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 [38]:
```
# 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 14.7781122  40.17107385]
env: MY_VARIABLE=123
/content
my shell variable 23
```

Get the material

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

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

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

```
gpu_colab  sample_data
```

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

```
/content/gpu_colab/code_samples
```

Create a file, compile & run!

In [2]:
```
%%file hello.cpp
#include <iostream>

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

```
Writing hello.cpp
```

In [10]:
```
%%shell
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===
/content
hello  hello.cpp  sample_data
===execute the program===
Hello World!
```

Out[10]:

cpp (auto) magic

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

In [13]:
```
from IPython.core.magic import register_cell_magic
```

In [14]:
```
@register_cell_magic
def cpp(line, cell):
  with open('a.cpp', 'w') as f:
    f.write(cell)
  !g++ a.cpp
  !./a.out
```

In [15]:
```
%%cpp
#include <iostream>
int main() {
    std::cout << "Hello World!";
    return 0;
}
```

```
Hello World!
```

In [16]:
```
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 [17]:
```
%%cpp
std::cout << "Hello World!";
```

```
Hello World!
```

In [25]:
```
%%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 [1]:
```
!nvidia-smi
```

```
Wed Feb 23 16:07:59 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| 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  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   68C    P8    32W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

In [42]:
```
%%file hello_cuda.cu

#include <stdio.h>

// functions qualifers:
// __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
    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 [43]:
```
%%shell

CUDA_SUFF=35
nvcc -gencode arch=compute_${CUDA_SUFF},code=sm_${CUDA_SUFF} ./hello_cuda.cu -o hello_cuda
./hello_cuda
```

```
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectu
res are deprecated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppr
ess warning).
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
-------------------------------
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
```

Out[43]:

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 [2]:
```
!pip install git+git://github.com/andreinechaev/nvcc4jupyter.git
```

```
Collecting git+git://github.com/andreinechaev/nvcc4jupyter.git
  Cloning git://github.com/andreinechaev/nvcc4jupyter.git to /tmp/pip-req-build-_3ffnuc8
  Running command git clone -q git://github.com/andreinechaev/nvcc4jupyter.git /tmp/pip-req-build-_3
ffnuc8
Building wheels for collected packages: NVCCPlugin
  Building wheel for NVCCPlugin (setup.py) ... done
  Created wheel for NVCCPlugin: filename=NVCCPlugin-0.0.2-py3-none-any.whl size=4306 sha256=2627b2ba
182e1c4d80f1730d4ad56d3341189b94129e76757038fb94cc338094
  Stored in directory: /tmp/pip-ephem-wheel-cache-0kz88oe7/wheels/c5/2b/c0/87008e795a14bbcdfc7c846a0
0d06981916331eb980b6c8bdf
Successfully built NVCCPlugin
Installing collected packages: NVCCPlugin
Successfully installed NVCCPlugin-0.0.2
```

In [3]:
```
%load_ext nvcc_plugin
```

```
created output directory at /content/src
Out bin /content/result.out
```

In [4]:
```
%%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
    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 <=> 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
```