SNT — Cryptage

Partie C1 — Pourquoi chiffrer ?

Du cadenas du navigateur à la scytale des Spartiates : garder un message secret

Tu envoies des messages, tu paies en ligne, tu te connectes à des comptes. À chaque fois, des informations voyagent par des câbles et des ondes que d'autres personnes peuvent capter. Pour qu'elles restent secrètes, on les **chiffre**. Ce thème t'apprend à chiffrer, puis à casser des chiffrements simples, et à comprendre pourquoi ceux d'aujourd'hui sont bien plus solides.

Cours · Le chiffrement est partout

- **Le cadenas du navigateur** : une adresse qui commence par `https` indique que les échanges entre ton navigateur et le site sont chiffrés.
- **Les messageries** : beaucoup chiffrent « de bout en bout », c'est-à-dire que seuls l'expéditeur et le destinataire peuvent lire le message.
- **Le paiement par carte** et **le Wi-Fi protégé** : les données sont chiffrées pendant leur trajet.
- **Les mots de passe** : un site sérieux ne les garde pas en clair.

Cours · Le vocabulaire

*On entend souvent « crypter » et « cryptage » dans la vie courante. Les spécialistes préfèrent **chiffrer** et **chiffrement** ; on garde « décrypter » pour le cas où l'on n'a pas la clé.*

Mot	Sens
Texte clair	le message avant chiffrement, lisible par tous
Chiffrer	transformer le texte clair en un texte illisible, à l'aide d'une méthode et d'une **clé**
Clé	l'information secrète qui règle la transformation (par exemple un nombre)
Déchiffrer	retrouver le texte clair **en connaissant la clé**
Décrypter	retrouver le texte clair **sans connaître la clé** : on « casse » le chiffrement
Cryptographie	la science des messages secrets

Cours · Coder n'est pas chiffrer

Un **codage** remplace une information par une autre selon une règle **connue de tous** : le code ASCII (chaque caractère a un numéro), le binaire, le code Morse, le QR code. Il sert à **représenter** ou **transmettre** une information, pas à la cacher.

> **À retenir**
>
> Un **chiffrement** repose sur une **clé secrète** : sans elle, le message reste illisible.
>
> Un **codage** n'a pas de secret : tout le monde peut le lire avec la table de correspondance.

Cours · Deux idées très anciennes

Idée	Principe	Exemple historique
Transposition	on **change l'ordre** des lettres, sans les modifier	la scytale, bâton autour duquel on enroule une bande (attribuée aux Spartiates)
Substitution	on **remplace** chaque lettre par une autre	le code de César : selon l'historien Suétone, Jules César décalait les lettres de 3 rangs

> **Un principe moderne (Kerckhoffs, 1883)**
>
> La sécurité d'un système de chiffrement doit reposer sur le secret de **la clé**, et non sur le secret de **la méthode**. On suppose que l'adversaire connaît la méthode.

Exercices · Premiers exercices

Exercice 1 — Où est le cadenas ? 🌶️

Ouvre trois sites que tu connais (un site d'actualités, un réseau social, le site de ton lycée) et regarde le début de chaque adresse.

1. Note pour chaque site si l'adresse commence par `http` ou par `https`.
2. Que signifie le **s** de `https` ?
3. Le chiffrement `https` protège-t-il **ce que fait le site de tes données** une fois qu'il les a reçues ? Justifie.

Réponses dans ton compte rendu (exercice 1)

Exercice 3 — Codé ou chiffré ? 🌶️

Dans le code ASCII, la lettre **A** a pour numéro 65, **B** a pour numéro 66, et ainsi de suite.

1. Retrouve les numéros des lettres **S**, **N** et **T**, puis écris le mot « SNT » sous forme de numéros.
2. Une personne qui n'a jamais vu ton message peut-elle le lire, si elle connaît le code ASCII ?
3. S'agit-il d'un codage ou d'un chiffrement ? Justifie.
4. Donne un autre exemple de codage qui n'est pas secret.

