1 Boucles for et boucles while

On **rappelle** la structure des boucles for (on connait le nombre d'itérations a priori) :

```
for x in "une chaine de caractères est itérable":
    print(x)

for x in [4,8,9]: # une liste aussi
    print(2*x)

# range(début,fin,écart) créé un intervalle entier.
for i in range(10):
    print(i)
# par défaut range(n) correspond à range(0,n,1)

for i in range(0,21,2):
    print(i)
```

Un autre type de boucle est possible, lorsqu'on ne connait pas le nombre d'itérations a priori : les **boucles conditionnelles** ou boucle **while** ("tant que") :

```
# Premier exemple
n=int(input('Un entier supérieur ou égal 2'))
k=2
while n%k!=0: # tant que k ne divise pas n
    k+=1
print("Plus petit diviseur (distinct de 1) de ",n,":",k)

# Deuxième exemple
n=1
f=1
while f<10**10: # tant que f est inférieur à 10 milliards
    n=n+1
    f=f*n
print("Plus petit n tel que n!>= 10^10 " : n)
```

2 Les fonctions

Une fonction est définie par 4 éléments :

- ▷ un nom, qu'il vaut mieux choisir explicite et court ;
- ▷ ses **arguments** ;
- ▷ un bloc d'instructions, marqué par l'indentation ;
- ▷ et **la valeur qu'elle renvoie**.

```
def <nom_de_fonction>(<argument_1>,...,<argument_n>):
        <instr 1>
        ...
        <instr p>
        return <donnée à renvoyer>
```

```
def carré(x):
    """renvoie le carré de x"""
    return x**2

def puissance4(x):
    """renvoie x^4"""
    return carré(carré(x))

def somme(L):
    """renvoie la somme des éléments de L"""
    s=0
    for x in L:
        s+=x
    return s

def diviseurs(n):
    """renvoie la liste des diviseurs de n"""
    L=[]
    for i in range(1,n+1):
        if n%i==0:
            L.append(i)
    return L

def mystere(epsilon):
    n=1
    s=1/n**2
    while 1/n>epsilon:
        n+=1
        s+=1/n**2
    return (6*s)**0.5
```

Exercice 1: (Solution)

Écrire une fonction `pairs(n)` d'argument un entier $n \in \mathbb{N}^*$ et renvoyant la liste de tous les entiers pairs inférieurs ou égaux à n.

Exercice 2: (Solution)

Écrire une fonction `diviseursImpairs` d'argument un entier $n \in \mathbb{N}^*$ et renvoyant la liste de tous les diviseurs impairs de n.

Exercice 3: (Solution)

Écrire un programme demandant deux entiers n, p à l'utilisateur et renvoyant le coefficient binomial $\binom{n}{p}$.

Exercice 4: (Solution)

Un nénuphar a une surface de $1m^2$ et double sa surface tous les jours.
Écrire un programme demandant un entier n à l'utilisateur et renvoyant la surface du nénuphar après n jours.

Exercice 5: (Solution)

La suite de Fibonacci est définie par $F_0 = F_1 = 1$ et $F_{n+2} = F_{n+1} + F_n$ pour tout entier $n \geqslant 0$.

1. Ecrire un programme demandant un entier $n \in \mathbb{N}$ à l'utilisateur et renvoyant le nombre F_n.
2. Écrire un programme demandant un entier $n \in \mathbb{N}$ à l'utilisateur et renvoyant la liste $[F_0, F_1, ..., F_n]$.
3. Déterminer à la main les racines du trinôme $X^2 - X - 1$ et stocker la plus grande racine dans une variable que l'on appellera `phi`.
4. Stocker les quotients $\dfrac{F_{n+1}}{F_n}$ dans une liste `L` pour $n \in [\![0, 49]\!]$.
5. Afficher les 10 derniers termes de cette liste. Conjecture ?

Exercice 6: (Solution)

On appelle suite de Syracuse toute suite d'entiers naturels vérifiant :
- $u_0 = a$ où a est un entier fixé dans $\mathbb{N}^*$
- pour tout $n \in \mathbb{N}^*$,

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si} \quad u_n \text{ est pair} \\ 3u_n + 1 & \text{si} \quad u_n \text{ est impair} \end{cases}$$

1. Écrire une fonction `suivant` qui prend comme argument un entier naturel et qui renvoie le terme suivant dans la suite. Par exemple `suivant(5)` renvoie 16 et `suivant(16)` renvoie 8.
2. Écrire une fonction `lesTermes` qui prend comme arguments deux entiers naturels $a \geqslant 1$ et p et qui renvoie la liste $[u_0, u_1, \ldots, u_p]$.
   Par exemple, `LesTermes(5,10)` renvoie `[5,16,8,4,2,1,4,2,1,4,2]`.
3. Tester la fonction `lesTermes` avec $p = 100$ et plusieurs valeurs de a. Que constate-t-on ?
   La conjecture de Syracuse affirme qu'une telle suite finit toujours par prendre la valeur 1. En dépit de la simplicité de son énoncé, cette conjecture défie depuis 1928 les mathématiciens.
4. Écrire une fonction `Temps` qui prend comme argument a et qui calcule la plus petite valeur de n telle que $u_n = 1$. Par exemple, `Temps(50)` renvoie 24.

