Topic 11– Data Structures Part 2 (Dictionary)

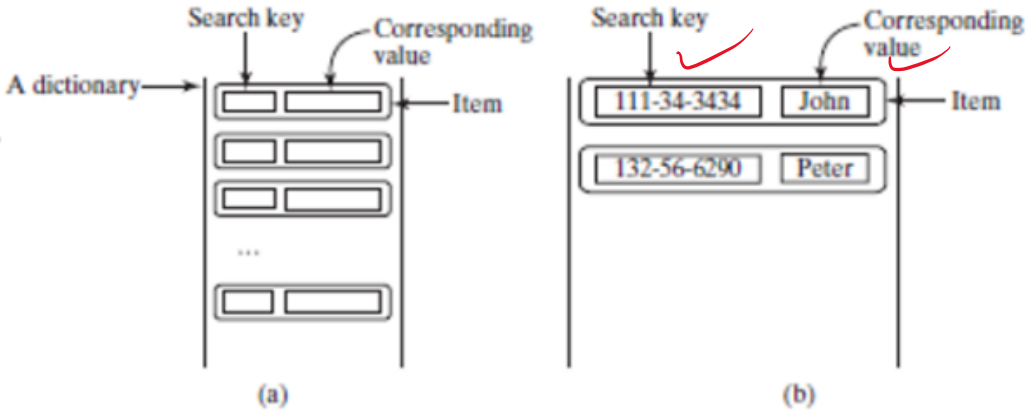
Python dictionary

A dictionary is a collection that stores the values along with the keys.

The keys are like an index operator.

In a dictionary, the key must be a hashable object.

A dictionary cannot contain duplicate keys.



Each key maps to one value.

A key and its corresponding value form an *item* (or *entry*) stored in a dictionary, as shown in below:

Creating a dictionary

- You can create a dictionary by enclosing the items inside a pair of curly braces ({}).

- Each item consists of a key, followed by a colon, followed by a value and items are separated by commas.

  Example:

```
students = {"111-34-3434":"John", "132-56-6290":"Peter"}
```

- The key in the first item is 111-34-3434, and its corresponding value is John.

- The key must be of a hashable type such as numbers and strings and the value can be of any type.

- To create an empty dictionary:

```
students = {} # Create an empty dictionary
```

Add |Modify |Retrieve VAlues

- To add an item to a dictionary, use the syntax: dictionaryName[key] = value

  Example:

  ```
  students["234-56-9010"] = "Susan“
  ```

- If the key is already in the dictionary, the preceding statement replaces the value for the key

- To retrieve a value, simply write an expression using dictionaryName[key].

- If the key is in the dictionary, the value for the key is returned, Otherwise, a KeyError exception is raised.

  Example:

  ```
  students = {"111-34-3434":"John", "132-56-6290":"Peter"}
  students["234-56-9010"] = "Susan" # Add a new item
  students["234-56-9010"] => Susan"
  students["111-34-3434"] = "John Smith"
  students["111-34-3434"] => "John Smith"
  students["343-45-5455"] => Key Error
  ```

Delete | Iterating Items

To delete an item from a dictionary, use the syntax:

del dictionaryName[key]

If the key is not in the dictionary, a KeyError exception is raised.

Traverse all keys in the dictionary using for loop to.

Example:

```
students = {"111-34-3434":"John", "132-56-6290":"Peter"}
for key in students:
    print(key + ":" + str(students[key])) # students[key] returns the value for the key
```

Output:

"111-34-3434":"John"

"132-56-6290":"Peter"

Using len | in |not |Equality (== and !=)

- You can find the *number of the items* in a dictionary by using len(dictionary)

  Example:

  ```
  students = {"111-34-3434":"John", "132-56-6290":"Peter"}
  len(students) => 2
  ```

- You can use the in or not in operator to determine whether a key is in the dictionary.

  Example:

  ```
  students = {"111-34-3434":"John", "132-56-6290":"Peter"}
  "111-34-3434" in students => True
  "999-34-3434" in students => False
  ```

- You can use the == and != operators to test whether two dictionaries contain the same items

  Example:

  ```
  d1 = {"red":41, "blue":3}
  d2 = {"blue":3, "red":41}
  d1 == d2 => True
  d1 != d2 => False
  ```

  In this example, d1 and d2 contain the same items regardless of the order of the items in a dictionary.

Dictionary Methods

Method	Return type	Explanation
keys()	tuple	Returns a sequence of keys.
values()	tuple	Returns a sequence of values.
items()	tuple	Returns a sequence of tuples. Each tuple is (key, value) for an item.
clear()	none	Deletes all entries.
get(key)	value	Returns the value for the key
pop(key)	value	Removes the item for the key and returns its value.
popitem()	tuple	Returns a randomly selected key/value pair as a tuple and removes the selected item.

- The **get(key)** method is similar to **dictionaryName[key]** except that the **get** method returns **None** if the key is not in the dictionary rather than raising an exception.

- The **pop(key)** method is the same as **del dictionaryName[key]**.

Other Operations on a Dictionary

- Finding the number of items in a Dictionary using len(dictionary)

```
>>> students = {"111-34-3434":"John", "132-56-6290":"Peter"}
>>> len(students)
2
>>>
```

- Testing the presence of a Key in a Dictionary using in or not in

```
>>> students = {"111-34-3434":"John", "132-56-6290":"Peter"}
>>> "111-34-3434" in students
True
>>> "999-34-3434" in students
False
```

- Equality Test using == and != operators to test if two dictionaries have the same items:

```
>>> d1 = {"red":41, "blue":3}
>>> d2 = {"blue":3, "red":41}
>>> d1 == d2
True
>>> d1 != d2
False
```

Items are matched regardless of the order they appear

You cannot use the comparison operators (>, >=, <=, and <) to compare dictionaries because the items are not ordered.

Assignment 2 – Discussion of Requirements

- Part A
- Part B
- Part C

Next Week – Data Structures –Part 3

- Linked Lists
- Stacks
- Queues
- Binary Trees