Homework (Simulation)-Tong Cui
This program, process-run.py, allows you to see how process states change as programs run and either use the CPU (e.g., perform an add instruction) or do I/O (e.g., send a request to a disk and wait for it to complete). See the README for details.

Questions

1. Run process-run.py with the following flags: -l 5:100,5:100. What should the CPU utilization be (e.g., the percent of time the CPU is in use?) Why do you know this? Use the -c and -p flags to see if you were right.

```
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu
  cpu

Process 1
  cpu
  cpu
  cpu
  cpu
  cpu

Important behaviors:
  System will switch whenthe current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO willrun LATER (when it is its turn)
```

The percent of the time the CPU is in use should be 100%. Because we passed at 5:100 and 5:100 when we run the program. According to the README file, the process we specified is "5:100" which means it should consist of 5 instructions, and the chances that each instruction is a CPU instruction is 100%.

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 5:100,5:100 -c
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2    RUN:cpu     READY         1
  3    RUN:cpu     READY         1
  4    RUN:cpu     READY         1
  5    RUN:cpu     READY         1
  6       DONE   RUN:cpu         1
  7       DONE   RUN:cpu         1
  8       DONE   RUN:cpu         1
  9       DONE   RUN:cpu         1
 10       DONE   RUN:cpu         1
```

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 5:100,5:100 -p
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu
  cpu

Process 1
  cpu
  cpu
  cpu
  cpu
  cpu

Important behaviors:
  System will switch whenthe current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO willrun LATER (when it is its turn)
```

I think my answer is correct. Because from the above pictures, we can see that Process 0 runs first and is done, and then Process 1 run and is done. There is no IOs. As a result, CPU utilization is 100%.

2. Now run with these flags: ./process-run.py -l 4:100,1:0.
These flags specify one process with 4 instructions (all to use the CPU), and one that simply issues an I/O and waits for it to be done. How long does it take to complete both processes? Use -c and -p to find out if you were right.

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 4:100,1:0
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu

Process 1
  io
```

It takes 10 clock ticks to finish both processes.

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 4:100,1:0 -c
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2    RUN:cpu     READY         1
  3    RUN:cpu     READY         1
  4    RUN:cpu     READY         1
  5       DONE    RUN:io         1
  6       DONE   WAITING                    1
  7       DONE   WAITING                    1
  8       DONE   WAITING                    1
  9       DONE   WAITING                    1
 10*      DONE      DONE
```

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 4:100,1:0 -p
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu

Process 1
  io
```

CPU percentage: 50%.
I/Os percentage: 40%.
According to the above pictures, Process 0 ran for 4 clock ticks and was done. Then Process 1 posted an I/O request and the CPU received and blocked Process 1, which takes 1 clock tick. Then the CPU waited for the I/O request for 4 clock ticks. Finally, Process 1 received the I/O result and the CPU ran and did Process 1 in 1 clock tick.

3. Switch the order of the processes: -l 1:0,4:100. What happens now? Does switching the order matter? Why? (As always, use -c and -p to see if you were right)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 1:0,4:100
Produce a trace of what would happen when you run these processes:
Process 0
  io

Process 1
  cpu
  cpu
  cpu
  cpu
```

Yes, switching the order matters. The total time to complete the two processes is 6 clock ticks.

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 1:0,4:100 -c
Time    PID: 0    PID: 1       CPU       IOs
  1     RUN:io     READY         1
  2    WAITING   RUN:cpu         1         1
  3    WAITING   RUN:cpu         1         1
  4    WAITING   RUN:cpu         1         1
  5    WAITING   RUN:cpu         1         1
  6*      DONE      DONE
```

CPU percentage: 83.33%.
I/Os percentage: 66.67%.
According to the above picture, Process 0 posted an I/O request. The CPU received and blocked Process 0 and started to run Process 1, which takes 1 clock tick. In the next 4 clock ticks, Process 0’s I/Os were waiting, while the CPU was running for Process 1. Finally, I/O

finished and the CPU unblocked Process 0 and Process 1 was done. There is a total of 6 clock ticks.

4. We'll now explore some of the other flags. One important flag is -S, which determines how the system reacts when a process issues an I/O. With the flag set to SWITCH ON END, the system will NOT switch to another process while one is doing I/O, instead waiting until the process is completely finished. What happens when you run the following two processes (-l 1:0,4:100 -c -S SWITCH ON END), one doing I/O and the other doing CPU Work?

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 1:0,4:100 -c -S SWITCH_ON_END
Time    PID: 0    PID: 1       CPU       IOs
  1     RUN:io     READY         1
  2    WAITING     READY                   1
  3    WAITING     READY                   1
  4    WAITING     READY                   1
  5    WAITING     READY                   1
  6*      DONE   RUN:cpu         1
  7       DONE   RUN:cpu         1
  8       DONE   RUN:cpu         1
  9       DONE   RUN:cpu         1
```

