Interrogation d'ITC n°1

Semaine du 22/09/2025

Durée : 30 minutes

Nom :		Prénom :	

Consignes

- Les codes doivent être présentés en mentionnant explicitement l'indentation, au moyen par exemple de barres verticales sur la gauche.
- Pour qu'un code soit compréhensible, il convient de choisir des noms de variables les plus explicites possibles.
- La performance du code proposé à chaque question entrera dans l'évaluation, de-même que l'utilisation de boucles appropriées.
- Vous pouvez bien sûr utiliser une feuille de brouillon en parallèle.

/20

Exercice 1 Manipulation de listes [Sol 1]

On considère la liste :

```
L = [6, 7.0, [1, 2, 3], -3]
```

1.
- /0.5 Que renvoie `int(L[1])` ?
- /0.5 Que renvoie `L[-2][-1]` ?
- /0.5 Que renvoie `len(L[-2])` ?
- /0.5 Que vaut `len(L)` ?
- /1 Que renvoie `L[2:][1]` ? *On expliquera le résultat.*

2. Comment, avec une ou plusieurs commandes Python, pouvez-vous :
- /1 faire en sorte que `L` devienne `[6, 7.0, [1, 3], -3]` ?
- /1 faire en sorte que `L` devienne `[6, 7.0, [1, 2, 3, 4], -3]` ? *on considère ici que `L` est à nouveau la liste initiale*
- /1 faire en sorte que `L` devienne `[6, [1, 2, 3]]` ? *on considère ici que `L` est à nouveau la liste initiale*

Exercice 2 **Suites** [Sol 2]

1. /3 Écrire le script d'une fonction `u(n:int)->int` qui renvoie le n-ième terme de la suite (u_n) définie par :

$$u_0 = 1, \quad \forall n \in \mathbb{N}, \quad u_{n+1} = 2u_n + 3.$$

2. /3 Écrire de même le script d'une fonction `v(n:int)->int` qui renvoie de n-ième terme de la suite (v_n) définie par :

$$v_0 = 1, \quad \forall n \in \mathbb{N}, \quad v_{n+1} = v_n^2 + n.$$

Exercice 3 **Étude de la 2-valuation d'un entier** [Sol 3]

Pour tout entier $n \geq 1$, on désigne par $v(n)$ le plus grand entier k tel que :

$$2^k \text{ divise } n, \quad \text{c'est-à-dire tel que : } \quad \frac{n}{2^k} \in \mathbb{N}.$$

On a ainsi : $v(2^3) = 3$, $v(12) = 2$, $v(3) = 0$.

1. On considère la fonction suivante :

```
1 def v(n:int)->int:
2     a = 1
3     k = 0
4     while n % a == 0 and a <= n:
5         k = k+1
6         a = 2*a
7     return k
```

1.1) /2 Compléter le tableau ci-après, détaillant l'évolution des variables `a` et `k` lors de l'appel `v(8)`.

i	a_i	k_i
0	1	0

Remarques : le tableau n'a pas forcément le bon nombre de lignes ... et i désigne le numéro de l'itération avec pour convention que $i = 0$ correspond à l'initalisation.

1.2) /2 Recopier et modifier le programme précédent afin qu'il renvoie la valeur de $v(n)$.

2. [4] Soit $N \in \mathbb{N}^\star$. On souhaite trouver la valuation maximale parmi les entiers de $[\![1\,,\,N]\!]$, c'est-à-dire $\max_{n\in[\![1\,,N]\!]} v(n)$. Compléter le code ci-après pour qu'il renvoie cet entier.

```
def max_val(N : int) -> int:
    maxi = ...........
    for k in range(.......) :
        if .............. :
            maxi = ..............
    return maxi
```

SOLUTIONS DES EXERCICES

Solution 1

```
>>> L = [6, 7.0, [1, 2, 3], -3]
>>> # QUESTION 1
>>> int(L[1]) # 2ème converti en entier
7
>>> L[-2][-1] # car L[-2] = [1, 2, 3]
3
>>> len(L[-2]) # longueur de [1, 2, 3]
3
>>> len(L)
4
>>> L[2:][1] # car :
-3
>>> L[2:] # on regarde donc le 2ème élément de cette liste
[[1, 2, 3], -3]
>>> # QUESTION 2
>>> del L[2][2]
>>> L
[6, 7.0, [1, 2], -3]
>>> L = [6, 7.0, [1, 2, 3], -3]
>>> L[2].append(4)
>>> L
[6, 7.0, [1, 2, 3, 4], -3]
>>> L = [6, 7.0, [1, 2, 3], -3]
>>> L = L[::2]
>>> L
[6, [1, 2, 3]]
```

Solution 2

1.
```
def u(n:int)->int:
    u = 1
    for k in range(1, n+1):
        u = 2*u+3
    return u
```

```
>>> u(0)
1
>>> u(4)
61
```

2.
```
def v(n:int)->int:
    v = 1
    for k in range(1, n+1):
        v = v**2 + k-1
    return v
```

```
>>> v(0)
1
>>> v(4)
39
```

Solution 3

1. 1.1)

i	a_i	k_i
0	1	0
1	2	1
2	4	2
3	8	3
4	16	4

a correspond aux puissances de 2 que l'on teste comme diviseur de n, et k la puissance associée. En fin de boucle, on peut observer que k vaut v(8)+1, on retourne donc plutôt k-1

1.2)
```
1 def v(n:int)->int:
2     a = 1
3     k = 0
4     while n % a == 0 and a <= n:
5         k = k+1
6         a = 2*a
7     return k-1
```

```
>>> v(8)
3
>>> v(7)
0
>>> v(1)
0
```

Autre possibilité.

```
def v(n:int)->int:
    a = 1
    k = -1
    while n % a == 0 and a <= n:
        k = k+1
        a = 2*a
    return k
>>> v(8)
3
>>> v(7)
0
>>> v(1)
0
```

Autre possibilité (mais peut-être moins intuitive) :

```
def v(n:int)->int:
    a = 2
    k = 0
    while n % a == 0 and a <= n:
        k = k+1
        a = 2*a
    return k
>>> v(8)
3
>>> v(7)
0
>>> v(1)
0
```

2.

```
def max_val(N : int) -> int:
    maxi = v(1)
    for k in range(2, N+1) :
        if v(k) >= maxi:
            maxi = v(k)
    return maxi
>>> max_val(10)
3
```