Classe de Terminale NSI

# Devoir surveillé

*Durée : 30 min + 30 min*

## Première partie (10 points)

### Exercice 1 (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)
```

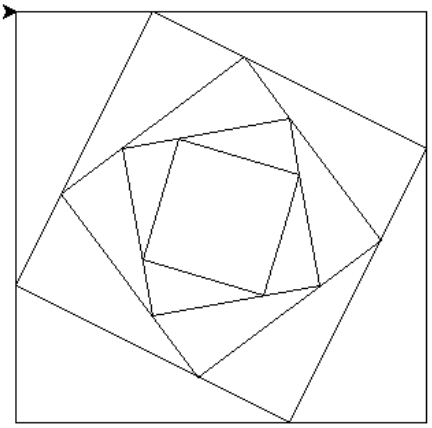
1. Justifier que `duree_de_vol` est une fonction récursive.
   Identifier le ou les cas récursifs et le ou les cas de base.

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. Donner le rôle de la fonction `duree_de_vol`.

### Exercice 2 (3 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`.

```
 1  def dessin(longueur, n):
 2      forward(longueur/3)
 3      right(26.57)
 4      if n > 0:
 5          dessin(longueur*5**0.5/3, n-1)
 6      left(26.57)
 7      forward(2*longueur/3)
 8      for k in range(3):
 9          right(90)
10          forward(longueur)
11      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.

3. Dessiner quelle figure est obtenue suite à l'appel `dessin(200, 1)`.

### Exercice 3 (3 points)

On donne la définition d'une classe `Pokemon`, puis la création d'une instance `mon_pokemon` de la classe `Pokemon`.

```
 1  class Pokemon:
 2      def __init__(self, identifiant):
 3          self.id = identifiant
 4          reponse = requests.get(f"https://pokeapi.co/api/v2/pokemon/{self.id}").json()
 5          self.nom = reponse["name"].capitalize()
 6          self.taille = round(reponse["height"] * 10)
 7          self.poids = round(reponse["weight"] / 10)
 8          self.xp = reponse["base_experience"]
 9
10      def modifier_xp(self, n):
11          self.xp = self.xp + n
12
13      def affronter(self, other):
14          """
15          Permet l'affrontement de deux Pokemon : le Pokemon qui a la plus faible expérience
16          perd 10 points d'expérience, l'autre Pokemon gagne 10 points d'expérience.
17          - Entrée : other (instance de la classe Pokemon)
18          """
19
20  mon_pokemon = Pokemon(25)
```

1. À quoi sert la ligne 4 ?
2. Lister les 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 (10 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.