

# LT10d : About Hashing

---

(Part 1 of 2)

# About Hashing ...

1. Hash Function
2. Hash Table
3. Collision Resolution
  - Open Hash
  - Separate Chain
4. Searching

# Some basic questions :

1. How do you store 1, 2, 3, 4, 5 in 5 boxes?

Box Index	
0	1
1	2
2	3
3	4
4	5

# Some basic questions :

1. How do you store 1, 2, 3, 4, 5 in 5 boxes?
2. How do you store 11, 12, 13, 14, 15 in 5 boxes?

```
def hash(n) :  
    return (n-11)
```

Box Index	
0	11
1	12
2	13
3	14
4	15

## Some basic questions :

1. How do you store 1, 2, 3, 4, 5 in 5 boxes?
2. How do you store 11, 12, 13, 14, 15 in 5 boxes?
3. How do you store 2, 5, 8, 14, 21 in 5 boxes?

# Using 'mod' :

The “*mod*” **modulo** operation in Computing, is also known as the **modulus** in Mathematics, finds the *remainder* after a number **a** (*dividend*) is divided by another number **n** (*divisor*).

Abbreviation : **a mod n**

Python code : **a % n**

# Using 'mod':

3. How to store 2, 5, 8, 14, 21 in 5 boxes?

```
def hash(n) :
```

```
    return n % 5
```

hash(2) = 2

hash(5) = 0

hash(8) = 3

hash(14) = 4

hash(21) = 1

Box Index	
0	5
1	21
2	2
3	8
4	14

So lucky there is no collision !

# Some basic questions :

1. How to store 1, 2, 3, 4, 5 in 5 boxes?
2. How to you store 11, 12, 13, 14, 15 in 5 boxes?
3. How to store 2, 5, 8, 14, 21 in 5 boxes?
4. How to store 'a', 'b', 'c', 'd', 'e' in 5 boxes?



# Using the ASCII\* Code :

\* **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

(<https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html>)

Python code : `ord('a') = 97`

(<https://docs.python.org/3/library/functions.html#ord>)

Python code : `chr(97) = 'a'`

(<https://docs.python.org/3/library/functions.html#chr>)

# Using the ASCII\* Code:

4. How to store 'a', 'b', 'c', 'd', 'e' in 5 boxes?

```
def hash(char) :
```

```
    return ord(char) % 5
```

So lucky there is no collision !

Box Index	
0	'd'
1	'e'
2	'a'
3	'b'
4	'c'

$\text{hash}('a') = 97\%5 = 2$

$\text{hash}('b') = 98\%5 = 3$

$\text{hash}('c') = 99\%5 = 4$

$\text{hash}('d') = 100\%5 = 0$

$\text{hash}('e') = 101\%5 = 1$

## Some basic questions :

1. How to store 1, 2, 3, 4, 5 in 5 boxes?
2. How to you store 11, 12, 13, 14, 15 in 5 boxes?
3. How to store 2, 5, 8, 14, 21 in 5 boxes?
4. How to store 'a', 'b', 'c', 'd', 'e' in 5 boxes?
5. How to store 'abc', 'bca', 'cab', 'cba', 'bac' in 5 boxes?

## Using the sum of the ASCII\* Codes:

```
def hash(string):  
    total = 0  
    for char in string:  
        total += ord(char)  
    return total%5
```

# Using the sum of the ASCII\* Codes:

5. How to store 'abc', 'bca', 'cab', 'cba', 'bac' in 5 boxes?

$$\text{hash('abc')} = 294\%5 = 4$$

$$\text{hash('bca')} = 294\%5 = 4$$

$$\text{hash('cab')} = 294\%5 = 4$$

$$\text{hash('cba')} = 294\%5 = 4$$

$$\text{hash('bac')} = 294\%5 = 4$$

Serious collision !!!  
So how?

Box Index	
0	
1	
2	
3	
4	???

Using the sum of the ASCII\* Codes with the index positions of the characters:

```
def hash(string):  
    total = 0  
    for i in range(len(string)):  
        total += ord(char) * (i+1)  
    return total%5
```

We will use this Hash Function  
from here on.

# Using the sum of the ASCII\* Codes with the index positions of the characters:

5. How to store 'abc', 'bca', 'cab', 'cba', 'bac' in 5 boxes?

$$\text{hash}(\text{'abc'}) = 590\%5 = 0$$

$$\text{hash}(\text{'bca'}) = 587\%5 = 2$$

$$\text{hash}(\text{'cab'}) = 587\%5 = 2$$

$$\text{hash}(\text{'cba'}) = 586\%5 = 1$$

$$\text{hash}(\text{'bac'}) = 589\%5 = 4$$

a collision here,  
so how?

