Exercice 1 (6 points)

Cet exercice porte sur les arbres binaires, la récursivité et la programmation orientée objet.

Cet exercice porte sur l'identification de végétaux (tilleul, ficus, …) à partir de caractéristiques de leurs *folia* (nom scientifique des feuilles d'un végétal) : simples ou complexes, disposées de façon alternée ou non, etc.

Par exemple, un tilleul a des *folia* simples, disposées de façon alternée mais pas en hélice, en forme de cœur et à bord denté. Un ficus a également des *folia* simples et disposées de façon alternée. Cependant elles sont insérées en hélice et sont de forme ovale. Un robinier a des *folia* complexes, disposées de façon alternée et non dentées.

Pour identifier un végétal à l'aide des caractéristiques de ses *folia*, on utilise un arbre binaire appelé **arbre de décision**. Un exemple de tel arbre de décision est partiellement représenté sur la figure 1 ci-dessous (les parties non représentées de cet arbre sont indiquées par des points de suspension).

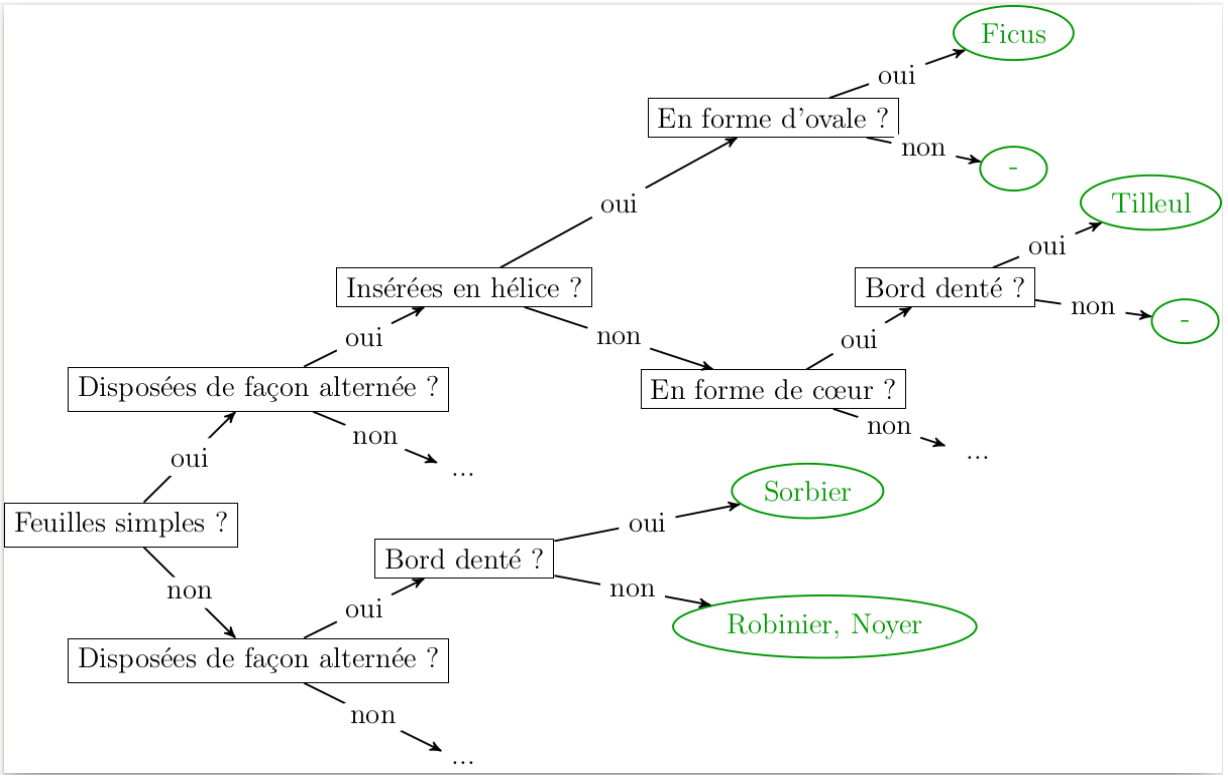

Figure 1. Extrait d'un arbre de décision aidant à reconnaître un végétal à partir des caractéristiques de ses *folia*.

