DS d'informatique : corrigé

PTSI Lycée Eiffel

18 novembre 2016

Exercice 1

1. Il suffit de calculer $2^6 + 2^4 + 2^1 = 64 + 16 + 2 = 82$.
2. On effectue les divisions succesives par 2 (que je ne présente pas avec des potences car c'est pénible à faire sur l'ordi : 141/2 donne 70 avec reste 1, puis 35 avec reste 0, puis 17 avec reste 1, puis 8 avec reste 1, 4 avec reste 0, 2 avec reste 0, 1 avec reste 0 et on s'arrête là. On remonte les restes pour obtenir 10001101.
3. Le principe consiste en fait à coder modulo $2^8 = 256$ les entiers de -128 à 127. Les entiers positifs (de 0 à 127) sont codés classiquement (et commenceront tous par un 0), un entier négatif a est codé par le code binaire classique de $256 + a$ (et commencera donc par un 1). Ainsi le code de 141 calculé ci-dessus 10001101 sera utilisé pour coder le nombre -115 puisque $141 = 256 - 115$.

Exercice 2

1. Elle sert à importer du module math la fonction usuelle cosinus et la constante π, qui ne sont pas disponibles par défaut en Python.
2. Il manque un :
3. Il faut définir la variable s en ajoutant une ligne s=0 avant la boucle for.
4. Le return s est mal indenté, il faut le mettre au niveau du for, sinon il sera exécuté dès le premier calcul et pas à la fin de la boucle.
5. Il faut en fait mettre un range(n) et pas un range(n-1) si on veut que la dernière valeur prise soit n-1.
6. Allons-y :

```
def verif() :
    for i in range(1,31) :
        if DMmaths(i) !=0 :
            return ('Ca ne marche pas !')
    return ('Vérification faite, pas de souci !')
```

7. Les erreurs d'arrondis font qu'on peut avoir une valeur très proche de zéro qui sera considérée comme différente de 0. Pour modifier le programme, au lieu de DMmaths(i) !=0, on mettra abs(DMmaths(i))>10**(-10).

Exercice 3

1. C'est une question de cours !

```
def factorielle(n) :
```

```
    p=1
    for i in range(1,n+1) :
        p=p*i
    return p
```

2.
```
def suiteun(n) :
    u=1
    for i in range(1,n+1) :
        u=u+1/factorielle(i)
    return u
```

3.
```
n=0
while 1/factorielle(n) >10**(-10) :
    n=n+1
print(suiteun(n))
```

4.
```
n=0
p=1
u=1
while p>10**(-10) :
    n=n+1
    p=p/n
    u=u+p
print(u)
```

Dans ce programme, p est égal à $\dfrac{1}{n!}$ à chaque étape de la boucle. Les seules opérations effectuées sont des comparaisons (une par boucle), des divisions (une par boucle) et des additions (une par boucle aussi), soit trois opérations élémentaires pour chaque tour de boucle. Par la méthode précédente, on recalculait à chaqe étape de la boucle une factorielle, ce qui demande n multiplications (avec n variant entre 1 et sa valeur finale qu'on notera n_0, cela fait de l'ordre de $\dfrac{n_0^2}{2}$ multiplications), et et on refait tous ces calculs une deuxième fois quand on calcule $suiteun(n_0)$. On a donc de l'ordre de n_0^2 calculs au lieu de $3n_0$, c'est évidemment beaucoup moins bon.