Examen Programmation Avancée en Python (PAPY)

Examinateur: Lucas Lestandi

27 Octobre 2022

Durée : 2h (2h40 pour les étudiants disposant d'un tiers temps)

Déroulement: L'examen se passe en deux parties:

1. Sur papier, vous répondrez aux questions ci-dessous. max 1h
2. **Seulement après** avoir rendu la partie écrite, vous répondrez aux questions sur machine à l'aide de votre machine personelle. Le notebook est à télécharger sur hippocampus et remettre sur cette même plateforme.

1 Partie 1: Connaissances générales [9pts]

1.1 Connaissance de python [3,5pts]

1. En python, il existe de types muables (*mutables*) et immuables (*immutable*).
    1. Donner la définition de chacun.
    2. Donner 2 exemples pour chaque.
2. Les `listes` et les `tuples` permettent de stocker des objets assimilables à des vecteurs.
    1. Si la variable `v` stocke un vecteur de taille 3, comment changer la seconde valeur et lui affecter `new_val` dans le cas ou `v` est un `tuple`? Faire de même si `v` est une liste.
    2. Comment allonger `v` d'un élément valant `new_val` dans les deux cas?
3. Présenter la structure du dictionnaire (`dict`) en python. On précisera la déclaration d'un dictionnaire qui stock deux éléments et comment afficher l'une de ces valeurs.
4. Qu'est-ce qu'un docstring?
5. Pour le code suivant:

```
a=3
b=2
c=a
a=4
print(c)
```

   Quelle est la valeur de `c` affichée par le print? Expliquer (ou pourra s'aider d'un schéma).

6. Soit L une liste: `L=[1,2,3]` et `L2=L`
    1. Que vaut L après l'opération suivante?

```
L2='a'
```

       Expliquer le résultat
    2. De même que vaut `L` après les opérations ci-dessous.

```
L=[1,2,3]
L2=L
L2[1]='a'
```

Expliquer le résultat.

7. Qu'est-ce qu'un module?
8. Comment importer le module `numpy` dans un autre script python et utiliser la fonction `linspace`. Donner deux façons de faire.

1.2 Être un bon développeur [2,5pts]

1. Pourquoi utiliser des environnements en python (par exemple avec le gestionnaire anaconda)?
2. Comment installer une bibliothèque python, par exemple `numpy`?
3. Qu'est-ce qu'un décorateur et à quoi sert-il? Comment l'utiliser?
4. Pourquoi tester son code? Proposer un méthodologie.
5. On souhaite créer une variable qui représente la constante gravitationnelle $g=9.81m/s^2$ dans tout notre programme. Proposer un nom. En cas de doute sur la casse des lettres, où trouver l'information sur l'usage recommandé en python.

1.3 Commentaire de code [3pts]

On se propose d'améliorer la fonction `LogSum` *suivante tant en termes d'efficacité que de bonnes pratiques. Elle calcule le logarithme des éléments d'un vecteur (itérable) et les additionne. On considère que* `log` *est bien définie dans le module.*

```
1 def LogSum(array):
2     res=0
3     for entry in array:
4         res+=log(entry)
5     return res
```

1. Pour l'appel de fonction suivante `logsum([-1,5,10])`
   - Quelle la première valeur d'`entry`?
   - Que se passera-t-il à la ligne 4 pour cette première valeur?
   - Proposer une solution
2. Cette fonction est simple, mais il n'est pas possible, en l'état, de l'utiliser sans lire attentivement le code source pour éviter des erreurs. Proposer une solution à ce problème. (ne pas réécrire le code)
3. Expliquer les grands principes de l'optimisation du temps de calcul en python.
4. Proposer une réécriture de la fonction `logsum` qui suit met en oeuvre les trois réponses aux 3 questions précédentes. *Le critère d'évaluation de cette question n'est pas l'exactitude de la syntaxe (ce sera évalué dans la partie sur machine. Ici on se concentre sur la conception du code.*

Lorsque vous avez répondu aux questions ci-dessus (parties 1,2 et 3). RENDRE VOTRE COPIE. Vous pourrez ensuite utiliser votre ordinateur. Et répondre aux questions suivantes.

