Classe de Terminale NSI | **Devoir surveillé** | *Durée : 60 min + 20 min*

# Première partie (15 points)

## Exercice 1 (5 points)

On donne en annexe la définition d'une classe. Attention, deux erreurs se sont glissées dans le code.

1. Donner :
   - le nom de la classe définie en annexe,
   - la liste des attributs des instances de cette classe, en précisant leur type,
   - la liste des méthodes,
   - le rôle des lignes `3` à `8`,
   - une explication et une correction des deux erreurs de code présentes en annexe.
2. Écrire des lignes de code permettant de créer une instance `naissance_Turing` de sorte que la commande `print(naissance_Turing)` entraîne l'affichage suivant : `23 juin 1912 (Naissance d'Alan Turing)`

On suppose qu'on dispose d'une instance `naissance_Hopper` telle que la commande `print(naissance_Hopper)` entraîne l'affichage suivant : `9 décembre 1906 (Naissance de Grace Hopper)`

3. Écrire des lignes de code permettant de savoir, entre les naissances d'Alan Turing et de Grace Hopper, lequel des deux événements s'est produit en premier.
4. Écrire la définition d'une méthode permettant de savoir si deux événements ont eu lieu le même jour ou pas.

## Exercice 2 (4 points)

La suite de Syracuse est définie, à partir d'une valeur initiale entière strictement positive, de la façon suivante :

- Si un terme est pair, on le divise par deux pour obtenir le terme suivant,
- Si un terme est impair, on le multiplie par trois et on ajoute un pour obtenir le terme suivant.

Par exemple, à partir de la valeur initiale 13, on obtient successivement les valeurs 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, etc.

On donne la définition d'une fonction `duree_de_vol` :

```
 1  def duree_de_vol(n):
 2      """
 3      Rôle ?
 4      - Entrée : n (entier strictement positif)
 5      - Sortie : (entier)
 6      """
 7      if n == 1:
 8          return 0
 9      elif n % 2 == 0:
10          return 1 + duree_de_vol(n // 2)
11      else:
12          return 1 + duree_de_vol(3*n + 1)
```

1. Justifier que `duree_de_vol` est une fonction récursive.
   Identifier le (ou les) cas récursif(s) et le (ou les) cas de base.
2. L'appel `duree_de_vol(10)` renvoie la valeur 6.
   Dessiner l'arbre des appels récursifs provoqués par l'appel `duree_de_vol(10)`.
3. Expliquer le rôle de la fonction `duree_de_vol` dans le contexte de la suite de Syracuse.

### Exercice 3 (2 points)

On donne la définition d'une procédure `dessin` et le dessin obtenu suite à l'appel `dessin(200, 4)`.

Les procédures `forward`, `left` et `right` ont été importées du module `turtle`.

```
def dessin(longueur, n):
    forward(longueur/3)
    right(26.57)
    if n > 0:
        dessin(longueur*5**0.5/3, n-1)
    left(26.57)
    forward(2*longueur/3)
    for k in range(3):
        right(90)
        forward(longueur)
    right(90)
```

1. Déterminer combien de fois a été appelée la procédure `dessin` pour obtenir le tracé ci-dessus.
2. Sur le dessin, numéroter les 10 premiers segments tracés, sachant que le début du dessin est indiqué par la flèche.

### Exercice 4 (4 points)

On donne le code suivant :

```
class Pokemon:
    def __init__(self, identifiant):
        self.id = identifiant
        reponse = requests.get(f"https://pokeapi.co/api/v2/pokemon/{self.id}").json()
        self.nom = reponse["name"].capitalize()
        self.taille = round(reponse["height"] * 10)
        self.poids = round(reponse["weight"] / 10)
        self.xp = reponse["base_experience"]

    def modifier_xp(self, n):
        self.xp = self.xp + n

    def affronter(self, other):
        """
        Permet l'affrontement de deux Pokemon : le Pokemon qui a la plus faible expérience
        perd 10 points d'expérience, l'autre Pokemon gagne 10 points d'expérience.
        Aucun Pokemon ne peut avoir un total de points d'expérience négatif.
        - Entrée : other (instance de la classe Pokemon)
        """
        # code à compléter...

mon_pokemon = Pokemon(25)
```

1. Expliquer le rôle de la ligne `4` puis le rôle de la ligne `22`.
2. Donner la liste des attributs de l'instance `mon_pokemon`.
3. Écrire la spécification de la méthode `modifier_xp`.
4. Écrire la définition de la méthode `affronter` conformément à sa spécification.

## Seconde partie (5 points)

Copier sur le bureau le fichier `devoir2.ipynb` depuis le dossier `Devoir` du réseau.

Lorsque vous aurez terminé, vous renommerez votre fichier `nom_prenom.ipynb` et vous le déposerez dans le dossier `Rendu` du réseau. Attention, une fois déposé dans `Rendu`, votre travail n'est plus modifiable.

NOM : **Annexe**

```
classe Date:
    def __init__(self, j, m, a):
        if j not in range(1, 32):
            raise ValueError("jour incorrect")
        if m not in range(1, 13):
            raise ValueError("mois incorrect")
        if a == 0:
            raise ValueError("année incorrecte")
        self.jour = j
        self.mois = m
        self.annee = a
        self.evenement = ""

    def definir_evenement(self, evenement):
        self.evenement = evenement

    def str(self):
        noms_mois = ["", "janvier", "février", "mars", "avril", "mai", "juin",
                     "juillet", "août", "septembre", "octobre", "novembre", "décembre"]
        chaine = f"{self.jour} {noms_mois[self.mois]} {self.annee}"
        if self.evenement != "":
            chaine = chaine + f" ({self.evenement})"
        return chaine

    def __lt__(self, other):
        if self.annee != other.annee:
            return self.annee < other.annee
        elif self.mois != other.mois:
            return self.mois < other.mois
        else:
            return self.jour < other.jour
```