# TD Méthodologie de la Programmation

3 juillet 2019

**Recommandations**

Pour votre convenance, l'énoncé est disponible au format .ipynb et .pdf. **Il s'agit du même exercice.**

Vous pouvez réaliser le devoir directement sur jupyter et rendre le fichier .ipynb modifié par vos soins. Alternativement, vous pouvez faire et rendre l'exercice directement en python (fichier .py). Dans tous les cas, pensez bien à indiquer votre nom, prénom et n. d'étudiant.

Les questions doivent être faites dans l'ordre, pour chaque question les fonctions précédentes devraient vous aider.

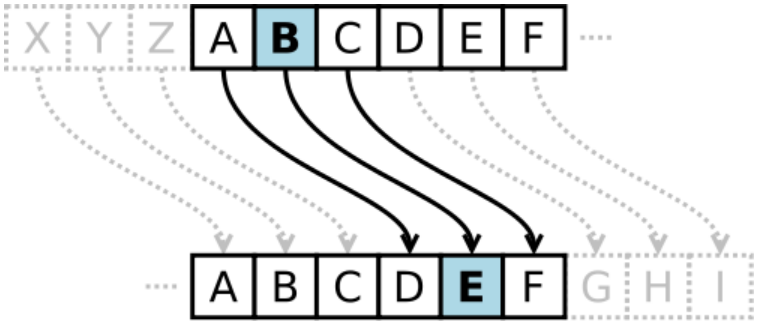
Vous devez envoyer votre travail (exercice + fichier de sortie) sur le mail `josue.melka02@univ-paris8.fr` à la fin de l'épreuve. Tout travail rendu après le délai fixé ne pourra être pris en compte.

## 1 Chiffre de César

Aussi connu sous le nom de « *chiffrement par décalage* », le chiffre de César est une méthode de chiffrement très simple, qui aurait été utilisée par Jules César dans ses correspondances secrètes.

Le texte chiffré s'obtient en remplaçant chaque lettre du texte original par une lettre à distance fixe dans l'ordre de l'alphabet. Pour les dernières lettres, on reprend au début.

La longueur du décalage constitue la clé du chiffrement qu'il suffit de transmettre au destinataire pour qu'il puisse déchiffrer le message.
Par exemple avec un décalage de 3 vers la droite, `A` est remplacé par `D`, `B` devient `E`, et ainsi de suite jusqu'à `W` qui devient `Z`, puis `X` devient `A` etc.




Exemple avec un décalage de 3

## Exercice

Pour tout décalage, il est possible de créer un dictionaire donnant la correspondance entre la lettre originale et la lettre codée, ce qui permet de réaliser l'encodage très simplement. Il suffit ensuite d'inverser la table de correspondance pour faire le décodage de la même façon.

Nous utiliserons cet alphabet:

```
ALPHABET = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

### 1.1 Décalage

Définir une fonction `decalage(distance)` permettant d'obtenir le code pour une distance donnée.
Par exemple, `decalage(6) == 'GHIJKLMNOPQRSTUVWXYZABCDEF'`.

### 1.2 Table de correspondance

Définir une fonction `table_cesar(distance)` permettant de générer la table de correspondance pour une distance donnée.
Par exemple, `table_cesar(6)` retournera:

```
{'A': 'G', 'B': 'H', 'C': 'I', ..., 'T': 'Z', 'U': 'A', 'V': 'B', ..., 'Z': 'F'}
```

### 1.3 Encodage

Définir une fonction `encode(txt, table)` permettant d'encoder en texte en fonction d'une table de correspondance.
Par exemple `encode('1234 ', {'1': 'A', '2': 'B', '3': 'C', '4': 'D'}) == 'ABCD '`

***Attention**: les caractères inconnus dans la table, comme les espaces, doivent rester fidèles à l'original.*

### 1.4 Chiffrement

Définir une fonction `chiffre_cesar(message, distance)` permettant de coder le message pour une clé donnée.
Par exemple, pour un décalage de 6, `chiffre_cesar('MESSAGE SECRET', 6) == 'SKYYGMK YKIXKZ'`.

### 1.5 Déchiffrement

Définir une fonction `dechiffre_cesar(cryptogramme, distance)` permettant de retrouver le message original à partir du cryptogramme.
Par exemple, `dechiffre_cesar('SKYYGMK YKIXKZ', 6) == 'MESSAGE SECRET'`.

### 1.6 Décoder un fichier

Le fichier `cryptogramme.txt` contient un message codé avec le chiffre de Cesar, mais la clé est inconnue.
A vous de trouver la clé et d'enregistrer le resultat dans un fichier.

*Indice: il suffit de tester l'ensemble des clés possible (il y en a 26) en affichant le resultat pour chaque clé en sortie, ce qui permet ainsi de trouver quelle est la bonne clé.*