NSI *Première* -> Terminale

Le programme de terminale est très chargé avec de nombreuses notions à voir (en particulier jusqu'à la mi-mars, date des épreuves écrites et pratiques)…

Nous vous proposons donc de reprendre et chercher quelques exercices de programmation en Python (fin août), afin d'être opérationnel dès la reprise.

- ➔ La correction de quelques exercices sera donnée à partir du lundi 23 août 2021 sur le site du lycée
- ➔ D'autres seront corrigés lors des premières séances de NSI
- ➔ **L'exercice 9** est à rendre lors de la première séance de NSI

Ces notions seront évaluées à l'occasion d'une épreuve pratique la semaine du 6 septembre.

Bonnes vacances à toutes et à tous !

Exercice 1

Objectif : *print | input | boucle*

1. Pour aider le fils de votre voisin qui est en CE1, écrire un script qui affiche la table de multiplication de n, où n est choisit par l'utilisateur.
Attention : n doit être compris entre 0 et 10 (compris)
Exemple : si on choisit n = 6, alors on obtient en console ➔

```
*** Console ***
>>>
6 * 0 = 0
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
>>>
```

2. Finalement, le petit voisin a besoin de toutes les tables de multiplication de 0 jusqu'à 10.
Écrire une autre version du script.

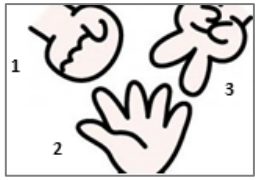
3. Apprendre par cœur les tables de multiplication 😊

Exercice 2

Objectif : *structure conditionnelle | input*

Réaliser le programme du jeu « pierre, feuille, ciseaux » :

- le joueur joue contre l'ordinateur.
- il peut faire 1, 2 ou 3 comme choix.



Exercice 3

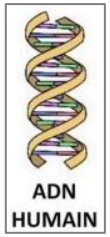
Objectif : *boucle | spécification, assertion | jeu de tests*

Écrire une fonction qui prend en argument un entier et qui renvoie combien de fois de suite cet entier est divisible par 2.

➔ On créera la docstring, des assertions en précondition et postcondition et un jeu de tests.

Exercice 4

Objectif : *chaîne de caractères | Recherche d'un élément*



Il s'agit d'écrire un programme qui demande de saisir une chaîne d'ADN valide et une séquence d'ADN valide (*valide* signifie qu'elles ne sont pas vides et sont formées exclusivement d'une combinaison arbitraire de "a", "t", "g" ou "c") et de dénombrer le nombre de séquences dans la chaîne.

- Écrire une fonction `valide` qui renvoie vrai si la saisie est valide, faux sinon.
- Écrire une fonction `saisie` qui effectue une saisie valide et renvoie la valeur saisie sous forme d'une chaîne de caractères.
- Écrire une fonction `nombre` qui reçoit deux arguments, la *chaîne* et la *séquence* et qui retourne le nombre de *séquence* dans la *chaîne*.
- Le programme principal appelle la fonction `saisie` pour la chaîne et pour la séquence et affiche le résultat.

Exemple d'affichage :

```
chaîne : attgcaatggtggtacatg
séquence : ca
Il y a 2 séquences "ca" dans la chaîne.
```

Exercice 5

Objectif : *liste*

Il s'agit d'écrire, d'une part, un programme principal et, d'autre part, deux fonctions utilisées dans le programme principal.

- La fonction `listAleaInt(n,a,b)` retourne une liste de n entiers aléatoires dans [a ; b] en utilisant la fonction `randint(a,b)` du module `random`.
- La fonction `mini_premierePosition(liste)` qui permet d'échanger le premier élément du tableau avec son minimum.

Prévoir un jeu de tests pour cette fonction ⚠

- Dans le programme principal :
  construire une liste en appelant la fonction `listAleaInt()` ;
  puis l'appliquer à la fonction `mini_premierePosition(liste)`

Sommaire

Exercice 6

Objectif : *listes | pop | append*

1. Préliminaire. On rappelle que :

- l'expression `T1 = list(T)` fait une copie de T indépendante de T,
- l'expression `x = T.pop()` enlève le sommet de la liste T et le place dans la variable x
- l'expression `T.append(v)` place la valeur v au sommet de la liste T.

Tester ⬇, éventuellement, en console les instructions suivantes :

Console

```
>>> T = [1, 2, 3]
>>> T2 = list(T)
>>> x = T2.pop()
>>> x
...
>>> T2.append(4)
>>> T2
...
>>> T
...
```

2. Compléter le code Python de la fonction positif ci-contre ➡

qui prend une liste T de nombres entiers en paramètre et qui renvoie la liste des entiers positifs dans le même ordre, sans modifier la variable T.

```
def positif(T):
    T2 = ...(T)
    T3 = ...
    while T2 != []:
        x = ...
        if ... >= 0:
            T3.append(...)
    T2 = []
    while T3 != ... :
        x = T3.pop()
        ...
    print('T =', T)
    return T2
```

Exemple ⬇

Console

```
>>> positif([-1, 0, 5, -3, 4, -6, 10, 9, -8])
T = [-1, 0, 5, -3, 4, -6, 10, 9, -8]
[0, 5, 4, 10, 9]
>>>
```

Exercice 7

Objectif : *dictionnaires*

Un professeur de NSI décide de gérer les résultats de sa classe sous la forme d'un dictionnaire :

- les clefs sont les noms des élèves .
- les valeurs sont des dictionnaires dont les clefs sont les types d'épreuves et les valeurs sont les notes obtenues associées à leurs coefficients.

