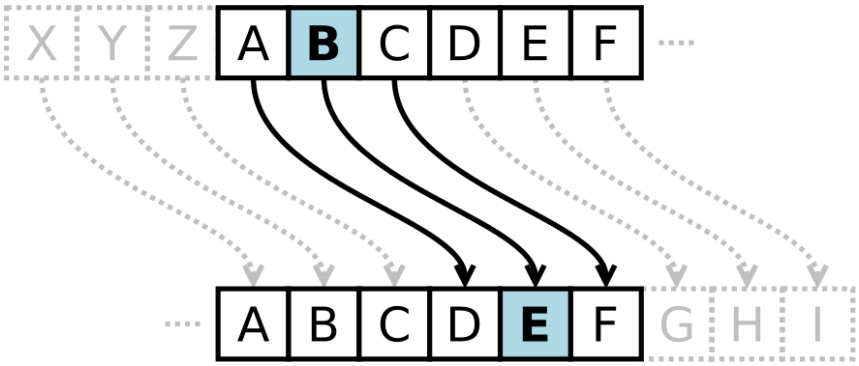
Exercice 1

Chiffre de César

Le chiffrement par décalage, ou chiffre de Caesar fonctionne par décalage des lettres de l'alphabet.



Cet algorithme de chiffrement utilise une fonction périodique pour transformer les rangs de chaque lettre. Le décalage est constant ; il s'agit de la **clé de chiffrement**.

Profitons de la table ascii qui fournit un alphabet et son code numerique pour réaliser le chiffrement de quelques caractères. Nous prendrons les caractères majuscules comme alphabet de depart et d'arrivée.

> Compléter la fonction `decalage` qui génère une lettre chiffrée à partir de la lettre du texte clair, et de la clé (un nombre entre 1 et 26) :

```
def decalage(lettre_clair, cle):
  ascii_clair = ...
  ascii_chiffre = ...
  lettre_chiffre = ...
  return ...
```

> Programmer et tester la fonction.

Exemple :

```
decalage('A',1)
# 'B'
```

Exercice 2

Exercice inversion de bits

L'inversion de bits est une technique employée pour le chiffrement de messages secrets.

On cherche à créer une fonction qui permet d'inverser les bits écrits dans une variable de type string.

On utilise une variable interne : `inv`, qui vaut au depart "".

On utilise une boucle bornée pour parcourir les bits de N, depuis le premier bit jusqu'au dernier.

Le variant de la boucle est l :

- Si l vaut "0", alors on ajoute "1" à la chaine `inv`.
- Sinon, si l vaut "1", on ajoute "0".

Les éléments du programme sont, dans le desordre, et sans respecter les indentations :

```
inv = ""
if l == "0" :
else :
inv = inv + "1"
inv = inv + "0"
return inv
for l in N:
def inverse_bits(N):
```

1. Utiliser les éléments de programme en base de page pour créer la fonction `inverse_bits`. Vous pouvez découper/coller.

2. Ajouter le docstring dans la fonction :

```
"""
inverse les bits "1" et "0" d'une chaine de caractères
params:
    N (str): chaine de caractères codant le binaire
variables:
    inv (str): chaine de caractères stockant les bits inversés
    l (str): variant de boucle prenant la valeur successive de
    chaque caractère de N
returns:
    inv (str)
"""
```

3. Variante utilisant des variables booléennes :

- La fonction `bool` transforme un entier 0 ou 1 en un booléen `False` ou `True` :

```
bit = 0
bool(bit)
# False
```

- l'opérateur `not` donne l'opposé du booléen :

```
bit = 0
not bool(bit)
# True
```

Modifier la fonction `inverse_bit` pour utiliser les fonction et opérateur `bool` et `not`.

```
if l == "0" :
```

```
else :
```

```
inv = inv + "1"
```

```
inv = inv + "0"
```

```
def inverse_bits(N):
```

```
for l in N:
```

```
return inv
```

```
inv = ""
```

```
"""
inverse les bits "1" et "0" d'une
chaine de caractères
params:
    N (str): chaine de caractères
codant le binaire
variables:
    inv (str): chaine de caractères
stockant les bits inversés
    l (str): variant de boucle prenant
la valeur successive de
    chaque caractère de N
returns:
    inv (str)
"""
```