Key Terms to Concurrency

There are few terms which we must understand and comprehend as our second skin. They are so critical to the discussion of Concurrency. The terms are:

1. Critical Section
2. Race condition
3. Indeterminate Program
4. Mutual Exclusion

Critical Section

A critical section is a piece of code that performs operations on shared resources, usually a shared variable or data structure.

Race Condition

A race condition occurs when multiple threads simultaneously execute the critical section. When multiple threads update the shared variable or data structures almost at the same time, then shocking and incorrect results come.

Indeterminate Program

An indeterminate program has one or more race conditions. The indeterminate program gives a different output on every run. The output depends on which thread ran when. For example: the following code which was explained in the previous video is an example of indeterminate program.
Code:

```
import threading

COUNT = 0

def calculate_count(arg):
  print("{}:begin".format(arg))
  global COUNT
  for _ in range(1000000):
      COUNT = COUNT + 1
  print("{}:ends".format(arg))

def main():
  print("main begin: COUNT = {}".format(COUNT))
  t1 = threading.Thread(target=calculate_count, args=("t1",))
  t2 = threading.Thread(target=calculate_count, args=("t2",))

  t1.start()
```

```
  t2.start()

  t1.join()
  t2.join()

  print("main done: COUNT = {}".format(COUNT))

if __name__ == "__main__":
  main()
```

Mutual exclusion

To avoid non deterministic outputs, threads must employ some kind of mutual exclusion primitives which ensure that only one thread enters and executes critical section at a time. This will give deterministic results.