Par DUBOIS Vincent (mis à jour le 23/03/2020)

JOURNAL DE BORD

- 23/03/2020 :

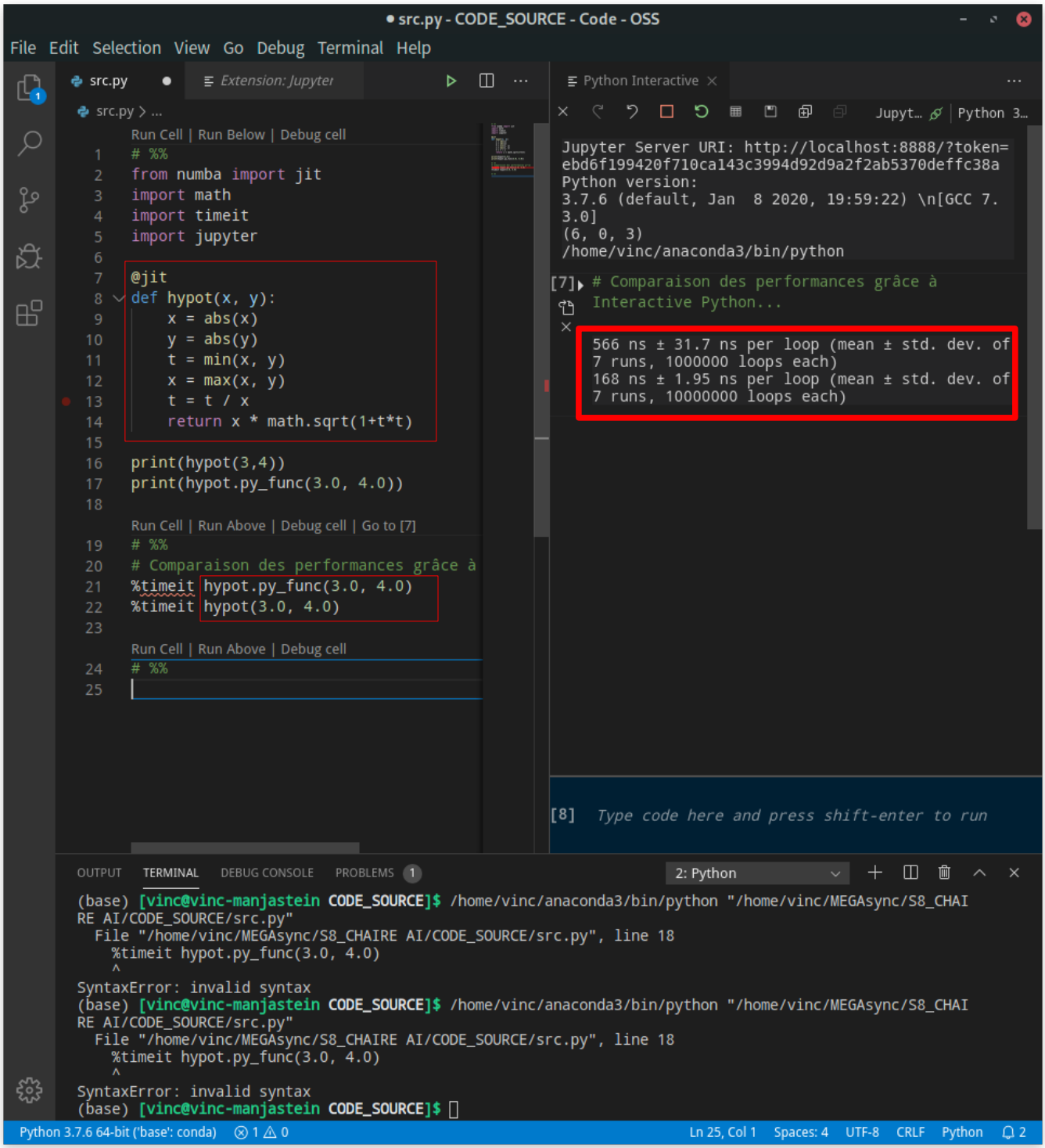
    - Suivi de la documentation du dépôt GitHub par ContnuumIO à propos de Numba et CUDA
    - Test de Benchmarking via Jupyter (en Interactive Python) :

    - Ipython peut représenter un outil de benchmark adapté, car il permet d'isolet une « cellule » de code, puis d'en mesurer une moyenne de rapidité d'exécution sur un nombre n d'itérations (commande %timeit).

      Dans le cas ci-dessous (voire page 2), une cellule contient une fonction (accompagnée de sa fonction décoratrice @jit qui permet de l'executer via Numba), et l'autre cellule contient deux appels de cette même fonction : l'une avec Numba, et l'autre sans :

      On peut ainsi noter un gain de performane notable, en passant de 31,7ns sans Numba à 1,95ns avec Numba (soit plus de 16 fois plus rapide).

      Numba devrait permettre dans mon cas de comparer l'exécution d'un programme (constitué de fonctions imbriquées) avec ou sans accélération matérielle CUDA, et même entre d'autres méthodes d'accélérations GPU.


src.py - CODE_SOURCE - Code - OSS
File Edit Selection View Go Debug Terminal Help
src.py
Extension: Jupyter
src.py > ...
Run Cell | Run Below | Debug cell
# %%
from numba import jit
import math
import timeit
import jupyter
@jit
def hypot(x, y):
x = abs(x)
y = abs(y)
t = min(x, y)
x = max(x, y)
t = t / x
return x * math.sqrt(1+t*t)
print(hypot(3,4))
print(hypot.py_func(3.0, 4.0))
Run Cell | Run Above | Debug cell | Go to [7]
%%
Comparaison des performances grâce à
%timeit hypot.py_func(3.0, 4.0)
%timeit hypot(3.0, 4.0)
Run Cell | Run Above | Debug cell
%%
Python Interactive
Jupyt... Python 3...
Jupyter Server URI: http://localhost:8888/?token=ebd6f199420f710ca143c3994d92d9a2f2ab5370deffc38a
Python version:
3.7.6 (default, Jan 8 2020, 19:59:22) \n[GCC 7.3.0]
(6, 0, 3)
/home/vinc/anaconda3/bin/python
[7] # Comparaison des performances grâce à Interactive Python...
566 ns ± 31.7 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
168 ns ± 1.95 ns per loop (mean ± std. dev. of 7 runs, 10000000 loops each)
[8] Type code here and press shift-enter to run
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1
2: Python
(base) [vinc@vinc-manjastein CODE_SOURCE]$ /home/vinc/anaconda3/bin/python "/home/vinc/MEGAsync/S8_CHAIRE AI/CODE_SOURCE/src.py"
File "/home/vinc/MEGAsync/S8_CHAIRE AI/CODE_SOURCE/src.py", line 18
%timeit hypot.py_func(3.0, 4.0)
^
SyntaxError: invalid syntax
(base) [vinc@vinc-manjastein CODE_SOURCE]$ /home/vinc/anaconda3/bin/python "/home/vinc/MEGAsync/S8_CHAIRE AI/CODE_SOURCE/src.py"
File "/home/vinc/MEGAsync/S8_CHAIRE AI/CODE_SOURCE/src.py", line 18
%timeit hypot.py_func(3.0, 4.0)
^
SyntaxError: invalid syntax
(base) [vinc@vinc-manjastein CODE_SOURCE]$
Python 3.7.6 64-bit ('base': conda) 1 0
Ln 25, Col 1 Spaces: 4 UTF-8 CRLF Python 2