université
PARIS-SACLAY

Programmation en Python
Master 2 Réseaux Télécoms
TP 1 : La prise en main

1 Nombres apocalyptiques

Cet exercice est consacré à l'étude de nombres apocalyptiques. Un nombre n est réputé apocalyptique si dans le développement décimal de 2^n apparaît au moins une fois le groupe de chiffres `"666"`. Ouvrez le fichier `apocaliptic.py` et complétez le code de la fonction

```
def is_apocaliptic(n):
    pass
```

Cette fonction retourne une valeur booléenne `True` si son argument est un nombre apocalyptique et `False` dans le cas contraire. *Indication :* pour obtenir une représentation décimale pensez à utiliser le constructeur `str()`. On peut tester la présence d'une séquence de caractères dans une chaîne avec l'opérateur `in`.

Le fichier `apocaliptic.py` contient le test pour la fonction `is_apocaliptic`. Exécutez ce fichier pour tester votre code.

2 Plus grand commun diviseur et l'indicatrice d'Euler

Ouvrez le fichier `pgcd.py` et complétez le code des fonctions suivantes :

```
def pgcd(a,b):
    pass

def phi(n):
    pass
```

La fonction `pgcd(a,b)` doit retourner le plus grand commun diviseur de ses arguments. Pour implémenter cette fonction, on utilisera l'algorithme d'Euler récursif.

La fonction `phi(n)` doit retourner la valeur de l'indicatrice d'Euler de son argument. Par définition, l'indicatrice d'Euler associe à tout entier naturel n non nul le nombre d'entiers compris entre 1 et n (inclus) et premiers avec n. *Rappel :* deux entiers sont premiers entre eux si et seulement si leur PGCD vaut 1.

Le fichier `pgcd.py` contient le test pour ces deux fonctions. Exécutez ce fichier pour tester votre code.

3 Chiffre de César

Ouvrez le fichier `caesar.py` et complétez le code de la fonction

```
def caesar(plain, n):
    pass
```

Cette fonction retourne la chaine de caractères `plain` encodée par le chiffre de César avec le décalage `n`. Le décalage s'applique séparément aux lettres minuscules et majuscules, les autres caractères de la chaîne `plain` restent inchangés. Vous pouvez utiliser la fonction auxiliaire suivante

```
def rotate(c, n):
    if c.isupper():
        return chr((ord(c)-ord('A')+n)%26+ord('A'))
    elif c.islower():
        return chr((ord(c)-ord('a')+n)%26+ord('a'))
    return c
```

qui réalise le chiffrement de César du caractère `c`.

Le fichier `caesar.py` contient le test pour la fonction `caesar`. Exécutez ce fichier pour tester votre code.

4 Analyse fréquentielle

Toujours dans le fichier `caesar.py`, complétez le code de la fonction

```
def freq(s):
    pass
```

qui retourne un dictionnaire avec les lettres (majuscules et minuscules) comme les clés, et leurs occurrences dans la chaîne de caractères `s` comme les valeurs. Vous pouvez utiliser la constante `ALPHABET` qui contient une liste de toutes les lettres.

Le fichier `caesar.py` contient le test pour la fonction `freq`. Exécutez ce fichier pour tester votre code.

5 Cryptanalyse du chiffre de César

Le but de cet exercice sera de déchiffrer le texte `ciphertext` dans le fichier `caesar.py`, en sachant que le texte d'origine est en anglais. Notez que le chiffre de César est inversible, notamment, le texte encodé avec le décalage n sera décodé par le même chiffre avec le décalage $26 - n$. Pour automatiser le travail vous utiliserez la fonction `freq` créée dans l'exercice précédant. Une approche possible consiste à utiliser un texte anglais de référence (un *corpus*) pour calculer les fréquences des lettres dans une langue naturelle, et ensuite tester différentes décalages du texte chiffré pour retrouver le décalage donnant une distribution des fréquences proche à celle du corpus.