Hash map usage in python

A hash map, also known as a hash table or dictionary in Python, is a data structure that allows you to store and retrieve values based on a key. Python's built-in dict type provides hash map functionality. Here's how you can use a hash map (dictionary) in Python:

Creating a Hash Map:

You can create a hash map using curly braces {} or the dict() constructor:

```
# Using curly braces
my_dict = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
# Using dict() constructor
my_dict = dict(key1='value1', key2='value2', key3='value3')
```

Accessing and Modifying Values:

You can access and modify values in a hash map using the keys:

```
# Accessing values
value = my_dict['key1']
print(value)  # Output: value1
# Modifying values
my_dict['key2'] = 'new_value'
```

Adding and Deleting Entries:

You can add and delete key-value pairs in a hash map:

```
# Adding a new entry
my_dict['key4'] = 'value4'
# Deleting an entry
del my_dict['key3']
```

Checking for Key Existence:

You can use the in operator to check if a key exists in the hash map:

```
if 'key1' in my_dict:
    print('Key exists')
```

Iterating Through a Hash Map:

You can iterate through the keys, values, or items (key-value pairs) of a hash map:

```
# Iterating through keys
for key in my_dict:
    print(key)
# Iterating through values
for value in my_dict.values():
    print(value)
# Iterating through items (key-value pairs)
for key, value in my_dict.items():
    print(key, value)
```

Handling Non-Existent Keys:

If you try to access a key that doesn't exist in the hash map, it will raise a KeyError. You can avoid this by using the get() method:

```
value = my_dict.get('nonexistent_key', 'default_value')
print(value)  # Output: default_value
```

Python's built-in hash map (dictionary) is a powerful and versatile data structure that can be used in various scenarios, such as storing data, caching, and quickly looking up values based on keys.