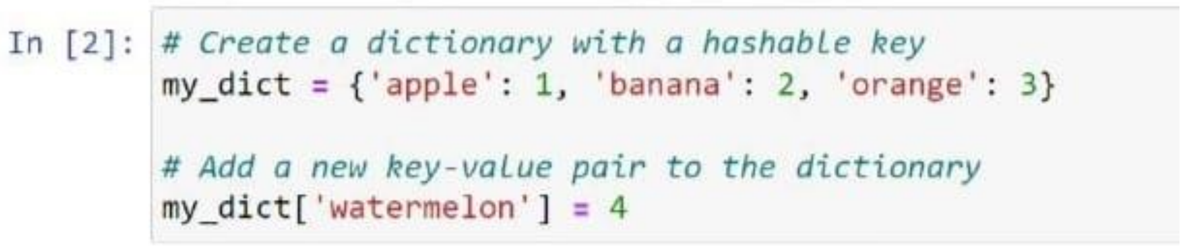
Usage of Hash Function in python data structure

A hash function is a mathematical function that takes an input (also known as the "key") and returns a fixed-size output, usually a string of characters. The output of the hash function is known as the hash value or hash code. The hash value is typically used to index into an array or hash table, allowing for efficient storage and retrieval of data.

Hash functions are commonly used in data structures to efficiently store and retrieve data. Here are some examples:

1. Dictionaries:

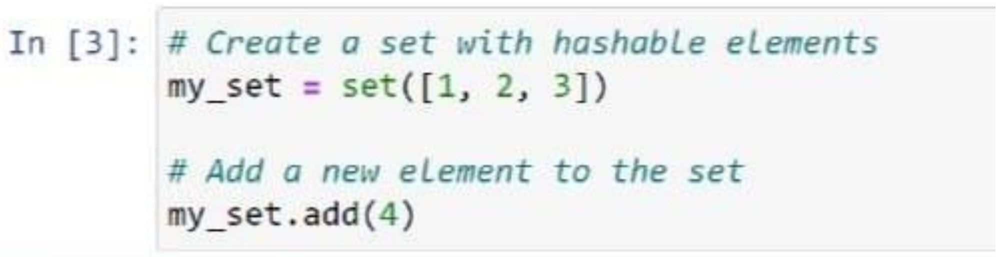


```
In [2]: # Create a dictionary with a hashable key
        my_dict = {'apple': 1, 'banana': 2, 'orange': 3}

        # Add a new key-value pair to the dictionary
        my_dict['watermelon'] = 4
```

In this example, the keys ('apple', 'banana', 'orange', 'Watermelon') are hashed using a built-in hash function, which maps each key to an index in the dictionary's underlying hash table. When a key-value pair is added to the dictionary, the hash function is used to determine the index where the value should be stored. When a value is looked up, the hash function is used again to find the index where the value was stored, allowing for fast retrieval.

2. Sets:

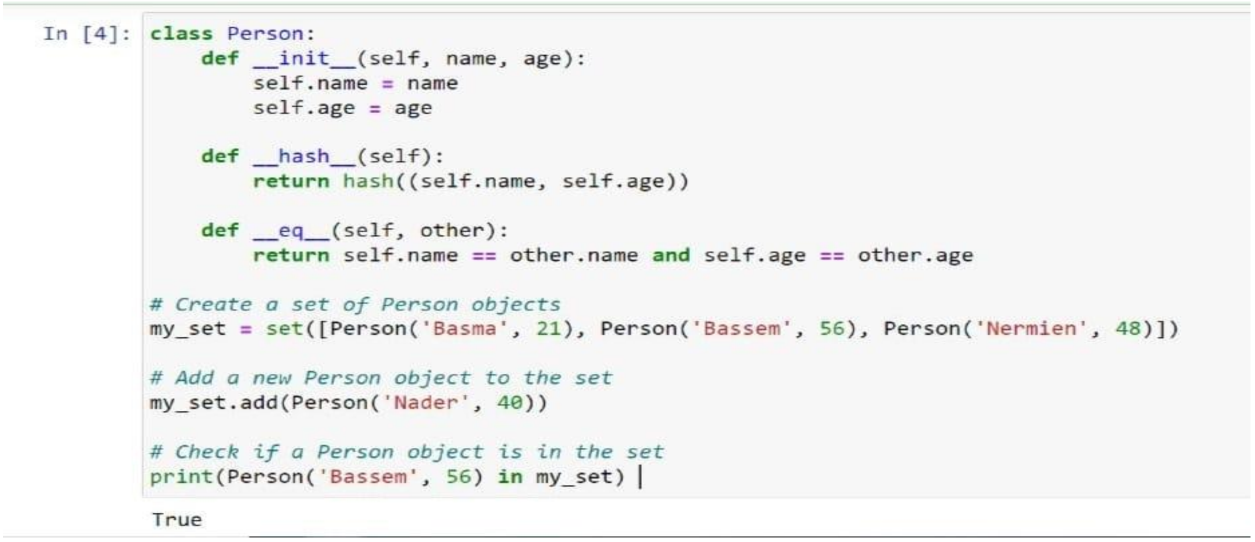


```
In [3]: # Create a set with hashable elements
        my_set = set([1, 2, 3])

        # Add a new element to the set
        my_set.add(4)
```

In this example, the elements (1, 2, 3, 4) are hashed using a built-in hash function, which maps each element to an index in the set's underlying hash table. When an element is added to the set, the hash function is used to determine the index where the element should be stored. When an element is looked up, the hash function is used again to find the index where the element was stored, allowing for fast membership testing.

3. Custom data structures:



```
In [4]: class Person:
            def __init__(self, name, age):
                self.name = name
                self.age = age

            def __hash__(self):
                return hash((self.name, self.age))

            def __eq__(self, other):
                return self.name == other.name and self.age == other.age

        # Create a set of Person objects
        my_set = set([Person('Basma', 21), Person('Bassem', 56), Person('Nermien', 48)])

        # Add a new Person object to the set
        my_set.add(Person('Nader', 40))

        # Check if a Person object is in the set
        print(Person('Bassem', 56) in my_set)

        True
```

In this example, the Person class defines a custom hash function that maps each object to a unique hash value based on its name and age attributes. This allows instances of the Person class to be stored and retrieved quickly and efficiently in a set.