[914-Magdici-Giorgia/LFTC (github.com)](https://github.com/914-Magdici-Giorgia/LFTC)

*SymbolTable* Class

The *SymbolTable* class is designed to create and manage a symbol table, which associates keys (symbols) with corresponding values using a hash table. In cases of collision, it employs chaining to store multiple key-value pairs in the same table slot.

* Constructor

def \_\_init\_\_(self, size: int)

- Initializes a new symbol table with a specified size for the underlying hash table.

- Parameters:

- **size** (int): The size of the hash table.

* Methods

def hash(key: str) -> int

- Computes a hash value for a given key using the sum of ASCII values of its characters, and returns the hash value modulo the size of the hash table.

- Parameters:

**key** (str): The key to be hashed.

- Returns:

An integer representing the index within the hash table where the key-value pair should be stored.

def insert(key: str, value: Any) -> None

- Inserts a key-value pair into the symbol table. Handles collisions using chaining.

- Parameters:

- **key** (str): The key to be inserted.

- **value** (Any): The corresponding value to be associated with the key.

- Returns: None

def lookup(key: str) -> Any

- Retrieves the value associated with a given key from the symbol table.

- Parameters:

- **key** (str): The key to look up.

- Returns:

The value associated with the key if it exists, or `None` if the key is not found in the symbol table.

def delete(key: str) -> None

- Deletes a key and its associated value from the symbol table.

- Parameters:

- key (str): The key to be deleted.

- Returns: None