Réponses dans ton compte rendu (exercice 3)

Exercice 5 — Un secret de fabrication ? 🌶️🌶️

Une entreprise affirme : « Notre système de chiffrement est sûr, car personne ne connaît notre méthode. »

1. Que lui répondrais-tu en utilisant le principe de Kerckhoffs ?
2. Que se passe-t-il si un employé mécontent diffuse la méthode, dans un système où seule la **clé** est secrète ?

Réponses dans ton compte rendu (exercice 5)

Partie C2 — Le code de César, à la main

Chiffrer et déchiffrer avec un tableau ou un disque, puis casser un message sans connaître la clé

Bien avant Sherlock Holmes, depuis l'Antiquité, les hommes modifient un texte pour le soustraire à la vue des personnes non autorisées. L'un des premiers procédés est le **code de César**, qui doit son nom à l'empereur romain Jules César.

Cours · Le principe

Il consiste à **décaler chaque lettre de l'alphabet de trois rangs** :

clé 3

```
clair   : ABCDEFGHIJKLMNOPQRSTUVWXYZ
chiffré : DEFGHIJKLMNOPQRSTUVWXYZABC
```

Ainsi le mot « ANTIQUITE » devient « DQWLTXLWH ». De même, le mot chiffré « UDQJHU » se déchiffre en « RANGER ».

Par extension, tout codage obtenu en décalant les lettres **du même nombre de rangs** s'appelle un code de César, et ce nombre s'appelle la **clé**. Avec une clé de 8, A devient I, B devient J, etc. :

clé 8

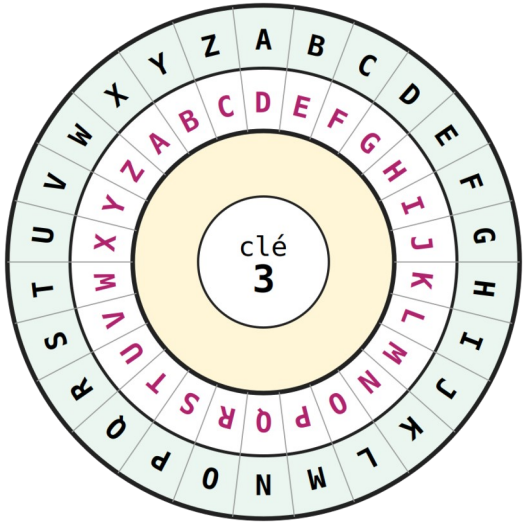
```
clair   : ABCDEFGHIJKLMNOPQRSTUVWXYZ
chiffré : IJKLMNOPQRSTUVWXYZABCDEFGH
```

À retenir

- **Chiffrer** avec la clé k : on décale chaque lettre de k rangs vers la fin de l'alphabet (après Z, on revient à A).
- **Déchiffrer** : on décale de k rangs dans l'autre sens.
- Il n'y a que **25 clés utiles** (la clé 26 ramène à la lettre de départ).

Cours · Le disque de César

Pour ne pas recopier un tableau à chaque nouvelle clé, on utilise **deux disques concentriques** : l'un fixe, l'autre mobile. On tourne le disque mobile pour obtenir le décalage.



Disque de César réglé sur la clé 3 : la lettre A du disque extérieur est en face de D sur le disque intérieur.

Tu peux utiliser le disque interactif : Disque de César. Sur papier, tu peux t'aider du disque dessiné ci-dessus (clé 3).

Exercices · À toi de chiffrer et de déchiffrer

Exercice 6 — Clé 8 🌶

À l'aide du tableau de la clé 8 :

1. Chiffre le texte « LES MATHEMATIQUES SONT LA REINE DES SCIENCES ».
2. Déchiffre le texte « YCQ DI BZMA TWQV UMVIOM AI UWVBCZM ».

