Lecture 21: Introduction to Computer Programming Course - CS1010

DEPARTMENT OF COMPUTER SCIENCE | 04/11/2019

Rensselaer

Announcements

- Homework 10 Will be Posted this weekend i.e. on Saturday
- Will be due on April 22nd.
- Last Class is on April 25th .
- Will be a review class - for the final exam.
- Final exam is April 30th .

Goals for Today

- **Dictionaries (Part 1):**
  - What are dictionaries?
  - Syntax
  - When to use dictionaries?
  - Implementing dictionaries.
  - Problems

What are dictionaries?

- Useful data type built into Python is the *dictionary*

- We've been learning about *sequences* in Python but now we're going to switch gears and learn about *mappings* in Python.

- If you're familiar with other languages you can think of these Dictionaries as hash tables.

What are Dictionaries

- Dictionaries are unordered mappings for storing objects.
  - Lists store objects in an ordered format
  - Dictionaries use key-value pairing

- Key-value pair allows users to quickly access objects without knowing the index location

Lists Vs. Dictionaries

A Linear collection of things that stay in order

A bag of values each with its own label

Lists Vs. Dictionaries

- The main difference is that items in dictionaries are accessed via keys and not via their position.
- A dictionary is an associative array (also known as **hashes**).
- Any key of the dictionary is associated (or mapped) to a value.
- The values of a dictionary can be any Python data type.
- So dictionaries are unordered key-value-pairs.

Dictionaries are unique!

- Dictionaries don't support the sequence operation of the sequence data types like strings, tuples and lists.
- Dictionaries belong to the built-in mapping type.
- They are the sole representative of this kind!

Mappings

- Mappings are a collection of objects that are stored by a *key*, unlike a sequence that stored objects by their relative position.
  - This is an important distinction, since mappings won't retain order since they have objects defined by a key.

- A Python dictionary consists of a key and then an associated value. That value can be almost any Python object.

Syntax

- Dictionaries use curly braces and colons to signify the keys and associated values.

- For example:
  - **{‘key1’: ‘value1’, ‘key2’: ‘value2’,}**
  - **food = {"ham" : "yes", "egg" : "yes", "spam" : "no" }**

Other ways

- Use the dict() built-in function.
- The following creates equivalent dictionaries:
- dict(Bob=‘508-2232', John=‘159-0055')
- dict([('Bob', ‘508-2232'), ('John', ‘159-0055')])

Implementing Dictionaries

- Lists index their entries based on the position in the list
- Dictionaries are like bags - no order
- So we index the things we put in the dictionary with a “lookup tag”

When to use Dictionaries

- When to use dictionary and when to use lists?
- **Dictionary:**
  - Objects retrieved by key name.
  - Unordered and cannot be sorted.

- **Lists:**
  - Objects retrieved by location
  - Ordered sequence that can be indexed and sliced

How will we use Dictionaries

1.) Constructing a Dictionary

2.) Accessing objects from a dictionary

3.) Nesting Dictionaries

4.) Basic Dictionary Methods

5.) Looping with Dictionaries

Properties

- Its important to note that dictionaries are very flexible in the data types they can hold.
- We can use arbitrary types as values in a dictionary, but there is a restriction for the keys.
- Only immutable data types can be used as keys, i.e. no lists or dictionaries can be used:

- If you use a mutable data type as a key, you get an error message.

Operators on Dictionaries

Operator	Explanation
len(d)	returns the number of stored entries, i.e. the number of (key,value) pairs.
del d[k]	deletes the key k together with his value
k in d	True, if a key k exists in the dictionary d
k not in d	True, if a key k doesn't exist in the dictionary d

Things to know: More about operators

- If you try to access a key which doesn't exist, you will get an error message.
  - words = {"house" : "Haus", "cat":"Katze"}
  - words["car"]
- You can prevent this by using the "in" operator:
  - if "car" in words: print words["car"]
  - if "cat" in words: print words["cat"]

Affect Values of a Dictionary

- This is similar to re-assignment in other object types.
- Python has a built-in method of doing a self subtraction or addition (or multiplication or division).
- We could also use += or -=

Problem 1

- Write a Python script to add a key to a dictionary.
- Sample Dictionary : {0: 10, 1: 20}
  Expected Result : {0: 10, 1: 20, 2: 30}

Problem 2

- Write a Python program to check if a given key already exists in a dictionary.

Problem 3

- Write a Python program to remove a key from a dictionary.

In Class Exercise

- Given In Class

Next lecture

- Nesting with Dictionaries
- Dictionary Methods
- Merging Dictionaries
- Iterating over dictionaries