Box Index	
0	'abc'
1	'cba'
2	'bca' or 'cab'
3	
4	'bac'

# Collision Resolution :

1. Open hashing : look for the nearest empty “slot”.

Box Index	
0	'abc'
1	'cba'
2	'bca' or 'cab'
3	
4	'bac'



Box Index	
0	'abc'
1	'cba'
2	'bca'
3	'cab'
4	'bac'



# Collision Resolution :

2. “Separate Chain” : create a separate list at the particular slot.

Box Index	
0	'abc'
1	'cba'
2	['bca', 'cab']
3	
4	'bac'

# Python codes

---

# The Hash Function

```
def hash(string):  
    total = 0  
    for i in range(len(string)):  
        total += ord(char) * (i+1)  
    return total%5
```

# Create an empty Hash Table:

```
def init_table(n):  
    table = []      # declare  
    table += [''] * n # initialize  
    return table
```

# Populate the Hash Table using the Hash Values:

```
def hashtable(seq):  
    tbl = init_table(len(seq))  
    for ele in seq:  
        i = hash(ele)  
        if tbl[i] == '':  
            tbl[i] = ele  
        else:  
            #collision resolution  
            print(ele, ' is not added.')  
    return tbl
```

# Using Hash Function and Table:

```
>>> lst = ['abc', 'bca', 'cab', 'cba', 'bac']  
>>> data_table = hashtable(lst)
```

## Some basic questions :

6. How to store 5 names in 5 boxes?

- Chloe Niu Man Yun  $\text{Sum} = 13465, \text{Mod}5 = 0$
- Ngyuen Hoang Minh  $\text{Sum} = 14092, \text{Mod}5 = 2$
- Poh Zheng Hong  $\text{Sum} = 9646, \text{Mod}5 = 1$
- Suresh Kannan Sakthieshwar  $\text{Sum} = 35379, \text{Mod}5 = 4$
- Wong Yong Xiang  $\text{Sum} = 11223, \text{Mod}5 = 3$

# Using Hash Function and Table:

```
>>> lst = ['Chloe Niu Man Yun', 'Ngyuen Hoang  
Minh', 'Poh Zheng Hong', 'Suresh Kannan  
Sakthieshwar', 'Wong Yong Xiang']
```

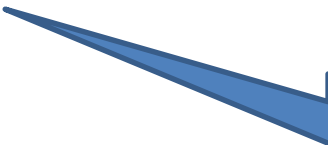
```
>>> data_table = hashtable(lst)
```



# Using the sum of the ASCII\* Codes with the index positions of the characters:

How are the 5 names stored in the 5 boxes?

Box Index	
0	'Chloe Niu Man Yun'
1	'Poh Zheng Hong'
2	'Ngyuen Hoang Minh'
3	'Wong Yong Xiang'
4	'Suresh Kannan Sakthieshwar'

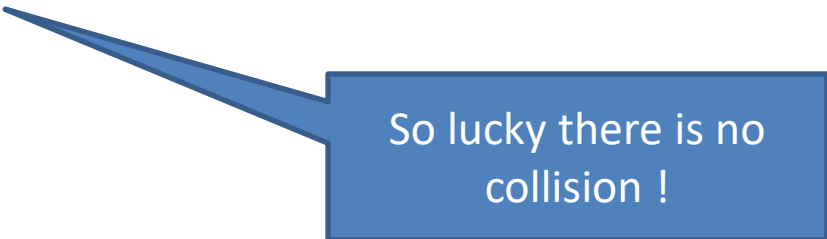


So lucky there is no collision !

## Some basic questions :

### 7. Why store the names in a hash table?

Box Index	
0	'Chloe Niu Man Yun'
1	'Poh Zheng Hong'
2	'Ngyuen Hoang Minh'
3	'Wong Yong Xiang'
4	'Suresh Kannan Sakthieshwar'



So lucky there is no collision !

## Searching for a name in Hash Table:

```
def search(table, name):  
    i = hash(name)  
    if table[i] == name:  
        return True  
    else:  
        return False
```

```
>>> search(data_table, 'Wong Yong Xiang')
```

To be continue ...

---