Réponses dans ton compte rendu (exercice 6)

Exercice 8 — Retrouver la clé 🌶🌶

Un code de César transforme « SUBSTITUTION » en « KMTKLALMLAGF ».

1. Quelle est la clé ?
2. Explique comment tu l'as trouvée.

Réponses dans ton compte rendu (exercice 8)

Exercices · Sans la clé

Dans les trois exercices suivants, **la clé n'est pas donnée**. Tu peux essayer toutes les clés (méthode appelée **force brute**), ou repérer un mot court ou une lettre fréquente.

Exercice 9 — Un mot de la langue française 🌶🌶🌶

Déchiffre ce texte chiffré par un code de César :

« KXDSMYXCDSDEDSYXXOVVOWOXD OCD VO WYD VO ZVEC VYXQ NO VK VKXQEO PBKXMKSCO »

Réponses dans ton compte rendu (exercice 9)

Partie C3 — Le code de César en Python

ord, chr et un petit problème de bouclage : écrire un crypteur-décrypteur

Demander à Python de décaler des lettres ne marche pas tout seul : il n'existe pas de fonction intégrée pour cela. **Il faut l'inventer.** Pour Python, les lettres n'ont pas de sens particulier : ce sont des caractères stockés sous forme de **nombres**. Notre but est d'écrire une fonction `cryptCesar(texte, cle)`.

Deux fonctions à connaître

- `ord("A")` donne le **numéro** du caractère A dans la table ASCII (ici 65).
- `chr(65)` fait l'inverse : elle donne le **caractère** qui a ce numéro (ici `"A"`).

Les lettres majuscules **A à Z** ont les numéros **65 à 90**.

La table ASCII est consultable sur ascii-code.com.

Exercice 13 — Des lettres aux nombres 🌶

Dans la console de Python :

1. Teste `chr(65)`, `chr(66)`, puis quelques nombres entre 65 et 91. Qu'obtiens-tu ?
2. Teste `ord("A")` et `ord("B")`. Devine ce que donneront `ord("F")` et `ord("Z")`, puis vérifie.
3. À quoi sert la fonction `ord` ? et la fonction `chr` ?

Réponses dans ton compte rendu (exercice 13)

Exercice 14 — Le décalage vu par Python 🌶

1. Compare `ord` d'une lettre de départ et `ord` de la lettre qu'elle devient avec la clé 3 (tableau du cours C2). Fais quelques exemples, puis généralise.
2. Complète l'expression qui transforme `lettre` en lettre chiffrée : `chr(ord(lettre) + ...)`. Vérifie avec `"G"`, `"M"` et `"R"`.

Réponses dans ton compte rendu (exercice 14)

Cours · Une fonction générale

On veut maintenant que la **clé** puisse être n'importe quel nombre. Si elle est **positive**, la fonction chiffre ; si elle est **négative**, elle déchiffre. Par exemple, un texte chiffré avec la clé 5 se déchiffre avec la clé −5.

Exercice 17 — La fonction cryptCesar 🌶🌶🌶

Enregistre le programme ci-dessous sous `NOM_Prenom_ex09.py` dans ton éditeur, puis complète les trois endroits marqués `...` en t'inspirant des exercices précédents. Les lettres sont en **majuscules**, sans accent.

cryptcesar.py

```
def cryptCesar(texte, cle):
    """la clé du codage sera comprise entre -26 et 26"""
    textdep = texte.upper()
    textfin = ""
    for lettre in textdep:
        if 65 <= ord(lettre) <= 90:   # on ne crypte que les lettres
            if ord(lettre) + cle > ...:          # à compléter
                textfin += chr(ord(lettre) + cle - 26)
            elif ...:                            # à compléter
                textfin += chr(ord(lettre) + cle + 26)
            else:
                ...                              # à compléter
        else:                           # la ponctuation est reproduite
            textfin += lettre           # sans changement
    return textfin
```

1. Complète le programme.
2. Teste-le avec `print(cryptCesar("ANTIQUITE", 3))`, puis `print(cryptCesar("DQWLTXLWH", -3))`.
3. Explique la ligne `textdep = texte.upper()` et la raison de sa présence.

