

Nous allons nous beaucoup nous servir des concepts utilisés dans le TD3. Commençons par un résumé des outils

## I Quelques outils

### 1.1 Sur les chaînes de caractères



Une chaîne de caractères est un objet informatique qui représente du texte. Techniquement, en python, on les distingue par des guillemets simples ou doubles.

```
1 s = 'Une chaîne d'exemple'
2 len(s) # le nombre de caractères, espaces compris
3 s[0] # le premier caractère
4 s[5] # le 6ème caractère
```

```
1 for i in range(len(s)):
2     # i est un entier, représentant un indice dans la chaîne.
3     # i va prendre les valeurs 0, 1, 2...
4     s[i] # il s'agit du caractère d'indice i
```

ou alors

```
1 for c in s:
2     # c est directement un caractère
3     # Avec notre exemple de chaîne s, c va prendre comme valeurs
4     # successives 'U', 'n', 'e', ' '...
```

### 1.2 Sur les dictionnaires



Un **dictionnaire** est un autre type de données utilisable en python. Il permet de stocker des **valeurs** associées à des **clés**.  
Plus précisément, un dictionnaire est un ensemble de (clé, valeur) où chaque valeur est associée à exactement une clé.

Pour créer un dictionnaire, on utilise des accolades de manière similaire aux crochets utilisés pour les listes.

```
1 d = {} # un dictionnaire vide
```

```
1 d = {'a': 12} # un dictionnaire contenant une paire de (clé, valeur)
2 # la clé est la chaîne 'a' et la valeur le nombre 12
3 d['a'] # c'est la valeur associée à la clé 'a', ici 12
4 d[0] = [2, 4, 6] # on crée une nouvelle paire (clé, valeur). La clé est un nombre
5 # et la valeur une liste.
```

- Les clés utilisables pour nous sont : des chaînes, des nombres. En toute généralité, tout objet non modifiable peut être utilisé comme clé.
- Les valeurs sont des objets python quelconques (nous connaissons : nombres entiers ou flottants, booléen, chaîne, liste, dictionnaire)

Pour tester si une clé est présente dans un dictionnaire (et avec le dictionnaire `d` du cadre précédent)

```
1 'a' in d # c'est le booléen True car la clé est présente dans d
2 2 in d # c'est le booléen False
```

## II Un texte chiffré

Le fichier `texte_chiffre.txt` contient un texte qui a été *chiffré* par un procédé assez simple : on a remplacé chaque occurrence de chaque lettre par une autre lettre. Par exemple tous les “a” auraient pu être remplacés par des “j”.

Le but de ce TD est de décrypter ce texte (retrouver le texte de départ, sans connaître a priori la manière exacte dont il a été chiffré). Pour simplifier l'étude, on considère que le de départ est entièrement en minuscules (mais on a conservé les caractères spéciaux comme les caractères accentués, la ponctuation...).

## 2.1 Analyse des occurrences

Une première approche simple est de comparer les fréquences d'apparition de chaque lettres dans le texte chiffré avec un tableau de référence. Nous utiliserons <sup>1</sup> [le tableau wikipedia](#).

Pour stocker les occurrences de chaque lettre, nous allons construire un dictionnaire dont les clés sont les lettres du texte et les valeurs le nombre d'occurrences. Par exemple

```
1 s = 'texte'
2 # et le dictionnaire que l'on veut calculer sera
3 d = {'t': 2, 'e': 2, 'x': 1}
```

### Exercice 1

Compléter la fonction `ajoute_occurrences(phrase: str, dico: dict) -> dict` qui prend comme argument une chaîne de caractère et un dictionnaire d'occurrences (éventuellement vide) et ajoute dans ce dictionnaire toutes les occurrences des caractères de `phrase`



Les dictionnaires, comme les listes, sont des objets modifiables et les modifications apportées à un de ces objets à l'intérieur d'une fonction sont conservées. Ceci est faux pour les nombres ou les chaînes de caractères.

### Exercice 2

Compléter la fonction `toutes_occurrences(texte: list) -> dict` qui prend comme argument un texte (une liste de chaînes de caractères) et retourne le dictionnaire des occurrences de ce texte (calculé comme occurrences dans toutes les phrases contenues dans `texte`).

## 2.2 Premier pas en décryptage

Le caractère d'espace est un caractère comme les autres en informatique. Il est logiquement le plus fréquent dans nos écrits.

### Exercice 3

En utilisant la fonction `toutes_occurrences` sur la variable appelées `lignes` dans votre fichier, construire le dictionnaire des occurrences de notre texte chiffré (et stocker ce dictionnaire dans une variable de la console, pour utilisation future).

Compléter maintenant la fonction `lettre_majoritaire(dico: dict) -> str` qui prend comme argument un dictionnaire d'occurrences et retourne la lettre ayant le plus d'occurrence.

Comment a été chiffré l'espace dans notre texte ?



Rappel : on peut facilement concaténer des chaînes de caractères et c'est la méthode à utiliser pour construire pas à pas une chaîne :

```
1 s1 = 'ab'
2 s2 = 'cd'
3 s = s1 + s2 # maintenant s contient 'abcd'
```

### Exercice 4

L'étape suivante consiste à échanger deux caractères dans notre texte : l'espace et le caractère trouvé précédemment, et noter sur une feuille qu'on a déjà décrypté le caractère d'espace.

Compléter les fonctions `remplace(chaine: str, car1: str, car2: str) -> str` et `remplace_texte(texte: list, car1: str, car2: str) -> list`

## 2.3 Finir le job

On peut maintenant trouver un texte où les espaces sont bien placés. Il nous reste à deviner pas à pas chacune des autres lettres en étant guider par le tableau donné en lien au début du TD. On peut trouver facilement la lettre "e" pour commencer, et ensuite c'est un jeu de devinette.



Votre feuille de script contient la fonction `tri_dico` qui permet d'avoir la liste des occurrences triées par ordre décroissant

Quelques commandes utiles.

1. A l'adresse [https://fr.wikipedia.org/wiki/Fr%C3%A9quence\\_d%27apparition\\_des\\_l lettres\\_en\\_fran%C3%A7ais](https://fr.wikipedia.org/wiki/Fr%C3%A9quence_d%27apparition_des_l lettres_en_fran%C3%A7ais)

```
1 L = tri_dico(toutes_occurrences(lignes)) # à refaire après chaque échange
2 lignes = remplace_texte(lignes, 'p', ' ') # changer le caractère
3 affiche_texte(lignes) # affiche les 150 premiers caractères
4 # ou affiche_texte(lignes, 250) pour les 250 premiers caractères.
```

Une dernière remarque : le caractère spécial ”

n” représente un passage à la ligne et est un caractère comme les autres, ne vous fiez pas aux séparations de lignes avant d’avoir décrypté ce caractère!