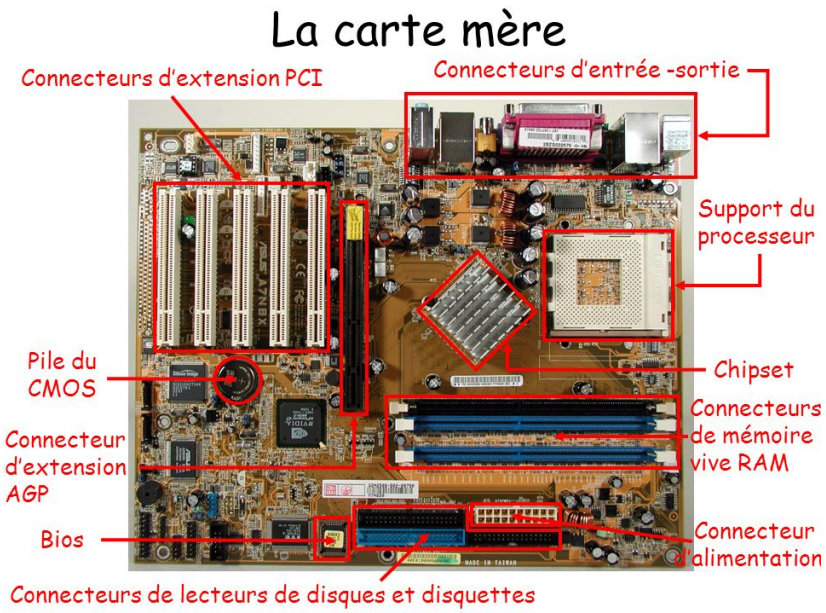
Devoir surveillé n° 1 — corrigé

Exercice 1



Exercice 2 *Débit binaire*

En 9 mois, il y a 23328000 s donc $23328 \cdot 10^6$ bits de transmis. On divise par 8 pour avoir des octets et on trouve 2916 Mo, soit $2916 \times 10^6 / 2^{20} \approx 2781$ Mio. C'est moins qu'un DVD !

Exercice 3 (*Image numérique*)

1. Chaque image est codée sur $1920 \times 1080 \times 3 = 6220800$ octets. Donc pour 1 heure d'image il faut $3600 \times 25 \times 6220800 = 559.872 \cdot 10^9$ octets, c'est-à-dire 559.872 Go. Pour le son, il faut $5 \times 44.1 \cdot 10^3 \times 2$ octet par seconde, soit, en multipliant par 3600, on trouve 1.5876 Go. Un film d'une heure prend donc 561.4596 Go, soit environ 523 Gio
2. La capacité d'un DVD est de l'ordre de 4.4 Gio (ou 8.0 pour un double couche), ce qui est nettement insuffisant. Un film de deux heures ne tient même pas sur un disque dur de 1 Tio.
3. Il faut compresser !

Exercice 4 *Moyenne des carrés des inverses*

Il faut bien faire attention aux bornes.

Python

```python
def moyenne_des_carres_des_inverses(n):
    """Moyenne des carrés des inverses des nombres entre `1` et `n > 0`."""
    somme = 0
    for i in range(1, n + 1):
        somme += 1 / (i ** 2)
    return somme / n
```

Exercice 5 *Calcul de variance* ♥

On commence par calculer la moyenne, comme dans le cours.

Python
```python
def moyenne(liste):
    """Moyenne d'une liste (non vide) de nombres"""
    return sum(liste) / len(liste)
```

On peut choisir une itération par éléments, puisque l'on n'a pas besoin des indices (et ainsi on ne risque pas de se tromper). Attention à ne calculer la moyenne qu'une seule fois (calcul qui coûte cher).

Python
```python
def variance(liste):
    """Variance d'une liste (non vide) de nombres."""
    moy = moyenne(liste)
    somme = 0
    for elt in liste:
        somme += (elt - moy) ** 2
    return somme / len(liste)
```

Exercice 6 *Bonus*

On vérifie pour chaque entier entre 1 et n (compris !) s'il divise n.

Python
```python
def diviseurs(n):
    """Calcule la liste des diviseurs de `n > 0`."""
    divs = []
    for i in range(1, n + 1):
        if n % i == 0:
            divs.append(i)
    return divs
```

On peut aussi s'arrêter à $\lfloor \sqrt{n} \rfloor$, ce qui améliore la complexité, en ajoutant à chaque étape `i` et `n // i`, que l'on calcule en une seule fois en utilisant `divmod`.

Python
```python
def diviseurs(n):
    """Calcule la liste des diviseurs de `n > 0`."""
    divs = []
    for i in range(1, int(math.sqrt(n)) + 1):
        q, r = divmod(n, i)
        if r == 0:
            divs += [i, q]
    return divs
```