# 1 Rappels sur les fichiers

## 1.1 Création d'un objet-fichier

En python, création et manipulation d'un fichier se font par l'intermédiaire d'un objet-fichier, généré par la fonction :

objet_fichier = **open**(nom du fichier, mode d'accès).

Les paramètres sont des chaînes de caractère :

– `nom du fichier` est le nom du fichier, avec son extension.

– `mode d'accès` peut être :

'w' : (write) ouverture pour écriture seule. Lorsque le fichier n'existe pas il est créé dans le répertoire courant ; lorsque le fichier existe il est écrasé.

'a' : (append) ouverture pour écriture seule. Lorsque le fichier n'existe pas il est créé dans le répertoire courant ; lorsque le fichier existe les données écrites le seront à la suite.

'r' : (read) ouverture pour lecture seule. Le ficher doit exister dans le répertoire courant.

## 1.2 Fermeture

Une fois la lecture et l'écriture dans le fichier terminés il faut refermer l'objet-fichier avec l'instruction :

```
objet_fichier.close()
```

## 1.3 Méthodes des objets-fichiers

Un objet-fichier admet les méthodes :

- Lorsque **ouvert en écriture** :
  `write()` écrit une chaine de caractère en fin de fichier ouvert en écriture.
- lorsque **ouvert en lecture** :
  `read()` lit l'intégralité du fichier. `read(n)` lit `n` caractères.

# 2 Chaines de caractères

## 2.1 Opérations communes aux listes et chaines de caractères

Les types **`int`**, **`float`**, **`bool`** sont des *types scalaires*.
Les types **`list`** (listes) et **`str`** (chaînes de caractère) sont des structures de données de *type séquentielles*.
Tous les objets de type séquentiel ont en commun :

| Opération | Résultat |
|---|---|
| `s[i]` | élément d'indice `i` de `s` |
| `s[i:j]` | Extraction de `i` (inclus) à `j` (exclus) |
| `s[i:j:k]` | Extraction de `i` à `j` par pas de `k` |
| **`len`**`(s)` | Longueur de `s` |
| `x` **`in`** `s` | `True` si `x` est dans `s`, `False` sinon |
| `x` **`not in`** `s` | `True` si `x` n'est pas dans `s`, `False` sinon |
| `s+t` | Concaténation de `s` et `t` |
| `s*n, n*s` | Concaténation de `n` copies de `s` |
| `s.index(x)` | Indice de la $1^{ere}$ occurrence de `x` dans `s` |

où `s` et `t` sont des objets séquentiels de même type, et `i, j, k, n` sont des entiers.

Les objets de type **`str`**, chaînes de caractère, sont des structures de données séquentielles :

```
In [1]: chaine = 'Amanda'
In [2]: len(chaine)
In [3]: print(chaine[::-1])
adnamA
In [4]: print(chaine*2)
AmandaAmanda
```

• On accède à leurs éléments par leurs indices entre crochets :

```
In [1]: ch = 'Amanda'
In [2]: print(ch[0], ch[-1])
A a
```

• On peut les parcourir à l'aide d'une boucle **`for`** :

```
In [1]: for c in  ch[:3]:
...         print(c)

A
m
a
```

• Attention : les chaînes sont **non-mutables** : changer un élément produit une erreur `TypeError` :

```
In [1]: chaine[0] = 'Z'
TypeError: 'str' object does not support item assignment
```

## 2.2 Méthode des chaînes de caractères

On peut utiliser l'une des méthodes `lower()` ou `upper()` des chaines de caractère, qui convertit en minuscule/majuscule :

```
>>> "toto".upper()
'TOTO'

>>> "Toto".lower()
'toto'
```

Les chaines de caractères sont munies de l'ordre lexicographique (ordre alphabétique de gauche à droite) :

```
>>> 'ABC' < 'ABD'
True
>>> 'ABC' < 'AAB'
False
```

En particulier on peut savoir si un caractère est une lettre minuscule/majuscule de la façon suivante :

```
>>> 'a' <=  'i' <= 'z'
True
>>> 'A' <= 'i' <= 'Z'
False
```

**Exercice 1** Les palindromes.

Un mot est un palindrome s'il est identique lu de gauche à droite et de droite à gauche (à la casse près) :

Exemples : radar, rotor, ressasser

1) Ecrire une fonction qui teste si une chaine est un palindrome :

2) La modifier pour qu'elle ne tienne pas compte de la casse (majuscule/minuscule).

Une phrase est un palindrome si après en avoir supprimé les espaces on obtient un mot palindrome.

Exemple : "Engage le jeu que je le gagne".

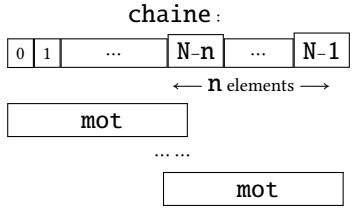
3) Écrire une fonction prenant en paramètre une chaine de caractère et qui renvoie `True` ou `False` selon si c'est ou non une phrase palindrome.

# 3 Recherche d'un mot dans une chaîne

La recherche d'un mot dans une chaîne de caractère est un problème essentiel en informatique qui admet des algorithmes élaborés et efficients. (Penser par exemple aux moteurs de recherche).

