LRU Cache Visualizer Documentation

Introduction

The **LRU Cache Visualizer** is a graphical tool built using Python and Tkinter to demonstrate the working of an **LRU (Least Recently Used) Cache**. This application allows users to interactively add, retrieve, and visualize cache entries while observing cache hits and misses in real-time.

Features

- **Interactive GUI** for managing LRU Cache.
- **Dark Mode & Theme Toggle** for better UI experience.
- **Cache Operations:** Initialize, Put, Get, Clear.
- **Statistics Panel:** View cache hits, misses, size, and capacity.
- **Graphical Representation:** Visualize the cache using a queue-like structure.
- **Logging Panel:** Logs operations with timestamps.
- **Audio Feedback:** Beeps on operations for user engagement.

Technologies Used

Technology	Description
Python	Programming language
Tkinter	GUI Framework
OrderedDict	Used for implementing LRU Cache
winsound	Generates beep sounds for feedback

Class Documentation

LRUCache Class

Description

Implements an LRU Cache using an **OrderedDict** to achieve O(1) time complexity for put and get operations.

Attributes

Attribute	Type	Description
capacity	int	Maximum number of elements in cache
cache	OrderedDict	Stores cache elements
hits	int	Tracks successful retrievals
misses	int	Tracks failed retrievals

Methods

Method	Description
`get(key: int) -> int`	Retrieves value for the given key and moves it to the most recently used position. Returns `-1` if key is not found.
`put(key: int, value: int) -> None`	Inserts/updates a key-value pair and removes the least recently used item if capacity is exceeded.
`clear() -> None`	Clears the cache and resets statistics.

CacheVisualizer Class

Description

A modern Tkinter GUI for **visualizing LRU Cache operations**.

Attributes

Attribute	Type	Description
cache	LRUCache	Instance of the cache
dark_mode	bool	Tracks the UI theme state
sound_enabled	bool	Enables or disables beep sounds
capacity_entry	ttk.Entry	Input field for cache capacity
key_entry	ttk.Entry	Input field for key
value_entry	ttk.Entry	Input field for value
canvas	tk.Canvas	Widget for visualizing cache elements
stats_labels	dict	Labels for displaying cache statistics
log	tk.Text	Logs operations

Methods

Method	Description
`_create_widgets()`	Initializes UI components
`_setup_styles()`	Configures UI styles and themes
`initialize_cache()`	Initializes the cache with user-defined capacity
`put_item()`	Handles put operation
`get_item()`	Handles get operation
`clear_cache()`	Clears the cache and updates UI
`_validate_inputs(need_value=False) -> bool`	Validates user input before performing cache operations
`_update_display()`	Refreshes UI elements
`_update_stats()`	Updates cache statistics in the UI
`_visualize_cache()`	Graphically represents cache elements
`_draw_node(x, y, key, value, color)`	Draws cache elements as graphical nodes
`_draw_arrow(x1, y1, x2, y2)`	Draws arrows between cache elements
`_draw_legend()`	Displays LRU and MRU labels
`_log_operation(message: str)`	Logs cache operations with timestamps
`toggle_theme()`	Switches between dark and light mode

`_play_sound(freq: int)`	Generates a beep sound for user feedback

GUI Layout

1. Control Panel

- Input fields for **cache capacity**, **key**, and **value**.
- Buttons for **Initialize Cache**, **Put**, **Get**, and **Clear** operations.

2. Visualization Canvas

- Displays cache elements as **circular nodes**.
- **Green Node:** Most recently used element.
- **Blue Nodes:** Other elements.
- **Arrows:** Indicate LRU-to-MRU flow.

3. Statistics Panel

- Shows **Hits, Misses, Current Size, and Capacity**.
- Includes a **Theme Toggle** button.

4. Log Panel

- Displays a log of cache operations with timestamps.

Example Usage

Running the Application

```
python lru_cache_visualizer.py
```

Initializing the Cache

1. Enter a capacity (e.g., 3) in the input field.

2. Click **Initialize Cache**.

Performing Cache Operations

- **Put Operation**: Enter a key-value pair and click **Put**.
- **Get Operation**: Enter a key and click **Get**.
- **Clear Cache**: Click **Clear** to reset the cache.

Notes

- The LRU Cache follows the **Least Recently Used eviction policy**.
- If the cache is full, the **oldest** (least recently used) item gets removed first.
- Beep sounds indicate cache operations (higher pitch for hits, lower pitch for misses).

Conclusion

This **LRU Cache Visualizer** provides an **interactive way** to understand and experiment with **cache replacement policies**. With an intuitive UI and real-time statistics, users can observe how cache performance is impacted by different access patterns.