Avec :

```
resultats = {'Dupont':{'DS1' : [15.5, 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [13, 4],
                       'PROJET1' : [16, 3],
                       'DS3' : [14, 4]},
             'Durand':{'DS1' : [6 , 4],
                       'DM1' : [14.5, 1],
                       'DS2' : [8, 4],
                       'PROJET1' : [9, 3],
                       'IE1' : [7, 2],
                       'DS3' : [8, 4],
                       'DS4' :[15, 4]}}
```

L'élève dont le nom est Durand a ainsi obtenu au DS2 la note de 8 avec un coefficient 4.

Le professeur crée une fonction `moyenne` qui prend en paramètre le nom d'un de ces élèves et lui renvoie sa moyenne arrondie au dixième.

Compléter le code du professeur ci-dessous :

```
def moyenne(nom):
    if nom in ...:
        notes = resultats[nom]
        total_points = ...
        total_coefficients = ...
        for ... in notes.values():
            note , coefficient = valeurs
            total_points = total_points + ... * coefficient
            total_coefficients = ... + coefficient
        return round( ... / total_coefficients , 1 )
    else:
        return -1
```

Exercice 8

Objectif : *écriture binaire / décimale d'un entier*

1. On modélise la représentation binaire d'un entier non signé par un tableau d'entiers dont les éléments sont 0 ou 1.

Par exemple, le tableau [1, 0, 1, 0, 0, 1, 1] représente l'écriture binaire de l'entier 1010011_2 dont l'écriture décimale est 2**6 + 2**4 + 2**1 + 2**0 = 83.

À l'aide d'un parcours séquentiel, écrire la fonction `convertir` répondant aux spécifications suivantes ⬇

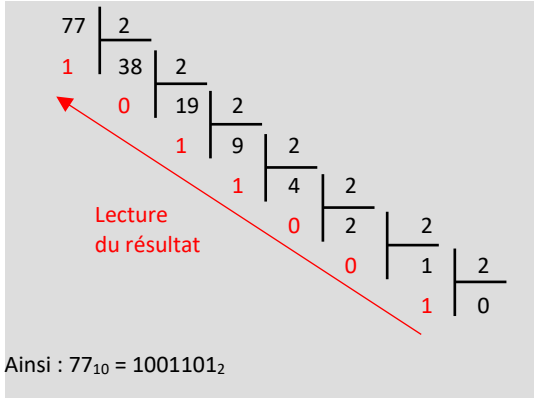
```
def convertir(T):
    """T est un tableau d'entiers, dont les éléments sont 0 ou 1 et
       représentant un entier écrit en binaire.
    Renvoie l'écriture décimale de l'entier positif dont la représentation
    binaire est donnée par le tableau T"""
```

Exemple ➡

Console

```
>>> convertir([1, 0, 1, 0, 0, 1, 1])
83
>>> convertir([1, 0, 0, 0, 0, 0, 1, 0])
130
```

2. Pour rappel, la conversion d'un nombre entier positif en binaire peut s'effectuer à l'aide des divisions successives comme illustré ici ⬇



Ainsi : $77_{10} = 1001101_2$

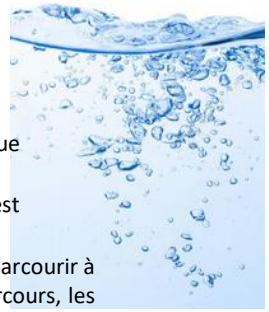
```
def binaire(a):
    bin_a = [...]
    a = a//2
    while a ... :
        bin_a = ... + ...
        a = ...
    return bin_a
```

Voici une fonction python basée sur la méthode des divisions successives ⬆ permettant de convertir un nombre entier positif en binaire.

Compléter la fonction `binaire`.

BINARY:
it's as easy as
01, 10, 11

Exercice à rendre



Exercice 9 **Tri à bulles** -> *bubble sort ou sinking sort*

Objectif : *trier une liste*

Le principe du tri à bulles

- L'algorithme parcourt la liste et compare les éléments consécutifs. Lorsque deux éléments consécutifs ne sont pas dans l'ordre, ils sont échangés.
- Après un premier parcours complet de la liste, le plus grand élément est forcément en fin de liste, à sa position définitive.
- Le reste du tableau est en revanche encore en désordre. Il faut donc le parcourir à nouveau, en s'arrêtant à l'avant-dernier élément. Après ce deuxième parcours, les deux plus grands éléments sont à leur position définitive.
- Il faut donc répéter les parcours du tableau, jusqu'à ce que les deux plus petits éléments soient placés à leur position définitive (*pas d'échange sur un parcours*).

Exemple avec le tableau suivant : L = [5, 3, 1, 6, 2, 7, 4]

L Init :	**5**	**3**	1	6	2	7	4	
Passage 1	3	**5**	**1**	6	2	7	4	
	3	1	**5**	**6**	2	7	4	
	3	1	5	**6**	**2**	7	4	
	3	1	5	2	**6**	**7**	4	
	3	1	5	2	6	**7**	**4**	
Pssg 2	**3**	**1**	5	2	6	4	7	
	1	**3**	**5**	2	6	4	7	
	1	3	**5**	**2**	6	4	7	
	1	3	2	**5**	**6**	4	7	
	1	3	2	5	**6**	**4**	7	
	1	3	2	5	4	**6**	**7**	
Pssg 3	**1**	**2**	3	5	4	6	7	
	1	**2**	**3**	5	4	6	7	
	1	2	**3**	**5**	4	6	7	
	1	2	3	**5**	**4**	6	7	
	1	2	3	4	**5**	**6**	**7**	etc

Plus de changement…

1. Écrire une fonction prenant pour argument la liste à trier et qui renvoie la liste triée par la méthode du tri à bulles.

[Une optimisation est possible afin d'éviter les derniers passages si la liste est déjà triée]

2. Quelle est le coût de cette méthode ?