**🔹 What is a Hash Map?**

A **Hash Map** is a data structure used to store **key-value pairs**.  
In Python, the built-in data type dict behaves like a hash map.

**🔹 Real-Life Analogy**

Think of a **contact list**:

* **Key**: Person's name
* **Value**: Phone number

"Rahul" → 9876543210

**🔹 Key Features:**

* Stores **(key, value)** pairs
* Fast **insertion**, **deletion**, and **lookup**
* Uses a **hash function** to compute an index for keys
* Handles **collisions** using techniques like chaining

**🔹 Creating a Hash Map in Python:**

# Define a dictionary (hash map)

my\_map = {

"apple": 10,

"banana": 20,

"orange": 15

}

**🔹 Common Operations:**

# Access value

print(my\_map["apple"]) # Output: 10

# Add or update

my\_map["banana"] = 25 # Update

my\_map["grape"] = 12 # Add new key

# Delete a key

del my\_map["orange"]

# Check if key exists

if "banana" in my\_map:

print("Found")

# Iterate over keys and values

for key, value in my\_map.items():

print(key, "→", value)

**🔹 Internal Working:**

1. Uses a **hash function** on the key
2. Hash value → index in an internal array
3. Key-value stored at that index
4. If two keys map to same index → **collision**
   * Handled using **chaining** (list at each index)

**🔹 Time Complexity:**

| **Operation** | **Average Time** | **Worst Case** |
| --- | --- | --- |
| Insert | O(1) | O(n) |
| Delete | O(1) | O(n) |
| Search/Get | O(1) | O(n) |

Worst case happens when many keys collide into one bucket.

**🔹 When to Use:**

* Fast lookups needed by key
* Data association (like name → age, item → price)
* Implementing caches, counters, or dictionaries

**🔹 Custom Hash Map (using List & Hashing):**

class HashMap:

def \_\_init\_\_(self):

self.size = 10

self.map = [[] for \_ in range(self.size)]

def \_get\_hash(self, key):

return hash(key) % self.size

def add(self, key, value):

h = self.\_get\_hash(key)

for pair in self.map[h]:

if pair[0] == key:

pair[1] = value

return

self.map[h].append([key, value])

def get(self, key):

h = self.\_get\_hash(key)

for pair in self.map[h]:

if pair[0] == key:

return pair[1]

return None

**🔹 Output Example:**

hm = HashMap()

hm.add("apple", 100)

hm.add("banana", 200)

print(hm.get("apple")) # Output: 100