---

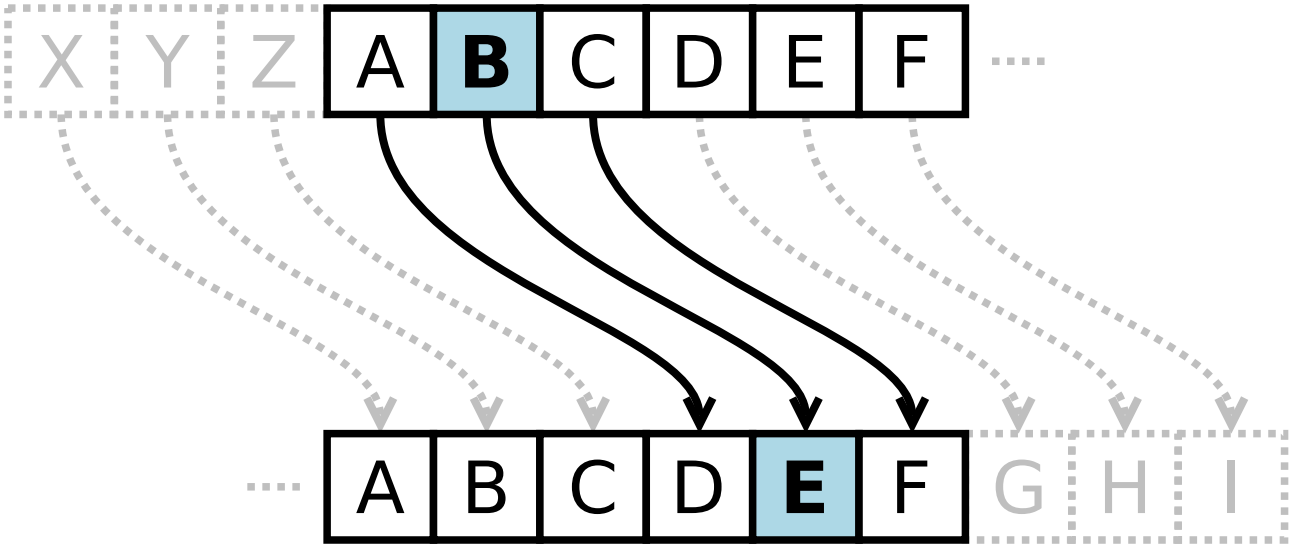
2 Partie 2: Programmation sur machine [12pts]

Vous avez terminé la partie connaissance du python de cet examen. Maintenant, vous devez proposer un programme qui réalise le chiffrement suivant.

2.1 Présentation du problème: Chiffrment par dictionnaire

2.1.1 Chiffrement de César

En cryptographie, le chiffrement par décalage, aussi connu comme le chiffre de César, est une méthode de chiffrement très simple utilisée par Jules César dans ses correspondances secrètes.



Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une permutation circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue la clé du chiffrement qu'il suffit de transmettre au destinataire pour que celui-ci puisse déchiffrer le message.

2.1.1.1 Example Le chiffrement peut être représenté par la superposition de deux alphabets, l'alphabet clair présenté dans l'ordre normal et l'alphabet chiffré décalé, à gauche ou à droite, du nombre de lettres voulu. Nous avons ci-dessous l'exemple d'un encodage de 3 lettres vers la droite. Le paramètre de décalage (ici 3) est la clé de chiffrement :

clair	ABCDEFGHIJKLMNOPQRSTUVWXYZ
chiffré	DEFGHIJKLMNOPQRSTUVWXYZABC

Pour encoder un message, il suffit de regarder chaque lettre du message clair, et d'écrire la lettre encodée correspondante. Pour déchiffrer, on fait tout simplement l'inverse.

clair	WIKIPEDIA L'ENCYCLOPEDIE LIBRE
chiffré	ZLNLSHGLD O'HQFBFORSHGLH OLEUH

Source: Wikipedia

2.1.2 Chiffre de Vigenère

Le chiffrage ci-dessus est très faible, même pour une longue table de caractères, par exemple ASCII.

Pour améliorer la sécurité de notre chiffrement, on introduit un chiffrage plus complexe (aussi connue sous le nom de chiffre de Vigenère). Cette fois-ci, au lieu d'utiliser le même décalage pour tous les caractères, on va faire varier le décalage (shift) à chaque caractère. On utilisera pour cela une clé de chiffrage sous la forme d'une chaine de caractères (un mot ou une phrase) pour laquelle chaque lettre détermine le décalage (place dans la table de caractères). Lorsque la clé est épuisée, on reprend au premier élément de celle-ci.