CPU percentage: 55.56%.

I/Os percentage: 44.44%.

It would take 9 clock ticks to finish. Process 0 was doing I/O first, which took 1 clock tick. The CPU won't switch to Process 1 until Process 0 is finished. Once Process 0 completed I/O in 4 clock ticks, the CPU started to run for Process 1. After 4 clock ticks, Process 1 finished. It took a total of 9 clock ticks.

5. Now, run the same processes, but with the switching behavior set to switch to another process whenever one is WAITING for I/O (-l 1:0,4:100 -c -S SWITCH ON IO). What happens now? Use -c and -p to confirm that you are right.

Because we switch it on when I/O starts, the system would switch when Process 0 issues an I/O. It turns to the same situation in question 3.

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 1:0,4:100 -c -S SWITCH_ON_IO
Time    PID: 0    PID: 1       CPU       IOs
  1     RUN:io     READY         1
  2    WAITING   RUN:cpu         1         1
  3    WAITING   RUN:cpu         1         1
  4    WAITING   RUN:cpu         1         1
  5    WAITING   RUN:cpu         1         1
  6*      DONE      DONE
```

CPU percentage: 83.33%.

I/Os percentage: 66.67%.

The total time to complete the two processes is 6 clock ticks.

According to the above picture, Process 0 posted an I/O request. The CPU received and blocked Process 0 and switched to run Process 1, which takes 1 clock tick. In the next 4 clock ticks, Process 0's I/Os were waiting, while the CPU was running for Process 1. Finally, I/O finished and the CPU unblocked Process 0 and Process 1 was done. There is a total of 6 clock ticks.

6. One other important behavior is what to do when an I/O completes. With -I IO RUN LATER, when an I/O completes, the process that issued it is not necessarily run right away; rather, whatever was running at the time keeps running. What happens when you run this combination of processes? (Run ./process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH ON IO -I IO RUN LATER -c -p) Are system resources being effectively utilized?

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_LATER
Produce a trace of what would happen when you run these processes:
Process 0
  io
  io
  io

Process 1
  cpu
  cpu
  cpu
  cpu
  cpu

Process 2
  cpu
  cpu
  cpu
  cpu
  cpu

Process 3
  cpu
  cpu
  cpu
  cpu
  cpu
```

