TP d'informatique n°6

" Boucles conditionnelles : Ecriture d'un entier dans une base autre que 10 "

But du TP
- Ecrire les premières structures conditionnelles et les premières instructions itératives
- Comprendre les instructions et la syntaxe **while**.
- Consolider les connaissances sur les structures conditionnelles et itératives.
- Découvrir la syntaxe des fonctions

Vous penserez **à sauvegarder régulièrement** votre travail dans un fichier dans votre espace personnel. Pour le nom de fichier, n'utilisez ni accent, ni espace, ni caractères spéciaux (autres que – et _) : en effet, certains projets nécessitent l'ouverture de fichiers que l'on aura écrits au préalable : or cet appel peut s'avérer délicat si le nom de ce fichier possède des caractères "non habituels"

A. Ecriture d'un entier en base autre que 10

1. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul, fournit le tuple de ses chiffres dans son écriture en base 10. L'algorithme est le suivant : On part de l'entier m = n et du tuple **Lchif** vide (avec l'instruction **Lchif** = tuple([ ]). On rajoute alors à gauche au tuple **Lchif** le reste de la division euclidienne de m par 10 et on transforme m en son quotient par 10, et on recommence. On itère ce processus jusqu'à ce que l'entier m soit égale à 0 (La procédure pourra utiliser les opérateurs **%** et //)
Tester pour n = 1, 9, 55555, 123456789 et 100!

2. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul, fournit le tuple de ses chiffres dans son écriture en base 8.
Tester pour n = 1, 9, 55555, 123456789 et 100!

3. Ecrire une séquence d'instructions qui, étant donné un entier naturel n compris entre 0 et 15 et qui affecte à la variable **c16** la chaine de caractères "n" si ce nombre est compris entre 0 et 9, "A" si n vaut 10, "B" si n vaut 11, … , "F" si n vaut 15.
Lorsque cette séquence d'instructions sera testée et validée, on l'insérera dans une fonction Python que l'on écrira sous la forme (en respectant les indentations)

```
def Chiffres16(n) :
    Vos instructions
    return c16
```

4. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul, donne le tuple de ses "chiffres" dans son écriture en base 16.
Tester pour n = 1, 9, 55555, 123456789 et 100!

5. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul écrit en base 8, donne l'écriture décimale de cet entier.
Tester pour n = $(1)_8$, $(45)_8$, $(55555)_8$, $(1234567)_8$ et $(345607)_8$

B. Quelques exercices de réécriture de boucles avec la structure while

On reprend les exercices du TP précédent mais cette fois, on utilise la structure **while** plutôt que la structure **for**

1. Ecrire des séquences d'instructions qui, étant donné un entier n naturel (qu'on supposera bien de type int et positif), calcule sa **factorielle**. Calculer 30 ! et déterminer les 3 premiers chiffres de 400 !

2. En utilisant la boucle précédente, ou en utilisant la fonction **factorial()** du module **math**, écrire une boucle permettant de calculer la somme des factorielles de tous les entiers n impairs compris entre 5 et 21.

3. Ecrire une boucle permettant de calculer la somme des 100 premiers termes de la suite définie par $\frac{(-1)^n}{2n+1}$

C. Multiplication égyptienne

On veut effectuer le produit d'entiers a par b.
Si b est pair, on divise b par 2 et on multiplie a par 2.
Si b est impair, on retranche 1 à b (qui de vient pair) et on ajoute a au résultat.
On recommence jusqu'à ce que b soit égal à 1.
Ex : $17 \times 30 = 34 \times 15$
$= 34 \times 14 + 34$
$= 68 \times 7 + 34$
$= 68 \times 6 + 102$
$= 136 \times 3 + 102$
$= 136 \times 2 + 238$
$= 272 \times 1 + 238$
$= 510$

Ecrire des séquences d'instructions qui, étant donnés les entiers a et b, donnent la liste des opérations effectuées

TP d'informatique n°6
(Python Maths)

" Boucles, instructions conditionnelles. Multiplication égyptienne, Ecriture d'un entier dans une base autre que 10 "

A. Quelques exercices de boucles et structures conditionnelles

1. Ecrire des séquences d'instructions qui, étant donné un entier n naturel (qu'on supposera bien de type int et positif), calcule sa **factorielle**. Calculer 30 ! et déterminer les 3 premiers chiffres de 400 !

30! = 265 252 859 812 191 058 636 308 480 000 000 400! commence par 640

2. En utilisant la boucle précédente, ou en utilisant la fonction **factorial()** du module **math**, écrire une boucle permettant de calculer la somme des factorielles de tous les entiers n impairs compris entre 5 et 21.

