Découverte du module numpy

Création d'une liste de nombres allant de 0 à 999.

Entrée [1]:
```
L = list(range(1000))
```

Visualisation des cinq premiers nombres de la liste L

Entrée [2]:
```
for i in range(5):
    print(L[i])
```

```
0
1
2
3
4
```

Visualisation des cinq premiers nombres de la liste L par slicing

Entrée [3]:
```
print(L[0:5])
```

```
[0, 1, 2, 3, 4]
```

Visualisation des cing derniers nombres par slicing

Entrée [4]:
```
print(L[-5:])
```

```
[995, 996, 997, 998, 999]
```

Visulalisation des nombres pairs entre 10 et 31.

Entrée [5]:
```
print(L[10:31:2])
```

```
[10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]
```

Création d'un tableau M à partir d'une liste L à l'aide du module numpy

Entrée [6]:
```
import numpy as np
L = list(range(1000))
N = np.array(L)
```

Entrée [7]:
```
N[5:25]
```

Out[7]:
```
array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
       22, 23, 24])
```

Entrée [8]:
```
N[:20:2]
```

Out[8]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

Comparaison des temps de construction d'une liste de carrés à partir de L et N

Entrée [9]:
```
%timeit [x**2 for x in L]
```

```
277 µs ± 1.09 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Entrée [10]:
```
%timeit [x**2 for x in N]
```

```
223 µs ± 614 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

On observe un gain de temps avec le tableau numpy

Vectorisation inhérente à numpy et optimisation du temps de calcul

Entrée [11]:
```
%timeit N**2
```

```
1.26 µs ± 6.89 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

On observe un gain de temps d'un facteur 200 phénoménal pour obtenir la liste des mille carrés