**Page Replacement Algorithm Simulator**

**Overview**

This project simulates three popular **page replacement algorithms** used in operating systems to manage memory efficiently:

1. **FIFO (First In, First Out):** Removes the oldest page in memory.
2. **LRU (Least Recently Used):** Removes the page that hasn't been used for the longest time.
3. **Optimal:** Removes the page that won't be used for the longest time in the future (ideal but impractical).

The simulator compares their performance by calculating:

* **Page Hits:** Number of times a page was found in memory, means we don’t have to fetch it from hard disk.
* **Page Faults:** Number of times a page was not found, requiring replacement, meaning we had to go fetch it from hard disk.
* **Hit Ratio** and **Fault Ratio.**

**Features**

* Configure **frame size** (memory capacity) to test algorithm behavior under different conditions.
* Display step-by-step progression of frames for each algorithm.
* Compare all three algorithms in a structured summary.

**How It Works**

1. **Input Parameters:**
   * Page sequence and frame size.
2. **Simulation:**
   * The program applies the algorithms to the page sequence.
3. **Output:**
   * A detailed breakdown of frame states and a comparison table of performance metrics.

**How to Run**

**Prerequisites**

Install Python: Make sure you have Python installed on your machine. You can download it from python.org. Ensure that you install Python 3.11.

1. How to RUN the Project from a ZIP file:
   * Download the ZIP and extract the ZIP file.
   * Navigate to the extracted ZIP folder location.
   * Open the terminal in the project location.
   * **Run the Program (by entering) : python main.py**
2. How to RUN the project from GitHub:
3. Follow the on-screen instructions to:
   * Choose an algorithm or compare all three.
   * Enter page sequence size and frame size.
   * View results and insights.

**Example Output**

**Applications**

This project is ideal for:

* Understanding memory management in operating systems.
* Comparing the effectiveness of different page replacement strategies.
* Educational purposes and algorithm analysis.