Nom :		Prénom :	

Informatique Pour Tous
Interrogation n°2

Conversions

1. Donner les résultats des conversions suivantes :	
a) Donner en base 10 la valeur de 10011000 considéré comme un octet signé en machine	**–104**
b) Donner en base 10 la valeur de 10100011 considéré comme un octet signé en machine	**–93**

Syntaxe Python

2. Pour arrêter l'exécution d'une boucle (qui n'est pas incluse dans une fonction), on utilise :			
☐ l'instruction **return**	☐ l'instruction **pass**	☐☒ l'instruction **break**	☐ aucun moyen

3. Préciser le type des variables dans la série d'instructions suivantes :			
`v0 = 2 == 3`	`type(v0): booléen`	`v5=(4,5)`	`type(v5): tuple`
`v1 = '3 > 2'`	`type(v1): str`	`v6 = v5[1]`	`type(v6): int`
`v2 = 4//2`	`type(v2): int`	`v7 = v5[0] < v5[1]`	`type(v7): booléen`
`v3 = input(' entrez un entier ')`	`type(v3): str`	`v8 = [v5]`	`type(v8): liste`
`v4 =(4 + 2)* 's'`	`type(v4): str`	`v9 = v2 / 2`	`type(v9): float`

4. Soit une liste `lst` *définie par :* `lst = ["mot",5 < 3, 5 - 3,(5,3)]` *Indiquez les valeurs des variables suivantes :*			
`v1 = lst[1]`	`v1 = False`	`v4 = lst[0]*lst[2]`	`v4 = 'motmot'`
`v2 = lst[3][1]`	`v2 = 3`	`v5 = lst[0][1]`	`v5 = 'o'`
`v3 = lst[-2]`	`v3 = 2`	`v6 = lst[1:3]*lst[2]`	`v6 = [False, 2, False, 2]`

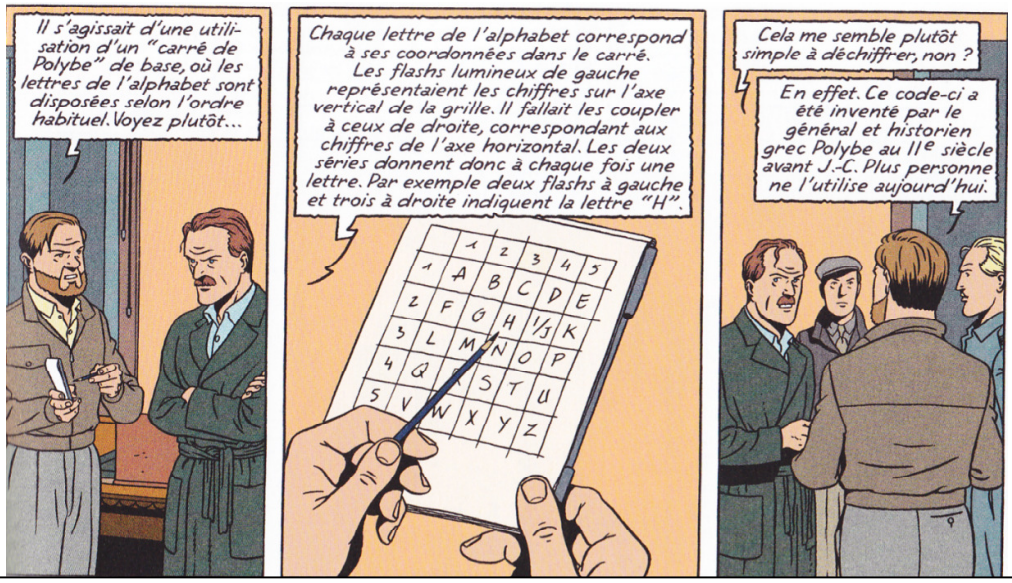
5. Quelles sont les valeurs entières produites par l'instruction `range(2,12,3)` ?
2, 5, 8, 11 **#range s'arrête à 12 – 1**

6. Quelle est l'affichage produit par le programme suivant :	

`def f(n) :`
`  s = 1`
`  for k in range(n+1) :`
`  s += k*s`
`  return s`
`print(f(4))`
	120

Algorithmique

Dans le dernier épisode paru des aventures de Blake et Mortimer, on trouve les explications suivantes :



extrait de « Le bâton de Plutarque » *Yves Sente et André Julliard* d'après les personnages de E.P.Jacobs

Le problème qui suit cherche à utiliser ce mode cryptage de manière informatique.

