CSC 171 - Introduction to Computer Programming

Lecture #10 – Dictionaries

What Is A Dictionary?

- In data structure terms, a dictionary is better termed an associative array, associative list or a map.
- You can think if it as a list of pairs, where the first element of the pair, the key, is used to retrieve the second element, the value.
- Thus we map a key to a value

Key Value Pairs

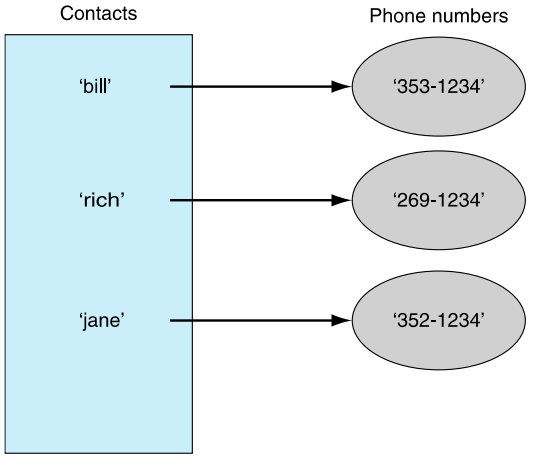
- The key acts as an index to find the associated value.
- Just like a dictionary, you look up a word by its spelling to find the associated definition
- A dictionary can be searched to locate the value associated with a key

Python Dictionary

- Use the { } marker to create a dictionary
- Use the : marker to indicate key: value pairs
- Example:

```
contacts = {'bill': '353-1234',\
   'rich': '269-1234', 'jane':'352-1234'}
print (contacts)
```

Phone Contact List: Names and Phone Numbers



Keys and Values

- Key must be immutable
  - strings, integers, tuples are fine
  - lists are NOT
- Value can be anything

Collections But Not a Sequence

- Dictionaries are collections but they are not sequences such as lists, strings or tuples
  - there is no order to the elements of a dictionary
  - in fact, the order (for example, when printed) might change as elements are added or deleted.
- So how to access dictionary elements?

Access dictionary elements

- Access requires `[ ]`, but the *key* is the index!

```
>>> my_dict = {}  an empty dictionary

>>> my_dict['bill'] = 25 #  added the pair 'bill':25

>>> print(my_dict['bill'])  # prints 25
25
```

Dictionaries Are Mutable

- Like lists, dictionaries are a mutable data structure
  - you can change the object via various operations, such as index assignment

```
>>> my_dict = {'bill':3, 'rich':10}
>>> print(my_dict['bill'])
3
>>> my_dict['bill'] = 100
>>> print(my_dict['bill'])
100
>>>
```

Again, Common Operators

- Like others, dictionaries respond to these
  `len(my_dict)`
- number of key: value **pairs** in the dictionary
  `element in my_dict`
- boolean, is `element` a **key** in the dictionary
  `for key in my_dict:`
- iterates through the **keys** of a dictionary

Fewer Methods

- Only 9 methods in total. Here are some:
  - **`key in my_dict -`** does the key exist in the dictionary?
  - **`my_dict.clear`** – empty the dictionary
  - **`my_dict.update(yourdict)`** **-** – for each key in **`yourDict`**, updates **`my_dict`** with that key-value pair
  - **`my_dict.copy`** – shallow copy
  - **`my_dict.pop(key)`** - remove **`key`**, return value

Dictionary Content Methods

- **`my_dict.items()`** – all the key/value pairs
- **`my_dict.keys()`** **-** all the keys
- **`my_dict.values()`** **-** all the values
- There return what is called a ***dictionary view***.
- The order of the views corresponds are dynamically updated with changes are iterable