Pour un indice donné `i`, on chiffrera le i-eme caractère du texte `texte[i]` en applicant le décalage donné par la position de la `clé[i]` dans le dictionnaire i.e. `shift[i]=position(clé[i],dic)`.

```
alpha="ABCDEFGHIJKLMNOPQRSTUVWXYZ"

Texte en clair :   j'adore ecouter la radio toute la journee
Clé répétée    :   M USIQU EMUSIQU EM USIQU EMUSI QU EMUSIQU
                   ^ ^^^
                   | ||Texte O, clé I : shift=8, on obtient le chiffré W
                   | |Texte D, clé S : shift=8, on obtient le chiffré L
                   | Texte A, clé U : shift=8, on obtient le chiffré I
                   Texte J, clé M : shift=8, on obtient le chiffré R
```

Dans le notebook d'examen, répondre aux questions suivantes

2.2 Implémentation du Code César [2pts]

1. Écrire une fonction `caesar(message, alpha, shift)-> str` du codage César. Elle prend pour argument le texte (`message`), la clé (`shift`) et l'alphabet `alpha` et renvoie le texte chiffré. Vous testerez de façon adéquate avec l'alphabet standard `"ABCDEFGHIJKLMNOPQRSTUVWXYZ"`.
2. Encoder le message suivant : "Tu quoque mi fili" avec un décalage de 8.
3. Décoder ce message (en anglais): "ROVVY GYBVN" et en donner la clé.

On va maintenant utiliser les caractères de la table ASCII imprimable qui va assurer le bon fonctionnement sur un texte en anglais (sans accent). On peut obtenir avec la commande suivante:

```
>>> [chr(i) for i in range(32,127)]
[' ', '!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',',
'-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':',
';', '<', '=', '>', '?', '@', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
 'V', 'W', 'X', 'Y', 'Z', '[', '\\', ']', '^', '_', '`', 'a', 'b',
 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '{', '|', '}
 ', '~']
```

2.3 Chiffre de Vigenère avec une classe [4pts]

1. Créer une classe de chiffrement `Vigenere` qui stocke la clé `key` et le tableau de chiffrement `alpha` et qui possède la méthode `code` et `decode`. On s'assurera que l'exemple d'implémentation suivant fonctionne bien:

```
>>> ASCII=[chr(i) for i in range(32,127)]
>>> enigma=Vignemere(alpha=ASCII,key="my fancy key!")
>>> print(enigma)
Vigenere encoder-decoder with
key="my fancy key!"
```

```
alpha=[' ', '!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',
', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
':', ';', '<', '=', '>', '?', '@', 'A', 'B', 'C', 'D', 'E', 'F', 'G',
'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
'V', 'W', 'X', 'Y', 'Z', '[', '\\', ']', '^', '_', '`', 'a', 'b',
'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '{', '|', '}', '~']
>>> coded_farm=enigma.code(animal_farm)
>>> enigma.decode(coded_farm)
Mr. Jones, of the Manor Farm, had locked the hen-houses for the
night, but was too drunk to remember to shut the pop-holes...
```

2. Exécuter l'algorithme sur l'exemple de texte ci-dessous avec la clé `"George Orwell wrote more than 1984"` et stocker le résultat dans la variable `code_farm`.

Exemple de texte suffisamment long pour obtenir de bonnes statistiques. Animal Farm, George Orwell (1945) en version originale:

Mr. Jones, of the Manor Farm, had locked the hen-houses for the night, but was too drunk to remember to shut the pop-holes. With the ring of light from his lantern dancing from side to side, he lurched across the yard, kicked off his boots at the back door, drew himself a last glass of beer from the barrel in the scullery, and made his way up to bed, where Mrs. Jones was already snoring.

... La suite est fournie dans le notebook.

2.4 Codage et décodage d'un vrai texte

On va maintenant démontrer la faiblesse du code César et la relative force du code de Vigenère.

