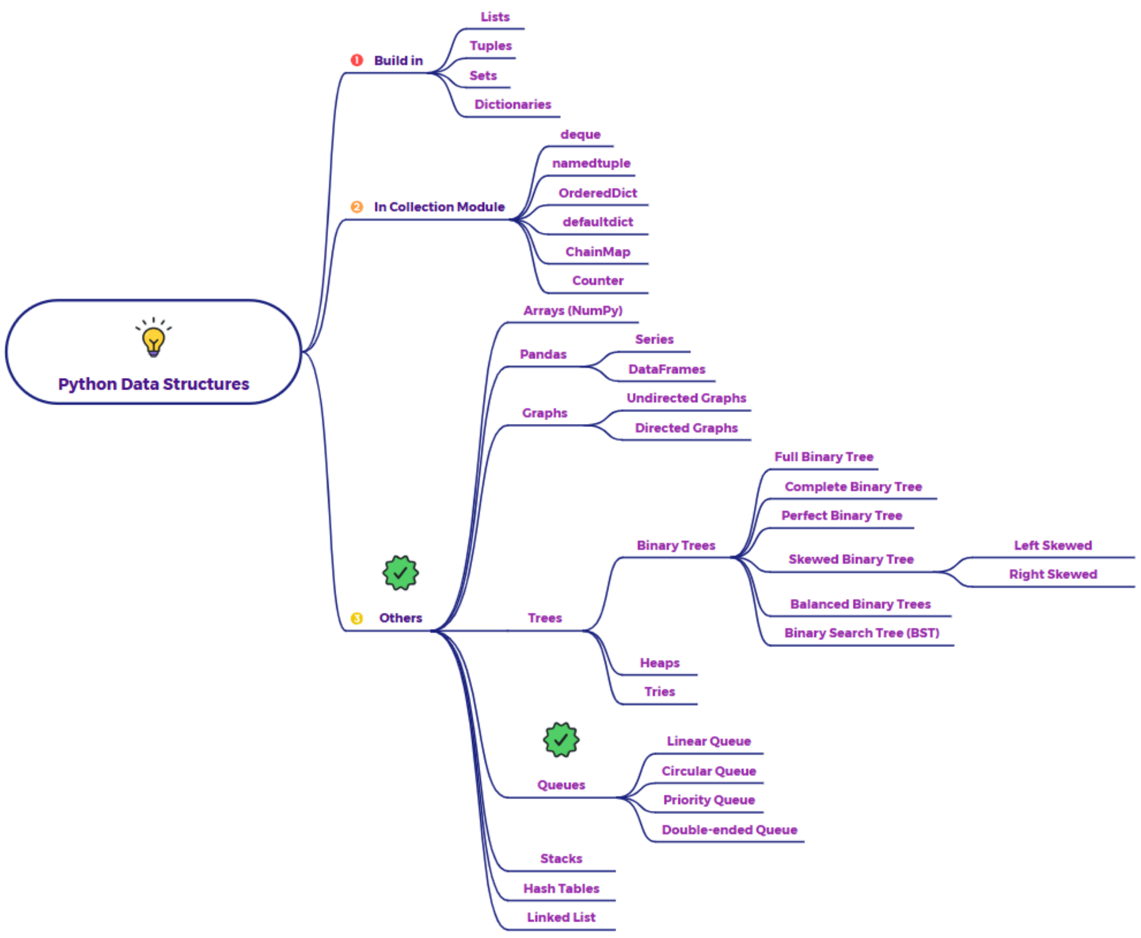
Explain Queues as a data structure in python



Imagine a line of people waiting to buy tickets for a movie. The first person in line is the first one to get their ticket and leave, and new people join at the back of the line. This "first-in, first-out" (FIFO) principle is the fundamental idea behind a **Queue** data structure.

What is a Queue?

A **Queue** is a linear data structure that follows the **First-In, First-Out (FIFO)** principle. This means that the element that was added first to the queue will be the first one to be removed. Think of it like a real-world queue or line.

Key Operations of a Queue:

- **Enqueue (or offer):** Adding a new element to the **rear** (end) of the queue.
- **Dequeue (or poll):** Removing and returning the element from the **front** (beginning) of the queue.

- **Peek (or front):** Viewing the element at the front of the queue without removing it.
- **isEmpty:** Checking if the queue is empty.
- **size:** Getting the number of elements in the queue.

Why Use Queues?

Queues are fundamental in many areas of computer science and real-world applications, including:

- **Task Scheduling:** Managing the order in which tasks are processed (e.g., in operating systems or print queues).
- **Breadth-First Search (BFS):** A graph traversal algorithm that explores nodes level by level.
- **Handling Requests:** Managing incoming requests in servers (e.g., web server request queues).
- **Simulations:** Modeling real-world queuing systems (e.g., customers waiting in a store).
- **Data Buffering:** Temporarily holding data that is being transferred between processes or systems.

In summary, a Queue is a linear data structure that follows the FIFO principle. In Python, collections.deque is the recommended and efficient way to implement a general-purpose queue due to its O(1) time complexity for both enqueue and dequeue operations.