```
%load_ext nvcc_plugin

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

Perform cuda operations with 1 thread and 1 block

[ ] ↳ *1 cell hidden*

Perform CUDA operations on blockIdx

```
%%cu
#include <stdio.h>
#define N 512

__global__ void
add(int *a, int *b, int *c) {
    c[blockIdx.x] = a[blockIdx.x] + b[blockIdx.x];
}

int main(void) {
    int *a, *b, *c;         // The arrays on the host CPU machine
    int *d_a, *d_b, *d_c;  // The arrays for the GPU device

    int size = N * sizeof(int);

    a = (int *)malloc(size);
    b = (int *)malloc(size);
    c = (int *)malloc(size);

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

    cudaMalloc((void **)&d_a, size);
    cudaMalloc((void **)&d_b, size);
    cudaMalloc((void **)&d_c, size);

    cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, size, cudaMemcpyHostToDevice);

    add<<<N, 1>>>(d_a, d_b, d_c);

    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < N; i++) {
      if (c[i] != a[i] + b[i]) {
```

✓ 1s completed at 3:05 PM

```
39        }
40      }
41
42      free(a);
43      free(b);
44      free(c);
45
46      cudaFree(d_a);
47      cudaFree(d_b);
48      cudaFree(d_c);
49
50      return 0;
51  }
```

Perform CUDA operations on threadIdx

```
 1 %%cu
 2 #include <stdio.h>
 3 #define N 512
 4
 5 __global__ void
 6 add(int *a, int *b, int *c) {
 7     c[threadIdx.x] = a[threadIdx.x] + b[threadIdx.x];
 8 }
 9
10 int main(void) {
11     int *a, *b, *c;         // The arrays on the host CPU machine
12     int *d_a, *d_b, *d_c;  // The arrays for the GPU device
13
14     int size = N * sizeof(int);
15
16     a = (int *)malloc(size);
17     b = (int *)malloc(size);
18     c = (int *)malloc(size);
19
20     for (int i = 0; i < N; i++) {
21         a[i] = i;
22         b[i] = i;
23     }
24
25     cudaMalloc((void **)&d_a, size);
26     cudaMalloc((void **)&d_b, size);
27     cudaMalloc((void **)&d_c, size);
28
29     cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
30     cudaMemcpy(d_b, b, size, cudaMemcpyHostToDevice);
31
32     add<<<1, N>>>(d_a, d_b, d_c);
33
34     cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);
```

```
    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < N; i++) {
      if (c[i] != a[i] + b[i]) {
          printf("Error in calculation;");
      }
    }

    free(a);
    free(b);
    free(c);

    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    return 0;
}
```

Perform CUDA operations with thread and blocks combined

```
%%cu
#include <stdio.h>
#define N (2048*2048)
#define TPB 512

__global__ void
add(int *a, int *b, int *c) {
    int index = threadIdx.x + blockIdx.x * blockDim.x;
    c[index] = a[index] + b[index];
}

int main(void) {
    int *a, *b, *c;         // The arrays on the host CPU machine
    int *d_a, *d_b, *d_c;  // The arrays for the GPU device

    int size = N * sizeof(int);

    a = (int *)malloc(size);
    b = (int *)malloc(size);
    c = (int *)malloc(size);

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

    cudaMalloc((void **)&d_a, size);
```

```
    cudaMalloc((void **)&d_a, size);
    cudaMalloc((void **)&d_b, size);
    cudaMalloc((void **)&d_c, size);

    cudaMemcpy(d_a, a, size, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, size, cudaMemcpyHostToDevice);

    add<<<N/TPB, TPB>>>(d_a, d_b, d_c);

    cudaMemcpy(c, d_c, size, cudaMemcpyDeviceToHost);

    for (int i = 0; i < N; i++) {
      if (c[i] != a[i] + b[i]) {
          printf("Error in calculation;");
      }
    }

    free(a);
    free(b);
    free(c);

    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    return 0;
}
```

```
1
```

Colab paid products - Cancel contracts here