TP – CHAPITRE 10

Sommaire

TP1 - Alphabet Majuscule

- Sur votre compte créer un dossier **TP-Chapitre910** dans lequel on va enregistrer les TP de ce chapitre. Récupérer le fichier **alphabet.py** et l'enregistrer dans le dossier **TP-Chapitre10**.
- Rappels :

  - Le **code ASCII** des majuscules va de 65 pour A à 90 pour Z .
  - La fonction python **chr(n)** retourne le caractère (type str) correspondant au code ASCII **n** de type int.
  - La fonction python **ord(c)** retourne le code ASCII (type int) correspondant au caractère **c** de type str.

- Dans le fichier **alphabet.py** compléter le code des fonctions **rang(c)** et **majuscule(n)** .
- Exemple de retours attendus dans la console.

```
>>> rang('A')
0
>>> rang('B')
1
>>> rang('Z')
25
>>> majuscule(0)
'A'
>>> majuscule(25)
'Z'
```

Lever la main pour valider ce TP.

TP2 - Chiffrer avec le code de César

- Récupérer le fichier **cesar.py** et l'enregistrer dans le dossier **TP-Chapitre10**.
- Compléter le code de la fonction **chiffrer_cesar(texte, cle)** pour que cette fonction retourne le texte chiffré avec le code de César et la clé passée en paramètre.
- Exemple de retours attendus dans la console.

```
>>> chiffrer_cesar('PYTHON',3)
'SBWKRQ'
>>> chiffrer_cesar('ZOO',3)
'CRR'
```

Lever la main pour valider ce TP.

TP3 - Déchiffrer avec le code de César

- Dans le fichier **cesar.py** rédiger une fonction **dechiffrer_cesar(texte, cle)** qui déchiffre le texte codé passé en paramètre avec le code de César et la clé passée en paramètre.
- Exemple de retours attendus dans la console.

```
>>> chiffrer_cesar('PYTHON',3)
'SBWKRQ'
>>> dechiffrer_cesar('SBWKRQ',3)
'PYTHON'
>>> chiffrer_cesar('BONJOUR',7)
'IVUQVBY'
>>> dechiffrer_cesar('IVUQVBY',7)
'BONJOUR'
```

Lever la main pour valider ce TP.

TP4 - Chiffrer avec le code de Vigenère

- Récupérer le fichier **vigenere.py** et l'enregistrer dans le dossier **TP-Chapitre10**.
- Compléter le code de la fonction **chiffrer_vigenere(texte, cle)** pour que cette fonction retourne le texte chiffré avec le code de Vigenère et la clé passée en paramètre.
- Exemple de retours attendus dans la console.

```
>>> chiffrer_vigenere('PYTHON', 'AB')
'PZTIOO'
>>> chiffrer_vigenere('PYTHON', 'NSI')
'CQBUGV'
>>> chiffrer_vigenere('MATH', 'BC')
'NCUJ'
```

Lever la main pour valider ce TP.

TP5 - Déchiffrer avec le code de Vigenère

- Dans le fichier **vigenere.py** rédiger une fonction **dechiffrer_vigenere(texte, cle)** qui déchiffre le texte codé passé en paramètre avec le code de Vigenère et la clé passée en paramètre.
- Exemple de retours attendus dans la console.

```
>>> chiffrer_vigenere('PYTHON', 'NSI')
'CQBUGV'
>>> dechiffrer_vigenere('CQBUGV', 'NSI')
'PYTHON'
>>> chiffrer_vigenere('MATH', 'BC')
'NCUJ'
>>> dechiffrer_vigenere('NCUJ', 'BC')
'MATH'
```

Lever la main pour valider ce TP.

TP6 - Chiffrer avec le codage affine

- Récupérer le fichier **affine.py** et l'enregistrer dans le dossier **TP-Chapitre10**.
- Compléter le code de la fonction **chiffrer_affine(texte, a, b)** pour que cette fonction retourne le texte chiffré avec le chiffrement affine avec la fonction affine f définie par $f(x) = ax + b$.
- Exemple de retours attendus dans la console.

```
>>> chiffrer_affine('NSI', 3, 2)
'PEA'
>>> chiffrer_affine('PYTHON', 5, 1)
'YRSKTO'
>>> chiffrer_affine('LAC', 3, 5)
'MFL'
```

Lever la main pour valider ce TP.

TP7 - Déchiffrer avec le codage affine

- Dans le fichier **affine.py** rédiger une fonction **inverser_modulo_26(n)** qui retourne l'inverse de l'entier n modulo 26 , c'est à dire l'entier m compris entre 0 et 25 tel que **n * m % 26 == 1** .
- Dans le fichier **affine.py** rédiger une fonction **dechiffrer_affine(texte, a, b)** qui retourne le texte déchiffré du texte codé passé en paramètre pour un chiffrement affine avec la fonction affine f définie par $f(x) = ax + b$. Pour cela, il faut calculer a1 avec la fonction **inverser_modulo_26(n)** puis calculer b1 (voir cours).
- Exemple de retours attendus dans la console.

```
>>> chiffrer_affine('LAC', 3, 5)
'MFL'
>>> dechiffrer_affine('MFL', 3, 5)
'LAC'
>>> chiffrer_affine('PYTHON', 5, 11)
'IBCUDY'
>>> dechiffrer_affine('IBCUDY', 5, 11)
'PYTHON'
```

Lever la main pour valider ce TP.

