CPU Scheduling Project (Done in Python)

CPU Scheduling:
CPU scheduling is an operating system function that determines the order in which tasks or processes are executed on a CPU. It aims to maximize system performance by efficiently allocating CPU time to different processes, balancing resource utilization, minimizing wait times, and ensuring fairness among competing tasks. It selects from a queue of tasks waiting to be executed Some scheduling algorithms that are implemented to do so are First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin (RR).

Round Robin:
Round Robin is a CPU scheduling algorithm where tasks are executed in a circular queue, with each task getting a unit of CPU time before being preempted to allow the next task to run. The fixed time given is called time quantums. This becomes a cycle until all processes execute till fully, to then be terminated. Round Robin ensures fairness by giving each task an equal opportunity to execute and is often used in time-sharing systems. Round Robin is a preempted process that warrants expected response time frames.

Project Implementation:
This was planned by separating process, process creator, and process queue. Process creator is responsible for creating the processes and adding them to the queue. Processes include process ID, run time, arrival time, and the number of turns. The process queue starts and terminates processes and it checks the clock.

Strictly process creators should only create processes. However due to the requirements of the project, this process creator has to have file input handling. Process creator reads in from a txt file. It takes the process ID, Arrival Time, and Burst Time. Furthermore this process creator has a list of processes that will arrive from the file that was read. The list of processes is also only for the simulation. A real CPU cannot predict what processes will come in, before they come in.

The most straightforward object is the process itself. The main purpose is to simulate a real process therefore it has the fields that the project required. The fields are process ID, arrival time, response time, total runtime, and burst time.

The process queue is in charge of the actual Round Robin algorithm. It checks the current queue of processes as well as the current process in a loop for as long as the program runs. This guarantees that the process will have a set time quantum. The loop is the equivalent of a clock.

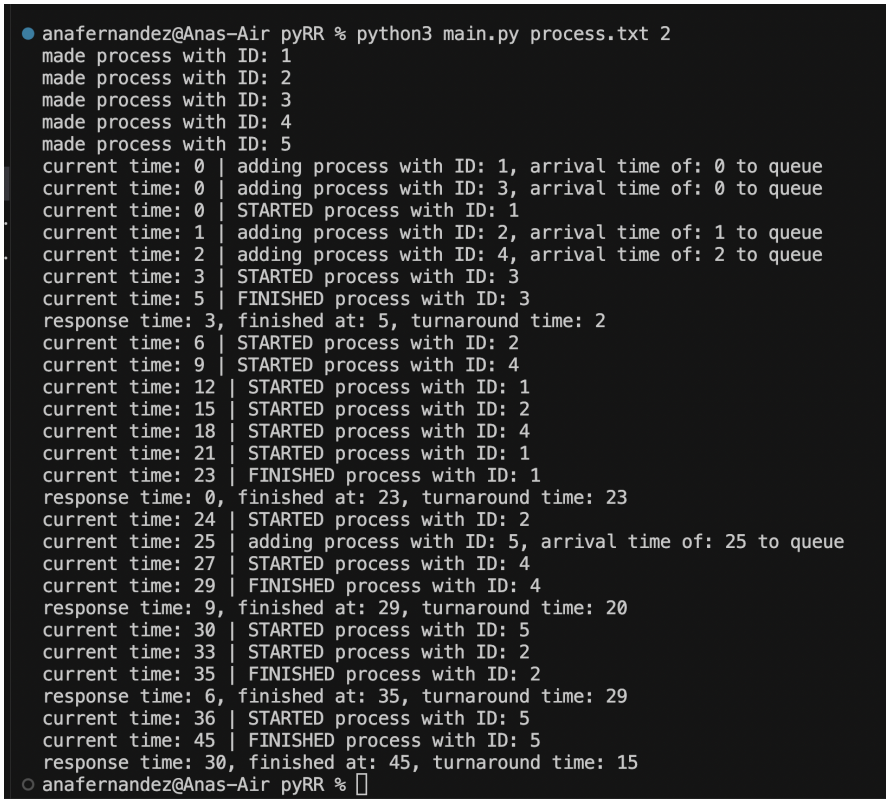
Because of the queue's design the scheduler is aware of the process queue. This means that if a process is executing and no other processes are waiting then the process is not evicted and can continue running until it finishes, but it will not run forever. In normal round robin scheduling it may be evicted and forced into a context switch even though it is the only process.