7. Création du carré de Polybe

a) Écrire le contenu du carré de Polybe(les lettres de `'A'` à `'Y'`, en distinguant `'I'` et `'J'`) sous la forme d'une liste de listes. Cette liste sera appelée `carrepol`.

carrepol = [['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']]

b) On rappelle que `chr(65)` renvoie `'A'`, `chr(66)` renvoie `'B'`,..., `chr(90)` renvoie `'Z'`. Écrire en Python une fonction `remplir_ligne()` qui renvoie la liste `['A','B','C','D','E']` sans écrire ses 5 éléments à la main.

```
1 def remplir_ligne() :
2     ligne = []
3     for i in range(5) :
4         ligne.append(chr(65 + i))
5     return ligne


ou bien ligne = [chr(65+i) for i in range(5)] à la place des lignes 2 à 4
```

c) En déduire la fonction `remplir_ligne(n)` qui renvoie la ligne d'indice n du tableau (n compris entre 0 et 4)

```
1 def remplir_ligne(n) :
2     ligne = []
3     for i in range(5) :
4         ligne.append(chr(65 + n*5 + i))
5     return ligne


ou bien ligne = [chr(65 + n*5+ i) for i in range(5)] à la place des
lignes 2 à 4.
```

d) Écrire en Python une fonction `remplir_carre()` *qui renvoie la liste* `carrepol`*, sans écrire les 25 éléments à la main.*

```
1 def remplir_carre() :
2          carre_pol=[]
3          for i in range(5) :
4                  ligne = remplir_ligne(i)
5                  carre_pol.append(ligne)
6          return carre_pol
```

8. Codage-décodage simple

a) Écrire une fonction `code(texte)` *qui transforme la chaîne de caractères* `texte` *en une liste de tuples, chacun d'eux étant une paire d'entiers compris entre 0 et 4, le premier indiquant la ligne et le second la colonne correspondant à la position de la lettre dans le carré de Polybe (au sens de la question précédente).*

Par exemple, `code('PYTHON')` *renvoie la liste* `[(3,0), (4,4), (3,4),(1,2),(2,4),(2,3)]`.

On rappelle que `ord('A')` *renvoie l'entier 65,* `ord('B')` *renvoie l'entier 66, ...,* `ord('Z')` *renvoie 90.*

Indication : cette fonction ne nécessite pas d'utiliser la variable `carrepol`*.*

```
1 def code(texte) :
2        result = []         #on crée une liste vide
3        for let in texte : # on prend chaque lettre de texte
4                  n = ord(let)-65 # on ramène les codes entre 0 et 25
5                  ligne = n //5   # indice de la ligne
6                  col = n%5       #indice de la colonne
7                  result.append((ligne,col))
8         return result
```

b) Écrire une fonction decode(liste) qui transforme une liste de tuples donnant chacun les coordonnées d'une lettre dans le carré de Polybe (au sens de la question précédente) en une chaîne de caractère.

Par exemple `decode([(3, 0), (2, 4), (3, 4), (1, 2), (1, 3), (0, 4),(3,2)])` *renvoie la chaîne* `'POTHIER'`.

```
1 def decode(liste) :
2     result = ''     #on crée une chaîne vide
3     for tup in liste:
4         let = chr(5*tup[0]+tup[1]+65) #on fabrique le code à partir des
                                        coordonnées
5         result += let  # on rajoute la lettre à la chaine
6     return result
```

9. Première variante

On applique la fonction `mystere` *ci-dessous à la variable* `carrepol` *obtenue ci-dessus.*

```
1 def mystere(T) :
2     NewT = [0]*len(T[0])
3     for i in range(len(T[0])):
4         NewT[i] = [0]*len(T[0])
5     for i in range(len(T[0])):
6         for j in range(len(T[0])):
7             NewT[i][len(T[0])-1-j] = T[j][i]
8     return NewT
```

a) Quelle est la valeur de la variable NewT après exécution de la ligne 4

```
NewT = [[0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0], [0,0,0,0,0]]
```

b) Pour la valeur i = 0 de la ligne 5, indiquer les valeurs successives de NewT[0] pour les différentes valeurs possibles de j de lignes 6 et 7.

j	NewT[0]
0	`[0, 0, 0, 0, 'A']`
1	`[0, 0, 0, 'F', 'A']`
2	`[0, 0, 'K', 'F', 'A']`
3	`[0, 'P', 'K', 'F', 'A']`
4	`['U', 'P', 'K', 'F', 'A']`

c) Indiquer la valeur de NewT[1] après exécution complète de la boucle des lignes 6-7.

```
NewT[1] =['V', 'Q', 'L', 'G', 'B']
```

d) En déduire la valeur de l'objet retourné par l'appel `mystere(carrepol)` *après avoir décrit en français l'opération réalisée par cette fonction.*

