Module 3.3

CUDA forces you to pre-declare the number of threads per block for each function. We also need to use blocks with a fixed amount of shared memory.

One challenge is that you need to be very aware of these sizes as you code or you will get very strange bugs! This exercise asks you to play through a given example and write down the sizes that you see.

```
THREADS_PER_BLOCK = 32

def map_fn(out, a):
   # Thread to process a[i]
   i = numba.cuda.blockIdx.x * THREADS_PER_BLOCK + numba.cuda.threadIdx.x

   if i < ???:
       out[i] = a[i]

kernel_fn = numba.cuda.jit()(kernel_fn)
a = np.random.rand(100) # shape (100,) tensor
out = np.zeros(100)     # shape (100,) tensor
blocks = ???
kernel_fn[blocks, THREADS_PER_BLOCK](out, a)
```

1 1 point

If we plan to process all of `a`, what should the value of `blocks` be?

Type your answer...

2 1 point

If we plan to process all of `a`, what should the value of `???` be in the map_fn?

Type your answer...

3 1 point

What should the value of `blocks` be if we instead decide to use 16 threads per block?

Type your answer...