Fichier : NOM_Prenom_ex17.py

> **Les accents trahissent**
>
> Un É a un numéro ASCII étendu qui n'est pas entre 65 et 90 : il est **recopié sans changement**. Un message chiffré contenant des accents laisserait donc du texte lisible. On écrit donc les messages **sans accent**.

Exercices · Je m'en sers

Exercice 18 — Les 25 clés d'un coup 🌶🌶🌶

Avec ta fonction `cryptCesar`, écris une boucle qui essaie les 25 clés :

brute.py

```
message = "HQ YROHU OD FOH HVW OH VHFUHW"
for cle in range(1, 26):
    print(cle, cryptCesar(message, -cle))
```

1. Exécute ce programme. Quelle est la clé utilisée pour chiffrer, et quel est le message ?
2. Pourquoi un ordinateur casse-t-il un code de César beaucoup plus vite qu'un humain ?

Fichier : NOM_Prenom_ex18.py

Partie C4 — Casser César : force brute et fréquences

Comme Sherlock : retrouver un message sans la clé, en comptant les lettres

Si on ne connaît pas la clé, le code de César est facile à casser. Deux méthodes :

- **La force brute** : essayer les **25 clés** une par une. C'est long à la main, instantané avec un ordinateur.
- **L'analyse des fréquences** : dans une langue donnée, les lettres n'apparaissent pas toutes aussi souvent. **Le décalage se voit sur la lettre la plus fréquente.**

Cours · Les lettres de la langue française

Le tableau donne la **fréquence d'apparition** (en pourcentage) de chaque lettre dans un texte français de longueur ordinaire (les lettres accentuées sont comptées avec leur lettre de base). Les valeurs sont **indicatives** : elles varient un peu d'une source et d'un texte à l'autre.

Lettre	E	S	A	I	N	T	R	U	L	O
Fréquence (%)	17,52	8,17	8,01	7,35	7,22	7,07	6,69	6,00	5,77	5,43

Lettre	M	V	Q	G	F	H	B	X	J	Y
Fréquence (%)	2,90	1,41	1,14	1,06	1,06	0,88	0,88	0,47	0,44	0,30

La méthode

1. Compter les lettres du message chiffré et calculer la fréquence de chacune : **fréquence (en %) = nombre d'apparitions ÷ nombre total de lettres × 100**.
2. Repérer la lettre la plus fréquente : c'est probablement le **E** du texte clair.
3. En déduire la clé (le décalage entre E et cette lettre), puis déchiffrer tout le message.

Pour aller plus vite, tu peux utiliser l'outil Analyseur de fréquences ou le tableur fourni (`SNT-cesar-frequence_lettres.xls`).

Attention aux unités

Dans le tableau ci-dessus, les fréquences sont en **pourcentage**. Le tableur donne des nombres entre 0 et 1 : multiplie par 100 pour comparer.

Exercices · Je casse des messages

Exercice 20 — Force brute : combien d'essais ? 🌶️

1. Combien de clés faut-il essayer, au maximum, pour casser un code de César ?
2. Si un humain met 20 secondes par essai, combien de temps lui faut-il au maximum ? Donne le résultat en minutes.
3. Un ordinateur effectue un million d'essais par seconde. Combien de temps lui faut-il ?

Réponses dans ton compte rendu (exercice 20)

Exercice 21 — Comme Sherlock 🌶️🌶️

Déchiffre le message suivant, chiffré par un code de César :

« YRF RYRIRF QR PRGGR PYNFFR BAG QR OBAF ERFHYGNGF RA ZNGURZNGVDHRF »

Marche à suivre : compte les lettres du message (il y en a 55), complète le tableau, calcule les fréquences **en pourcentage, arrondies à 0,1 %**, puis compare avec le tableau du français.

