

Première NSI (A, B, & C)	Épreuve blanche d'informatique n° 1 Durée : 3h30	novembre 2023
--------------------------	---	---------------

**Documents et calculatrice non autorisée**

## Exercice 1

1. Écrire les instructions permettant d'afficher les séquences suivantes, sous la forme de votre choix (affichage à l'écran, liste, tuple, ...) \_\_\_\_\_/ 4 p.
  - a) La liste des 25 premiers multiples de 3.
  - b) Une suite de 50 éléments alternés 1, 2
  - c) Les entiers relatifs de -12 à 6.
2. Écrire une fonction qui renvoie le minimum d'une liste ou d'un tuple passé en argument. On n'utilisera pas la fonction native `min`.  
Fournir des exemples d'utilisation de la fonction.
3. Écrire une fonction qui renvoie la position du maximum d'une liste ou d'un tuple.  
Fournir des exemples d'utilisation de la fonction.
4. Écrire une fonction qui renvoie la moyenne d'une liste ou d'un tuple d'éléments numérique.  
Fournir des exemples d'utilisation de la fonction.

## Exercice 2

1. Donner, en écrivant les calculs, \_\_\_\_\_/ 3 p.
  - a) l'écriture binaire de 181.
  - b) l'écriture décimale du nombre  $(10011111)_2$
2. Écrire, en langage algorithmique, en pseudo code ou en Python, l'algorithme permettant de transformer un nombre écrit en base 10 en sa représentation en base 2.

## Exercice 3

1. On donne le code HTML suivant : \_\_\_\_\_/ 3 p.

```

1 <html>
2   <head>
3     <titre>Une page à corriger</titre>
4     <meta charset="utf-8" />
5   </head>
6   <body>
7     <titre>Chapitre à corriger</titre>
8     <p>Dans cette page, il faut corriger certaines balises qui sont
9     <span style="text-color:red ;">incorrectes</span> :
10    <li>balises</li>
11    <ul>incorrectes</ul>
12    <a text="La Madone">https://www.lamadone.fr/</a>
13  </body>

```

Corriger ce code HTML en indiquant sur votre copie le numéro de la ligne et les corrections apportées.

2. Expliquer la différence entre HTML et CSS.

## Exercice 4

L'objectif de ce problème est de mettre en place une modélisation d'un jeu de labyrinthe avec le langage Python.

\_\_\_\_\_/10 p.

On décide de représenter un labyrinthe par un tableau carré de taille  $n = 10$ , dans lequel les cases contiendront 0 si on peut s'y déplacer et 1 s'il s'agit d'un mur. Voici un exemple de représentation d'un labyrinthe.



```
laby = [[0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

1. Proposer un dessin de labyrinthe simple de taille  $n = 6$  et sa représentation associée sous forme de liste de liste.
2. Compléter la fonction `mur` qui prend comme paramètres un tableau représentant un labyrinthe et deux entiers  $i$  et  $j$  compris entre 0 et  $n - 1$  et qui renvoie un booléen indiquant la présence ou non d'un mur.

Exemple :

```
>>> mur(laby, 2, 3)
```

```
True
```

```
>>> mur(laby, 1, 8)
```

```
False
```

```
1 def mur(laby, i, j)
2     if laby[i][j] ...:
3         return ...
4     else:
5         return ...
```

Un parcours de labyrinthe va être représenté par une liste de cases. Il s'agit de couples  $(i, j)$  où  $i$  et  $j$  correspondent respectivement aux numéros de ligne et colonne des cases successivement visitées au long du parcours. Ainsi, la liste suivante  $[(1,4), (1,5), (1,6), (2, 6), (3,6), (3,5), (3,4)]$  correspond au parcours repéré par des étoiles.

