Série 3

Dans cette série on verra quelques fonctions de chiffrage. On va commencer avec les plus simples pour passer aux chiffrages utilisés de nos jours.

Exercice 1

On va commencer très simple avec une fonction de substitution et on va l'utiliser pour faire des dechiffrages simples.

1. Connaissance

Cette première partie définit une fonction qui fait simplement la substitution linéaire dans l'alphabet. Elle ignore toutes les lettres autres que a-z minuscules et majuscules.

Si vous arrivez à comprendre comment ceci est fait dans la fonction, félicitations. J'étais trop fatigué pour faire quelque chose de plus joli...

En tout cas vous voyez que le déchiffrage se fait simplement en passant la phrase chiffré avec l'index négatif. Si vous y pensez, ceci fait du sens, car on inverse le cercle de substitution.

2. Compréhension

Je vous laisse découvrir la phrase secrète qui ne donne pas le sens de la vie, mais le message tout aussi important que le créateur à laissé à sa création.

Pour trouver le message, vous devez trouver la valeur qui vous déchiffre (ou *décode* dans cet example...) la phrase.

Une fois que vous avez trouvé la valeur `x`, vous pouvez essayer la valeur `26 + x`, qui devrait donner le même résultat. Est-ce que vous arrivez à comprendre pourquoi?

Un cas spécial est la substitution avec un `shift` de 13. Dans l'internet ceci est appelé `ROT-13`, et c'est souvent utilisé par des vieilles personnes. Encore plus vieilles que moi...

Essayez de déchiffrer les messages suivant:

```
Wsvvc jsv xli mrgsrzirmirgi
Jung vf 6 gvzrf 7?
```

3. Application

Pour le décryptage on a vu qu'on doit passer par les 26 valeurs possibles. Vous pouvez écrire la fonction `decrypt` qui prend une phrase et une liste de lettres, suivant la fréquence d'apparition dans la langue cible.

Testez-le avec différentes phrases que vous chiffrez avec la fonction `substitution`. A partir de quelle longueur de phrase ça marche bien?

In [2]:

```python
# Exercice 1 - Partie 1

# Takes a phrase and a rotation-indicator, then returns the new phrase.
# All characters outside of [A-Za-z] are ignored.
def substitution(phrase: str, shift: int) -> str:
  temp = list(phrase)
  for i, c in enumerate(temp):
    if c.isalpha():
      char_case = ord(c) & 0xe0
      char_index = ord(c) & 0x1f
      char_new_index = ( char_index + shift - 1 ) % 26 + 1
      temp[i] = chr(char_case + char_new_index)

  return ''.join(temp)

print(substitution("Don't panic", 2))
print(substitution("Fqp'v rcpke", -2))
```

```
Fqp'v rcpke
Don't panic
```

In [25]:

```python
# Exercice 1 - Partie 2

print(substitution("Wsvvc jsv xli mrgsrzirmirgi", 22))
print(substitution("Jung vf 6 gvzrf 7?", 13))
```

```
Sorry for the inconvenience
What is 6 times 7?
```

In [ ]:

```python
# Exercice 1 - Partie 3

# Decrypt takes an encrypted string where the original phrase has the letters
# that follow the position in frequency: the first position gives the most
# frequent letter.
def decrypt(encrypted: str, frequency: str) -> str:
    pass
```