On trouve 51 212 944 273 488 041 640

3. Ecrire une boucle permettant de calculer la somme des 100 premiers termes de la suite définie par $\frac{(-1)^n}{2n+1}$

On trouve 0.782 898 225 889 638 4

4. Ecrire une séquence d'instructions qui, étant donnée une année n (un entier), affecte à la variable **bissextile** le booléen True si l'année n est bissextile et le booléen False sinon. On rappelle que l'année n est bissextile si n est un multiple de 400 ou s'il s'agit d'une année non séculaire multiple de 4.

```
bissextile = n%400 == 0 or (n%100 != 0 and n%4 == 0)
```

B. Multiplication égyptienne

On veut effectuer le produit d'entiers a par b.
Si b est pair, on divise b par 2 et on multiplie a par 2.
Si b est impair, on retranche 1 à b (qui de vient pair) et on ajoute a au résultat.
On recommence jusqu'à ce que b soit égal à 1.
Ecrire des séquences d'instructions qui, étant donnés les entiers a et b, donnent la liste des opérations effectuées

Ex : 17 × 30 = 34 × 15
= 34 × 14 + 34
= 68 × 7 + 34
= 68 × 6 + 102
= 136 × 3 + 102
= 136 × 2 + 238
= 272 × 1 + 238
= 510

```
s, L = 0, ([a,b,s],)
while b > 1:
    if  b%2 == 0: b, a = b//2 , a * 2
    else :  b, s = b – 1, s + a
    L += ([a,b,s],)
print (L + ([a,0, a+s],))
```

C. Ecriture d'un entier en base autre que 10

1. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul, fournit le tuple de ses chiffres dans son écriture en base 10. L'algorithme est le suivant : On part de l'entier m = n et du tuple **Lchif** vide (avec l'instruction **Lchif** = tuple([ ]). On rajoute alors à gauche au tuple **Lchif** le reste de la division euclidienne de m par 10 et on transforme m en son quotient par 10, et on recommence. On itère ce processus jusqu'à ce que l'entier m soit égale à 0 (La procédure pourra utiliser les opérateurs **%** et //)
Tester pour n = 1, 9, 55555, 123456789 et 100!

```
Lchif , m = tuple([]) , n
while m>0 : m, Lchif = m // 10 , (m%10,) + Lchif
print(Lchif)
```

2. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul, fournit le tuple de ses chiffres dans son écriture en base 8.
Tester pour n = 1, 9, 55555, 123456789 et 100!

```
Lchif8 , m = tuple([]) , n
```

```
while m>0 : m, Lchif8 = m // 8 , (m%8,) + Lchif
print(Lchif)
```

On trouve à partir de 1, 9, 55555, 123456789, respectivement :
(1,) , (1, 1) , (1, 5, 4, 4, 0, 3) et (7, 2, 6, 7, 4, 6, 4, 2, 5)

3. Ecrire une séquence d'instructions qui, étant donné un entier naturel n compris entre 0 et 15 et qui affecte à la variable **c16** la chaine de caractères "n" si ce nombre est compris entre 0 et 9, "A" si n vaut 10, "B" si n vaut 11, … , "F" si n vaut 15.
Lorsque cette séquence d'instructions sera testée et validée, on l'insérera dans une fonction Python que l'on écrira sous la forme (en respectant les indentations)

```
def Chiffres16(n):
    LC = ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F']
    return LC[n]
```

4. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul, donne le tuple de ses "chiffres" dans son écriture en base 16.
Tester pour n = 1, 9, 55555, 123456789 et 100!

```
Lchif16 , m = '' , n
while m>0 : m, Lchif16 = m // 16 , Chiffres16(m%16) + Lchif
print(Lchif16)
```

On trouve à partir de 1, 9, 55555, 123456789, respectivement :
1 , 9, D903, 75BCD15,

5. Ecrire une séquence d'instructions qui, étant donné un entier naturel non nul écrit en base 8, donne le tuple de ses chiffres dans son écriture en base 10.
Tester pour n = 1, 45, 55555, 1234567 et 345607 (écritures en base 8)

```
s, chif  = 0 , str(n)
for k in range(len(chif)):
      s +=  int(chif[k]) * 8**( len(chif) – k – 1)
print(s)
```

On trouve à partir des nombres octaux $(1)_8$, $(45)_8$, $(55555)_8$, $(1234567)_8$, $(345607)_8$
respectivement : 1 , 37, 23405, 342391, 117639