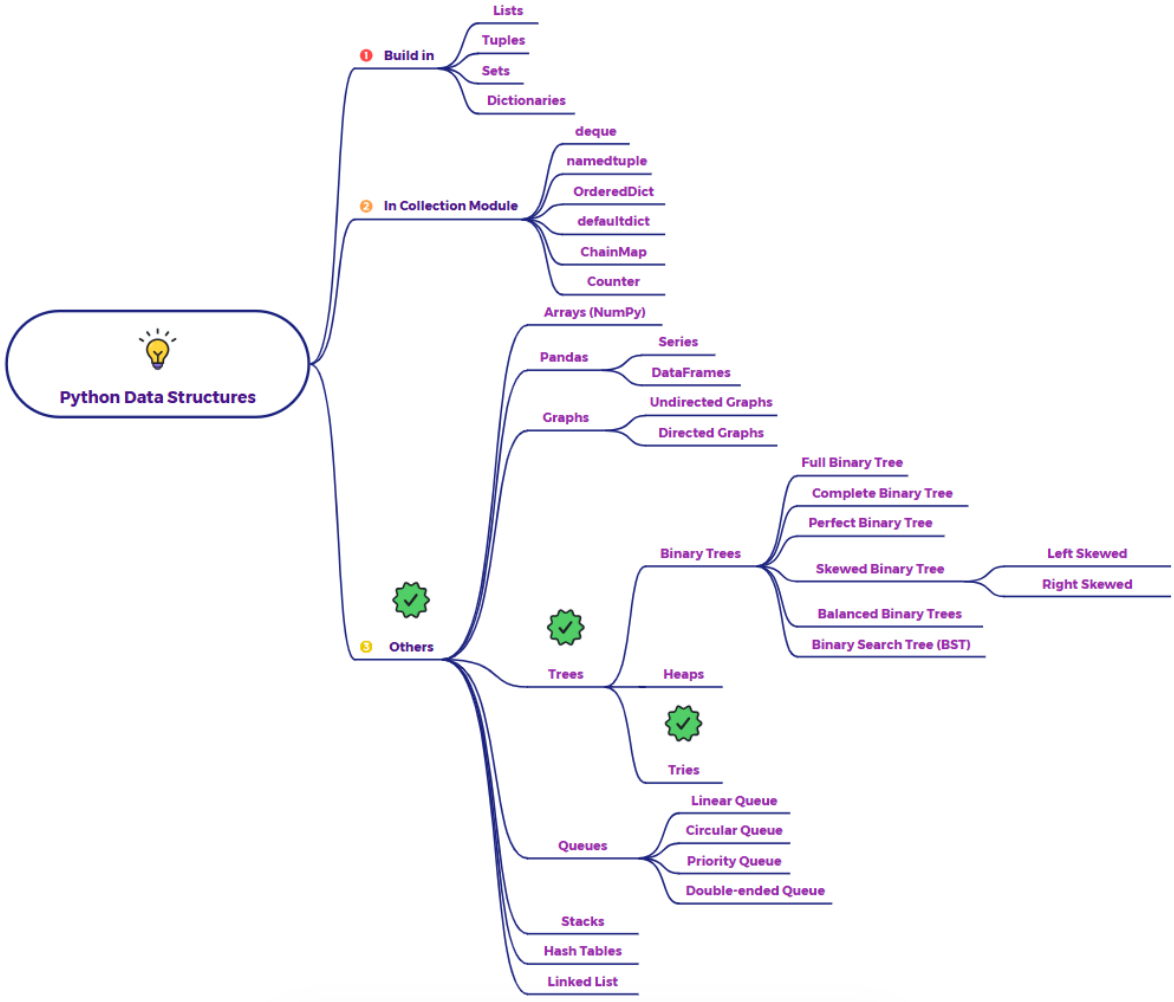
## Explain Tries as a data structure in python




Imagine an English dictionary. All words share common prefixes. For example, "car", "card", and "care" all start with "car". A **Trie** (also known as a prefix tree) is a tree-like data structure that exploits this shared prefix property to store and retrieve strings efficiently.

### What is a Trie?

A **Trie** is a tree-based data structure that is used to store a collection of strings. Each node in a trie represents a prefix of a string, and the edges leading down from a node represent possible next characters in the strings. The root node represents an empty string.

### Key Concepts of a Trie:

- **Nodes:** Each node in a trie (except the root) represents a character.
- **Edges:** Edges connect nodes and represent the sequence of characters forming a prefix or a word.
- **Root:** The root node represents an empty string.
- **Word Termination:** Nodes can be marked to indicate the end of a valid word in the stored set.
- **Prefix Sharing:** Strings with common prefixes share the same path from the root down to the node representing that prefix.

### Why Use Tries?

- **Efficient Prefix Searching:** Tries excel at prefix-based search operations like autocomplete or spell checking. Finding all words with a given prefix is very efficient.
- **Space Efficiency for Shared Prefixes:** If many words share common prefixes, the trie structure can save space by storing the prefix only once.
- **Faster String Searching (in some cases):** For a set of strings, searching for a specific string in a trie can be faster than other data structures like hash sets, especially for prefix-based searches. The time complexity is often proportional to the length of the word being searched.

### Common Use Cases of Tries:

- **Autocomplete and Suggestions:** Predicting and suggesting words as the user types.
- **Spell Checkers:** Finding words with similar prefixes.
- **IP Routing:** Looking up network prefixes.
- **Dictionary Implementation:** Storing and searching words.
- **Pattern Matching:** Finding patterns in text.

In summary, a Trie is a tree-based data structure optimized for storing and retrieving strings based on their prefixes. It's particularly efficient for prefix searching and scenarios where many strings share common starting sequences.