Ch.7 Simulation hw
Tyler Holmquist

1. SJF

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p SJF -l 200,200,200
ARG policy SJF
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


Compute the turnaround time, response time, and wait time for each job.
When you are done, run this program again, with the same arguments,
but with -c, which will thus provide you with the answers. You can use
-s <somenumber> or your own job list (-l 10,15,20 for example)
to generate different problems for yourself.

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```

Turnaround = 200+400+600/3 = **400** response = 0+200+400/3 = **200**

FIFO

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p FIFO -l 200,200,200
ARG policy FIFO
ARG jlist 200,200,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 200.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 200.0 )


Compute the turnaround time, response time, and wait time for each job.
When you are done, run this program again, with the same arguments,
but with -c, which will thus provide you with the answers. You can use
-s <somenumber> or your own job list (-l 10,15,20 for example)
to generate different problems for yourself.

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```

Turnaround = 200+400+600/3 = **400** response = 0+200+400/3 = **200** since they all arrive at the same time and are the same length we get the same results for FIFO and SJF

2. SJF

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p SJF -l 100,200,300
ARG policy SJF
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )


Compute the turnaround time, response time, and wait time for each job.
When you are done, run this program again, with the same arguments,
but with -c, which will thus provide you with the answers. You can use
-s <somenumber> or your own job list (-l 10,15,20 for example)
to generate different problems for yourself.

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```

Turnaround = 100+300+600/3 = **333.33** response = 0+100+300/3= **133..33**

FIFO

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p FIFO -l 100,200,300
ARG policy FIFO
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )


Compute the turnaround time, response time, and wait time for each job.
When you are done, run this program again, with the same arguments,
but with -c, which will thus provide you with the answers. You can use
-s <somenumber> or your own job list (-l 10,15,20 for example)
to generate different problems for yourself.

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```

Turnaround = 100+300+600/3 = **333.33** response = 0+100+300/3 = **133.33** again FIFO and SJF will have the same results because everything is coming in at the same time as well as the shortest jobs being put in the front of the list so nothing changes in how they are processed.

3. RR

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p RR -l 300,100,200 -q 1
ARG policy RR
ARG jlist 300,100,200

Here is the job list, with the run time of each job:
  Job 0 ( length = 300.0 )
  Job 1 ( length = 100.0 )
  Job 2 ( length = 200.0 )


Compute the turnaround time, response time, and wait time for each job.
When you are done, run this program again, with the same arguments,
but with -c, which will thus provide you with the answers. You can use
-s <somenumber> or your own job list (-l 10,15,20 for example)
to generate different problems for yourself.

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```

Turnaround = 298+499+600/3 = **465.67** response = 0+1+2 /3 = **1**

FIFO

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p FIFO -l 100,200,300 -q
 1
ARG policy FIFO
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )


Compute the turnaround time, response time, and wait time for each job.
When you are done, run this program again, with the same arguments,
but with -c, which will thus provide you with the answers. You can use
-s <somenumber> or your own job list (-l 10,15,20 for example)
to generate different problems for yourself.

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```

Turnaround = 100+300+600/3 = **333.33** response = 0+100+300/3 =**133.33**

SJF

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p SJF -l 100,200,300 -q
1
ARG policy SJF
ARG jlist 100,200,300

Here is the job list, with the run time of each job:
  Job 0 ( length = 100.0 )
  Job 1 ( length = 200.0 )
  Job 2 ( length = 300.0 )


Compute the turnaround time, response time, and wait time for each job.
When you are done, run this program again, with the same arguments,
but with -c, which will thus provide you with the answers. You can use
-s <somenumber> or your own job list (-l 10,15,20 for example)
to generate different problems for yourself.

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```

Turnaround = 100+300+600 = **333.33** response = 0+100+300/3 = **133.33**

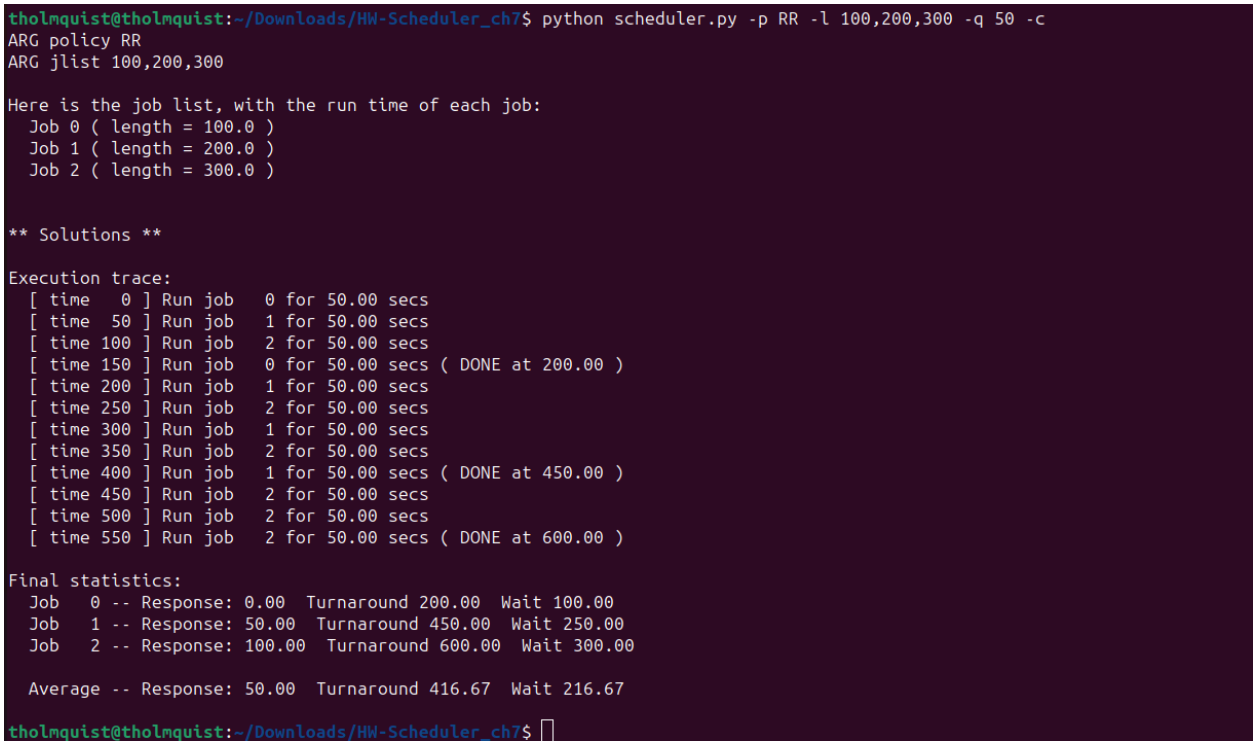
The time slice and round robin method creates an amazing response time but an awful turnaround time. The time slice doesn't affect the FIFO or SJF methods because they both run the job it is on until completion before moving on to the next

