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 21 août 2023 sur le site du lycée
- ➔ D'autres seront corrigés lors des premières séances de NSI
- ➔ **Les exercices 9 et 10** sont à rendre lors de la première séance de NSI

Ces notions seront évaluées, notamment à l'occasion d'une épreuve pratique, la semaine du 4 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.

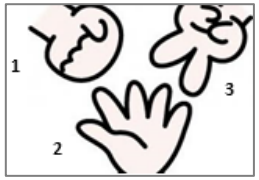
3. Apprendre par cœur les tables de multiplication 😊

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 non nul 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.

Exemples

```
>>> nb_de_div_par_2(24)
3
```

```
>>> nb_de_div_par_2(5)
0
```

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 prend en argument une chaine de caractère et qui renvoie `True` si la saisie est valide, `False` sinon.

Exemples

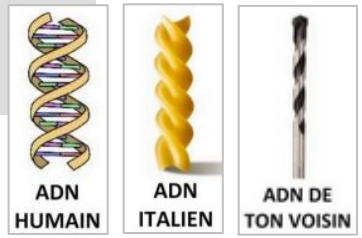
```
>>> valide("agtcc")
True
```

```
>>> valide("accbtg")
False
```

- Écrire une fonction `nombre` qui reçoit deux arguments, la *chaîne* et la *séquence* et qui retourne le nombre de *séquence* dans la *chaîne* [ne pas oublier la documentation et les assertions].

Exemple :

```
>>> chaine = "attgcaatcgtggtacatgc"
>>> sequence = "ca"
>>> nombre(chaine, sequence)
2
```



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)`

Exemples

```
>>> liste = listAleaInt(5,1,6)
>>> liste
[5,3,4,1,3]
>>> mini_premierePosition(liste)
>>>liste
[1,3,4,5,3]
```

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

```
>>> 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*

Partie 1

On considère le dictionnaire suivant :

```
mondict = {"device": "laptop" , "constructeur": "acer" , "ram": "8G" ,
           "processeur": "Intel core i5", "stockage": "1 T"}
```

Écrire les inscriptions permettant :

- de corriger l'erreur `"stockage": "2 T"`
- d'afficher ➡ la liste des clés,
  ➡ la liste des valeurs
  ➡ la liste des paires de clés et valeurs
- d'ajouter la pair clé-valeur : `"Système d'exploitation" : "Windows 11"`

Partie 2

Écrire une fonction Python qui prend en paramètre un entier n et qui renvoie un dictionnaire formé des entiers de 1 à n et de leurs carrées.

Exemple

Pour n = 7 le dictionnaire sera : `{1: 1, 2: 4, 3: 9, 4: 16, 5: 25 , 6: 36 , 7: 49}`

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.

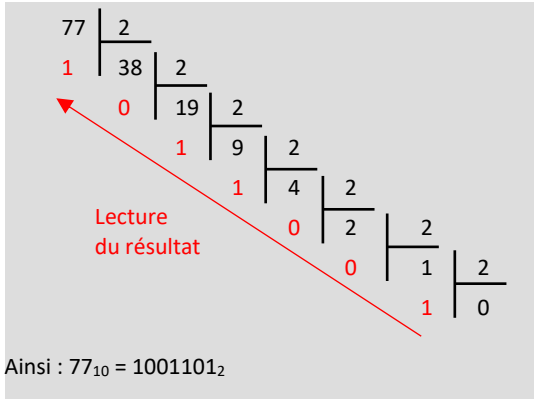
À l'aide d'un parcours séquentiel, écrire la fonction `convertir` répondant aux spécifications suivantes ⬇

```
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

```
>>> 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`.

Exercice à rendre

Exercice 9

Objectif : *implémenter en Python un algorithme*

Écrire les fonctions en Python les trois algorithmes du cours :

tri par sélection, tri par insertion et recherche par dichotomie

Exercice 10

Objectif : *dictionnaire*

Le but de l'exercice consiste à écrire une fonction en Python `nb_occurrences` qui prend en paramètre une chaine de caractère, et qui renvoie un dictionnaire dont les clés sont les caractères de la chaine et les valeurs sont les nombres d'occurrences des caractères dans la chaine.

Exemple

```
>>> chaine = "vacances"
>>> nb_occurrences(chaine)
{'n': 1, 'e': 1, 'v': 1, 'c': 2, 'a': 2, 's': 1}
```

Compléter le code de cette fonction.

```
def nb_occurrences(chaine):
    dico =
    for caractere in
        if dico.get(caractere) !=
            dico[caractere] =
        else:

    return dico
```