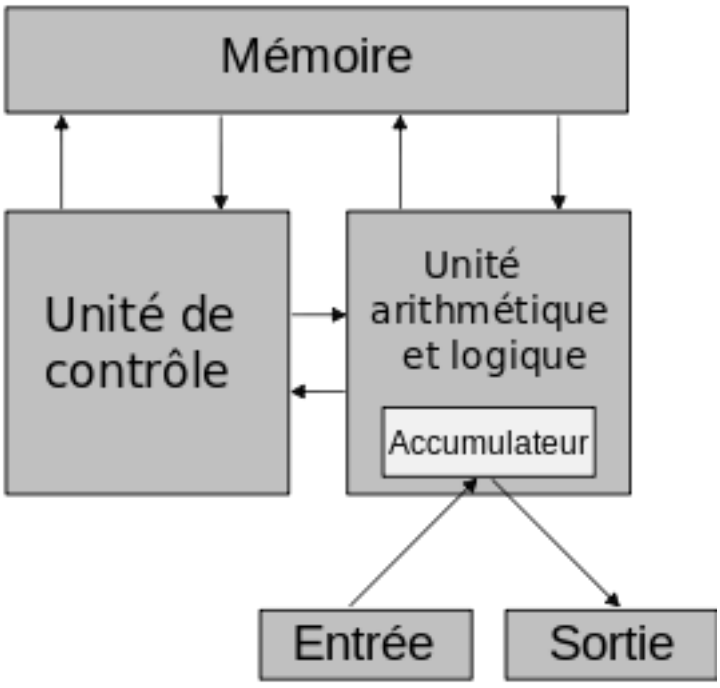
Devoir Surveillé n° 1 : corrigé

PTSI Lycée Eiffel

15 novembre 2019

Exercice 1

1. On cite généralement le nom de Blaise Pascal avec sa pascaline en 1642 (17ème siècle).
2. Pour changer de la méthode des divisions successives vues en cours, on peut aussi se contenter de décomposer directement le nombre en utilisant des puissances de 3 : les puissances de 3 utiles vont être $3^0 = 1$, $3^1 = 3$, $3^2 = 9$, $3^3 = 27$ et $3^4 = 81$ ($3^5 = 243$ est trop grand), donc on calcule $232 = 162+70 = 2*3^4+54+16 = 2*3^4+2*3^3+9+7 = 2*3^4+2*3^3+1*3^2+6+1 = 2*3^4+2*3^3+1*3^2+2*3^1+1*3^0$, donc 232 s'écrit en base 3 sous la forme 22121.
3. Le nombre codé sera négatif puisqu'il a un bit de signe égal à 1. On renverse la méthode vue en cours : on soustrait 1 pour obtenir 10110100, puis on change tous les chiffres, et on décode : $01001011 = 2^6 + 2^3 + 2^1 + 2^0 = 64 + 8 + 2 + 1 = 75$, donc l'entier codé est -75.
4. Je vous remets simplement le schéma (honteusement piqué sur Wikipedia, d'ailleurs) :



Exercice 2

1. ```
   >     for i in range(1,n+1) :
   >         if n%i==0 :
   >             print(i)
   ```

   Il faut absolument commencer la boucle avec la valeur $i = 1$ pour éviter un problème de division par 0, et il est logique dans cette question d'aller jusqu'à $i = n$ inclus puisque n fait
   ```

partie de la liste des diviseurs de n. Pour ceux qui voudraient optimiser un peu, on peut faire beaucoup moins de divisions en s'arrêtant beaucoup plus tôt mais en affichant à chaque fois deux diviseurs d'un coup, par exemple :

```
>    for i in range(1,int(n**0.5)+1) :
>        if n%i==0 :
>            print(i)
>            print(n//i)
```

Deux légers inconvénients à cette dernière solution : les diviseurs ne seront pas affichés dans l'ordre croissant, et l'un des diviseurs sera affiché deux fois dans certains cas particuliers (lequel et pourquoi ? Exercice laissé au lecteur).

2. Cette fois-ci il ne faut évidemment pas prendre en compte n comme diviseur, donc modifier légèrement la boucle :

```
>    def parfait(n) :
>    s=0
>        for i in range(1,n+1) :
>            if n%i==0 :
>                s=s+i
>        return(s==n)
```

La dernière ligne est une petite astuce de rédaction : l'expression $s == n$ va tester si s est égal à n ou non, et le résultat de ce test sera égal à True ou False, c'est-à-dire exactement ce qu'on veut retourner, pas besoin donc de rajouter un if autour.

3. On part de 29 et on augmente n jusqu'à trouver un entier parfait, on va donc utiliser un while :

```
>    n=29
>    while not parfait(n) :
>        n=n+1
>    print(n)
```

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. Initialement, les variables ont donc les valeurs suivantes : $n = 4$ (celle-ci ne sera pas modifiée au cours du programme), $p = 1$, $a = 1$. On entre ensuite dans la boucle while et on effectue les étapes suivantes :
   - $1 < 4$, donc on effectue les instructions : $a = 1 + 1 = 2$ et $p = 1 * 2 = 2$
   - $2 < 4$, donc on recommence : $a = 2 + 1 = 3$ et $p = 2 * 3 = 6$
   - $3 < 4$, donc on recommence : $a = 3 + 1 = 4$ et $p = 6 * 4 = 24$
   - la condition $4 < 4$ n'est pas vérifiée, on sort de la boucle.

Le programme s'arrête et retourne la valeur de p, c'est-à-dire 24.

2. Le programme calcule simplement $n!$. Si n n'est pas entier, le programme tournera sans problème, mais s'arrêtera quand a deviendra supérieur ou égal à n. Comme a est lui toujours entier, en pratique, on calculera toujours une factorielle, celle du plus petit entier supérieur ou égal à n (autrement dit, la partie entière de $n+1$).

3. (a)
```
> def sommeexpo(x,n) :
>     s=0
>     for i in range(n+1) :
>         s=s+(x**i)/mystere(i)
> return s
```

   (b)
```
> def expoapprochee(x) :
>     n=0
>     while (x**n)/mystere(n) > 10**(-20) :
>         n=n+1
> return sommeexpo(x,n)
```

   Cette version du programme est particulièrement atroce puisqu'elle recalcule à chaque étape les puissances et les factorielles en repartant de zéro, il vaut en fait nettement mieux les calculer au fur et à mesure, quitte à utiliser un peu plus de variables, par exemple :

```
> def expoapprochee(x) :
>     n,p,f=0,1,1
>     while p/f > 10**(-20) :
>         n,p,f=n+1,p*x,f*n
> return sommeexpo(x,n)
```

4. (a)
```
> def coefbin(n,k) :
>     return mystere(n)/(mystere(k)*mystere(n-k))
```

   (b)
```
> n=input('Choisissez un entier n')
> s=0
> for i in range(n+1) :
>     s=s+coefbin(n,i)
> if s==2**n :
>     print('Les matheux sont vraiment trop balaises')
> else :
>     print('Rien ne va plus, à partir de maintenant on ne fait plus que de la SII en prépa')
```