hashtable

April 15, 2021

[ ]:
```
ord('h')
```

[ ]:
```
sum(map(ord,'hello world'))
```

[ ]:
```
sum(map(ord,'world hello'))
```

[ ]:
```
sum(map(ord,'gello xorld'))
```

[ ]:
```
def myhash(s):
    mult = 1
    hv = 0
    for ch in s:
        hv += mult * ord(ch)
        mult += 1
    return hv
```

[ ]:
```
print(myhash('hello world'))
print(myhash('world hello'))
print(myhash('gello xorld'))
```

[ ]:
```
print(myhash('ad'))
print(myhash('ga'))
```

0.0.1 Class for Hash item

[ ]:
```
class HashItem:
    def __init__(self, key, value):
        self.key = key
        self.value = value
```

0.0.2 Class for Hash Table

0.0.3 Hash table with linear probing as conflict addressing method.

[ ]:
```
class HashTable:
    def __init__(self):
        self.size = 256
        self.slots = [None for i in range(self.size)]
```

```
        self.count = 0

    def _hash(self, key):
        mult = 1
        hv = 0
        for ch in key:
            hv += mult * ord(ch)
            mult += 1
        return hv % self.size

    def put(self, key, value):
        item = HashItem(key, value)
        h = self._hash(key)

        while self.slots[h] is not None:
            if self.slots[h].key is key:
                break
            h = (h + 1) % self.size
        if self.slots[h] is None:
            self.count += 1
        self.slots[h] = item

    def get(self, key):
        h = self._hash(key)
        while self.slots[h] is not None:
            if self.slots[h].key is key:
                return self.slots[h].value
            h = (h+ 1) % self.size
        return None

    def __setitem__(self, key, value):
        self.put(key, value)

    def __getitem__(self, key):
        return self.get(key)
```

[ ]:
```
ht = HashTable()
```

[ ]:
```
ht.put("good","eggs")
ht.put("better","ham")
ht.put("best","spam")
ht.put("ad","do not")
ht.put("ga","collide")
ht.put("data","value")
```

[ ]:
```
for key in ("good", "better", "best", "worst", "ad", "ga"):
    v = ht.get(key)
```

```
    print(v)
```

[ ]:
```
print("The number of elements is: {}".format(ht.count))
```