Lettre	A	B	C	D	E	F	G	H	I	J
Nombre										
Fréquence (%)										

Lettre	N	O	P	Q	R	S	T	U	V	W
Nombre										
Fréquence (%)										

1. Quelle est la lettre la plus fréquente ? À quelle lettre du texte clair correspond-elle sans doute ?
2. Quelle est la clé ? Déchiffre le message.

Réponses dans ton compte rendu (exercice 21)

Exercices · Compter avec Python

Exercice 23 — Un compteur de lettres 🌶🌶🌶

Le programme suivant compte les lettres d'un texte avec un **dictionnaire** (clé : la lettre, valeur : le nombre d'apparitions). Complète-le, puis enregistre-le sous `NOM_Prenom_ex00.py` (avec le numéro de l'exercice).

compteur.py

```
texte = "YRF RYRIRF QR PRGGR PYNFFR BAG QR OBAF ERFHYGNGF RA
ZNGURZNGVDHRF"
compte = {}
for lettre in texte:
    if lettre != " ":
        if lettre in compte:
            compte[lettre] = compte[lettre] + ...      # à
compléter
        else:
            compte[lettre] = ...                       # à
compléter

print(compte)
```

1. Complète les deux lignes, puis exécute. Retrouve-t-il tes comptages de l'exercice 'Comme Sherlock' ?
2. Ajoute à la fin une boucle qui affiche, pour chaque lettre, sa **fréquence en pourcentage** : `for lettre in compte:` puis un `print(lettre, round(100 * compte[lettre] / 55, 1))`.
3. Pourquoi utiliser 55 est peu pratique ? Propose une façon de calculer le nombre total de lettres dans le programme.

Fichier : NOM_Prenom_ex23.py

Exercice 24 — La lettre la plus fréquente 🌶🌶🌶

À partir de ton dictionnaire `compte`, trouve la lettre la plus fréquente, puis la clé.

max.py

```
lettre_max = ""
nb_max = 0
for lettre in compte:
    if compte[lettre] > nb_max:
        nb_max = ...                                   # à
compléter
        lettre_max = ...                               # à
compléter
cle = (ord(lettre_max) - ord("E")) % 26
print(lettre_max, cle)
print(cryptCesar(texte, -cle))
```

1. Complète les deux lignes. Que fait ce programme ?
2. Pourquoi utilise-t-on `% 26` ?

Fichier : NOM_Prenom_ex24.py

Exercices · Quand l'analyse échoue

Exercice 25 — Les limites 🌶🌶🌶

1. L'analyse des fréquences marche-t-elle bien sur un message de 10 lettres ? Pourquoi ?
2. En 1969, Georges Perec a écrit un roman entier, *La Disparition*, **sans jamais utiliser la lettre E**. Que donnerait la méthode sur un tel texte ?
3. Quelle faiblesse du code de César cette méthode exploite-t-elle ?

Réponses dans ton compte rendu (exercice 25)

Partie C5 — Vigenère : plusieurs décalages

Comment résister à l'analyse des fréquences : changer de clé à chaque lettre

Le code de César est faible car **une lettre est toujours remplacée par la même lettre**. Au XVI^{e} siècle, le diplomate **Blaise de Vigenère** décrit une méthode qui change de décalage à chaque lettre, à l'aide d'un **mot-clé**.

Cours · Le principe

La clé est un **mot**. Chaque lettre du mot donne un décalage : A = 0, B = 1, C = 2, … Z = 25. On utilise les décalages **l'un après l'autre, en boucle**.

Exemple : on chiffre « ECOLE » avec la clé « CLE » (décalages 2, 11 et 4).

