Open in Colab

Introduction - Syntax sugar

When you run a command with

- `!` it directly executes a bash command in a **subshell**.
- `%` it executes one of the magic commands defined in IPython.
- `%% my_native_language` defines the language used to interpret the cell

Some of the magic commands defined by IPython deliberately mirror bash commands, but they differ in the implementation details.

For example, running the !cd bash command does not persistently change your directory, because it runs in a temporary subshell. However, running the %cd magic command will persistently change your directory:

```
.sh
!pwd
# /content

!cd sample_data/
!pwd
# /content

%cd sample_data/
!pwd
# /content/sample_data
```

Reference https://ipython.readthedocs.io/en/stable/interactive/magics.html

In [1]:
```
# an example of mixing python an shell in one cell

# this is python (default interpreter)
import numpy as np
print(2*np.exp([1,2,3]))

# this is bash shell
%env MY_VARIABLE=123
!pwd
!echo "my shell variable ${123}"
```

```
[ 5.43656366 14.77811222  40.17107385]
env: MY_VARIABLE=123
/home/grzegorz/GITHUB/CUDA/gpu_colab
my shell variable
```

Get the material

In [2]:
```
!git clone https://github.com/ggruszczynski/gpu_colab.git
```

```
fatal: destination path 'gpu_colab' already exists and is not an empty directory.
```

In [3]:
```
!ls
```

```
10_intro_setup.ipynb                    a.cpp          pdfy
20_vector_add.ipynb                     a.out          README.md
30_element_wise_matrix_add.ipynb        experimental   requirements.txt
30_matrix_matrix_multiplication.ipynb   gpu_colab      solutions
40_parallel_reduction.ipynb             hello          src
50_thrust.ipynb                         hello.cpp      to_pdf_1by1.sh
60_python_cuda_intro.ipynb              hello_cuda
70_heat_diffusion2D.ipynb               hello_cuda.cu
```

In [4]:
```
%cd gpu_colab/code_samples
```

```
[Errno 2] No such file or directory: 'gpu_colab/code_samples'
/home/grzegorz/GITHUB/CUDA/gpu_colab
```

Create a file, compile & run!

In [5]:
```
%%file hello.cpp
#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

```
Overwriting hello.cpp
```

In [6]:
```
%%bash
g++ hello.cpp -o hello
echo "===print working directory and its content==="
pwd
ls
echo "===execute the program==="
./hello
```

```
===print working directory and its content===
/home/grzegorz/GITHUB/CUDA/gpu_colab
10_intro_setup.ipynb
20_vector_add.ipynb
30_element_wise_matrix_add.ipynb
30_matrix_matrix_multiplication.ipynb
40_parallel_reduction.ipynb
50_thrust.ipynb
60_python_cuda_intro.ipynb
70_heat_diffusion2D.ipynb
a.cpp
a.out
experimental
gpu_colab
hello
hello.cpp
hello_cuda
hello_cuda.cu
pdfy
README.md
requirements.txt
solutions
src
to_pdf_1by1.sh
===execute the program===
Hello World!
```

cpp (auto) magic

This section explains how to create a wrapper for your cell.

In [7]:
```
from IPython.core.magic import register_cell_magic
```

In [8]:
```
@register_cell_magic
def cpp(line, cell):
    with open('a.cpp', 'w') as f:
        f.write(cell)
    !g++ a.cpp
    !./a.out
```

In [9]:
```
%%cpp
#include <iostream>
int main() {
    std::cout << "Hello World!";
    return 0;
}
```

```
Hello World!
```

In [10]:
```
cpp_header = """
#include <iostream>
#include <string>
#include <iterator>
#include <utility>
#include <map>
using namespace std;
"""

@register_cell_magic
def cpp(line, cell):
    if '-main()' not in cell:
        cell = "int main(){" + cell + "}"
    with open('a.cpp', 'w') as f:
        f.write(cpp_header + cell)
    !g++ a.cpp
    !./a.out
```

In [11]:
```
%%cpp
std::cout << "Hello World!";
```

```
Hello World!
```

In [12]:
```
%%cpp
for(int i=0; i<5; i++) {
    cout << i;
}

cout << endl;
pair <int, string> PAIR1;

PAIR1.first = 100;
PAIR1.second = "lat!" ;

cout << PAIR1.first << " ";
cout << PAIR1.second << endl;
```

```
01234
100 lat!
```

Activate GPU

- To get access to a GPU, click on the *Runtime* menu and select *Change runtime type*. Choose GPU as a Hardware accelerator. It might take a minute for your notebook to connect to a GPU.
- To check whether a GPU has been connected to your session, run the code cell below with the `!nvidia-smi` command by hitting `SHIFT-ENTER` on it.

In [13]:
```
!nvidia-smi
```

