CONTROLE FLASH - PYTHON *(45 minutes)*

DATE : Novembre 2016 **GROUPE MI**

NOM : **PRENOM :**

Q1. Complétez les lignes vides dans la session ci-dessous au toplevel.

```
>>> L1 = [2*i for i in range(2)]
>>> L2 = [L1,L1,L1 + L1]
>>> print('L1 =',L1,'et L2 =',L2)

>>> L1[0] = 10
>>> print('L1 =',L1,'et L2 =',L2)
```

Q2. Programmez une fonction **que_des_pairs(n)** prenant un entier n > 0 et retournant True si n ne contient que des chiffres pairs (en base 10), et False sinon. Exemples :

`que_des_pairs(214)` ☞ `False` `que_des_pairs(2046)` ☞ `True`

```
def que_des_pairs(n) :          # on suppose n entier > 0
```

Q3. Programmez une fonction **list2int(L)** prenant une liste L d'entiers entre 0 et 9. Cette fonction retournera l'entier dont les chiffres sont les éléments de L, dans le même ordre. Exemple :

`list2int([1,0,3,4,0])` ☞ `10340`

```
def list2int(L) :
```

Q4. Avec une boucle **for**, programmez la fonction **remplace(x, y, L)** retournant une *copie* de la liste L dans laquelle on aura remplacé chaque apparition de x par y. Aucune mutation sur L. Exemple : `remplace(3,5,[1,8,3,6,7,3,3,9])` ☞ `[1,8,5,6,7,5,5,9]`

```
def remplace(x,y,L) :
```

Q5. Une **matrice** à p lignes et q colonnes - on dit *de format (p,q)* - est représentée par une liste de p lignes, chaque ligne étant une liste de q nombres. Par exemple : `M1 = [[2,3,-1],[5,0,6]]` et `M2 = [[5,1,0],[4,2,2]]` sont deux matrices de format (2,3) qui s'écrivent en maths :

$$M_1 = \begin{pmatrix} 2 & 3 & -1 \\ 5 & 0 & 6 \end{pmatrix} \quad M_2 = \begin{pmatrix} 5 & 1 & 0 \\ 4 & 2 & 2 \end{pmatrix} \quad M_1+M_2 = \begin{pmatrix} 7 & 4 & -1 \\ 9 & 2 & 8 \end{pmatrix}$$

Programmez une fonction **add_mat(A, B)** prenant deux matrices A et B. Cette fonction déclenchera l'**erreur** *Formats incompatibles* dans le cas où A et B n'ont pas les mêmes formats, et sinon retournera la matrice somme A + B. On suppose qu'il existe déjà des fonctions **nblig(M)** et **nbcol(M)** retournant respectivement le nombre de lignes et de colonnes d'une matrice M.

```
def add_mat(A,B) :
    if
```

Q6. Programmez la fonction **poly_rac(L)** prenant une liste L de nombres réels $[a_1,..,a_n]$ et retournant le polynôme creux $(X - a_1)...(X - a_n)$. Par exemple, poly_rac([2,1]) retournera le polynôme $(X - 2)(X - 1) = X^2-3X+2$ sous la forme [[1, 2], [-3, 1], [2, 0]].

```
def poly_rac(L) :          # on suppose polycreux déjà importé
```

Q7. Un épicier gère en Python son stock de fruits dans un **dictionnaire**, *par exemple* :

```
stock = {'pommes':65, 'poires':18, 'bananes':25, 'pêches':65, 'abricots':12, etc...}
```

Ecrivez en *une ligne* chaque fois ce qu'il faut pour calculer à partir de la variable **stock** :

a) la liste de tous les fruits en quantité > 20 (ici ['pommes', 'bananes', 'pêches', ...]) :

b) le nombre total de fruits :