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_LATER -c
Time    PID: 0    PID: 1    PID: 2    PID: 3       CPU       IOs
  1     RUN:io     READY     READY     READY         1
  2    WAITING   RUN:cpu     READY     READY         1         1
  3    WAITING   RUN:cpu     READY     READY         1         1
  4    WAITING   RUN:cpu     READY     READY         1         1
  5    WAITING   RUN:cpu     READY     READY         1         1
  6*     READY   RUN:cpu     READY     READY         1
  7      READY      DONE   RUN:cpu     READY         1
  8      READY      DONE   RUN:cpu     READY         1
  9      READY      DONE   RUN:cpu     READY         1
 10      READY      DONE   RUN:cpu     READY         1
 11      READY      DONE   RUN:cpu     READY         1
 12      READY      DONE      DONE   RUN:cpu         1
 13      READY      DONE      DONE   RUN:cpu         1
 14      READY      DONE      DONE   RUN:cpu         1
 15      READY      DONE      DONE   RUN:cpu         1
 16      READY      DONE      DONE   RUN:cpu         1
 17     RUN:io      DONE      DONE      DONE         1
 18    WAITING      DONE      DONE      DONE                   1
 19    WAITING      DONE      DONE      DONE                   1
 20    WAITING      DONE      DONE      DONE                   1
 21    WAITING      DONE      DONE      DONE                   1
 22*    RUN:io      DONE      DONE      DONE         1
 23    WAITING      DONE      DONE      DONE                   1
 24    WAITING      DONE      DONE      DONE                   1
 25    WAITING      DONE      DONE      DONE                   1
 26    WAITING      DONE      DONE      DONE                   1
 27*      DONE      DONE      DONE      DONE
```

CPU percentage: 66.67%.

I/Os percentage: 44.44%.
When we add IO_RUN_LATER, we can see the system resources are not utilized efficiently. There are 27 clock ticks. In the second diagram, when Process 0's first I/O and Process 1's first task was done, the second I/O for Process 0 should have been done immediately. However, it didn't. The same situation happened for the third I/O for Process 0. As a result, it took more clock ticks than the situation that I/O can run immediately.

7. Now run the same processes, but with -I IO RUN IMMEDIATE set, which immediately runs the process that issued the I/O. How does this behavior differ? Why might running a process that just completed an I/O again be a good idea?

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_IMMEDIATE -c
Time    PID: 0    PID: 1    PID: 2    PID: 3       CPU       IOs
  1      RUN:io     READY     READY     READY         1
  2     WAITING   RUN:cpu     READY     READY         1         1
  3     WAITING   RUN:cpu     READY     READY         1         1
  4     WAITING   RUN:cpu     READY     READY         1         1
  5     WAITING   RUN:cpu     READY     READY         1         1
  6*     RUN:io     READY     READY     READY         1
  7     WAITING   RUN:cpu     READY     READY         1         1
  8     WAITING      DONE   RUN:cpu     READY         1         1
  9     WAITING      DONE   RUN:cpu     READY         1         1
 10     WAITING      DONE   RUN:cpu     READY         1         1
 11*     RUN:io      DONE     READY     READY         1
 12     WAITING      DONE   RUN:cpu     READY         1         1
 13     WAITING      DONE   RUN:cpu     READY         1         1
 14     WAITING      DONE      DONE   RUN:cpu         1         1
 15     WAITING      DONE      DONE   RUN:cpu         1         1
 16*       DONE      DONE      DONE   RUN:cpu         1
 17        DONE      DONE      DONE   RUN:cpu         1
 18        DONE      DONE      DONE   RUN:cpu         1
```

CPU percentage: 100%.
I/Os percentage: 66.67%.
There are only 18 clock ticks when adding IO_RUN_IMMEDIATE flag. Because I/O can be run immediately after the last I/O is finished. The CPU can accept the new I/O request or run the next process instruction after the last I/O is finished. Then I/O operations and CPU can run simultaneously. Also, because we turn the mode SWITCH_ON_IO on, the CPU starts to run instructions and I/O operations can be executed at the same time, which would save 1 clock tick.

8. Now run with some randomly generated processes: -s 1 -l 3:50,3:50 or -s 2 -l 3:50,3:50 or -s 3 -l 3:50,3:50. See if you can predict how the trace will turn out. What happens when you use the flag -I IO RUN IMMEDIATE vs. -I IO RUN LATER? What happens when you use -S SWITCH ON IO vs. -S SWITCH ON END?

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 1 -l 3:50,3:50 -I IO_RUN_IMMEDIATE -c
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING   RUN:cpu         1         1
  4    WAITING   RUN:cpu         1         1
  5    WAITING   RUN:cpu         1         1
  6    WAITING      DONE                   1
  7*    RUN:io      DONE         1
  8    WAITING      DONE                   1
  9    WAITING      DONE                   1
 10    WAITING      DONE                   1
 11    WAITING      DONE                   1
 12*      DONE      DONE
```

Stats: Total Time 12

Stats: CPU Busy 6 (50.00%)

Stats: IO Busy 8 (66.67%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 1 -l 3:50,3:50 -I IO_RUN_LATER -c
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING   RUN:cpu         1         1
  4    WAITING   RUN:cpu         1         1
  5    WAITING   RUN:cpu         1         1
  6    WAITING      DONE                   1
  7*    RUN:io      DONE         1
  8    WAITING      DONE                   1
  9    WAITING      DONE                   1
 10    WAITING      DONE                   1
 11    WAITING      DONE                   1
 12*      DONE      DONE
```

