Exercice 1

4 points

1) Avec $n = 0$ puis $n = 1$ dans la relation de récurrence :

$$u_1 = \frac{1}{5}u_0 + \frac{1}{3^0} = \frac{8}{5} + 1 = \boxed{\frac{13}{5}}$$

$$u_2 = \frac{1}{5}u_1 + \frac{1}{3^1} = \frac{13}{25} + \frac{1}{3} = \frac{39}{75} + \frac{25}{75} = \boxed{\frac{64}{75}}$$

2) **a.** Pour $n = 2$, `range(1,2)` ne contient que la valeur 1 : la boucle est exécutée une seule fois, avec `i = 1`. La variable `u` part de 8 et devient

$$\frac{1}{5} \times 8 + \frac{1}{3^1} = \frac{8}{5} + \frac{1}{3} = \frac{24}{15} + \frac{5}{15} = \frac{29}{15}.$$

`terme(2)` renvoie donc $\boxed{\frac{29}{15}}$ (Python affiche la valeur approchée `1.9333333333333333`).

b. On devrait obtenir $u_2 = \frac{64}{75}$ et non $\frac{29}{15}$: la fonction ne convient pas, pour deux raisons.

- `range(1,n)` ne donne que $n - 1$ passages dans la boucle, alors qu'il faut appliquer n fois la relation de récurrence pour passer de u_0 à u_n (par exemple `terme(1)` renvoie $8 = u_0$ au lieu de u_1).
- Pour calculer u_{k+1} à partir de u_k, on ajoute $\frac{1}{3^k}$, où k est l'indice du terme *précédent* : le premier passage doit utiliser 3^0, pas 3^1.

Il suffit de faire varier `i` de 0 à $n - 1$: au passage numéro `i`, `u` contient u_i et reçoit $u_{i+1} = \frac{1}{5}u_i + \frac{1}{3^i}$.

```
1  def terme(n):
2      u=8
3      for i in range(n):
4          u=1/5*u+1/(3**i)
5      return(u)
```

Vérification : `terme(1)` renvoie `2.6` $= \frac{13}{5}$ et `terme(2)` renvoie `0.8533...` $= \frac{64}{75}$.

Exercice 2

2 points

Les n premiers termes de la suite sont $u_0, u_1, \ldots, u_{n-1}$. Au départ, `U` contient $u_0 = 1$, `S` contient la somme $u_0 = 1$ et `K` vaut 0 : la variable `K` est l'indice du terme stocké dans `U`.

```
1  def somme(n):
2      U=1
3      S=1
4      K=0
5      while K<n-1:
6          U=2*U-K+1
7          S=S+U
8          K=K+1
9      return(S)
```

- Ligne 6 : `U` contient u_K et reçoit $u_{K+1} = 2u_K - K + 1$; c'est pourquoi `K` n'est augmenté qu'*après* ce calcul (ligne 8).
- Ligne 7 : on ajoute ce nouveau terme à la somme.
- Ligne 5 : la boucle s'arrête quand `K` vaut $n-1$; à ce moment `S` vaut $u_0 + u_1 + \cdots + u_{n-1}$, la somme des n premiers termes. La condition est donc `K<n-1`.

Vérification pour $n = 3$: $u_0 = 1$, $u_1 = 2 \times 1 - 0 + 1 = 3$, $u_2 = 2 \times 3 - 1 + 1 = 6$; la boucle tourne deux fois (`K` $= 0$ puis 1) et `somme(3)` renvoie $1 + 3 + 6 = 10$.

Attention : avec la condition `K<n`, la boucle tournerait une fois de trop et la fonction renverrait la somme des $n + 1$ premiers termes (`somme(3)` renverrait $1 + 3 + 6 + 11 = 21$).

Exercice 3

2 points

On part de $v_0 = 1$ et on applique n fois la relation $v_{k+1} = \dfrac{9}{6 - v_k}$ (on admet que $6 - v_k$ ne s'annule jamais).

```
1  def terme3(n):
2      v=1
3      for i in range(n):
4          v=9/(6-v)
5      return(v)
```

Vérification : $v_1 = \dfrac{9}{6-1} = \dfrac{9}{5}$ et $v_2 = \dfrac{9}{6 - \frac{9}{5}} = \dfrac{9}{\frac{21}{5}} = \dfrac{45}{21} = \dfrac{15}{7}$; `terme3(1)` renvoie `1.8` et `terme3(2)` renvoie `2.142857`…, soit $\dfrac{15}{7}$.

Exercice 4

2 points

À chaque étape, u_{n+1} *et* v_{n+1} se calculent tous deux à partir des *anciennes* valeurs u_n et v_n. Il faut donc faire une affectation simultanée (ou garder l'ancienne valeur de `u` dans une variable auxiliaire).

Affectation simultanée :

```
def terme4(n):
    u=2
    v=10
    for i in range(n):
        u,v=(2*u+v)/3,(u+3*v)/4
    return(u,v)
```

Avec une variable auxiliaire :

```
def terme4(n):
    u=2
    v=10
    for i in range(n):
        a=u
        u=(2*u+v)/3
        v=(a+3*v)/4
    return(u,v)
```

Vérification : $u_1 = \dfrac{2\times 2+10}{3} = \dfrac{14}{3}$ et $v_1 = \dfrac{2+3\times 10}{4} = 8$; `terme4(1)` renvoie bien `(4.666…, 8.0)`.

Erreur classique : écrire `u=(2*u+v)/3` puis `v=(u+3*v)/4` sans précaution. Le calcul de `v` utilise alors u_{n+1} au lieu de u_n : on obtiendrait

$$v_1 = \frac{\frac{14}{3}+30}{4} = \frac{26}{3} \quad \text{au lieu de } 8.$$