Analyzing Program:

This project will take in a process txt file that includes process ID, Arrival Time and Burst Time. The delimiter in the txt file is "|". Users can input a time quantum into the terminal to run the algorithm and find the best CPU utilization. This is done by entering:
"python3 main.py <input file path> <time quantum>"
If the time quantum is too short, a lot of time will be wasted on context switching, this reduces CPU utilization. On the other hand if the time quantum is too long processes will end up with a lot of extra time on their turns that they might not need. It is difficult to find a good time quantum because different processes take different times. What is too long for one process may be too short for another

Screenshots with Different Time Quantums:

Time quantum 2:



Time quantum 3:

```
anafernandez@Anas-Air pyRR % python3 main.py process.txt 3
made process with ID: 1
made process with ID: 2
made process with ID: 3
made process with ID: 4
made process with ID: 5
current time: 0 | adding process with ID: 1, arrival time of: 0 to queue
current time: 0 | adding process with ID: 3, arrival time of: 0 to queue
current time: 0 | STARTED process with ID: 1
current time: 1 | adding process with ID: 2, arrival time of: 1 to queue
current time: 2 | adding process with ID: 4, arrival time of: 2 to queue
current time: 4 | STARTED process with ID: 3
current time: 7 | FINISHED process with ID: 3
response time: 4, finished at: 7, turnaround time: 3
current time: 8 | STARTED process with ID: 2
current time: 12 | STARTED process with ID: 4
current time: 16 | STARTED process with ID: 1
current time: 19 | FINISHED process with ID: 1
response time: 0, finished at: 19, turnaround time: 19
current time: 20 | STARTED process with ID: 2
current time: 24 | STARTED process with ID: 4
current time: 25 | adding process with ID: 5, arrival time of: 25 to queue
current time: 27 | FINISHED process with ID: 4
response time: 12, finished at: 27, turnaround time: 15
current time: 28 | STARTED process with ID: 2
current time: 31 | FINISHED process with ID: 2
response time: 8, finished at: 31, turnaround time: 23
current time: 32 | STARTED process with ID: 5
current time: 43 | FINISHED process with ID: 5
response time: 32, finished at: 43, turnaround time: 11
anafernandez@Anas-Air pyRR %
```

Time quantum 4:

```
anafernandez@Anas-Air pyRR % python3 main.py process.txt 4
made process with ID: 1
made process with ID: 2
made process with ID: 3
made process with ID: 4
made process with ID: 5
current time: 0 | adding process with ID: 1, arrival time of: 0 to queue
current time: 0 | adding process with ID: 3, arrival time of: 0 to queue
current time: 0 | STARTED process with ID: 1
current time: 1 | adding process with ID: 2, arrival time of: 1 to queue
current time: 2 | adding process with ID: 4, arrival time of: 2 to queue
current time: 5 | STARTED process with ID: 3
current time: 9 | FINISHED process with ID: 3
response time: 5, finished at: 9, turnaround time: 4
current time: 10 | STARTED process with ID: 2
current time: 15 | STARTED process with ID: 4
current time: 20 | STARTED process with ID: 1
current time: 24 | FINISHED process with ID: 1
response time: 0, finished at: 24, turnaround time: 24
current time: 25 | STARTED process with ID: 2
current time: 25 | adding process with ID: 5, arrival time of: 25 to queue
current time: 29 | FINISHED process with ID: 2
response time: 10, finished at: 29, turnaround time: 19
current time: 30 | STARTED process with ID: 4
current time: 34 | FINISHED process with ID: 4
response time: 15, finished at: 34, turnaround time: 19
current time: 35 | STARTED process with ID: 5
current time: 46 | FINISHED process with ID: 5
response time: 35, finished at: 46, turnaround time: 11
anafernandez@Anas-Air pyRR %
```

