fcfs(processes) Function:

Purpose:

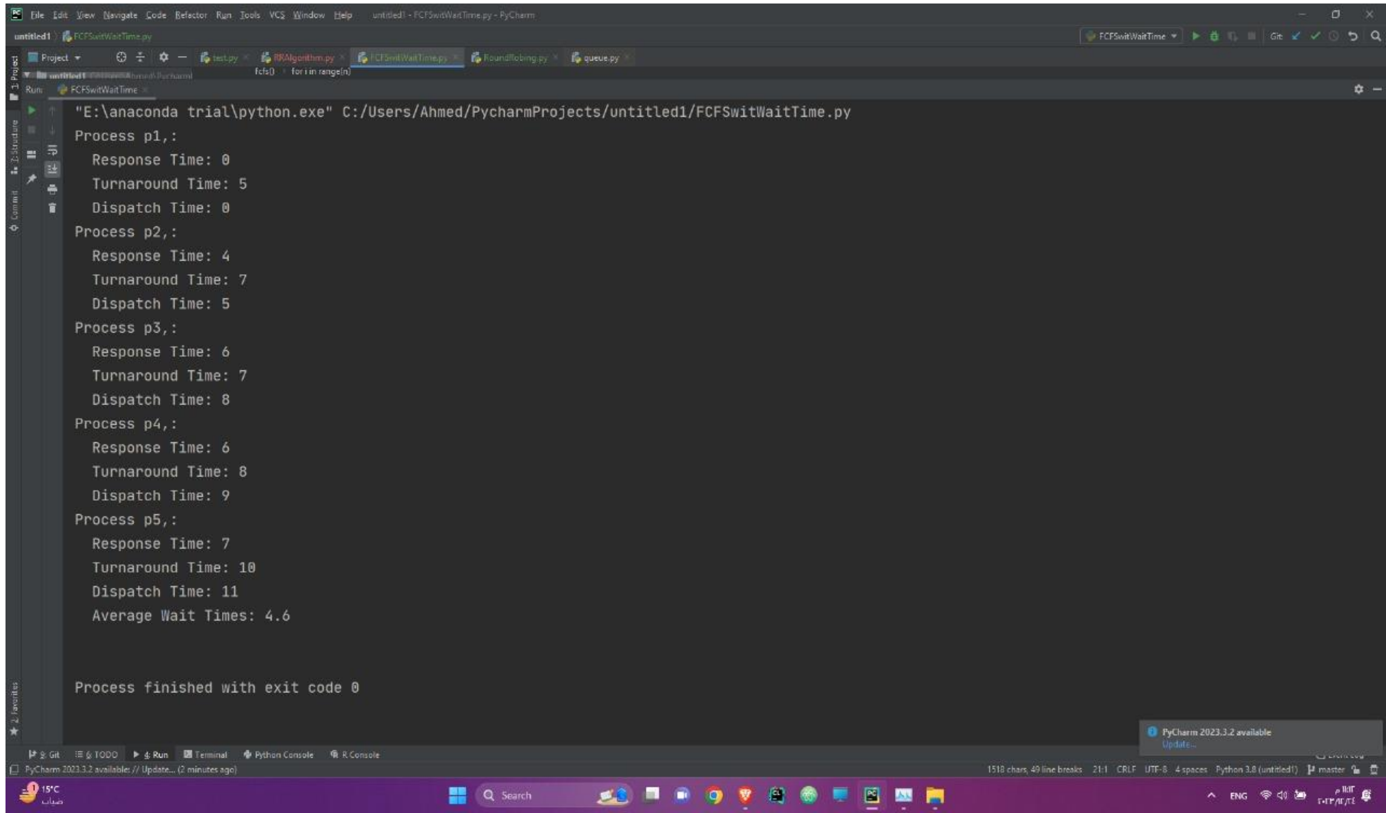
- This function takes a list of processes as input, where each process is represented as a tuple (process_name, arrival_time, burst_time).
- It calculates and prints the response time, turnaround time, and dispatch time for each process using the FCFS algorithm.
- Additionally, it calculates and prints the average wait time for all processes.

read_input_from_file(file_path) Function:

Purpose:

- Reads input data from a file specified by the file_path.
- Each line in the file represents a process, and the format is assumed to be process_name burst_time arrival_time.
- Processes are then converted into tuples and stored in a list.

