Hashing

March 31, 2020

1 Dictionaries in Python

[1]:
```
fruits={
    "banana": 2.95,
    "kiwi": 0.70,
    "strawberry": 9.95,
    "pear": 4.20,
    "apple": 3.95
}
```

[2]:
```
print(fruits)
```

```
{'banana': 2.95, 'kiwi': 0.7, 'strawberry': 9.95, 'pear': 4.2, 'apple': 3.95}
```

[3]:
```
fruits["melon"] = 3.95
fruits["banana"] = 1.90
```

[4]:
```
print("melon in fruits?", "melon" in fruits)
print("onion in fruits?", "onion" in fruits)
print("strawberry", fruits["strawberry"])
```

```
melon in fruits? True
onion in fruits? False
strawberry 9.95
```

[5]:
```
del fruits["strawberry"]
```

[6]:
```
def fruit_price(name):
    if name in fruits:
        print(name,"->",fruits[name])
    else:
        print(name,"not in database")
```

[7]:
```
fruit_price("onion")
fruit_price("banana")
```

```
onion not in database
banana -> 1.9
```

[8]:
```
for name,price in fruits.items():
    print(name,"->",price)
```

```
banana -> 1.9
kiwi -> 0.7
pear -> 4.2
apple -> 3.95
melon -> 3.95
```

2 How can this be possibly implemented?

2.1 Naive attempt

[9]:
```
class Fruit:
    name = ""
    price = 0.0

    def __init__(self, name, price):
        self.name = name
        self.price = price

class Fruits:
    fruits = []

    def __init__(self):
        self.fruits = []

    def add(self, name, price):
        self.fruits.append(Fruit(name,price))

    def find(self, name):
        for i in range(len(self.fruits)):
            if self.fruits[i].name == name:
                return self.fruits[i]
        return None
    def price(self,name):
        fruit = self.find(name)
        if fruit is None:
            print(name,"is not in database")
        else:
            print(fruit.name,"->",fruit.price)

    def print(self):
        for i in range(len(self.fruits)):
            print(self.fruits[i].name, "->", self.fruits[i].price)
```

[10]:
```
myFruits = Fruits()
myFruits.add("banana", 2.95)
myFruits.add("kiwi", 0.70)
myFruits.add("strawberry", 9.95)
myFruits.add("pear", 4.20)
myFruits.add("apple", 3.95)
```

[11]:
```
myFruits.price("onion")
myFruits.price("banana")
```

```
onion is not in database
banana -> 2.95
```

[12]:
```
myFruits.print()
```

```
banana -> 2.95
kiwi -> 0.7
strawberry -> 9.95
pear -> 4.2
apple -> 3.95
```

[13]:
```
import random
n = 10
for i in range(n):
    myFruits.add(random.randint(0,n),i)
```

[14]:
```
myFruits.print()
```

```
banana -> 2.95
kiwi -> 0.7
strawberry -> 9.95
pear -> 4.2
apple -> 3.95
5 -> 0
5 -> 1
5 -> 2
0 -> 3
4 -> 4
5 -> 5
5 -> 6
8 -> 7
3 -> 8
3 -> 9
```

[15]:
```
n = 1000000
for i in range(n):
    myFruits.add(random.randint(0,n),i)
```

[16]:
```
myFruits.price("onion")
myFruits.price("banana")
```

```
onion is not in database
banana -> 2.95
```

[17]:
```
import time
def measure_time(algo,x):
    start=time.time()
    algo(x)
    end = time.time()
    print("execution time", end-start, "s")
```

[18]:
```
measure_time(myFruits.price,"onion")
measure_time(myFruits.price,"banana")
myFruits.add("cherry",3.8)
measure_time(myFruits.price,"cherry")
```

```
onion is not in database
execution time 0.24211406707763672 s
banana -> 2.95
execution time 9.775161743164062e-05 s
cherry -> 3.8
execution time 0.2295229434967041 s
```

[19]:
```
fruits={
    "banana": 2.95,
    "kiwi": 0.70,
    "strawberry": 9.95,
    "pear": 4.20,
    "apple": 3.95
}
```

[20]:
```
n = 1000000
for i in range(n):
    fruits[random.randint(0,n)] = i
```

[21]:
```
measure_time(fruit_price,"onion")
measure_time(fruit_price,"banana")
fruits["cherry"]= 3.7
measure_time(fruit_price,"cherry")
```

```
onion not in database
execution time 0.00032806396484375 s
banana -> 2.95
execution time 0.0003616809844970703 s
cherry -> 3.7
execution time 0.0002543926239013672 s
```

3 (Pre-)Hash Funktionen

3.1 Self-made

[22]:
```
def hash_name(name):
    result = 0
    for i in range(len(name)):
        result = result*31;
        result = result + ord(name[i])
    return result
```

[23]:
```
print("Anna","->", hash_name("Anna"))
print("Jacqueline","->",hash_name("Jacqueline"))
```

```
Anna -> 2045632
Jacqueline -> 2042089953442505
```

[24]:
```
hash_name("Jacqueline") % 2**32
```

[24]:
```
507919049
```

[25]:
```
def hash_name_trunc(name):
    result = 0
    for i in range(len(name)):
        result = result*31 % 2**32;
        result = result + ord(name[i])
    return result
```

[26]:
```
print("Anna","->", hash_name_trunc("Anna"))
print("Jacqueline","->",hash_name_trunc("Jacqueline"))
```

```
Anna -> 2045632
Jacqueline -> 507919049
```

3.2 Offered by (and used in) Python

[27]:
```
print("Anna","->",hash("Anna"))
print("annA","->",hash("Jacqueline"))
```

```
Anna -> -861198018981117959
annA -> 3330533257378696375
```

[ ]:
```

```