Department Of Computer Science and Engineering

Course Title: Operating System Lab

Course Code: CSE 406

Title: Optimal Page Replacement Algorithm

Submitted To,

Atia Rahman Orthi
Lecturer
Department Of Computer Science &
Engineering

Submitted By,

Md. Atikul Islam Atik
Reg No:21201063
Roll:63
Sec:B1

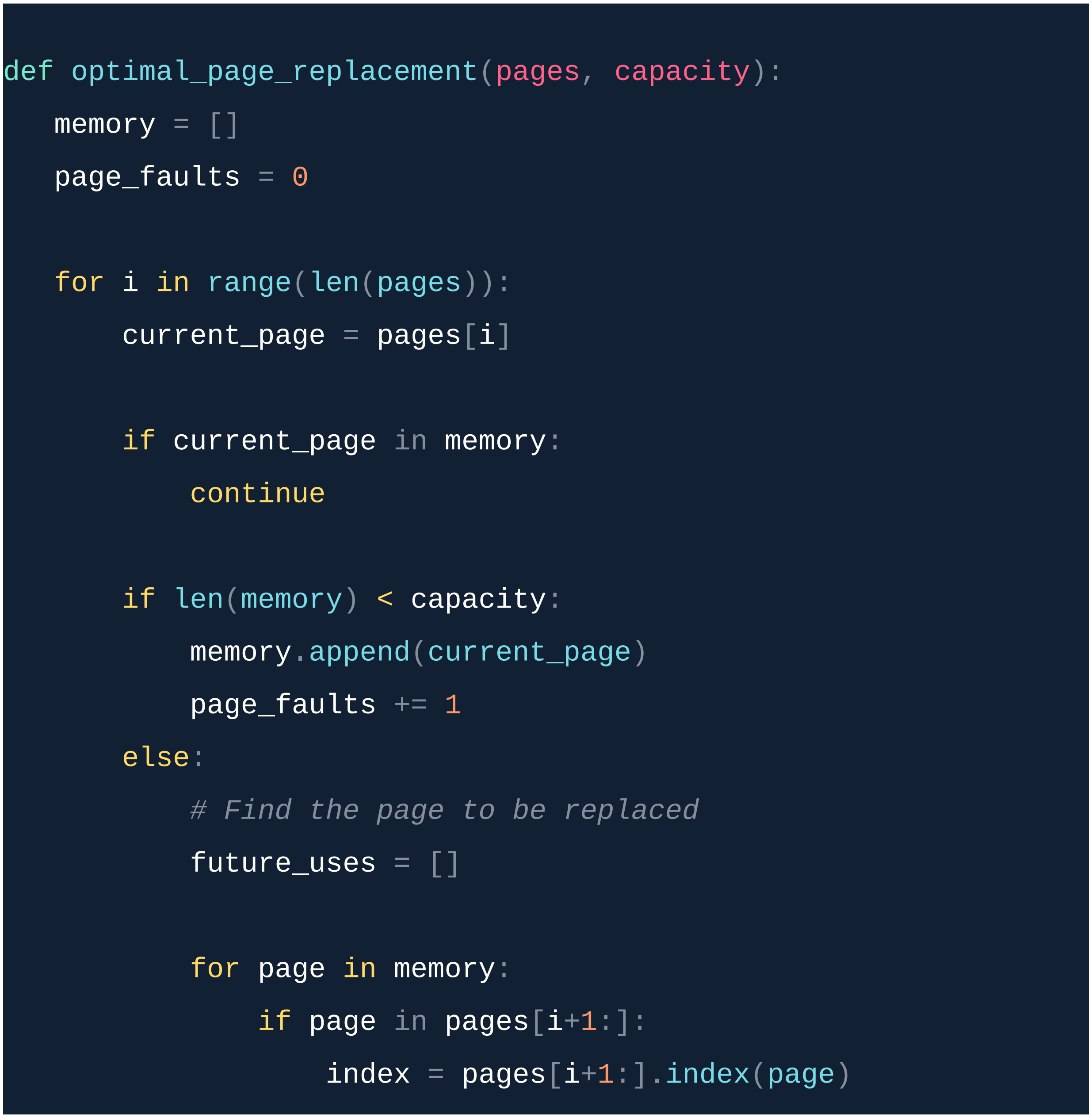
Submission Date: 24-04-2025

Input: **7 0 1 2 0 3 0 4 2 3 0 3 2 3**

Frame Size= 4

Output: Total Page Faults = 6.

Code Snapshot:

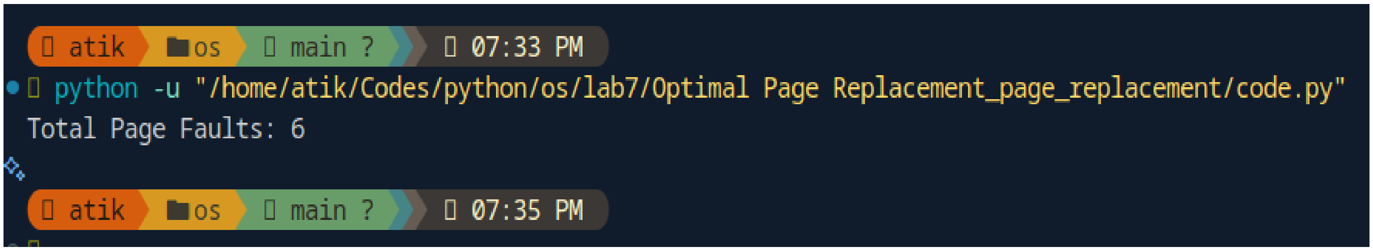


```python
def optimal_page_replacement(pages, capacity):
    memory = []
    page_faults = 0

    for i in range(len(pages)):
        current_page = pages[i]

        if current_page in memory:
            continue

        if len(memory) < capacity:
            memory.append(current_page)
            page_faults += 1
        else:
            # Find the page to be replaced
            future_uses = []

            for page in memory:
                if page in pages[i+1:]:
                    index = pages[i+1:].index(page)
```

```python
               else:
                   index = float('inf')
               future_uses.append(index)

           replace_index = future_uses.index(max(future_uses))
           memory[replace_index] = current_page
           page_faults += 1

   print("Total Page Faults:", page_faults)

pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3]
capacity = 4
optimal_page_replacement(pages, capacity)
```

Code in Github

Output:



```
 atik  os  main ?  07:33 PM
● python -u "/home/atik/Codes/python/os/lab7/Optimal Page Replacement_page_replacement/code.py"
 Total Page Faults: 6

 atik  os  main ?  07:35 PM
```

Code Working Principle:

1. Initialize an empty list to act as the memory frames.
2. Set a counter to track the number of page faults.
3. Loop through each page in the reference string:
    - If the page is already in memory, skip it.
    - If there is still space in memory, add the page and increase the page fault count.
    - If the memory is full:
        - For each page in memory, check when it will next be used in the future.
        - If a page does not appear again, assign it the highest possible index value.
        - Identify the page with the farthest or no future use and replace it with the current one.
        - Increase the page fault count.
4. After processing all pages, display the total number of page faults.

Conclusion:

The Optimal Page Replacement algorithm selects the page that won’t be needed for the longest time in the future. It minimizes page faults by making the best possible decision at each step based on future knowledge. While it provides excellent performance, it is mainly theoretical, as future page references are usually unknown in real-time systems.