Les rectangles sont les **nœuds** de l'arbre de décision. Ils correspondent chacun à une question. Les ovales sont les **feuilles** de l'arbre de décision. Ils correspondent chacun à un ensemble de végétaux. Pour chaque question, il est possible de répondre par oui ou par non ce qui permet d'atteindre soit un nouveau nœud, c'est-à-dire une nouvelle question, soit une feuille de l'arbre de décision. Cette feuille contient le plus souvent

un seul végétal, éventuellement plusieurs si leurs *folia* ont les mêmes caractéristiques, et éventuellement aucun si aucun végétal connu ne présente ces caractéristiques. Par exemple, robinier et noyer ont tous les deux des *folia* complexes (non simples), disposées de façon alternée et non dentées : ils sont donc dans la même feuille de l'arbre de décision de la figure 1.

1. On observe un végétal dont les *folia* sont complexes (non simples), disposées de façon alternée et à bord denté. D'après l'arbre de décision de la figure 1, peut-on identifier ce végétal ? Si oui, quel est-il ?

2. On observe un végétal dont les *folia* sont simples, disposées de façon alternée, insérées en hélice et ne sont pas de forme d'ovale. D'après l'arbre de décision de la figure 1, peut-on identifier ce végétal ? Si oui, quel est-il ?

L'arbre de décision est représenté en langage Python en utilisant une classe `Noeud` et une classe `Feuille_resultat` dont les définitions sont données ci-dessous.

```
1 class Noeud:
2     def __init__(self, question, sioui, sinon):
3         self.question = question
4         self.sioui = sioui
5         self.sinon = sinon
```

```
1 class Feuille_resultat:
2     def __init__(self, vegetaux):
3         self.vegetaux = vegetaux
```

La classe `Noeud` a trois attributs :

- un attribut `question`, qui est une chaîne de caractères représentant une question ;
- un attribut `sioui`, qui peut être soit un objet de la classe `Noeud` représentant une autre question, soit un objet de la classe `Feuille_resultat` ;
- un attribut `sinon`, qui peut être soit un objet de la classe `Noeud` représentant une autre question, soit un objet de la classe `Feuille_resultat`.

La classe `Feuille_resultat` a un seul attribut, `vegetaux`, qui est une liste (éventuellement vide) de chaînes de caractères, dans laquelle chaque chaîne est le nom d'un végétal.

Par exemple, pour l'arbre de décision de la figure 1, pour le `Noeud` dont la question est `'En forme d'ovale?'`, l'attribut `sioui` est un objet de la classe `Feuille_resultat` dont l'attribut `vegetaux` est la liste `['Ficus']` alors que l'attribut `sinon` de ce nœud est un objet de la classe `Feuille_resultat` dont l'attribut `vegetaux` est la liste vide.

3. Écrire en langage Python le code permettant de construire l'arbre de décision de la **figure 2** ci-dessous et de l'affecter à une variable nommée `arbre_2`.

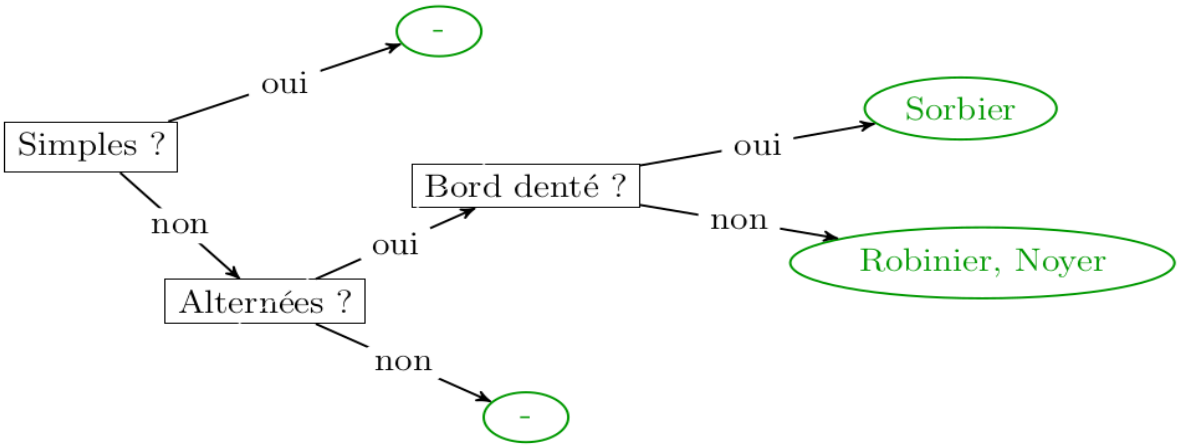

Figure 2. Arbre de décision 2.