Views are Iterable

```
for key in my_dict:
     print(key)
```

prints all the keys

```
for key, value in my_dict.items():
     print(key, values)
```

prints all key/value pairs

```
for value in my_dict.values():
     print(values)
```

prints all the values

Operators

```
>>> my_dict = {'a':2, 3:['x', 'y'], 'joe': 'smith'}
>>> dict_value_view = my_dict.values()
>>> dict_value_view
dict_values([2, ['x', 'y'], 'smith'])
>>> type(dict_value_view)
<class 'dict_values'>
>>> for val in dict_value_view:
     print(val)

2
['x', 'y']
smith
```

```
>>> my_dict['new_key'] = 'new_value'
>>> dict_value_view
dict_values([2, ['x', 'y'], 'smith', 'new_value'])
>>> dict_key_view = my_dict.keys()
>>> dict_key_view
dict_keys(['a', 3, 'joe', 'new_key'])
>>> dict_value_view
dict_values([2, ['x', 'y'], 'smith', 'new_value'])
>>>
```

Membership Test

```
word_list = ["apple", "pear", "fruit", "pear", \
       "cantelope", "melon", "melon", "banana", \
       "orange", "banana", "orange", "grape", \
       "mango", "tangerine", "watermelon"]

count_dict = {}

for word in word_list:
    if word in count_dict:
        count_dict[word] += 1
    else:
        count_dict[word] = 1

print(count_dict)
```

```
count_dict["pineapple"] =
count_dict.get("pineapple", 12)
print(count_dict)
```

`get()` Method

- **`get()`** method returns the value associated with a dict key or a default value provided as second argument. Below, the default is 0

```
count_dict = {}
for word in word_list:
    count_dict[word] = count_dict.get(word,0) + 1
```

Example – Counting Word Occurrences in the Gettysburg Address

- We can use dictionary for many tasks, but let's start with a basic idea: using it to count how many occurrences there are of various words in Abraham Lincoln's Gettyburg Address

Text of the Gettysburg Address

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

- Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we can not dedicate—we can not consecrate—we can not hallow—this ground. The brave men, living and dead, who struggled here, have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember what we say here, but it can never forget what they did here. It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced.

It is rather for us to be here dedicated to the great task remaining before us—that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion—that we here highly resolve that these dead shall not have died in vain—that this nation, under God, shall have a new birth of freedom—and that government of the people, by the people, for the people, shall not perish from the earth.

Initial Algorithm

1. Initialize our dictionary (**`word_count_dict`**) to empty
2. Split the speech into a list of words (**`speech_list`**)
3. For every word in speech_list:
   a. If the word is in the dictionary:
      Add 1 to its value
   b. Else
      Insert it in the dictionary with a value of 1.

Counting Words in a String

```
speech = "to be or not to be"
speech_list = speech.split()

word_count_dict = {}
for word in speech_list:
    if word in word_count_dict:
        word_count_dict[word] += 1
    else:
        word_count_dict[word] = 1
print(word_count_dict)
```

Output

```
{'to': 2, 'be': 2, 'or': 1, 'not': 1}
```

The Functions In Our Program

- **`add_word(word, word_count_dict)`** – *Either updates the number of occurences or adds the word to the dictionary*
- **`process_line(line, word_count_dict)`** – *Processes the line so there is a sequence of lower-case words to add to the dictionary*
- **`pretty_print(word_count_dict)`** – *Print words from the most frequent to the least frequent*
- **`main()`** – *Calls all these functions*

add_word()

```
def add_word(word, word_count_dict):
    # Update the frequency count
    #  Add the word if it isn't in the dictionary
    if word in word_count_dict:
        word_count_dict[word] += 1
    else:
        word_count_dict[word] = 1
```

process_line()

```
import string

def process_line(line, word_count_dict):
    # Process the line so there is a list of
    # lower-case words
    line = line.strip()
    word_list = line.split()
```

```
    for word in word_list:
        # Ignore the "--" in the file
        if word != '--':
            word = word.lower()
            word = word.strip()

            # Get out punctuation
            word = word.strip(string.punctuation)
            print(word)
            add_word(word, word_count_dict)
```

pretty_print()

```
def pretty_print(word_count_dict):
    # Print nicely from highest to lowest
    # frequency
    value_key_list = []
    for key, val in word_count_dict.items():
        value_key_list.append((val, key))

    # Sort method sorts on list's first element
    # frequency
    # reverse means highest are first
    value_key_list.sort(reverse = True)
```

```
    print('{:11s}{:11s}'.format('Word', 'Count'))
    #print('%11s%11s' %('Word', 'Count'))
    print('_'*21)
    for val, key in value_key_list:
        print('{:12s} {:3d}'.format(key, val))
```

main()

```
def main():
    word_count_dict = {}
    gba_file = open('gettysburg.txt', 'r')
    for line in gba_file:
        process_line(line, word_count_dict)
    print('Length of the dictionary:', \
          len(word_count_dict))
    pretty_print(word_count_dict)

main()
```

