String Key Hash Table

May 20, 2020

1 String Key Hash Table

1.0.1 Problem Statement

In this quiz, you'll write your own hash table and hash function that uses string keys. Your table will store strings in the buckets. The (bucket) index is calculated by the first two letters of the string, according to the formula below:

```
Hash Value = (ASCII Value of First Letter * 100) + ASCII Value of Second Letter
```

In the formula above, the generated hash value is the (bucket) index.

Example: For a string "UDACITY", the ASCII value for letters 'U' and 'D' are 85 and 68 respectively. The hash value would be: `(85 *100) + 68 = 8568`.

You can use the Python function `ord()` to get the ASCII value of a letter, and `chr()` to get the letter associated with an ASCII value.

Assumptions 1. The string will have at least two letters, 2. The first two characters are uppercase letters (ASCII values from 65 to 90).

Rules - Do not use a Python dictionary—only lists! - Store `lists` at each bucket, and not just the string itself. For example, you can store "UDACITY" at index 8568 as ["UDACITY"].

1.0.2 Instructions

Create a `HashTable` class, with the following functions: - `store()` - a function that takes a string as input, and stores it into the hash table. - `lookup()` - a function that checks if a string is already available in the hash table. If yes, return the hash value, else return -1. - `calculate_hash_value()` - a helper function to calculate a hash value of a given string.

1.0.3 Exercise - Try building a string hash table!

```
In [ ]: """Write a HashTable class that stores strings
        in a hash table, where keys are calculated
        using the first two letters of the string."""

        class HashTable(object):
            def __init__(self):
                self.table = [None]*10000

            def store(self, string):
```

```
        """TODO: Input a string that's stored in
        the table."""
        pass

    def lookup(self, string):
        """TODO: Return the hash value if the
        string is already in the table.
        Return -1 otherwise."""
        return -1

    def calculate_hash_value(self, string):
        """TODO: Helper function to calulate a
        hash value from a string."""
        return -1
```

1.0.4 Test Cases - Let's test your function

```
In [ ]: # Setup
        hash_table = HashTable()

        # Test calculate_hash_value
        print (hash_table.calculate_hash_value('UDACITY'))    # Should be 8568

        # Test lookup edge case
        print (hash_table.lookup('UDACITY'))                  # Should be -1

        # Test store
        hash_table.store('UDACITY')
        print (hash_table.lookup('UDACITY'))                  # Should be 8568

        # Test store edge case
        hash_table.store('UDACIOUS')
        print (hash_table.lookup('UDACIOUS'))                 # Should be 8568
```

Show Solution