```
Sun Apr 16 17:39:28 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.108.03   Driver Version: 510.108.03   CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:01:00.0  On |                  N/A |
|  0%   57C    P5    31W / 250W |   1473MiB /  8192MiB |     14%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1703      G   /usr/lib/xorg/Xorg                 96MiB |
|    0   N/A  N/A      2624      G   /usr/lib/xorg/Xorg                684MiB |
|    0   N/A  N/A      2819      G   /usr/bin/gnome-shell              604MiB |
|    0   N/A  N/A      3712      G   ...RendererForSitePerProcess       3MiB |
|    0   N/A  N/A      8175      G   ...feats=BackForwardCache         10MiB |
|    0   N/A  N/A      8543      G   ...957248867340528764,131072      237MiB |
|    0   N/A  N/A     13016      G   ...AAAAAAAAA= --shared-files       35MiB |
|    0   N/A  N/A     14339      G   ...b/thunderbird/thunderbird      121MiB |
|    0   N/A  N/A     15643      G   ...RendererForSitePerProcess      169MiB |
+-----------------------------------------------------------------------------+
```

In [14]:
```
%%file hello_cuda.cu

#include <stdio.h>

// functions qualifiers:
// __global__ launched by CPU on device (must return void)
// __device__ called from other GPU functions (never CPU)
// __host__ can be executed by CPU
// (can be used together with __device__)

// kernel launch:
// f_name<<<blocks,threads_per_block>>>(p1,,,, pN)

__global__ void print_from_gpu(void) {
    int tidx = blockIdx.x*blockDim.x+threadIdx.x;
    printf("Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> %d = %d * %d + %d\n",
    tidx, blockIdx.x, blockDim.x, threadIdx.x);
}

int main(void) {
    printf("Hello World from host!\n");

    print_from_gpu<<<2,3>>>();  // <<<blocks, threads_per_block>>>
    cudaDeviceSynchronize();
    printf("----------------------------\n");
    dim3 grid_dim(2,1,1);
    dim3 block_dim(3,1,1);
    print_from_gpu<<<grid_dim, block_dim>>>();  // <<<blocks, threads_per_block>>>
    cudaDeviceSynchronize();
    return 0;
}
```

```
Overwriting hello_cuda.cu
```

Check version of your GPU card

if you received an older gpu like Tesla K80 (check the output of `!nvidia-smi` command) add the `-gencode arch=compute_35,code=sm_35` flags to nvcc compiler.

In [15]:
```
%%bash

CUDA_SUFF=70
nvcc -gencode arch=compute_${CUDA_SUFF},code=sm_${CUDA_SUFF} ./hello_cuda.cu -o hello_cuda
./hello_cuda
```

```
Hello World from host!
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 0 = 0 * 3 + 0
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 1 = 0 * 3 + 1
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 2 = 0 * 3 + 2
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 3 = 1 * 3 + 0
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 4 = 1 * 3 + 1
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 5 = 1 * 3 + 2
----------------------------
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 3 = 1 * 3 + 0
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 4 = 1 * 3 + 1
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 5 = 1 * 3 + 2
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 0 = 0 * 3 + 0
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 1 = 0 * 3 + 1
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 2 = 0 * 3 + 2
```

if you were lucky to get a more recent gpu (like Tesla T4)...

you can install a python wrapper to run `%%cu` cells directly

```
.sh
!pip install git+git://github.com/andreinechaev/nvcc4jupyter.git
%load_ext nvcc_plugin
```

then,

```
%%cu

your cell with cuda code...
```

In [ ]:
```
!pip install git+git://github.com/andreinechaev/nvcc4jupyter.git
```

In [17]:
```
%load_ext nvcc_plugin
```

```
directory /home/grzegorz/GITHUB/CUDA/gpu_colab/src already exists
Out bin /home/grzegorz/GITHUB/CUDA/gpu_colab/result.out
```

In [18]:
```
%%cu

#include <stdio.h>

__global__ void print_from_gpu(void) {
    int tidx = blockIdx.x*blockDim.x+threadIdx.x;
    printf("Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> %d = %d * %d + %d\n",
    tidx, blockIdx.x, blockDim.x, threadIdx.x);
}

int main(void) {
    printf("Hello World from host!\n");

    print_from_gpu<<<2,3>>>();  // <<<blocks, threads_per_block>>>

    cudaDeviceSynchronize();
    return 0;
}
```

```
Hello World from host!
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 3 = 1 * 3 + 0
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 4 = 1 * 3 + 1
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 5 = 1 * 3 + 2
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 0 = 0 * 3 + 0
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 1 = 0 * 3 + 1
Hello from device! My threadId = blockIdx.x *blockDim.x + threadIdx.x <=> 2 = 0 * 3 + 2
```