TP8 - Convertir du décimal au binaire

- Récupérer le fichier **xor.py** et l'enregistrer dans le dossier **TP-Chapitre10**.
- Dans le fichier **xor.py** dans la fonction **binaire(n)** rédiger une instruction **assert** pour vérifier que le paramètre est compris entre 0 et 255 .
- Exemple de retours attendus dans la console.

```
>>> binaire(400)
...
AssertionError: Le paramètre doit être entre 0 et 255.
```

- Rédiger ensuite le code de la fonction binaire. pour cela, il faut utiliser la fonction **bin(n)** qui retourne l'écriture binaire de n précédée de '0b' .

```
>>> binaire(15)
'0b1111'
```

- Utiliser un slice pour enlever les deux premiers caractères '0b' puis utiliser une boucle **while** pour compléter l'écriture binaire à 8 bits en rajoutant si nécessaire des '0' a gauche.
- Exemple de retours attendus dans la console.

```
>>> binaire(15)
'00001111'
>>> binaire(32)
'00100000'
>>> binaire(65)
'01000001'
```

Lever la main pour valider ce TP.

TP9 - Chiffrer avec le codage XOR

- Dans le fichier **xor.py** compléter le code de la fonction **chiffrer_xor(texte, cle)** pour qu'elle retourne la chaîne binaire de type str cryptée du texte avec le code XOR et la clé passée en paramètre.
- Exemple de retours attendus dans la console.

```
>>> chiffrer_xor('BAC', 'DE')
'000001100000010000000111'
>>> chiffrer_xor('PYTHON', 'NSI')
'000111100000101000011101000001100001110000000111'
```

Lever la main pour valider ce TP.

TP10 - Déchiffrer avec le codage XOR

- Dans le fichier **xor.py** compléter le code de la fonction **dechiffrer_xor(texte, cle)** pour qu'elle retourne la chaîne de type str donnant le texte déchiffré correspondant au texte chiffré passé en paramètre avec le code XOR et la clé passée en paramètre.
- On pourra utiliser la fonction **int( )** qui permet de convertir une chaîne binaire en entier de type int.

```
>>> int('101', 2)
5
```

- Exemple de retours attendus dans la console.

```
>>> dechiffrer_xor('000001100000010000000111', 'DE')
'BAC'
```

Lever la main pour valider ce TP.

TP11 - Chiffrer et déchiffrer avec le code RSA

- Récupérer le fichier **rsa.py** et l'enregistrer dans le dossier **TP-Chapitre10**.
- Compléter le code de la fonction **puissance_mod(m, e, n)** qui retourne le calcul de m^e modulo n en utilisant une boucle while.
- Exemple de retours attendus dans la console :

```
>>> puissance_mod(13, 7, 33)
7
>>> puissance_mod(102, 41, 1147)
262
>>> puissance_mod(262, 1001, 1147)
102
```

- On considère la clé publique (43, 527) .
  En faisant des appels dans la console avec la fonction **puissance_mod( )** compléter le tableau suivant qui chiffre des nombres avec cet clé publique du code rsa.

Nombre clair	**80**	**65**	**82**	**67**
Nombre chiffré	516	520	10	67

- La clé privée correspondante est (67, 527) .
  Recopier la deuxième ligne obtenue dans le tableau ci-dessus, dans la première ligne du tableau ci-dessous.
  Puis en faisant des appels dans la console avec la fonction **puissance_mod( )** compléter le tableau suivant qui déchiffre les nombres avec cet clé privée du code rsa.

Nombre clair	516	520	10	67
Nombre chiffré	80	65	82	67

Lever la main pour valider ce TP.

TP12 - Chiffrer un texte avec le code RSA

- Dans le fichier **rsa.py** **c**ompléter le code de la fonction **chiffrer_rsa(texte, e, n)** pour que cette fonction retourne le chiffrement des codes ASCII de chaque caractères du texte en utilisant le code RSA avec la clé publique (e, n) . Cette fonction retournera une chaîne de type str qui contient les codes chiffrés séparés par des espaces.
- Exemple de retours attendus dans la console.

```
>>> chiffrer_rsa('PARC', 43, 527)
'516 520 10 67'
>>> chiffrer_rsa('LAC', 43, 527)
'121 520 67'
```

Lever la main pour valider ce TP.

TP13 - Déchiffrer un texte avec le code RSA

- Dans le fichier **rsa.py** **c**ompléter le code de la fonction **dechiffrer_rsa(texte, d, n)** pour que cette fonction retourne le text clair déchiffré en utilisant la clé privée (d, n) du code RSA.
- Exemple de retours attendus dans la console.

```
>>> dechiffrer_rsa('121 520 67', 67, 527)
'LAC'
>>> dechiffrer_rsa('516 520 10 67', 67, 527)
'PARC'
```

Lever la main pour valider ce TP.

TP14 - Principe du protocole HTTPS

- Compléter les zones dans le schéma ci-dessous.

1.

Client

Requête HTTPS

Serveur

2.

Client

Clé publique RSA : (43, 527)

Serveur

3.

Client

Clé symétrique code XOR : 'BAC'

Clé symétrique chiffrée avec le code RSA :

281 520 67

Serveur

Clé symétrique déchiffrée avec la clé privée (67, 527) : 'BAC'

4.

Client

Envoie du message 'SALUT' chiffrée avec la clé symétrique:

0001000100000000000011111000101110001010 1

Serveur

Lever la main pour valider ce TP.