(FCFS Algorithm)



```
"E:\anaconda trial\python.exe" C:/Users/Ahmed/PycharmProjects/untitled1/FCFSwitWaitTime.py
Process p1,:
  Response Time: 0
  Turnaround Time: 5
  Dispatch Time: 0
Process p2,:
  Response Time: 4
  Turnaround Time: 7
  Dispatch Time: 5
Process p3,:
  Response Time: 6
  Turnaround Time: 7
  Dispatch Time: 8
Process p4,:
  Response Time: 6
  Turnaround Time: 8
  Dispatch Time: 9
Process p5,:
  Response Time: 7
  Turnaround Time: 10
  Dispatch Time: 11
  Average Wait Times: 4.6


Process finished with exit code 0
```

Process Class:

Purpose:

- Represents a process with attributes related to process scheduling.

Attributes:

- arrivalTime: The time at which the process arrives.
- burstTime: The time required by the process to complete its execution.
- startTime: A list to store start times (to handle preemption in Round Robin).
- waitTime: The time the process has to wait before starting its execution.
- responseTime: The time taken from the arrival of the process to its first execution.
- finalTime: The time at which the process completes its execution.
- turnAroundTime: The total time taken from the arrival of the process to its completion.
- pname: The name or identifier of the process.

round_robin_scheduling Function:

Purpose:

- Implements the Round Robin scheduling algorithm and calculates associated metrics.
- Parameters:
- n: Number of processes.
- processes: List of Process objects.
- quantum: Time quantum for each process.

Details:

- Initializes variables and lists for storing scheduling information.
- Iterates through processes, schedules them using Round Robin, and calculates metrics.
- Prints Gantt Chart, average wait time, average turnaround time, and average response time.

read_input_from_file Function:

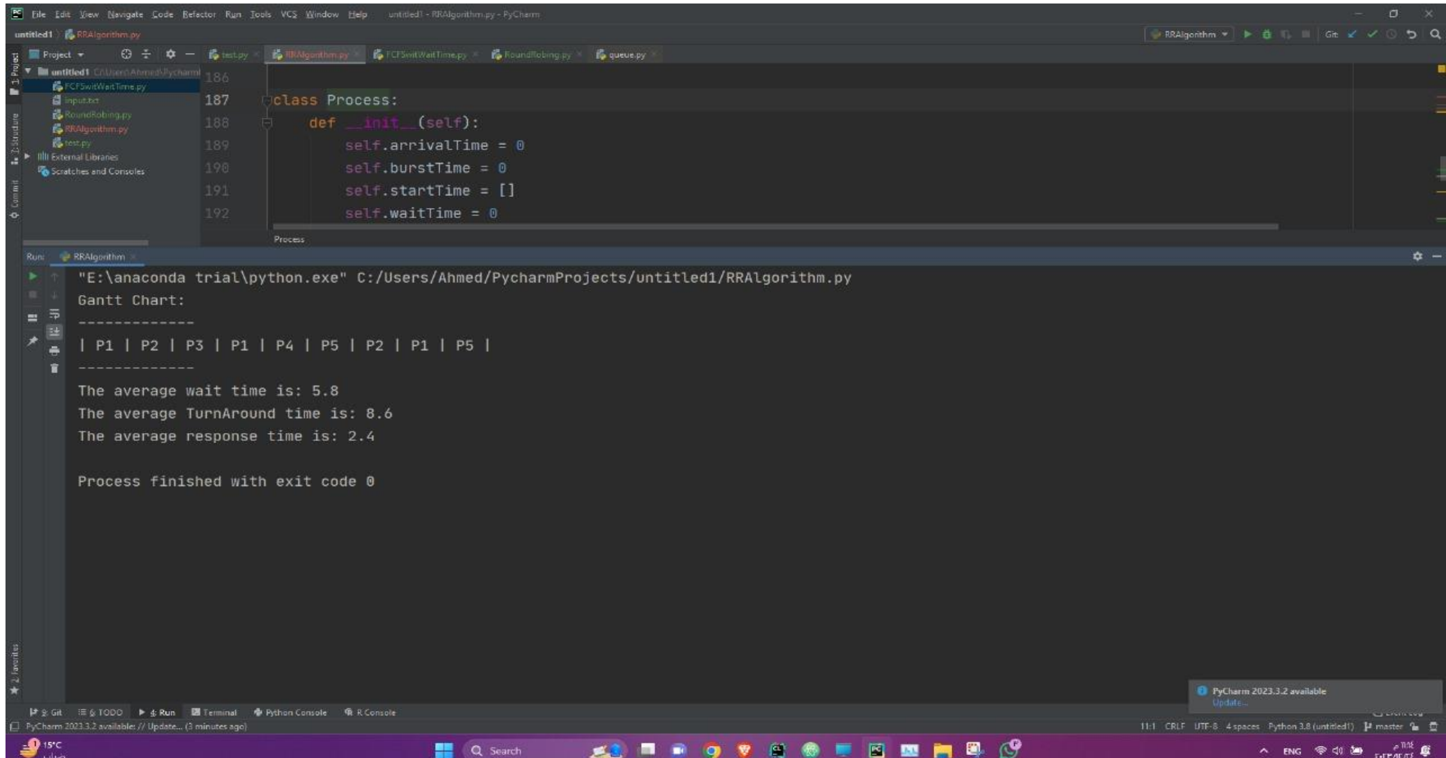
Purpose:

- Reads input data from a file specified by file_path and creates Process objects.
- Parameters:
- file_path: Path to the input file.

Details:

- Opens the specified file and reads lines.
- For each line, extracts process attributes and creates a Process object.
- Returns a list of Process objects.

(Round Robin Algorithm)



```python
class Process:
    def __init__(self):
        self.arrivalTime = 0
        self.burstTime = 0
        self.startTime = []
        self.waitTime = 0
```

```
"E:\anaconda trial\python.exe" C:/Users/Ahmed/PycharmProjects/untitled1/RRAlgorithm.py
Gantt Chart:
-------------
| P1 | P2 | P3 | P1 | P4 | P5 | P2 | P1 | P5 |
-------------
The average wait time is: 5.8
The average TurnAround time is: 8.6
The average response time is: 2.4

Process finished with exit code 0
```

Process Class:

Purpose:

- Represents a process with attributes related to SJF scheduling.

Attributes:

- name: Process identifier.
- arrival_time: The time at which the process arrives.
- burst_time: The time required by the process to complete its execution.
- wait_time: The time the process has to wait before starting its execution.
- start_time: The time at which the process starts execution.
- end_time: The time at which the process completes execution.

SJF_algorithm Function:

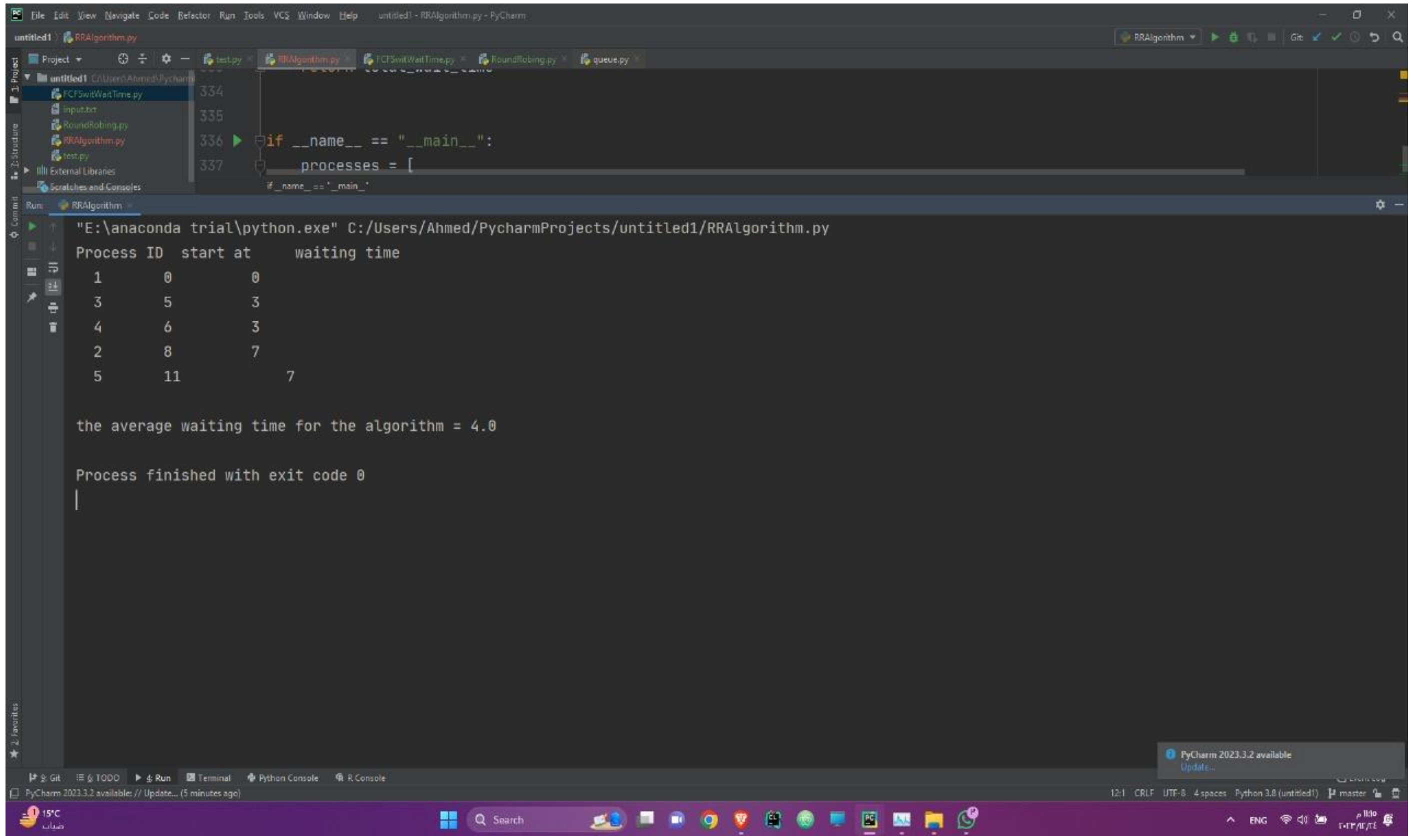
Purpose:

- Implements the SJF scheduling algorithm and calculates the average waiting time.
- Parameters:
- processes: List of Process objects.

Details:

- Initializes variables to track total time, current time, and total wait time.
- Sorts the list of processes by arrival time and burst time.