Comma Separated Values (CSV)

- CSV files are a text format that are used by many applications (especially spreadsheets) to exchange data as text.
- Row oriented representation where each line is a row, and elements of the row (columns) are separated by a comma.
- Despite the simplicity, there are variations and we'd like Python to help.

CSV Module

- **`csv.reader`** takes an opened file object as an argument and reads one line at a time from that file.
- Each line is formatted as a list with the elements (the columns, the comma separated elements) found in the file.

Encodings Other Than UTF-8

- This example uses a csv file encoded with characters other than UTF-8 (our default)
  - In particular, the symbol ± occurs
- Can solve by opening the file with the correct encoding, in this case **`windows-1252`**

Example: Periodic Table

```
import csv
periodic_file = open("Periodic-Table.csv", "r",\
encoding="windows-1252")
reader = csv.reader(periodic_file)
for row in reader:
    print(row)
```

Sample Output from the Periodic Table

```
['1', 'H', '1', 'I A', '1', 'hydrogen', …]
['2', 'He', '18', 'VIII A', '1', 'helium', …]
['3', 'Li', '1', 'I A', '2', 'lithium', …]
['4', 'Be', '2', 'II A', '2', 'beryllium',…]
['5', 'B', '13', 'III A', '2', 'boron', …]
['6', 'C', '14', 'IV A', '2', 'carbon', …]
['7', 'N', '15', 'V A', '2', 'nitrogen', …]
['8', 'O', '16', 'VI A', '2', 'oxygen', …]
['9', 'F', '17', 'VII A', '2', 'fluorine',…]
['10', 'Ne', '18', 'VIII A', '2', 'neon', …]
```

Parsing the Periodic Table

- Our program will read in basic information about the elements from a periodic table file.
- It will need the following functions
  - **`read_table()`** – reads the file into a dictionary
  - **`parse_element()`** – parses the element into a symbol and quantity

read_table()

```
import csv

def read_table(a_file, a_dict):
    # Read Periodic Table file into a dictionary
    # with element symbol as key.
    data_reader = csv.reader(a_file)

    for row in data_reader:
        # Ignore header rows:
        # elements begin with a number
        if row[0].isdigit:
            symbol_str = row[1]

            #ignore end of row
            a_dict[symbol_str] = row[:8]
```

parse_element()

```
def parse_element(element_str):
    # Parse element string ito symbol and
    # quantity eg, Si2 returns ('si", 2)
    symbol_str = ""
    quantity_str = ""
    for ch in element_str:
        if ch.isalpha():
            symbol_str = symbol_str + ch
        else:
            quantity_str = quantity_str + ch

    # If no number, the default is 1
    if quantity_str == "":
        quantity_str = "1"
    return symbol_str, int(quantity_str)
```

pertable.py

```
# 1. Read file
periodic_file = open("Periodic-Table.csv", "r", \
encoding="windows-1252")

# 2. Create dictionary of Periodic Table using
#    element symbols as keys
periodic_dict = {}
read_table(periodic_file, periodic_dict)

# 3. Prompt for input and convert compound into a
#    list of elements
compound_str = input("Input a chemical compound,\
hyphenated, eg, C-O2:")
compound_list= compound_str.split("-")




# 4. Initialize atomic mass
mass_float = 0.0
print("The compound is composed of: ", end=" ")

# 5. Parse compound list into symbol-quantity
pairs,
#    print name, and add mass
for c in compound_list:
    symbol_str, quantity_int = parse_element(c)
    print(periodic_dict[symbol_str][5], end=' ')

    #  Add atomic mass
    mass_float = mass_float + quantity_int*\
            float(periodic_dict[symbol_str][6])
```

```
print("\n\nThe atomic mass of the compound is",\
 mass_float)

periodic_file.close()
```