Lettre du texte	E	C	O	L	E
Lettre de la clé	C	L	E	C	L
Décalage	2	11	4	2	11
Lettre chiffrée	G	N	S	N	P

À retenir

Le **E** est chiffré en **G** la première fois, en **P** la seconde : **une même lettre du texte clair peut donner des lettres différentes.** Inversement, deux lettres différentes (ici C et L) peuvent donner la même lettre chiffrée (N).

Tu peux essayer avec la machine de Vigenère.

Pour déchiffrer

On connaît la clé : on **décale dans l'autre sens**, avec les mêmes décalages dans le même ordre.

Exercices · Chiffrer à la main, puis comparer

Exercice 26 — Un premier message

1. Chiffre « BONJOUR » avec la clé « LUNE » (décalages 11, 20, 13, 4).
2. Déchiffre « XEL TSFHL ZGZT TTMEAGAWE » avec la clé « MATHS » (décalages 12, 0, 19, 7, 18).

Réponses dans ton compte rendu (exercice 26)

Exercice 27 — César contre Vigenère

Voici le même texte clair, puis deux chiffrements. Le premier est un César de clé 7, le second un Vigenère de clé « LUNE ».

- **César :** THPAYL JVYILHB ZBY BU HYIYL WLYJOL ALUHPA LU ZVU ILJ BU MYVTHNL THPAYL YLUHYK WHY S VKLBY HSSLJOL SBP APUA H WLB WYLZ JL SHUNHNL
- **Vigenère :** XUVXCY PSCVREF MHV FH NVMLR TPLPLP NRRLCG IY MBR MYP YY ZESXUTI XUVXCY EIYUEH AUE P ZXRYC UYPPWUI WOV XTHG E AYH TCYF GP FNRRUTI

1. Avec l'Analyseur de fréquences, colle chaque texte. Quelle est la lettre la plus fréquente, et combien de fois apparaît-elle ?
2. Sur lequel des deux textes la méthode de la partie C4 fonctionne-t-elle ? Pourquoi ?
3. Pourquoi, avec Vigenère, le E n'est-il plus toujours chiffré par la même lettre ?

Réponses dans ton compte rendu (exercice 27)

Cours · Peut-on casser Vigenère ?

Oui, mais c'est plus difficile : le chiffrement de Vigenère a longtemps été surnommé « le chiffre indéchiffrable ». Il a pourtant été cassé au XIX^{e} siècle : si on retrouve la **longueur de la clé** (méthode de Kasiski, 1863), on se ramène à plusieurs César successifs, que l'on casse avec les fréquences.

Une idée pour la suite

Plus la clé est **longue**, plus le chiffrement est solide. Une clé **aussi longue que le message, choisie au hasard et utilisée une seule fois** donne un chiffrement impossible à casser (le « masque jetable »). Mais il faut alors échanger cette clé en secret… ce qui pose un autre problème.

Exercice 29 — Pourquoi c'est plus solide

1. Que se passe-t-il si on choisit une clé d'**une seule lettre** ?
2. Pourquoi une clé plus longue rend-elle l'analyse de fréquences plus difficile ?

3. Pourquoi une clé aussi longue que le message est-elle difficile à utiliser en pratique ?

Réponses dans ton compte rendu (exercice 29)

Partie B1 — Bonus : mots de passe et hachage

Pourquoi un site ne garde pas ton mot de passe, et pourquoi la longueur compte

Pour se connecter, il faut un mot de passe. Un site sérieux **ne le garde pas en clair** : il en garde une **empreinte**. Cette partie, facultative, te montre pourquoi, et pourquoi la **longueur** d'un mot de passe compte plus que sa « difficulté ».

Cours · L'empreinte d'un mot de passe

Une **fonction de hachage** transforme n'importe quel texte en une suite de caractères de longueur fixe, appelée **empreinte** (ou *hash*). Elle a deux propriétés :

- elle est **à sens unique** : connaître l'empreinte ne permet pas de retrouver le mot de passe ;
- un **changement minuscule** du texte change **complètement** l'empreinte.

