Épreuve pratique BAC NSI – Sujet 1 – Cycle menstruel

CORRECTION

1. Écrire en Python une fonction nommée `est_bissextile` qui prend en paramètre un entier correspondant à une année et qui renvoie un booléen indiquant si elle est bissextile, en appliquant la règle donnée ci-dessus.

```
###########################################################################
# Écrire le code de la fonction est_bissextile de la question 1           #
###########################################################################
def est_bissextile(annee) :  #--> par la négation
    if annee%4 != 0 : return False
    if annee%100 == 0 and annee%400 != 0: return False
    return True

def est_bissextile(annee) :  #--> par la définition
    if annee%4 == 0 :
        if annee%100 == 0 and annee%400 != 0 :
            return False
        return True
    return False

#tests
assert est_bissextile(2024) ,"Le test avec année = 2024 a échoué"
assert not est_bissextile(2010),"Le test avec année = 2010 a échoué"
assert est_bissextile(2000),"Le test avec année = 2000 a échoué"
```

2. Écrire en Python une fonction nommée `determiner_phase` qui prend en paramètre un entier (compris entre 1 et 28 inclus) qui correspond au jour d'un cycle et qui renvoie un entier correspondant au numéro de la phase associée. À l'aide d'une assertion, on garantira que l'entier donné en argument est compris entre 1 et 28 inclus.

```
###########################################################################
# Écrire le code de la fonction determiner_phase de la question 2         #
###########################################################################
def determiner_phase(jour) :
    assert jour >=1 and jour <= 28, 'le jour doit être compris entre 1 et 28'
    if jour <= 5 : return 1
    elif jour <= 13 : return 2
    elif jour <= 14 : return 3
    else : return 4

#tests
assert determiner_phase(2) == 1 , "Le test 2ième jour en phase 1 a échoué"
assert determiner_phase(7) == 2 , "Le test 7ième jour en phase 2 a échoué"
assert determiner_phase(14) == 3 , "Le test 14ième jour en phase 3 a échoué"
assert determiner_phase(20) == 4 , "Le test 20ième jour en phase 4 a échoué"
```

3. La fonction `ajouter_jours`, dont le code est déjà fourni, prend en paramètres une date et un entier représentant un nombre de jours. Elle renvoie la nouvelle date obtenue après ajout de ces jours.

   Compléter la fonction `test_ajouter_jours` en ajoutant au moins trois autres tests pertinents. Pour chaque test ajouté, une brève justification doit être donnée afin d'expliquer pourquoi ce cas est important à vérifier.

```
def test_ajouter_jours():
    assert ajouter_jours((7, 9, 2025), 3) == (10, 9, 2025) , "Le test 'sans changement de mois' a échoué"
    assert ajouter_jours((29, 9, 2025), 3) == (2, 10, 2025) , "Le test 'avec changement de mois' a échoué"
    assert ajouter_jours((28, 2, 2024), 3) == (2, 3, 2024) , "Le test 'avec changement de mois' en année bissextile a échoué"
    assert ajouter_jours((30, 12, 2025), 3) == (2, 1, 2026), "Le test 'avec changement d'année' a échoué"
```

4. La fonction `calendrier_cycles`, présente dans le fichier fourni, prend en paramètre une date correspondant au premier jour des dernières règles et renvoie la liste chronologique des dates de début de règles qui se présentent dans les 100 jours suivant cette date, date incluse, au format `iCalendar` sous la forme d'une chaîne de caractères.

   Observer avec la fonction `test_calendrier_cycles` que le calendrier renvoyé par la fonction `calendrier_cycles` n'est pas dans un format valide.

   Identifier le problème dans la fonction `calendrier_cycles`, proposer une démarche de résolution et la mettre en œuvre.

Code initial

```
while jours_ecoules + 28 <= 100:
    jour, mois, annee = date_courante
    cal_lignes.append('BEGIN:VEVENT')
    cal_lignes.append('SUMMARY: Règles')
    date = str(annee)+str(mois)+str(jour)
    cal_lignes.append('DTSTART:'+date)
    cal_lignes.append('END:VEVENT')
    date_courante = ajouter_jours(date_courante, 28)
    jours_ecoules += 28
```

Code modifié

```
while jours_ecoules + 28 <= 100:
    jour, mois, annee = date_courante
    cal_lignes.append('BEGIN:VEVENT')
    cal_lignes.append('SUMMARY: Règles')
    m = str(mois)
    j = str(jour)
    if mois <= 9 :
        m = '0'+str(mois)
    if jour <= 9 :
        j = '0'+str(jour)
    date = str(annee)+m+j
    cal_lignes.append('DTSTART:'+date)
    cal_lignes.append('END:VEVENT')
    date_courante = ajouter_jours(date_courante, 28)
    jours_ecoules += 28
```