```
#Amruta Sool
#BBCO19128
```

```
!nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

```
#Run the given command to install a small extension to run nvcc from the Notebook cells.
!pip install git+https://github.com/andreinechaev/nvcc4jupyter.git
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+https://github.com/andreinechaev/nvcc4jupyter.git
  Cloning https://github.com/andreinechaev/nvcc4jupyter.git to /tmp/pip-req-build-m4c8w3v3
  Running command git clone --filter=blob:none --quiet https://github.com/andreinechaev/nvcc4jupyter.git /tmp/pip-req-build-m4c8w3v3
  Resolved https://github.com/andreinechaev/nvcc4jupyter.git to commit aac710a35f52bb78ab34d2e52517237941399eff
  Preparing metadata (setup.py) ... done
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=4287 sha256=b452f361e46779a8d4516fcab4888a56884d37934232dd9dbff6c4
  Stored in directory: /tmp/pip-ephem-wheel-cache-6789c91n/wheels/a8/b9/18/23f8ef71ceb0f63297dd1903aedd067e6243a68ea756d6feea
Successfully built NVCCPlugin
Installing collected packages: NVCCPlugin
Successfully installed NVCCPlugin-0.0.2
```

```
# Commented out IPython magic to ensure Python compatibility.
#Load the extension
%load_ext nvcc_plugin
```

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

```
#Commented out IPython magic to ensure Python compatibility.
%%cu
#include<stdio.h>
#include<cuda.h>
#include<stdlib.h>
#include<time.h>
#define N 500

 __global__ void add(int *a, int *b, int *c){

 int tid = threadIdx.x;
 if (tid < N){
  c[tid]= a[tid] + b[tid];
  }

 }

  int main (void){
 int a[N], b[N], c[N];
 int *dev_a, *dev_b,*dev_c;
 cudaError_t err=cudaSuccess;
 err=cudaMalloc((void**)&dev_a,N * sizeof(int));
 if (err !=cudaSuccess)
 {  printf("failed to  allocate on device \n");
  printf("error is:\n",cudaGetErrorString(err));
 exit(EXIT_FAILURE);
 }
 cudaMalloc((void**)&dev_b,N * sizeof(int));
 cudaMalloc((void**)&dev_c,N * sizeof(int));

 for(int i=0;i<N;i++){
 a[i] =i;
 b[i] = i*i;

 c[i]=0;
 }

 /*for(int i=0;i<N;i++){
 printf(" c conttents are =%d\n",  c[i]);
 } */
 cudaEvent_t start, end;
 cudaEventCreate(&start);
 cudaEventCreate(&end);
 cudaEventRecord(start);

 cudaMemcpy(dev_a,a,N*sizeof(int), cudaMemcpyHostToDevice);
 cudaMemcpy(dev_b,b,N*sizeof(int), cudaMemcpyHostToDevice);
 cudaMemcpy(dev_c,c,N*sizeof(int), cudaMemcpyHostToDevice);
 add<<<1,N>>>(dev_a,dev_b,dev_c);
```

```
err=cudaMemcpy(c,dev_c,N*sizeof(int), cudaMemcpyDeviceToHost);
if (err !=cudaSuccess)
{  printf("failed to  copy from device \n");
exit(EXIT_FAILURE);
}

cudaEventRecord(end);
cudaEventSynchronize(end);
float time = 0;
cudaEventElapsedTime(&time, start, end);
printf("execution time=%f\n",time);

for(int i=0;i<N;i++){
printf("%d +%d=%d\n", a[i], b[i], c[i]);
}

cudaFree(dev_a);
cudaFree(dev_b);
cudaFree(dev_c);

return 0;
}
```

```
execution time=0.060192
0 +0=0
1 +1=2
2 +4=6
3 +9=12
4 +16=20
5 +25=30
6 +36=42
7 +49=56
8 +64=72
9 +81=90
10 +100=110
11 +121=132
12 +144=156
13 +169=182
14 +196=210
15 +225=240
16 +256=272
17 +289=306
18 +324=342
19 +361=380
20 +400=420
21 +441=462
22 +484=506
23 +529=552
24 +576=600
25 +625=650
26 +676=702
27 +729=756
28 +784=812
29 +841=870
30 +900=930
31 +961=992
32 +1024=1056
33 +1089=1122
34 +1156=1190
35 +1225=1260
36 +1296=1332
37 +1369=1406
38 +1444=1482
39 +1521=1560
40 +1600=1640
41 +1681=1722
42 +1764=1806
43 +1849=1892
44 +1936=1980
45 +2025=2070
46 +2116=2162
47 +2209=2256
48 +2304=2352
49 +2401=2450
50 +2500=2550
51 +2601=2652
52 +2704=2756
53 +2809=2862
54 +2916=2970
55 +3025=3080
56 +3136=3192
```