

Devoir Surveillé n°2 d'Informatique

Samedi 10 Janvier 2026 (*Durée : 2 heures*)

Exercice 1

Soit $n \in \mathbb{Z}$. On compte calculer 2^n à l'aide de la fonction suivante :

```
def f(n):  
    c=n  
    p=1  
    while c!=0:  
        p=p*2  
        c=c-1  
    return p
```

- 1) On tape $f(3)$. Qu'obtient-on?
On dressera le tableau d'exécution de la fonction pour vérifier que tout marche bien.
- 2) Que se passe-t-il si l'on tape $f(-4)$?
- 3) Modifiez la fonction afin qu'elle renvoie 2^n quel que soit le signe de l'entier n .

Exercice 2

Soit x un nombre de l'intervalle $]0;1[$. On suppose que son écriture binaire est finie.

- 1) On dispose d'une chaîne de caractères ch contenant la représentation de x en base 2.
Écrire un programme qui calcule et affiche alors x .

Exemple : si $ch = '0.101'$, on doit obtenir $x=0.625$.

- 2) Réciproquement, on dispose en mémoire de la variable x .
Écrire un programme qui calcule et affiche la chaîne ch contenant sa représentation en base 2.

Exemple : si $x=0.375$, on doit obtenir $ch='0.011'$.

Exercice 3

- 1) Écrire une procédure $\text{Tableur}(f, a, b, n)$ qui renvoie la liste des valeurs prises par une fonction f en n points régulièrement espacés de l'intervalle $[a; b]$.
- 2) On définit la fonction $f : x \mapsto 2x^3 + x - 5$ sur $\mathbb{R}$.
Utiliser la procédure Tableur pour afficher la liste $[f(0), f(0.1), f(0.2), \dots, f(1)]$.

Exercice 4

Écrire une fonction `Somme_Inverses(n)` qui renvoie la somme $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$.

Remarque : on partira du principe que n appartient à $\mathbb{N}^*$, sans le vérifier dans le programme.

Exercice 5

On dispose en mémoire d'une variable entière $p > 0$.

On exécute alors le programme suivant :

```
c=0
while p>0:
    if c==0:
        p=p-2
        c=1
    else:
        p=p+1
        c=0
```

- 1) Dressez le tableau d'exécution de ce programme dans le cas où $p = 5$.
- 2) On désire montrer que la boucle `while` se termine toujours.
Pour cela, prouvez que $v = 2p + 3c$ est un variant de boucle.

Exercice 6

- 1) Écrire *en français* un algorithme qui calcule le produit des entiers de 1 à 99.
- 2) Traduire cet algorithme en langage Python.

Exercice 7

On considère la fonction suivante :

```
def fonction(debut=100):
    i = debut
    while i >= 1:
        i = i-1
    return i
```

- 1) Justifier que l'appel `fonction()` se termine en exhibant un variant de boucle.
- 2) Quelle est la complexité de `fonction(N)` ?

Exercice 8

On considère la fonction suivante :

```
def fonction(n):  
    i = 1  
    while i < n:  
        i = i*2  
    return i
```

- 1) Montrer que la proposition $\mathcal{P}(k)$: « à l'issue du k^e passage dans la boucle, on a $i = 2^k$ » est un invariant de boucle.
- 2) Utiliser cet invariant pour expliquer ce que fait cette fonction.

Exercice 9

On considère la fonction suivante, où L est une liste de nombre réels :

```
def fonction(L):  
    i , n = 0, len(L)  
    while i+1 < n and L[i]<L[i+1]:  
        i = i+1  
    return i
```

- 1) Justifier que l'exécution de cette fonction se termine.
- 2) Donner sa complexité en fonction de $n=\text{len}(L)$.
- 3) Montrer que la proposition $\mathcal{P}(i)$: « à l'issue du i^e passage dans la boucle, on est sûrs que $L[0] < L[1] < \dots < L[i]$ » est un invariant de boucle.
- 4) Que fait donc cette fonction ?

Exercice 10

On définit la suite de Héron¹ par $u_0 = 1$ et $u_{n+1} = \frac{u_n}{2} + \frac{1}{u_n}$ pour $n \in \mathbb{N}$.

- 1) Écrire une fonction `SuiteU(n)` qui renvoie la liste $[u_0, u_1, \dots, u_n]$.
- 2) Écrire une fonction `MoyenneU(n)` qui renvoie la quantité $\frac{u_0 + u_1 + \dots + u_n}{n+1}$.

1. Héron d'Alexandrie, mathématicien grec contemporain de Pline l'Ancien ... qui n'a évidemment aucun rapport avec les petits patapons.