La fonction transpose les colonnes en lignes, la première colonne devenant la première ligne et ainsi de suite. on obtient donc :

```
[['U', 'P', 'K', 'F', 'A'] , ['V', 'Q', 'L', 'G', 'B'] ,
['W','R','M','H','C'], ['X','S','N','I','D'], ['Y','T','O','J','E']]
```

e) Écrire une fonction `mystere_bis(T)` *qui réalise l'opération inverse de* `mystere(T)` .

```
1 def mystere_bis(T)
2     NewT = [0]*len(T[0])
3     for i in range(len(T[0])):
4         NewT [i] = [0]*len(T[0])
5     for i in range(len(T[0])):
6         for j in range(len(T[0])):
7             NewT [len(T[0])-1-i][len(T[0])-1-j] = T[len(T[0])-1-j][i]
8     return NewT
```

10. Une variante utilisée en 1918

Le système de chiffrement mis au point et utilisé par l'armée allemande en mars 1918 repose sur le principe du carré de Polybe : un carré de cinq cases de côté, avec en haut et à gauche une ligne supplémentaire où sont placées les lettres ADFGX. Ce système à 25 cases est remplacé le 1er juin 1918 par un système à trente-six cases, avec les lettres ADFGVX.

Dans l'intérieur du carré sont placées toutes les lettres de l'alphabet et les 10 chiffres. L'ordre est déterminé par un mot de code (clé) : on remplit les cases du carré d'abord avec les différentes lettres du nom de code et ensuite avec toutes les autres lettres de l'alphabet et avec les chiffres…

Si MARGUERITE' est la clé, on obtient le carré ci-contre :

(lire la suite à http://j.poitou.free.fr/pro/html/crypt/polybe.html)

	A	D	F	G	V	X
A	M	A	R	G	U	E
D	I	T	B	C	D	F
F	H	J	K	L	N	O
G	P	Q	S	V	W	X
V	Y	Z	0	1	2	3
X	4	5	6	7	8	9

Dans la variante qui suit, on reste avec un carré 5×5 qui ne contient pas la lettre 'Z' ni la ligne et la colonne 'ADFGV'

a) Écrire une fonction `nettoie(texte)` ***qui crée une chaîne formée des caractères de*** `texte` ***écrits une seule fois. Par exemple, 'MARGUERITE' devient 'MARGUEIT'***

```
1 def nettoie(texte) :
2       result =''
3       for lettre in texte :
4            if lettre not in result :
5                 result += lettre
6       return result
```

b) Écrire une fonction `reste_alpha(texte)` ***qui crée la liste des lettres de l'alphabet qui ne sont pas dans*** `texte`.

```
1 def reste_alpha(texte) :
2       liste=[]
3       for j in range(65,91) :
4               lettre = chr(j)
5               if lettre not in texte :
6                       liste.append(lettre)
7        return liste
```

c) Expliquer les différentes étapes d'une fonction `remplir_carre_bis(cle)` ***qui crée le carré répondant au cahier des charges.***

```
     def remplir_carre_bis(cle) :
         cle= nettoie(cle)
         reste = reste_alpha(cle)
         carre_pol=[]
         nbligne = len(cle)//5   #nombre de lignes entières occupées par
la clé
         nbcol = len(cle)%5        #nombre de colonne occupées dans la
dernière ligne incomplète
```

```
        icle = 0                #indice des lignes occupées par la clé
        while icle < nbligne :  #remplissage des lignes completes avec
la clé
            ligne=[]
            for jcle in range(5) :
                let = cle[5*icle + jcle]
                ligne.append(let)
            carre_pol.append(ligne)
            icle =icle + 1


        jcle = 0
        ligne=[]
        while jcle < nbcol :     #remplissage de la ligne incomplète
avec la fin de la clé
            let = cle[5*icle + jcle]
            ligne.append(let)
            jcle = jcle + 1
        if jcle != 0 :           #si il reste des cases à remplir dans
cette ligne
            for  col  in  range(jcle,5):    #complément  de  la  ligne
incomplète
                ligne.append(reste[col-jcle])
            carre_pol.append(ligne)
        else :
            jcle = -1           #si la ligne est complète avec la clé



        for i in range(4-icle) :  # les lignes qui restent à remplir
            ligne=[]
            for j in range(5):     # remplissage d'une ligne avec les
lettres qui ne sont pas dans la clé
                ligne.append(reste[5*i+j + 5-nbcol])
            carre_pol.append(ligne)


    return carre_pol
```