Devoir Surveillé n° 1

PTSI Lycée Eiffel

15 novembre 2019

Exercice 1

Les questions de cet exercice sont indépendantes.

1. Qui est généralement considéré comme l'inventeur de la première machine à calculer ? De quand date celle-ci (donner au moins le siècle) ?
2. Convertir en base 3 le nombre décimal 232 (en détaillant les calculs).
3. Déterminer l'entier relatif qui est codé en complément à 2 sur 8 bits par l'entier binaire 10110101.
4. Expliquer rapidement l'architecture interne d'un ordinateur selon le modèle de von Neumann.

Exercice 2

Un nombre entier naturel n est un nombre **parfait** s'il est égal à la somme de ses diviseurs autres que lui-même. Ainsi, le nombre 28 est divisible par 1, 2, 4, 7, 14 et 28. Comme $14+7+4+2+1 = 28$, le nombre 28 est un nombre parfait.

On rappelle pour cet exercice qu'en Python, la commande $n\%p$ renvoie le reste de la division de l'entier n par l'entier p (autrement dit, n est divisible par p si et seulement si $n\%p = 0$).

1. Écrire une suite de commandes Python qui affiche à l'écran tous les diviseurs d'un entier n (pas besoin de manipuler des listes, on veut simplement afficher les diviseurs l'un à la suite de l'autre).
2. Écrire une fonction Python **parfait(n)** qui prend comme argument un entier n et détermine s'il s'agit ou non d'un entier parfait. On se contentera de parcourir tous les entiers inférieurs à n et d'ajouter l'entier à une variable seulement s'il s'agit d'un diviseur de n. La fonction renverra True si le nombre est parfait, False sinon.
3. En utilisant la fonction précédente, écrire une suite d'instructions en Python qui permet de calculer le plus petit entier strictement supérieur à 28 qui soit un nombre parfait.

Exercice 3

On s'intéresse pour débuter cet exercice au petit programme suivant :

```
>    def mystere(n) :
>        p=1
>        a=1
>        while a<n :
>            a=a+1
>            p=p*a
>        return p
```

1. Détailler l'exécution par Python de la commande mystere(4) (on précisera bien entendu la valeur que calculera le programme lorsque $n = 4$, mais on détaillera aussi les étapes de calcul en précisant toutes les valeurs intermédiaires prises par les variables p et a).
2. Que calcule plus généralement mystere(n) ? Que se passera-t-il si on utilise le programme mystere avec une valeur du paramètre n qui n'est pas entière (par exemple, que renverra la commande mystere(3.5) ?) ?
3. On **admet** que, pour tout réel x, le nombre e^x peut être calculé de la façon suivante : en posant $S_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$, alors $\lim_{n\to+\infty} S_n(x) = e^x$.
   (a) Écrire une fonction Python **sommeexpo(x,n)** qui prend comme paramètres un réel x et un entier n, et qui calcule la valeur de $S_n(x)$. On a le droit pour ce programme de calculer les puissances à l'aide de l'opérateur **.
   (b) Comme on ne peut évidemment pas programmer un calcul de limites simplement, on décide de calculer une valeur approchée de e^x de la façon suivante : on calcule la valeur de $S_n(x)$ pour le plus petit entier n pour lequel $\frac{x^n}{n!} < 10^{-20}$ (on supposera ici x positif pour ne pas avoir de problèmes de signe). Écrire une fonction Python **expoapprochee(x)** prenant comme paramètre un réel x et calculant une valeur approchée de e^x en utilisant cette méthode (on pourra bien sûr réutiliser si on le souhaite la fonction de la question précédente, même si on n'a pas réussi à programmer cette dernière correctement).
4. Les coefficients binômiaux sont des nombres très importants en mathématiques, définis à partir de deux entiers k et n par la formule $\frac{n!}{k!(n-k)!}$.
   (a) Écrire une fonction Python **coefbin(n,k)** qui calcule la valeur de ce nombre (on pourra bien sûr utiliser la fonction mystere définie plus haut dans l'énoncé).
   (b) On peut prouver mathématiquement que la somme de ces nombres lorsque k varie entre 0 et n (l'entier n restant quant à lui fixé) vaut 2^n. Écrire un programme Python demandant à l'utilisateur une valeur de n, calculant et affichant la valeur de la somme des coefficients binômiaux à n fixé, puis vérifiant si cette valeur est bien égale à 2^n (on affichera un message du genre « Oui, les matheux ont raison » pour conclure le programme si le test est effectivement concluant).