- Iterates through processes, assigns start times, wait times, and calculates total wait time.
- Prints process information, including process ID, start time, and waiting time.
- Returns the total wait time.

(SJF Algorithm)



```
if __name__ == "__main__":
    processes = [
```

```
"E:\anaconda trial\python.exe" C:/Users/Ahmed/PycharmProjects/untitled1/RRAlgorithm.py
Process ID  start at     waiting time
  1       0          0
  3       5          3
  4       6          3
  2       8          7
  5       11             7

the average waiting time for the algorithm = 4.0

Process finished with exit code 0
```

Process Class:

Purpose:

- Represents a process with attributes related to SRTF scheduling.
- Attributes:
- name: Process identifier.
- arrival_time: The time at which the process arrives.
- burst_time: The time required by the process to complete its execution.
- wait_time: The time the process has to wait before starting its execution.
- start_time: The time at which the process starts execution.
- end_time: The time at which the process completes execution.
- remaining_time: The remaining time for the process to complete.

SRTF_algorithm Function:

Purpose:

- Implements the SRTF scheduling algorithm.
- Parameters:
- processes: List of Process objects.

Details:

- Uses a priority queue to keep track of processes with the shortest remaining time.
- Processes are added to the ready queue based on their arrival time.
- Processes with the shortest remaining time are selected for execution.
- The algorithm updates waiting times, start times, and end times for each process.

read_input_from_file Function:

Purpose:

- Reads process information from a file.

