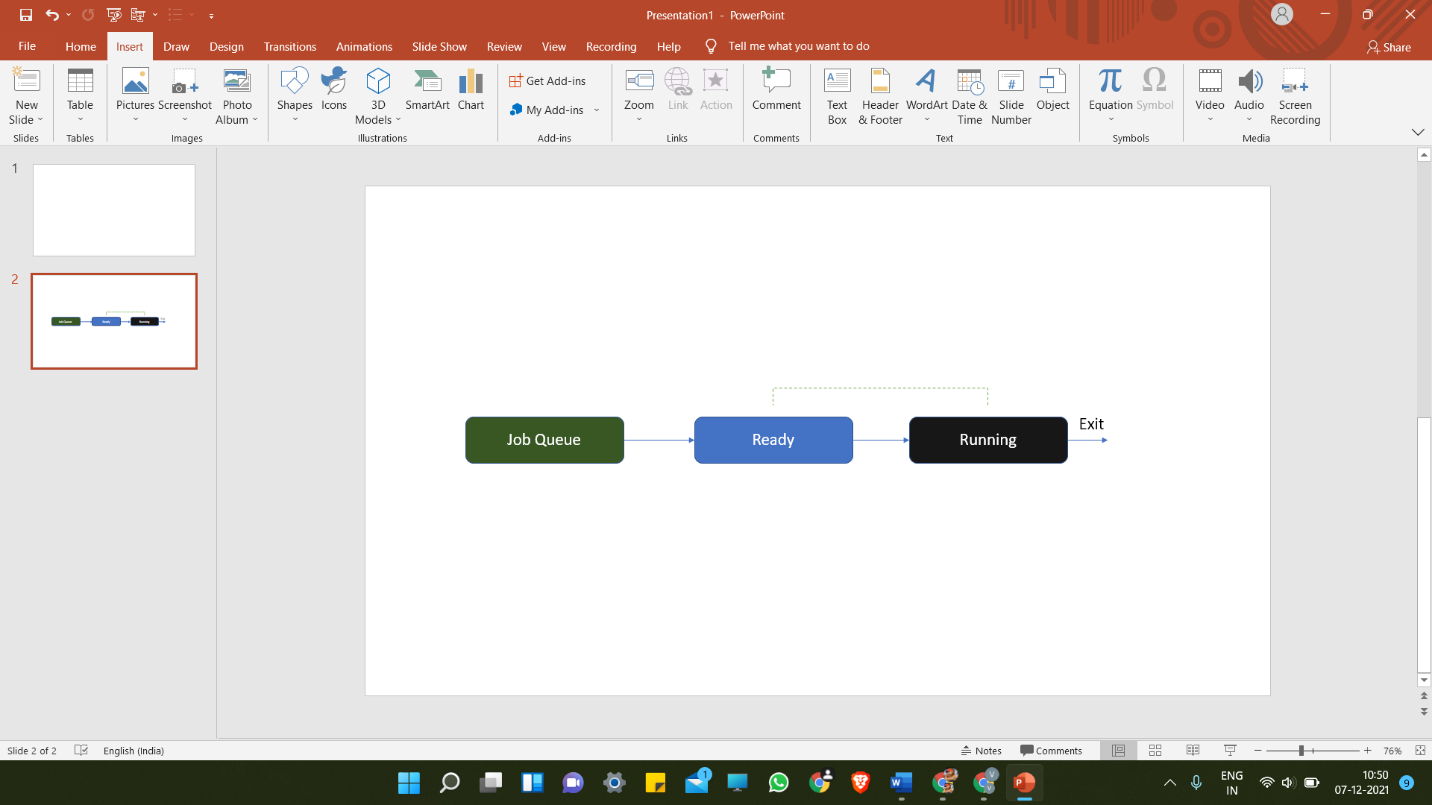
**Time Quantum**

A preemptive scheduler will allow a particular process to run for a short amount of time called a quantum (or time slice). After this amount of time, the process is placed back in the ready queue and another process is placed into the run state (i.e., the scheduler ensures that the processes take turns to run). 

* The size of a quantum has to be selected carefully. Each time the operating system makes a scheduling decision, it is itself using the processor. This is because the operating system comprises one or more processes and it has to use system processing time to do its own computations in order to decide which process to run and actually move them from state to state;
* If the time quanta are too short, the operating system has to perform scheduling activities more frequently, and thus, the overheads are higher as a proportion of total system processing resource.
* On the other hand, if the quanta are too long, the other processes in the ready queue must wait longer between turns, and there is a risk that the users of the system will notice a lack of responsiveness in the applications to which these processes belong.

**Round Robin (RR)**

* Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta)are assigned to each process in equal portions and in circular order, handling all processes without priority (also known as cyclic executive).
* Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept.
* Real time Example:
* If the time slot is 100 milliseconds, and job1 takes a total time of 250 ms to complete, the round-robin scheduler will suspend the job after 100 ms and give other jobs their time on the CPU. Once the other jobs have had their equal share (100 ms each), job1 will get another allocation of CPU time and the cycle will repeat. This process continues until the job finishes and needs no more time on the CPU.
* Job1 = Total time to complete 250 ms (quantum 100 ms).
* First allocation = 100 ms.
* Second allocation = 100 ms.
* Third allocation = 100 ms but job1 self-terminates after 50 ms.
* Total CPU time of job1 = 250 ms
* Consider the following table with the arrival time and execute time of the process with the quantum time of 100 ms to understand the round-robin scheduling:

|  |  |  |
| --- | --- | --- |
| **Process name** | **Arrival time** | **Execute time** |
| P0 | 0 | 250 |
| P1 | 50 | 170 |
| P2 | 130 | 75 |
| P3 | 190 | 100 |
| P4 | 210 | 130 |
| P5 | 350 | 50 |

|  |  |  |
| --- | --- | --- |
| **Process name** | **Arrival time** | **Execute time** |
| P1 | 0 | 5 |
| P2 | 1 | 4 |
| P3 | 2 | 2 |
| P4 | 4 | 1 |

* ------ \* -------
* Ready Queue

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P1 | P4 | P2 | P1 |  |

* Running Queue(GANTT chart)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P1 | P4 | P2 | P1 |  |

* 0 2 4 6 8 9 11 12

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process No | Arrival Time | Burst Time | Completion Time | TAT | WT | RT |
| P1 | 0 | 5 | 12 | 12 | 7 | 0 |
| P2 | 1 | 4 | 11 | 10 | 6 | 1 |
| P3 | 2 | 2 | 6 | 4 | 2 | 2 |
| P4 | 4 | 1 | 9 | 5 | 4 | 4 |

* Average Waiting Time = (7+6+2+4)/4 = 19/4 units