2.4.1 Casser César [4pts]

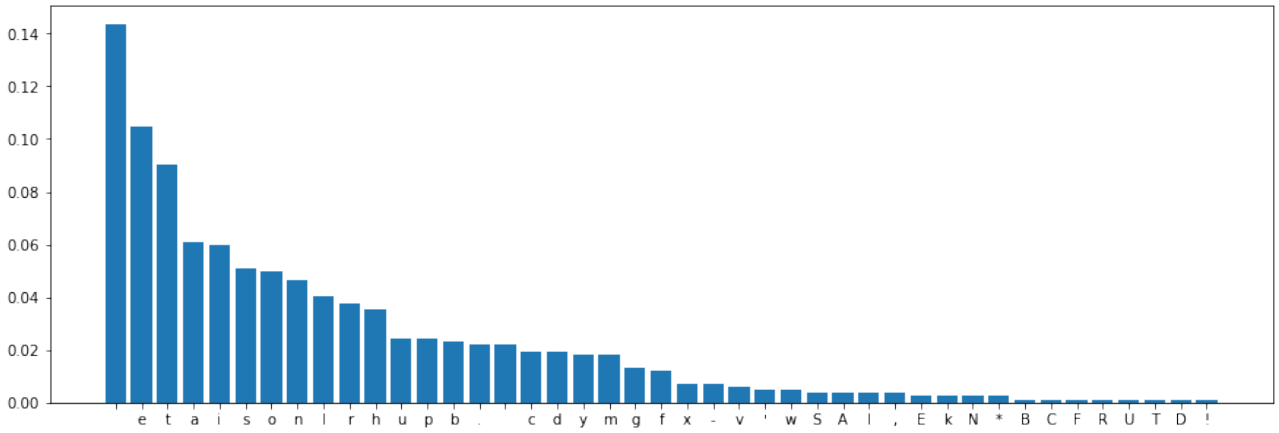
Dans cet exercice, on va attaquer le texte chiffré (avec la table `ASCII`) par la méthode César suivant:

```
message_secret="""Lok ~sp v*s}*lo~~o|*~rkx* qv$8
O#zvsms~*s}*lo~~o|*~rkx*swzvsms~8
]swzvo*s}*lo~~o|*~rkx*mywzvo#8
Mywzvo#*s}*lo~~o|*~rkx*mywzvsmk~on8
Pvk~*s}*lo~~o|*~rkx*xo}~on8
]zk|}o*s}*lo~~o|*~rkx*nox}o8
\oknklsvs~$*my x~}8
]zomskv*mk}o}*k|ox1~*}zomskv*oxy qr*~y*l|oku*~ro*| vo}8
Kv~ry qr*z|km~smkvs~$*lok~}*z |s~$8
O||y|}*}ry vn*xo!o|*zk}}*}svox~v$8
_xvo}}*o#zvsms~v$*}svoxmon8
Sx*~ro*pkmo*yp*kwlsq s~$6*|op }o*~ro*~owz~k~syx*~y*q o}}8
^ro|o*}ry vn*lo*yxo77*kxn*z|opo|klv$*yxv$*yxo*77yl!sy }*"k$*~y*ny*s~8
Kv~ry qr*~rk~*"k$*wk$*xy~*lo*yl!sy }*k~*ps|}~* xvo}}*$y 1|o*N ~mr8
Xy"*s}*lo~~o|*~rkx*xo!o|8
Kv~ry qr*xo!o|*s}*yp~ox*lo~~o|*~rkx*4|sqr~4*xy"8
Sp*~ro*swzvowox~k~syx*s}*rk|n*~y*o#zvksx6*s~1}*k*lkn*snok8
Sp*~ro*swzvowox~k~syx*s}*ok}$*~y*o#zvksx6*s~*wk$*lo*k*qyyn*snok8
Xkwo}zkmo}*k|o*yxo*ryxusxq*q|ok~*snok*77*vo~1}*ny*wy|o*yp*~ry}o+"""
```

Puisqu'il est mono-dictionnaire, on propose de l'attaquer par une analyse fréquentielle. En effet, la fréquence des lettres dans une langue donnée tend vers une valeur connue si le texte est suffisamment long et varié. Par exemple, on retrouve les 5 lettres les plus utilisées dans les textes en anglais.

Lettre	Fréquence
E	13%
T	9.1%
A	8.2%
O	7.5%
I	7%

On aura par exemple la répartition suivante sur le texte clair de notre `message_secret`.



1. Écrire une fonction `compute_frequency` qui calcule la fréquence des caractères dans un texte donné.

```
>>> compute_frequency(message)
'*': 0.14355231143552297,
 'o': 0.10462287104622861,
 '~': 0.09002433090024323,
 'k': 0.06082725060827245,
 's': 0.059610705596107004
```

2. Grâce à `matplotlib` et en particulier `plt.bar(coord,height, label=...)`, proposer un graphe similaire à celui présenté au-dessus.
3. À partir de `compute_frequency`, décoder le `message_secret` sans utiliser la force brute.

2.4.2 Se casser les dents sur Vigenère [2pts]

À l'inverse, le code de Vigenère ne peut pas être attaqué directement par recherche fréquentielle.

1. Utiliser `compute_frequency` sur le résultat de codage Vigenère `code_farm`. Expliquer le résultat.
2. À la lumière de cette analyse, proposer une méthode ou un critère qui assure une bonne robustesse pour le code de Vigenère.
3. Proposer un algorithme d'attaque par force brute.
    1. Quel est le nombre d'évaluations en fonction de la longueur de la clé?
    2. Le tester sur le texte chiffré suivant (clé à deux caractères). `'!)9-Jy4(4C!-.-'`