Entraînement 1

Écrire une fonction qui renvoie le maximum de trois nombres.

Solution :

Il s'agit évidemment de n'utiliser que des comparaisons et des affectations.

La bonne solution ne comporte que deux comparaisons. C'est le minimum car on peut montrer que tout algorithme de recherche de maximum sur n éléments comporte au moins $n-1$ comparaisons.

```
def plusGrand(x, y, z):

    m = x

    if y >= m:
        m = y
    if z >= m:
        m = z
    return m
```

Ce principe se généralise pour obtenir le maximum d'une liste.

Entraînement 2

En 2020, le mois de juin commence un lundi. À l'aide de deux boucles imbriquées, faire afficher les jours des quatre premières semaines du mois sous la forme :

```
lundi 1 juin
mardi 2 juin
```

Solution :

```
jours = ('lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi',
'dimanche')

for sem in range(4):
    for j in range(7):
        print(jours[j], sem*7 + j + 1, 'juin')
```

Entraînement 3

Écrire une fonction qui calcule la somme $\displaystyle\sum_{i=0}^{n}\left(\prod_{j=1}^{p}(i+j)\right)$.

Solution :

```
def Somme(n, p):

    S = 0

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

Remarque :

Lorsqu'une boucle à besoin d'une initialisation, on place celle-ci juste avant la mise en place de la boucle.

Entraînement 4

Écrire une fonction `valuation2(n)` qui calcule la valuation 2-adique d'un entier n.

On rappelle que la valuation 2-adique d'un entier n est l'exposant de 2 dans la décomposition en facteur premier de n.

Solution :

```
def valuation2(n):

    val = 0

    while n % 2 == 0:
        val += 1
        n = n // 2
    return val
```

Entraînement 5

La conjecture de Syracuse énonce que, pour toute suite (u_n) définie par son premier élément $u_0 \in \mathbb{N}^*$ et la relation de récurrence

$$u_{n+1} = \begin{cases} u_n/2 & \text{si} \quad u_n \text{ est pair,} \\ 3u_n + 1 & \text{sinon,} \end{cases}$$

il existe un indice n tel que $u_n = 1$.

Écrire une fonction `tempsVol(u)` qui prend en paramètre un entier positif u qui est le premier terme de la suite et renvoie le premier indice n tel que $u_n = 1$.

Solution :

```
def tempsVol(u):

    n = 0

    while u != 1:
        if u%2 == 0:
            u = u//2
        else:
            u = 3*u + 1
        n += 1
    return n
```