4. FJS delivers the same turnaround times as FIFO when all the jobs come in at the same time and in ascending order
5. FJS can deliver a similar response to RR when the workloads are uniform in length and fairly short while the quantum length also resembles around the same length as the workload of each process.
6.

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p SJF -l 1000,2000,3000 -c
ARG policy SJF
ARG jlist 1000,2000,3000

Here is the job list, with the run time of each job:
  Job 0 ( length = 1000.0 )
  Job 1 ( length = 2000.0 )
  Job 2 ( length = 3000.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job 0 for 1000.00 secs ( DONE at 1000.00 )
  [ time 1000 ] Run job 1 for 2000.00 secs ( DONE at 3000.00 )
  [ time 3000 ] Run job 2 for 3000.00 secs ( DONE at 6000.00 )

Final statistics:
  Job   0 -- Response: 0.00  Turnaround 1000.00  Wait 0.00
  Job   1 -- Response: 1000.00  Turnaround 3000.00  Wait 1000.00
  Job   2 -- Response: 3000.00  Turnaround 6000.00  Wait 3000.00

  Average -- Response: 1333.33  Turnaround 3333.33  Wait 1333.33
```

As the length of processes gets longer the response time when using SJF grows with the process length. So we end up getting extremely bad response times as a result.

7.



When you increase the quantum length in round robin the response time grows to that same length. So when the quantum length is 1 the response time will be one similarly when the quantum length is 50 as seen in the image above, the response time is now 50. This can also be seen when changing the lengths of the processes from 100,200, and 300 to 1000, 2000, and 3000 the response time still stays the same. An easy equation for response time would just be R = Q where R is the response time and Q is the quantum length.

```
tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$ python scheduler.py -p RR -l 1000,2000,3000 -q 50 -c
ARG policy RR
ARG jlist 1000,2000,3000

Here is the job list, with the run time of each job:
  Job 0 ( length = 1000.0 )
  Job 1 ( length = 2000.0 )
  Job 2 ( length = 3000.0 )


** Solutions **

Execution trace:
  [ time   0 ] Run job   0 for 50.00 secs
  [ time  50 ] Run job   1 for 50.00 secs
  [ time 100 ] Run job   2 for 50.00 secs
  [ time 150 ] Run job   0 for 50.00 secs
  [ time 200 ] Run job   1 for 50.00 secs
  [ time 250 ] Run job   2 for 50.00 secs
  [ time 300 ] Run job   0 for 50.00 secs
  [ time 350 ] Run job   1 for 50.00 secs
  [ time 400 ] Run job   2 for 50.00 secs
  [ time 450 ] Run job   0 for 50.00 secs
  [ time 500 ] Run job   1 for 50.00 secs
  [ time 550 ] Run job   2 for 50.00 secs
  [ time 600 ] Run job   0 for 50.00 secs
  [ time 650 ] Run job   1 for 50.00 secs
  [ time 700 ] Run job   2 for 50.00 secs
  [ time 750 ] Run job   0 for 50.00 secs
  [ time 800 ] Run job   1 for 50.00 secs
  [ time 850 ] Run job   2 for 50.00 secs
  [ time 900 ] Run job   0 for 50.00 secs
  [ time 950 ] Run job   1 for 50.00 secs
  [ time 1000 ] Run job   2 for 50.00 secs
  [ time 1050 ] Run job   0 for 50.00 secs
  [ time 1100 ] Run job   1 for 50.00 secs
  [ time 1150 ] Run job   2 for 50.00 secs
  [ time 1200 ] Run job   0 for 50.00 secs
Final statistics:
  Job   0 -- Response: 0.00  Turnaround 2900.00  Wait 1900.00
  Job   1 -- Response: 50.00  Turnaround 4950.00  Wait 2950.00
  Job   2 -- Response: 100.00  Turnaround 6000.00  Wait 3000.00

  Average -- Response: 50.00  Turnaround 4616.67  Wait 2616.67

tholmquist@tholmquist:~/Downloads/HW-Scheduler_ch7$
```