Stats: Total Time 12

Stats: CPU Busy 6 (50.00%)

Stats: IO Busy 8 (66.67%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 2 -l 3:50,3:50 -I IO_RUN_IMMEDIATE -c
Time    PID: 0    PID: 1       CPU       IOs
  1     RUN:io     READY         1
  2    WAITING   RUN:cpu         1         1
  3    WAITING    RUN:io         1         1
  4    WAITING   WAITING                   2
  5    WAITING   WAITING                   2
  6*    RUN:io   WAITING         1         1
  7    WAITING   WAITING                   2
  8*   WAITING    RUN:io         1         1
  9    WAITING   WAITING                   2
 10    WAITING   WAITING                   2
 11*   RUN:cpu   WAITING         1         1
 12       DONE   WAITING                   1
 13*      DONE      DONE
```

Stats: Total Time 13

Stats: CPU Busy 6 (46.15%)

Stats: IO Busy 11 (84.62%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 2 -l 3:50,3:50 -I IO_RUN_LATER -c
Time    PID: 0    PID: 1       CPU       IOs
  1     RUN:io     READY         1
  2    WAITING   RUN:cpu         1         1
  3    WAITING    RUN:io         1         1
  4    WAITING   WAITING                   2
  5    WAITING   WAITING                   2
  6*    RUN:io   WAITING         1         1
  7    WAITING   WAITING                   2
  8*   WAITING    RUN:io         1         1
  9    WAITING   WAITING                   2
 10    WAITING   WAITING                   2
 11*   RUN:cpu   WAITING         1         1
 12       DONE   WAITING                   1
 13*      DONE      DONE
```

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy 11 (84.62%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 3 -l 3:50,3:50 -I IO_RUN_IMMEDIATE -c
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING    RUN:io         1         1
  4    WAITING   WAITING                   2
  5    WAITING   WAITING                   2
  6    WAITING   WAITING                   2
  7*   RUN:cpu   WAITING         1         1
  8*      DONE    RUN:io         1
  9       DONE   WAITING                   1
 10       DONE   WAITING                   1
 11       DONE   WAITING                   1
 12       DONE   WAITING                   1
 13*      DONE   RUN:cpu         1
```

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy 9 (69.23%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 3 -l 3:50,3:50 -I IO_RUN_LATER -c
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING    RUN:io         1         1
  4    WAITING   WAITING                   2
  5    WAITING   WAITING                   2
  6    WAITING   WAITING                   2
  7*   RUN:cpu   WAITING         1         1
  8*      DONE    RUN:io         1
  9       DONE   WAITING                   1
 10       DONE   WAITING                   1
 11       DONE   WAITING                   1
 12       DONE   WAITING                   1
 13*      DONE   RUN:cpu         1
```

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy 9 (69.23%)

We can see there is no obvious difference between IO_RUN_IMMEDIATE and IO_RUN_END in the above cases. Because the default mode is SWITCH_ON_IO. Usually, the difference between IO_RUN_IMMEDIATE and IO_RUN_END is whether I/O can always grab the CPU once the CPU is idle. However, in these cases, I/Os are not conflicted with CPU executing instructions. Thus, there is no obvious difference in these cases.

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 1 -l 3:50,3:50 -S SWITCH_ON_IO -c -p
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING   RUN:cpu         1         1
  4    WAITING   RUN:cpu         1         1
  5    WAITING   RUN:cpu         1         1
  6    WAITING      DONE                   1
  7*    RUN:io      DONE         1
  8    WAITING      DONE                   1
  9    WAITING      DONE                   1
 10    WAITING      DONE                   1
 11    WAITING      DONE                   1
 12*      DONE      DONE
```

Stats: Total Time 12
Stats: CPU Busy 6 (50.00%)
Stats: IO Busy 8 (66.67%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 1 -l 3:50,3:50 -S SWITCH_ON_END -c -p
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING     READY                   1
  4    WAITING     READY                   1
  5    WAITING     READY                   1
  6    WAITING     READY                   1
  7*    RUN:io     READY         1
  8    WAITING     READY                   1
  9    WAITING     READY                   1
 10    WAITING     READY                   1
 11    WAITING     READY                   1
 12*      DONE   RUN:cpu         1
 13       DONE   RUN:cpu         1
 14       DONE   RUN:cpu         1
```

