Fiche 6 : Algorithmes

La syntaxe à connaître

```
def nom (variable): #On définit une fonction avec la commande def
    if ... : #Si la condition est vraie
        ... # alors
    else ... # sinon
    while ... : #tant que On inverse l'inégalité souhaitée
        ... #On continue à faire le calcul
    for k in range (n) : #On sait combien on doit faire de calculs ici n

    return ... # affiche le résultat et arrête la boucle
    print(...) #affiche le résultat sans le garder en mémoire
```

```
#3,5 s'écrit 3.5
# 3x s'écrit 3*x
# 32 s'écrit 3**2
# 5/6 est la division décimale de 5 par 6
# 5//6 est la division euclidienne (entier) de 5 par 6
# 5%6 est le reste de la division euclidienne (entier) de 5 par 6
# pour utiliser la racine carrée, il faut :
from math import *
# sqrt(x) est la racine carrée de x
# = signifie l'affectation (on affecte une valeur à une variable)
# == est un test : les deux valeurs à g et à d sont-elles égales ?
```

Les algorithmes à comprendre et savoir compléter :

Soit la suite (u_n) définie par $u_{n+1}=3u_n + 2$

1. Algorithme en python affichant la liste des premiers termes d'une suite

```
def liste (n):
    u=5.2 #on initialise u
    for k in range (n):
        u=3*u+2 #on écrit la formule de récurrence sans les indices
        print(u)
```

2. Algorithme en python calculant la somme des premiers termes d'une suite

```
def somme (n):
    u=5.2 #on initialise u
    s = u
    for k in range (n):
        u=3*u+2 #on écrit la formule de récurrence sans les indices
        s= s+u
    return s
```

3. Algorithme en python de seuil

```
def seuil ():
    u=5.2 #on initialise u
    n=0
    while u<10**9 : # on continue de calculer tant la condition est vraie
        u=3*u+2 #on écrit la formule de récurrence sans les indices
        n=n+1
    return n
```

Pour faire fonctionner un programme, on l'appelle dans la console :

```
Shell
>>> %Run 'premiere_lexique.py'
>>> somme(20)
32427094905.2
>>> %Run 'premiere_lexique.py'
>>> %Run 'premiere_lexique.py'
>>> seuil()
18
```