Exercice 7: (Solution)

Soit n un entier naturel $n \leqslant 26$. On souhaite écrire un programme qui code un mot en décalant chaque lettre de l'alphabet de n lettres.
Par exemple pour $n = 3$, le décalage sera le suivant :

avant décalage	a	b	c	d	...	...	x	y	z
après décalage	d	e	f	g	...	...	a	b	c

Le mot `oralensam` devient ainsi `rudohqvdp`.

1. Définir la chaîne de caractères `alphabet='abcdefghijklmopqrstuvwxyz'`
2. Définir une fonction `positition` qui prend comme argument une lettre en minuscule et qui renvoie sa position dans `alphabet`. Par exemple, `position('d')` renvoie `3` ("indice python").
3. Écrire une fonction `decalage` qui prend comme arguments un entier n et une lettre minuscule et qui renvoie la lettre codée avec un décalge de n dans `alphabet`.
   Par exemple,

— `decalage(3,'o')` renvoie 'r'
— `decalage(3,'y')` renvoie 'b'

4. Écrire une fonction `codage` qui prend comme arguments un entier naturel n et une chaîne de caractères phrase et renvoyant la phrase codée où tous les caractères appartenant à `alphabet` sont décalés de n unités, les autres étant inchangés. Par exemple, `codage(3,'TEST : oralensam')` renvoie 'TEST : rudohqvdp'.
5. Comment peut-on décoder un mot codé ?

Exercice 8: (Solution)

On appelle nombre d'Armstrong un entier naturel qui est égal à la somme des cubes de ses chiffres en écriture décimale. Par exemple, $153 = 1^3 + 5^3 + 3^3$ est un nombre d'Armstrong.

1. Soit $n = 1234$. Quel est le quotient, noté q, dans la division euclidienne de n par 10 ? Quel est le reste ? Que se passe-t-il si on recommence la division par 10 à partir de q ?
2. Écrire une fonction `somCube` qui prend pour argument n, qui renvoie la somme des cubes des chiffres de l'entier n en utilisant les observations de la question précédente.
   Par exemple, `somCube(123)` renvoie $36 = 1^2 + 2^3 + 3^3$.
3. Écrire une fonction `ListeArmstrong` qui prend comme argument n et qui renvoie la liste des nombres d'Armstrong compris entre 1 et n.
   Tester avec $n = 1000$.
4. En modifiant les instructions de la fonction `somCube`, écrire une fonction `somCube2` qui convertit l'entier n en une chaîne de caractères permettant ainsi la récupération de ses chiffres sous forme de caractères. Cette fonction renvoie toujours la somme des cubes des chiffres de l'entier n.

Exercice 9: (Solution)

On raconte qu'il y a environ 14 siècles, un Prince de Perse voulait récompenser l'inventeur du jeu des échecs. Ce dernier lui demanda le cadeau suivant : 1 grain de blé sur la première case de l'échiquier, 2 sur la deuxième, 4 sur la troisième et ainsi de suite en doublant jusqu'à la 64 ème case.
Sachant qu'un grain de blé pèse en moyenne un quart de gramme, combien de temps aurait-il fallu pour satisfaire le vœu de l'inventeur au rythme actuel de la production mondiale de blé qui est d'environ 800 millions de tonnes par an ?

Exercice 10: (Solution)

Écrire une fonction `comptage(L)` d'argument une liste de nombres et renvoyant un dictionnaire dont les clés sont les éléments de `L` et les valeurs sont leurs occurrences dans la liste `L`.

Exercice 11: (Solution)

Écrire une fonction `recherche(L,x)` d'arguments une liste `L` et un élément `x` et renvoyant un booléen indiquant si l'élément `x` appartient à `L` ou non.

Exercice 12: (Solution)

1. Écrire une fonction `maximum(L)` renvoyant le maximum de la liste `L`.
2. Écrire une fonction `premierMax(L)` renvoyant la position de la première occurence du maximum dans la liste `L`.
   Par exemple `premierMax([0,3,2,1,3,2])` renvoie 1.
3. Écrire une fonction `dernierMax(L)` renvoyant la position de la dernière occurence du maximum dans la liste `L`.
   Par exemple `dernierMax([0,3,2,1,3,2])` renvoie 4.
4. Écrire une fonction `lesMax(L)` renvoyant la liste des position de chaque occurence du maximum dans la liste `L`.
   Par exemple `lesMax([0,3,2,1,3,2])` renvoie `[1,4]`.