Stats: Total Time 14
Stats: CPU Busy 6 (42.86%)
Stats: IO Busy 8 (57.14%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 2 -l 3:50,3:50 -S SWITCH_ON_IO -c -p
Time    PID: 0    PID: 1       CPU       IOs
  1     RUN:io     READY         1
  2    WAITING   RUN:cpu         1         1
  3    WAITING    RUN:io         1         1
  4    WAITING   WAITING                   2
  5    WAITING   WAITING                   2
  6*    RUN:io   WAITING         1         1
  7    WAITING   WAITING                   2
  8*   WAITING    RUN:io         1         1
  9    WAITING   WAITING                   2
 10    WAITING   WAITING                   2
 11*   RUN:cpu   WAITING         1         1
 12       DONE   WAITING                   1
 13*      DONE      DONE
```

Stats: Total Time 13

Stats: CPU Busy 6 (46.15%)

Stats: IO Busy 11 (84.62%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 2 -l 3:50,3:50 -S SWITCH_ON_END -c -p
Time    PID: 0    PID: 1       CPU       IOs
  1     RUN:io     READY         1
  2    WAITING     READY                   1
  3    WAITING     READY                   1
  4    WAITING     READY                   1
  5    WAITING     READY                   1
  6*    RUN:io     READY         1
  7    WAITING     READY                   1
  8    WAITING     READY                   1
  9    WAITING     READY                   1
 10    WAITING     READY                   1
 11*   RUN:cpu     READY         1
 12       DONE   RUN:cpu         1
 13       DONE    RUN:io         1
 14       DONE   WAITING                   1
 15       DONE   WAITING                   1
 16       DONE   WAITING                   1
 17       DONE   WAITING                   1
 18*      DONE    RUN:io         1
 19       DONE   WAITING                   1
 20       DONE   WAITING                   1
 21       DONE   WAITING                   1
 22       DONE   WAITING                   1
 23*      DONE      DONE
```

Stats: Total Time 23

Stats: CPU Busy 6 (26.09%)

Stats: IO Busy 16 (69.57%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 3 -l 3:50,3:50 -S SWITCH_ON_IO -c -p
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING    RUN:io         1         1
  4    WAITING   WAITING                   2
  5    WAITING   WAITING                   2
  6    WAITING   WAITING                   2
  7*   RUN:cpu   WAITING         1         1
  8*      DONE    RUN:io         1
  9       DONE   WAITING                   1
 10       DONE   WAITING                   1
 11       DONE   WAITING                   1
 12       DONE   WAITING                   1
 13*      DONE   RUN:cpu         1
```

Stats: Total Time 13
Stats: CPU Busy 6 (46.15%)
Stats: IO Busy 9 (69.23%)

```
1e18@Zuojuns-MacBook-Air file-devices % ./process-run.py -s 3 -l 3:50,3:50 -S SWITCH_ON_END -c -p
Time    PID: 0    PID: 1       CPU       IOs
  1    RUN:cpu     READY         1
  2     RUN:io     READY         1
  3    WAITING     READY                   1
  4    WAITING     READY                   1
  5    WAITING     READY                   1
  6    WAITING     READY                   1
  7*   RUN:cpu     READY         1
  8       DONE    RUN:io         1
  9       DONE   WAITING                   1
 10       DONE   WAITING                   1
 11       DONE   WAITING                   1
 12       DONE   WAITING                   1
 13*      DONE    RUN:io         1
 14       DONE   WAITING                   1
 15       DONE   WAITING                   1
 16       DONE   WAITING                   1
 17       DONE   WAITING                   1
 18*      DONE   RUN:cpu         1
```

Stats: Total Time 18
Stats: CPU Busy 6 (33.33%)
Stats: IO Busy 12 (66.67%)

The difference between SWITCH_ON_IO and SWITCH_ON_END is whether the system can switch when a process issues an I/O. When the mode is SWITCH_ON_END, if there is an I/O operation, there won't be other CPU executions or I/O operations. Thus, the time utilization of SWITCH_ON_END would be less efficient.