécuter les instruction. La condition *d'arrêt* est la négation de *c*.

△ Il est facile de produire ainsi des programmes qui ne s'arrêtent pas. Lorsque l'on utilise une boucle *while*, il faut avoir un argument permettant de justifier que la *condition* deviendra nécessairement fausse.

Exemple Python. On calcule la somme $1^2 + 2^2 + 3^2 + \dots + n^2$ et on cherche la première valeur de *n* pour laquelle cette somme est strictement supérieure à 1000.

```
s = 0
n = 0
while s <= 1000:
    n = n + 1
    s = s + n ** 2
print("La somme est >1000 lorsque n =", n)
```

La somme est >1000 lorsque n = 14

△ Les instructions à répéter sont repérées par leur indentation. On notera que la dernière instruction de l'exemple ci-dessus ne fait pas conséquent pas partie de la boucle (elle n'est exécutée qu'une seule fois).

Exemple Python. On considère la suite $(u_n)_{n \geq 0}$ définie par :

$$u_0 = 1 \quad \text{et} \quad \forall n \in \mathbb{N}, u_{n+1} = 1 + u_n^2$$

On admet que $u_n \xrightarrow{n \rightarrow +\infty} +\infty$. Déterminer le premier entier *n* tel que $u_n \geq 1000$.

```
n = 0
u = 1
while u < 1000:
    u = 1 + u ** 2
    n = n + 1
print(n, u)
```

5 458330

Choisir entre une boucle *for* et une boucle *while*

- Si on connaît à l'avance le nombre d'itérations qui doivent être effectuées, on utilise une boucle *for*. Exemple : pour calculer la somme des nombres de 1 à *n*, on sait à l'avance qu'il faudra effectuer *n* itérations et on peut donc utiliser une boucle *for*.
- Si on ne connaît pas à l'avance le nombre d'itérations à effectuer mais que l'on dispose d'une condition permettant de s'arrêter, on utilise une boucle *while*. Exemple : dans l'exemple donné plus haut (somme des carrés), on ne sait pas à l'avance jusqu'à quelle valeur de *n* il faudra aller mais on sait que les itérations s'arrêteront dès que la somme sera strictement supérieure à 1000.
- Ces deux situations ne s'excluent pas. D'ailleurs toute boucle *for* peut également s'écrire comme une boucle *while*.

Justifier la terminaison d'une boucle *while*

◊ En présence de boucle *while*, il est possible de réaliser des programmes qui bouclent indéfiniment. Il est donc important d'avoir le moyen de justifier que le programme va en fait s'arrêter. Démontrer la *terminaison* d'une boucle *while*, c'est établir que cette boucle ne va pas tourner indéfiniment.

◊ On considère le programme ci-contre qui calcule A^N pour une variable *A* de type *float* et *N* de type *int* avec $N \geq 0$. On peut justifier la terminaison de la boucle de la manière suivante : à chaque étape de la boucle, *n* décroît strictement. Par ailleurs, *n* reste entier et positif. Or, il n'existe pas de suites infinies d'entiers positifs qui soit strictement décroissante. Cette suite est donc *finie* et la boucle se termine nécessairement.

```
def puiss(A, N):
    x = 1
    n = N
    while n > 0:
        x = x * A
        n = n - 1
    return x
print(puiss(2, 5))
```

32

◊ Reprenons le premier l'exemple de la première page (somme des carrés). La quantité $d = 1000 - s$ reste toujours entière et positive au cours des itérations de la boucle. De plus, à chaque itération, *s* augmente strictement donc *d* diminue strictement. La suite des valeurs prises par *d* au cours des itérations de la boucle est donc une suite d'entiers positifs strictement décroissante. Cette suite est donc nécessairement finie ce qui assure la terminaison de la boucle *while*.

Exercices pour la semaine prochaine

Question 1. On suppose qu'une variable n (entière, positive) est définie. À l'aide d'une boucle *while*, écrire du code permettant d'afficher successivement n , $n - 1$, etc. jusqu'à 1 puis "zéro !".

Question 2. On suppose qu'une variable a (de type *float* et telle que $a \geq 1$) est définie. Écrire un programme permettant d'afficher le premier entier k tel que $2^k \geq a$.

Question 3.

- En utilisant une boucle *while*, écrire une fonction **somme (n)** prenant en paramètre un nombre entier (positif) n qui calcule et renvoie la somme des carrés des nombres compris entre 1 et n (inclus).
- Utiliser cette fonction pour afficher la valeur de :

$$\sqrt{1^2 + 2^2 + 3^2 + \dots + 10^2}$$

Corrections

Q1.

```
while n>0:
    print(n)
    n = n-1
# Fin de la boucle while : n est nul
print("zéro !")
```

Q2.

```
k = 0
while 2**k<a:
    # On veut que la boucle s'arrête quand 2**k>=a
    # On continue donc tant que 2**k<a
    k = k+1
print(k)
```

Q3.

```
def somme(n):
    s = 0
    k = 1
    while k<=n:
        s = s+k**2
        k = k+1
    return s

print(somme(10))
```