# Examen final

Lundi 8 décembre 2025

Initiation à l'algorithmique et la programmation (TALA346A/B)

**Sujet unique** — **Durée :** 1h45

### Problème 1: Questions conceptuelles – 4 points

Vous souhaiteriez participer à la maintenance d'un logiciel de textométrie développé par un groupe de développeurs installés en Laponie. Grâce à votre profil, vous parvenez à obtenir un premier entretien : pour le mener à bien, il vous faudra être en mesure de répondre aux questions ci-dessous !

1. Qu'est-ce qu'une fonction ?
2. Soit deux variables, `a` et `b`. Comment permuter les deux variables ?
3. Soit $n$ un nombre entier. Que renvoie `range(n)` ? À quoi servent les objets de type `range` ?
4. Qu'est-ce qu'une liste ? Comment accède-t-on à un élément d'une liste ?

### Problème 2: Double trouble! – 4 points

Il semblerait que vous soyez parvenu.e à épater ces développeurs polaires lors de votre entretien : bravo ! Après un déménagement précipité en Laponie, vous recevez par courrier, le code d'une fonction issue du logiciel interne de textométrie.

```
    def function(text):
        vowels = "aeiouy"
        i = 0

        while i < len(text) - 1:
            if text[i] in vowels and text[i+1] in vowels:
                return True
            i += 1

        return False
```

1. Expliquer le comportement de la fonction. Suggérer un nom plus adéquat.
2. Prédire ce que renvoie la fonction pour les entrées suivantes :
   (a) `"snowflake"`
   (b) `"tree"`

(c) `"oeuf"`

Vous avez finalement comme tâche de revoir la fonction ! Vos nouveaux collègues vous indiquent que la prochaine version du logiciel ne devra plus contenir la moindre boucle while...

3. Quelle est la différence entre une boucle while et une boucle for ?
4. Réécrire le code ci-dessus avec une boucle for.

**Problème 3: Compter les suffixes – 4 points**

Visiblement, vos talents de programmation se sont révélés lors de votre première tâche ! Vous êtes à présent en charge de coder la prochaine fonctionnalité du logiciel de textométrie (félicitations !). Votre objectif est d'écrire une fonction `suffix_counter(words, s)` qui prend en entrée une liste de mots et un suffixe (chaîne de caractères) et renvoie le nombre de mots se terminant par ce suffixe.

**Exemples :**

- `suffix_counter(["chanter", "danse", "parler"], "er")` renvoie 2
- `suffix_counter(["maison", "raison", "chat", "son"], "son")` renvoie 3

1. Écrire une première fonction, `has_string` qui prend en entrée une liste de chaînes `l`, et une chaîne de caractère `s` et qui renvoie le nombre de chaînes qui contiennent la chaîne `s`.
2. Modifier la fonction `has_string` pour obtenir la fonction `suffix_counter`.
3. Modifier la fonction `suffix_counter` pour qu'elle retourne le nombre de mots avec le suffixe ET la liste de ces mots.

**Problème 4: Déboggage – 4 points**

Par un jour de grand froid (un jour comme les autres d'après les Samis), on vous transmet une autre fonction du logiciel de textométrie à relire : `lists_of_words`, qui prend en entrée une liste de mots et un entier $n$ et renvoie deux listes : la liste des mots de longueur $n$, ET la liste des premiers caractères pour chacun des mots de longueur $n$. Malheureusement, le code original est corrompu et le comportement de la fonction ne correspond pas à ce qui est documenté ! Voici la fonction :

```
1    def lists_of_words(words, n=7):
2      long_words = {}
3      first_letters = {}
4      for i in range(len(words)-1):
5        word = words[i]
6        if len(word) > n:
7          long_words.append(i)
8          first_letters.append(words[1])
9      return long_words
```

Sur votre bureau, quelqu'un a laissé un post-it : *la méthode* `.append()` *permet d'ajouter un élément à une liste prédéfinie. Elle prend en argument l'élément à ajouter.*

1. Identifier et expliquer brièvement chaque erreur présente dans la fonction.
2. Écrire une version corrigée de la fonction. Assurez-vous que la fonction aie le comportement attendu.
3. Prédire la sortie de `lists_of_words(["ordinateur", "clé", "programme", "code"], 5)`
4. Modifiez la fonction pour qu'elle prenne un paramètre optionnel supplémentaire `include_indices` (booléen, `False` par défaut). Lorsque `include_indices=True`, la fonction doit retourner trois éléments : la liste des mots de longueur $n$, la liste de leurs premiers caractères, et la liste de leurs indices d'origine dans la liste d'entrée.

**Problème 5: Premier indice – 4 points**

La fin de la semaine arrive : il vous tarde de rentrer chez vous et de profiter de votre cheminée avec un bon chocolat chaud... Cependant, il y a encore du travail à fournir pour partir l'esprit tranquille : cette fois-ci, il vous faut écrire une fonction `first_index(l, item)` qui prend en entrée une liste et un élément, et renvoie l'indice de la première occurrence de cet élément dans la liste. Si l'élément n'est pas présent, la fonction doit renvoyer `-1`.

**Exemple :**

- `first_index([3, 7, 2, 7, 9], 7)` renvoie `1`
- `first_index([3, 7, 2, 7, 9], 5)` renvoie `-1`

1. Écrire la fonction `is_value` qui prend en entrée une liste `l`, et `item`, et renvoie un booléen pour indiquer si `item` est dans `l` (`True` si `item` est dans `l`, `False` sinon). (*Indice : itérer sur les indices de la liste pour accéder aux éléments de* `l`)
2. Modifier la fonction `is_value` pour qu'elle ait le comportement attendu de la fonction `first_index` (renvoie l'indice de la première occurence de `item` dans `l`).
3. Modifier votre fonction pour qu'elle prenne un paramètre optionnel `start` (par défaut 0) qui indique à partir de quel indice commencer la recherche.

*Fin de l'examen. Bon courage !*