Time quantum 5:

```
anafernandez@Anas-Air pyRR % python3 main.py process.txt 5
made process with ID: 1
made process with ID: 2
made process with ID: 3
made process with ID: 4
made process with ID: 5
current time: 0 | adding process with ID: 1, arrival time of: 0 to queue
current time: 0 | adding process with ID: 3, arrival time of: 0 to queue
current time: 0 | STARTED process with ID: 1
current time: 1 | adding process with ID: 2, arrival time of: 1 to queue
current time: 2 | adding process with ID: 4, arrival time of: 2 to queue
current time: 5 | FINISHED process with ID: 1
response time: 0, finished at: 5, turnaround time: 5
current time: 6 | STARTED process with ID: 3
current time: 11 | FINISHED process with ID: 3
response time: 6, finished at: 11, turnaround time: 5
current time: 12 | STARTED process with ID: 2
current time: 18 | STARTED process with ID: 4
current time: 24 | STARTED process with ID: 2
current time: 25 | adding process with ID: 5, arrival time of: 25 to queue
current time: 29 | FINISHED process with ID: 2
response time: 12, finished at: 29, turnaround time: 17
current time: 30 | STARTED process with ID: 4
current time: 35 | FINISHED process with ID: 4
response time: 18, finished at: 35, turnaround time: 17
current time: 36 | STARTED process with ID: 5
current time: 47 | FINISHED process with ID: 5
response time: 36, finished at: 47, turnaround time: 11
anafernandez@Anas-Air pyRR %
```

Time quantum 6:

```
anafernandez@Anas-Air pyRR % python3 main.py process.txt 6
made process with ID: 1
made process with ID: 2
made process with ID: 3
made process with ID: 4
made process with ID: 5
current time: 0 | adding process with ID: 1, arrival time of: 0 to queue
current time: 0 | adding process with ID: 3, arrival time of: 0 to queue
current time: 0 | STARTED process with ID: 1
current time: 1 | adding process with ID: 2, arrival time of: 1 to queue
current time: 2 | adding process with ID: 4, arrival time of: 2 to queue
current time: 6 | FINISHED process with ID: 1
response time: 0, finished at: 6, turnaround time: 6
current time: 7 | STARTED process with ID: 3
current time: 13 | FINISHED process with ID: 3
response time: 7, finished at: 13, turnaround time: 6
current time: 14 | STARTED process with ID: 2
current time: 21 | STARTED process with ID: 4
current time: 25 | adding process with ID: 5, arrival time of: 25 to queue
current time: 27 | FINISHED process with ID: 4
response time: 21, finished at: 27, turnaround time: 6
current time: 28 | STARTED process with ID: 2
current time: 34 | FINISHED process with ID: 2
response time: 14, finished at: 34, turnaround time: 20
current time: 35 | STARTED process with ID: 5
current time: 46 | FINISHED process with ID: 5
response time: 35, finished at: 46, turnaround time: 11
anafernandez@Anas-Air pyRR %
```

From the processes in the process list the total burst time is 31 seconds. With a time quantum of 6 seconds the total runtime was 46 seconds. In here there were 6 context switches. The number of processes was 5. We know at least one took up more than one turn (exceeded 6 seconds). The CPU utilization is $\frac{31}{46}$ which is roughly 68%. The CPU was in use for two thirds of the time which means on average it was mostly in use. This is not optimal. There are other scheduling

algorithms or optimizations that improve performance. For example if the process finishes before the time quantum is up, it can eagerly terminate the process and not make it wait and waste time.