Analyze then speed-up your Python codes !

Code profiling : Identify code bottlenecks, analyse memory use, compute function's call duration etc..

Tools :
- cProfile
- line_profiler

```
Generated by Cython 0.29.21

Yellow lines hint at Python interaction.
Click on a line that starts with a "+" to see the C code

Raw output: sum_of_squares_cython.c

+01: def square(a):
+02:     return a**2
 03:
+04: def sum_of_squares():
+05:     s = 0
+06:     for i in range(1, 10**6):
+07:         s += square(i)
+08:     return s
 09:
+10: if __name__ == '__main__':
+11:     print(sum_of_squares())
```

```python
# pure python version
import math


def f(x):
    return math.exp(- x * x)


def integrate_f(a, b, n):
    s = 0
    dx = (b - a) / n
    for i in range(n):
        s += f(a + i * dx)
    return s * dx
```

```python
# cython version
from libc import math


cdef double f(double x):
    return math.exp(- x * x)


cdef integrate_f(double a, double b, int n):
    cdef double s = 0
    cdef double dx = (b - a) / n
    cdef int i
    for i in range(n):
        s += f(a + i * dx)
    return s * dx
```

Code wrapping : Speed-up vectorial code

Tools :
- Numba (AOT & JIT)
- Cython
- Pybind

Numba

py Cython

Parallel computing : « Split » a code in several independent blocks which will be processed simultaneously

Tools :
- multiprocessing (Python)
- DASK (Python, C/C++, Fortran)

DASK