Polynésie – 2026 – sujet1 - Correction

Exercice 1 (6 points)

1. Fonction récursive suite

```
def suite(n):
    if n < 2:
        valeur = n
    else:
        valeur = suite(n - 1) + suite(n - 2)
    return valeur
```

👉*Commentaire : pour n = 0 et n = 1, U_n vaut n : c'est le cas de base.*

2. Fonction nb_additions_suite

```
def nb_additions_suite(n):
    if n < 2:
        return 0
    else:
        return 1 + nb_additions_suite(n - 1) + nb_additions_suite(n - 2)
```

3. Valeurs de p(2), p(4) et p(7)

n	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p(n)	0	0	**1**	2	**4**	7	12	**20**

p(2) = 1 + 0 + 0 = **1**, p(4) = 1 + p(3) + p(2) = 1 + 2 + 1 = **4**, p(7) = 1 + p(6) + p(5) = 1 + 12 + 7 = **20**.

👉*Commentaire : p(n) + 1 suit la même relation que la suite : p(n) = U(n+1) − 1. Le nombre d'additions croît donc de façon exponentielle.*

4. Nombre d'additions de suite2

n	nombre d'additions
2	1
3	2
4	3
5	**4**
6	**5**
7	**6**

Chaque terme de U_2 à U_n n'est calculé qu'une fois, avec une seule addition : **q(n) = n − 1**.

👉*Commentaire : attention à la valeur par défaut valeurs_calculees = [0, 1] : elle est créée une seule fois et conservée d'un appel à l'autre. Si l'on appelle suite2(5) puis suite2(7), le second appel ne fait que 2 additions. Le tableau correspond à des appels faits avec une liste neuve (on a vérifié ces valeurs en Python).*

5. Pourquoi suite2 est plus efficace

suite recalcule de nombreuses fois les mêmes termes (suite(5) appelle suite(3) deux fois, suite(2) trois fois…) : son coût est exponentiel. suite2 **mémorise** chaque terme calculé dans la liste valeurs_calculees et le réutilise : chaque terme est calculé une seule fois, le coût devient linéaire. Ce procédé s'appelle la **mémoïsation** (programmation dynamique).

6. Version itérative suite3

```
def suite3(n):
    a, b = 0, 1          # a = U_0, b = U_1
    for i in range(n):
        a, b = b, a + b
    return a
```

👉*Commentaire : après k tours de boucle, a vaut U(k) et b vaut U(k+1). suite3(5) renvoie 5.*

7. Pourquoi préférer la version itérative

La version itérative utilise une **mémoire constante** (deux variables) et fait n additions. Les versions récursives empilent un appel par terme : pour n grand, on dépasse la **limite de récursion** de Python (environ 1 000 appels) et le programme s'arrête avec une erreur RecursionError. De plus, chaque appel de fonction coûte du temps et de la mémoire.