2026 sujet 3

Question 1

```
def est_bissextile(annee):
    return annee % 400 == 0 or (annee % 4 == 0 and annee % 100 != 0)
```

Question 2

```
def determiner_phase(n):
    assert n >= 1 and n <= 28, "le nombre doit être compris entre 1 et 28"

    if n <= 5:
        return 1
    elif n <= 13:
        return 2
    elif n == 14:
        return 3
    else:
        return 4
```

Question 3

On teste un changement d'année, une année non bissextile puis une année bissextile :

```
def test_ajouter_jours():
    assert ajouter_jours((7, 9, 2025), 3) == (10, 9, 2025)
    # Test de changement d'année
    assert ajouter_jours((31, 12, 2025), 1) == (1, 1, 2026)
    # Test d'ajout d'une année non bissextile
    assert ajouter_jours((15, 2, 2025), 365) == (15, 2, 2026)
    # Test d'ajout d'une année bissextile
    assert ajouter_jours((15, 2, 2024), 366) == (15, 2, 2025)
```

Question 4

Narrivant pas à installer le module `ics`, on peut quand même voir le résultat en commentant les lignes utilisant ce module :

```
def test_calendrier_cycles():
    '''Crée un calendrier et le charge avec le module ics pour vérifier sa
    validité.

    Nécessite que le module ics soit présent sur la machine (pip install ics).
    '''
    #from ics import Calendar
```

```
    c = calendrier_cycles( (12,3,2026) )
    print(c)
    #cal = Calendar(c)
    #print(cal.events)
```

Le résultat est alors :

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:
BEGIN:VEVENT
SUMMARY: Règles
DTSTART:2026312
END:VEVENT
BEGIN:VEVENT
SUMMARY: Règles
DTSTART:202649
END:VEVENT
BEGIN:VEVENT
SUMMARY: Règles
DTSTART:202657
END:VEVENT
END:VCALENDAR
```

Il manque certains zéros au début du mois et du jour, on ajoute `zfill(2)`
Il manque également la dernière date, il faut donc enlever le `+28` dans le `while`.
Le code final est donc :

```
def calendrier_cycles(date_regles):
    """Renvoie une chaîne de caractère contenant au format iCalendar, l'ensemble
    des dates de début de règles qui se présentent dans les 100 jours suivants
    `date_regles`, date incluse.

    Hypothèse : cycle régulier de 28 jours. """

    cal_lignes = ['BEGIN:VCALENDAR', 'VERSION:2.0', 'PRODID:Règles']

    date_courante = date_regles
    jours_ecoules = 0

    # On ajoute les dates tant que l'on ne dépasse pas 100 jours écoulés
    while jours_ecoules <= 100:
        jour, mois, annee = date_courante
        cal_lignes.append('BEGIN:VEVENT')
        date = str(annee)+str(mois).zfill(2)+str(jour).zfill(2)
        cal_lignes.append('DTSTART:'+date)
        cal_lignes.append('SUMMARY: Règles')
        cal_lignes.append('END:VEVENT')
        date_courante = ajouter_jours(date_courante, 28)
        jours_ecoules += 28
```

```
    cal_lignes.append('END:VCALENDAR')

    # La méthode join va renvoyer ici une unique chaîne contenant toutes les
    # chaînes de la liste séparées par des sauts de lignes.
    return '\n'.join(cal_lignes)
```

Et le résultat final est :

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:Règles
BEGIN:VEVENT
DTSTART:20260312
SUMMARY: Règles
END:VEVENT
BEGIN:VEVENT
DTSTART:20260409
SUMMARY: Règles
END:VEVENT
BEGIN:VEVENT
DTSTART:20260507
SUMMARY: Règles
END:VEVENT
BEGIN:VEVENT
DTSTART:20260604
SUMMARY: Règles
END:VEVENT
END:VCALENDAR
```