On souhaite écrire une méthode `est_resultat` pour chacune des classes `Noeud` et `Feuille_resultat`. Un objet de la classe `Feuille_resultat` est un résultat, la méthode doit renvoyer `True`. Un objet de la classe `Noeud` n'est pas un résultat, la méthode doit renvoyer `False`.

4. Écrire le code de la méthode `est_resultat` pour la classe `Noeud`.

5. Écrire le code de la méthode `est_resultat` pour la classe `Feuille_resultat`.

On souhaite connaître le nombre de végétaux identifiables par un arbre de décision.

6. Écrire le code de la méthode `nb_vegetaux` pour la classe `Feuille_resultat`.

7. Écrire le code de la méthode `nb_vegetaux` pour la classe `Noeud`, qui prend en compte tous les végétaux identifiables à partir de ce nœud.

On souhaite enfin écrire une méthode `liste_questions` pour chacune des classes `Noeud` et `Feuille_resultat` afin d'obtenir la liste des questions présentes dans un arbre de décision. L'ordre des éléments dans cette liste n'a pas d'importance, de plus elle peut contenir des doublons. On remarque que :

- si `f` est un objet de la classe `Feuille_resultat`, alors `f.liste_questions()` est la liste vide ;

- le résultat de l'appel `arbre_2.liste_questions()` est la liste `['Simples ?', 'Alternées? ', 'Bord denté ?']` (ou une liste avec les mêmes éléments mais dans un ordre différent).

8. Écrire le code de la méthode `liste_questions` pour la classe `Feuille_resultat`.

9. Écrire le code de la méthode `liste_questions` pour la classe `Noeud`, qui prend en compte toutes les questions accessibles à partir de ce nœud. On rappelle que l'opérateur `+` en Python permet de concaténer des listes, par exemple la valeur de l'expression `[1,2]+[3,4,5]` est la liste `[1,2,3,4,5]`.

Pour représenter les caractéristiques des *folia* d'un végétal, on utilise un dictionnaire. Les clés du dictionnaire sont les questions de l'arbre de décision et les valeurs sont `True` ou `False` selon la réponse.

Par exemple, le dictionnaire décrivant les *folia* du sorbier pour l'arbre de décision de la figure 2 est le dictionnaire `folia_sorbier` défini ci-dessous.

```
folia_sorbier = {
  'Simples ?': False,
  'Alternées ?': True,
  'Bord denté ?': True
}
```

En revanche le dictionnaire `folia_tilleul` ci-dessous qui décrit (partiellement) les *folia* du tilleul n'est pas adapté pour l'arbre de décision de la figure 1 car des données sont manquantes. Par exemple, la question `'Feuilles simples ?'` n'est pas une clé de ce dictionnaire alors que c'est une question présente dans l'arbre.

```
folia_tilleul = {
  'En forme d'ovale ?': False,
  'Disposées de façon alternée ?': True,
  'Bord denté ?': True
}
```

On cherche à éviter ce genre de cas, afin de ne pas d'utiliser un arbre de décision pour classifier un végétal à partir d'un dictionnaire qui n'est pas assez renseigné.

10. Écrire une fonction `est_bien_renseigne` qui prend en paramètres :
    - un dictionnaire `dico_vegetal` qui donne les caractéristiques des *folia* d'un végétal ,
    - un arbre de décision représenté par un objet `arbre` de la classe `Feuille_resultat` ou de la classe `Noeud`,

et qui renvoie `True` si toutes les questions présentes dans `arbre` sont des clés de `dico_vegetal`.

11. Écrire une fonction `identifier_vegetaux` qui prend en paramètres :

    – un dictionnaire `dico_vegetal` qui donne les caractéristiques des *folia* d'un végétal,

    – un arbre de décision représenté par un objet `arbre` de la classe `Feuille_resultat` ou de la classe `Noeud`,

    et qui renvoie la liste, éventuellement vide, des noms des végétaux dont les *folia* correspondent aux caractéristiques du dictionnaire.

    Par exemple l'appel `identifier_vegetaux(arbre_2,folia_sorbier)` devra renvoyer la liste `['Sorbier']`.

    On suppose que toutes les questions de l'arbre de décision `arbre` apparaissent comme des clés dans le dictionnaire `dico_vegetal`.