Data Collections - part 2

- Using Python to Analyze Larger Data Sets

CMSC 104 Section 2
November 18, 2025

Administrative Notes

Classwork 8 is today

- Due next Monday, November 24

Python is useful in analyzing large collections of data

Because the built-in list and dictionary data types can be the foundation of many solutions

Today we will write a program that can analyze any textfile and identify the most commonly used words

- Based on lists and dictionaries

List methods (from last lecture)

Lists - Pre-defined list methods

Method	Meaning	Result
append	Add one element to the end of the list	
sort	Puts the list elements in order, smallest to largest	
reverse	Reverses the elements of the list	
index	Returns the index of the first occurrence of the indicated value	
insert	Inserts the item into the list at the designated index	
count	Returns the number of occurrences of the value in the list	
remove	Deletes the first occurrence of the item in the list	
pop	Deletes the list element at the indicated index	

And now for dictionaries

Built-in dictionary methods

Method	Meaning
key in dict	Is the key in the dictionary.keys()
dict.keys()	Returns a sequence of the keys in the dictionary
dict.values()	Returns a sequence of the values in the dictionary
dict.items()	Returns a sequence of tuples - key,value pairs
dict.get(key, default)	If key is in the dictionary, returns the associated value. Otherwise, returns the defaults
Del dict[key]	Deletes the specified entry
dict.clear()	Deletes all entries, leaving an empty dictionary
for var in dict	Loops over all the keys in the dictionary

And now for the program

Create a text file from the article or book

Now read in the file

Read it as a string

Get rid of *almost* all punctuation

```
def getwordlist(filename):

    #first read in the file contents as a single string
    with open(filename, "r") as f:
        data = f.read()

        # we have the words as a single string. We are
going to remove all punctuation, because
        # that could mess up our word counts

        #print(data[1:100])

        for ch in '!"#$%&()*+/,.:;<=>?@[]{}||_-|':
            data = data.replace(ch, '')
        #print(data[1:100])
        list_of_words = data.split()
        return list_of_words
```

Create a dictionary of words & their frequencies

```
def create_dictionary(list_of_words):

   dictionary = {}

   for word in list_of_words:

       dictionary[word] = dictionary.get(word, 0) + 1

   return dictionary
```

Copy the dictionary into a list, and sort it

```
def organize (d):

   items = list(d.items())

   items.sort()

   items.sort(key=byFrequency, reverse=True)

   return items
```

List.sort() puts the items smallest-to-largest. Reverse makes it largest-to-smallest

Remember what lists.items() does

A function we'll need…

```
def byFrequency(list_of_words):

   return list_of_words[1]
```

Now put it all together

```
if __name__ == '__main__':
    fname = input('Enter file name: ')
    contents = getwordlist(fname)
    #contents now contains the list of words in the file
    #create a dictionary that contains a list of the words in the dictionary and the
    #number of times each word has appeared
    word_count = create_dictionary(contents)
    #now create a list of the word, count pairs in the dictionary, and put them in
order
    #from most common to least common
    common_list = organize(word_count)

    #print 10 items
    for i in range(10):
        print(common_list[i])
```