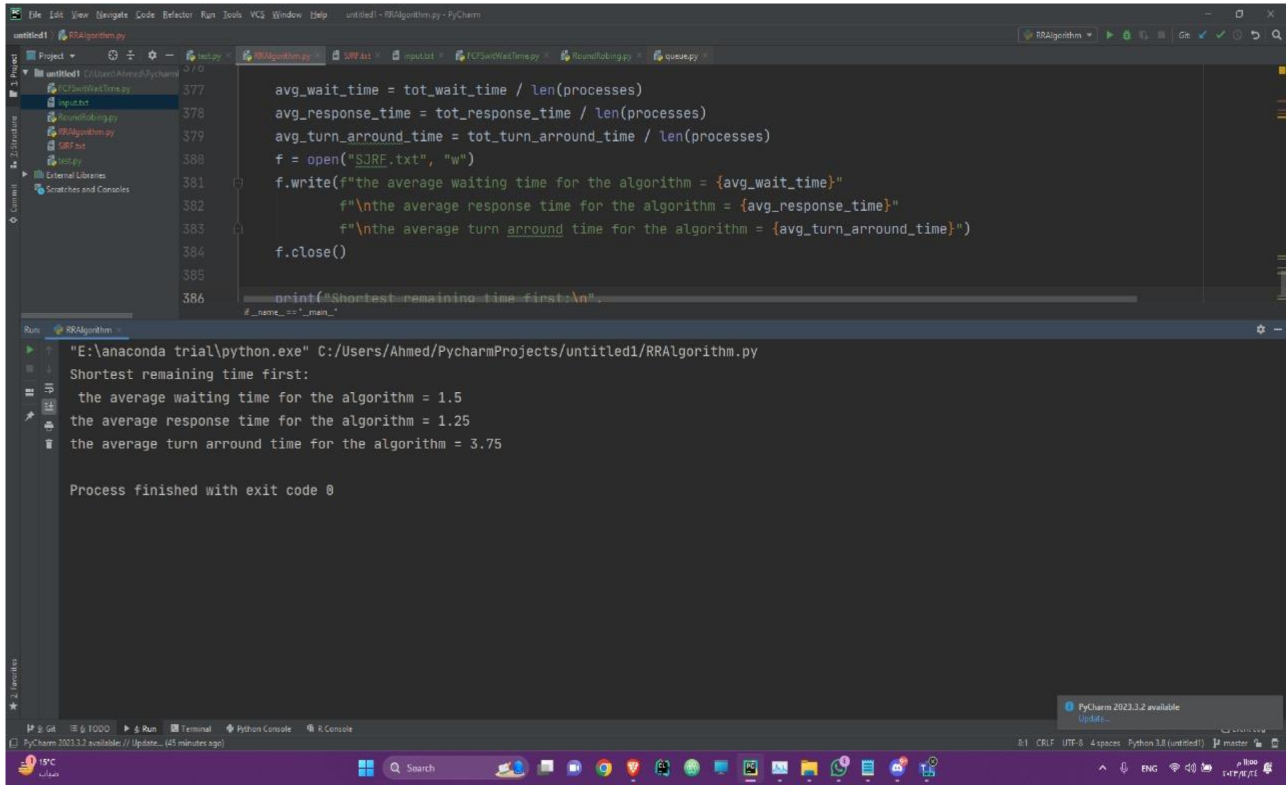
Parameters:

- file_path: Path to the input file.

Details:

- Parses the file to extract process information and creates Process objects.

(SRTF Algorithm)



```
    avg_wait_time = tot_wait_time / len(processes)
    avg_response_time = tot_response_time / len(processes)
    avg_turn_arround_time = tot_turn_arround_time / len(processes)
    f = open("SJRF.txt", "w")
    f.write(f"the average waiting time for the algorithm = {avg_wait_time}"
            f"\nthe average response time for the algorithm = {avg_response_time}"
            f"\nthe average turn arround time for the algorithm = {avg_turn_arround_time}")
    f.close()

    print("Shortest remaining time first:\n",
```

```
"E:\anaconda trial\python.exe" C:/Users/Ahmed/PycharmProjects/untitled1/RRAlgorithm.py
Shortest remaining time first:
 the average waiting time for the algorithm = 1.5
the average response time for the algorithm = 1.25
the average turn arround time for the algorithm = 3.75

Process finished with exit code 0
```

Check me 😉