À la connexion, le site calcule l'empreinte de ce que tu tapes et la compare à celle qu'il a gardée : **il n'a pas besoin de connaître ton mot de passe.**

hash.py

```
import hashlib

def empreinte(texte):
    return hashlib.sha256(texte.encode()).hexdigest()

print(empreinte("snt2nde"))
print(empreinte("snt2ndf"))
```

Cours · Combien de mots de passe possibles ?

Un mot de passe de n caractères choisis parmi a symboles possibles donne a^n mots de passe différents. Pour 8 lettres minuscules : $26^8 = 208\ 827\ 064\ 576$ possibilités.

> **Hypothèse de travail**
>
> On suppose qu'un attaquant teste **un milliard (10^9) de mots de passe par seconde**. C'est une hypothèse de travail pour calculer des ordres de grandeur, pas une mesure.

Exercices · Exercices

Exercice 31 — Le temps de casser

On suppose un milliard d'essais par seconde.

1. Calcule le nombre de mots de passe possibles pour **6**, **8**, **10** puis **12** lettres minuscules (tu peux utiliser Python : `26 ** 8`).
2. Pour chacun, calcule la durée maximale pour tous les tester, en secondes, puis dans une unité adaptée (minutes, jours, années).
3. Un mot de passe de 8 caractères peut utiliser des minuscules, des majuscules et des chiffres, soit 62 symboles : calcule 62^8 et la durée correspondante.
4. Que préfères-tu, pour la sécurité : ajouter des types de caractères ou ajouter des caractères ? Justifie avec les résultats.

Réponses dans ton compte rendu (exercice 31)

Partie B2 — Bonus : cacher un message dans une image

La stéganographie : ne pas chiffrer, mais dissimuler

Chiffrer un message le rend **illisible**, mais cela se **voit**. La **stéganographie** fait autre chose : elle **cache l'existence même** du message, par exemple dans une photo. Cette partie, facultative, réutilise ce que tu as vu dans le thème Photographie numérique.

Cours · Le dernier bit

Dans une image RVB, chaque couleur d'un pixel est un nombre de 0 à 255, c'est-à-dire un **octet**. Changer ce nombre de **1** ne se voit pas à l'œil : 200 et 201 donnent

deux rouges quasi identiques. On peut donc utiliser **le dernier bit** de chaque nombre pour stocker un bit du message.

Le dernier bit, en Python

- `200 % 2` donne `0` (nombre pair) ; `201 % 2` donne `1` (nombre impair).
- Le dernier bit d'un nombre est donc `nombre % 2`.

Dans ce bonus, on cache le message dans le **canal rouge** des pixels, en lisant l'image ligne par ligne. Le message est écrit en **code ASCII**, 8 bits par caractère. Le fichier `image_secrete.png` (dans l'archive à télécharger : Cryptage.zip) contient un message caché. *On utilise un fichier PNG : un fichier JPEG modifierait les valeurs des pixels.*

Exercice 34 — Lire le message caché 🌶🌶

Enregistre `image_secrete.png` dans le même dossier que ton programme, puis complète les deux lignes du programme (l'extraction du dernier bit, puis la conversion en caractère).

lire.py

```
from PIL import Image

img = Image.open("image_secrete.png").convert("RGB")
largeur, hauteur = img.size

bits = ""
for y in range(hauteur):
    for x in range(largeur):
        r, v, b = img.getpixel((x, y))
        bits = bits + str(r % ...)                # à compléter : le dernier bit de r

message = ""
for i in range(0, len(bits) - 7, 8):
    octet = bits[i:i + 8]
    code = int(octet, 2)
    if code == 0:
        break
    message = message + chr(...)                  # à compléter

print(message)
```

1. Complète, exécute : quel est le message caché ?
2. À quoi sert `int(octet, 2)` ?
3. Que signifie `if code == 0: break` ?

Fichier : NOM_Prenom_ex34.py