Structures: Dictionaries

CPSC 231: Introduction to Computer Science for Computer Science Majors I
Fall 2021

Jonathan Hudson, Ph.D.
Instructor
Department of Computer Science
University of Calgary

Wednesday, August 25 2021

UNIVERSITY OF CALGARY

Dictionary

- A mutable data structure that maps ***unique keys*** to ***values***
  - Dictionaries support many of the same operations as lists
  - They can contain mutable types, such as lists.
  - They are **unordered** (insertion order isn't naturally preserved)*
  - Keys are unique, values can be duplicates, using same key overwrites previous value

  - *newer versions of Python track an insertion order secondarily, but this is metadata tracked in addition to the data structure and not inherent to dictionaries
  - Therefore, we will maintain the definition that dictionaries (AKA hash tables) are unordered data structures by definition

Dictionary

- Think of a dictionary like a post office where your mailbox can hold one item
- Your address is a unique location (key) where mail (value) can be put
- In Python to store multiple letters we need to store a list in our mailbox (otherwise previous mail falls out)

Creating a Dictionary

- Keys must be unique
- Keys should be of an immutable type: int, bool, string, float (since floats are approximated, then it is unwise to use them).
  - They could of any type, but you must do some extra work to make it work.

- Hardcoding:

```
<dict name> = {key1:value,... , keyn:value}
```

Examples:

```
students = {123:'Alice',124:'Bob',125:'Charles'}
```

Adding Elements to Dictionary

- Keys must be unique, but not values. If a duplicate key is found, then its value is overwritten:

```
myDictionary = {}
myDictionary[123] = 'Alice'
myDictionary[124] = 'Bob'
myDictionary[125] = ‘Alice'
print(myDictionary)
myDictionary[123] = ‘Charles‘
print(myDictionary)
```

```
{123: 'Alice', 124: 'Bob', 125: ‘Alice'}
{123: 'Charles', 124: 'Bob', 125: ‘Alice'}
```

Dictionary Operations

Dictionary operations

Operation	Example	Description
Indexing	`students[123]`	Access an element by the key
Membership	`if 123 in students` `if 124 not in students`	Query whether or not an item is in the dictionary by the key
Length	`len(names)`	Get the number of items in a list
Add	`students[126] = 'Daniel'`	Add an item using the key
Delete	`del students[123]`	Delete an item busing the key
List	`list(students.keys())`	List keys in the dictionary
Sort	`sorted(students.keys()`	Sort the dictionary by the keys
Keys	`students.keys()`	Get all keys in the dictionary
Items	`students.items()`	Get all items in the dictionary
Clear	`students.clear()`	Clear the dictionary

Create and size

```
#Make empty dictionary
students = {}
print('students = %s' % students)

#Make an initally filled dictionary
students = {101:'Ken', 102:'Tony', 100:'Marc'}
print('students = %s' % students)

#Get size
print('size = %s' % len(students))
```

Add, update, remove, get and remove

```
#Add items to dictionary
students[103] = 'Maryam'
print('students = %s' % students)

#Update item in dictionary
students[101] = 'Jim'
print('students = %s' % students)

#Remove key and value
del students[102]
print('students = %s' % students)

#Pop key and return value
print(students.pop(103))
print('students = %s' % students)
```

Keys, values, items

```
#Print keys (unsorted)
print('keys = %s' % students.keys())

#Print values (unsorted)
print("values = %s" % students.values())

#Print item tuples (key, value) (unsorted)
print("items = %s" % students.items())
```


UNIVERSITY OF CALGARY

Get item, sort, empty

```
#Get an item by key
num = int(input('Please an item number to search: '))
if (num in students):
        print ("Found the item", students[num])
else:
        print ("Not found")

#Sort the keys
print('sorted = %s' % sorted(students.keys()))

#Empty dictionary
students.clear()
print('cleared dicts= %s' % students)
```


UNIVERSITY OF CALGARY

Usage

What are dictionaries for?

1. Sometimes having an order does not apply for certain data.

2. Sometimes we don't care if something is stored in ordered structure or not.

3. Sometimes an unordered structure makes things faster/easier.
   - If data is unsorted how long does it take to find something in list of length n?
   - If data is sorted how long does it take to find something?

What are dictionaries for?

- **In lists the only way to find things fast, was to keep a list sorted**.
  - Sorting has a cost of efficiency. (and even then there are delays)

- **Dictionaries** use a mathematical trick called **hashing** which applies mathematical function to key and changes it to a hidden index. The math works the same way every time. Essentially a **constant lookup time**.
  - There is a cost every time we store too many things and are hash table has to grow, but once we stop adding. Lookup is fast.

Onward to … strings.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~hudsonj/

UNIVERSITY OF CALGARY