Donner une solution naïve, pas très efficiente n'est pas difficile :

```
def contient(chaine,mot):
    N = len(chaine)
    n = len(mot)
    for i in range(N-n+1):
        if (mot == chaine[i:i+n]):
            return True
    return False
```




Le résultat est lent est couteux en mémoire car la fonction doit effectuer jusqu'à $(N-n)$ copies d'une liste de longueur $n$, et pour chacune $n$ comparaison, soit de l'ordre de $(N-n)\times 2n$ opérations et $(N-n)\times n$ fois l'emplacement nécessaire au stockage d'un caractère utilisé.

Une meilleure solution, (mais toujours naive) est :

```
def cherche(chaine,mot):
    N = len(chaine)
    n = len(mot)
    for i in range(N-n+1):
        k = 0
        while k<n:
            if mot[k] != chaine[i+k]:
                break
            k += 1
        if k == n:
            return True
    return False
```

On compare à chaque position, mais on passe à la position suivante dès que 2 caractères diffèrent. On n'effectue plus les copies en mémoires. Dans le pire des cas on effectue de l'ordre de $(N-n)\times n$ opérations (de comparaisons).

Il existe des algorithmes beaucoup plus efficients (et plus compliqués : Boyer-Moore, Knuth-Bendix-Pratt).

# 4 Application : Recherche dans un fichier PDF

**Exercice 2.** Recherche dans un fichier PDF

Le format de fichier PDF (*Portable Document Format*) permet de stocker des documents constitués de textes et d'images dans un format compressé, léger et portable (un document PDF peut être ouvert, et avoir une apparence identique sur la plupart des systèmes d'exploitation).
Un fichier PDF est un fichier texte commençant par la chaîne de caractères `%PDF` et finissant par `%%EOF` (il peut y avoir d'autres caractères avant et après, qui seront ignorés par le lecteur PDF). Il utilise un jeu de caractères plus large que les tables de caractères ASCII et Unicode.
Pour l'ouvrir sous Python, on pourra utiliser l'option `errors` de la fonction **`open()`** avec pour valeur `'surogateescape'` ; tous les caractères du fichier ne figurant pas dans la table Unicode seront alors ignorés, mais l'ouverture s'exécutera sans aucun message d'erreur d'encodage.

```
fichier = open('nom_du_fichier', 'r', errors="surrogateescape")
texte = fichier.read()
fichier.close()
```

1. Écrire une fonction `indice(chaine,mot)` prenant en paramètres deux chaînes de caractères `chaine` et `mot`, et qui retourne $-1$ si `chaine` ne contient pas `mot` comme sous-chaîne, et sinon l'indice dans `chaine` où la première occurence de `mot` a été trouvée.
2. Écrire une fonction `estPDF()` qui permet de vérifier qu'un fichier `fichier` placé dans le répertoire de travail soit bien au format PDF. Elle écrira dans la console le résultat trouvé.

Un fichier PDF contient sous forme de chaînes de caractères, des commandes PDF donnant, lorsqu'elles sont renseignées, des informations sur le fichier.
Le nom du créateur du fichier (ou du logiciel ayant encodé le fichier) apparaît souvent, entre parenthèses, après la chaîne de caractères `'/Creator'`, éventuellement séparés d'un ou plusieurs espaces.
Par exemple, si le fichier contient la chaîne de caractères `'/Creator (Jean Martin)'`, son créateur est Jean Martin.

Les date, heure et fuseau horaire de création du fichier sont souvent renseignés après la chaîne de caractères `/CreationDate` sous la forme :

`/CreationDate(D:aaaammjjhhmmss+XX'00')`

où `aaaa` : année, `mm` : mois, `jj` : jour, `hh` : heure, `mm` : minutes, `ss` : secondes, `+XX` décalage GMT. La commande `/CreationDate` et ses paramètres entre parenthèses `(...)` peuvent être séparés d'aucun, un ou plusieurs espaces. Par exemple :

`/CreationDate(D:20150611214310+01'00')`

décrit une fichier créé le 11 juin 2015, à 21h 43min 10s, à la longitude du méridien de Paris (GMT+01).

3. Écrire une fonction `createurPDF(fichier)` où `fichier` est un fichier PDF placé dans le répertoire courant ; la fonction écrira dans la console le nom de son créateur, s'il y figure, et la chaîne de caractères `'Auteur non trouve'` sinon.
4. Écrire une fonction `datePDF()` qui prend en paramètre le nom d'un fichier PDF placé dans le répertoire courant qui, si possible, écrit dans la console, dates et heures de création (avec décalage GMT).

*Indications :*
**1.** *Adapter l'algorithme de recherche dans une chaîne de caractères d'une sous-chaîne.*
**2.** *Comparer les indices où apparaissent les sous-chaînes* `'%PDF'` *et* `'%%EOF'` *dans le texte du fichier.*
**3.** *Copier la chaîne après* `/Creator` *et délimitée par deux parenthèses.*
**4.** *On supposera que dès que la chaîne* `/CreationDate` *apparaît et est suivie de* `(D:` *la date est présente. Extraire alors les champs significatifs par slicing.*