More aggregate data types: Dictionaries

- another useful data type in Python
- unordered set of *key:value* pairs

```
In [6]: x = {} # here is an empty dict

In [7]: type(x)
Out[7]: dict

In [8]: d = { 'Sinan' : 705, 'Brian' : 867 }

In [9]: d['Sinan']
Out[9]: 705
```

- sequences: indexed by numbers

  dictionary: indexed by keys

- keys must be (i) of immutable type and (ii) unique!!

Aside: known aliases to dictionaries

- Dictionary: Python, Smalltalk, Objective-C, .NET.
- **Hash** or **hash table**: C, C++, Perl, Ruby, Visual Basic, Common Lisp.
- **Collection**: Visual Basic
- **Map**: C++, Java.
- **Associative array**: Javascript, AWK

Why use dictionaries?

- dictionary lends itself naturally to a mapping situation

example: a pair of dice roll odds

```
16 d = {2:     29,
17      3:     66,
18      4:     71,
19      5:     100,
20      6:     137,
21      7:     173,
22      8:     132,
23      9:     122,
24      10:    78,
25      11:    59,
26      12:    33
27      }
28
29 rollsum = dice_roll() + dice_roll()
30
31 d[rollsum] = d[rollsum] + 1
```

Dice outcome		Counter
2	→	counters[0]
3	→	counters[1]
4	→	counters[2]
...	...	...
11	→	counters[9]
12	→	counters[10]

Why use dictionaries?

- performance wise dictionaries are absolutely superior to lists in look up situations where there is no simple way to map key to values
  - dictionary
  - flight number + day -> arrival time
  - name -> phone number
- in a look up scenario, computational time
  - increases linearly as a function of problem size (N) in **lists**
  - nearly independent of problem size in **dict**s **!!!!**

Working with dictionary type

```
len(d)        # returns number of items in d
d.keys()      # returns a view of keys in d
d.values()    # returns a view of values in d
k in d        # returns True if k in d
d[k]          # returns value associated with key k
d[k] = v      # associates value v with key k
del d[k]      # removes entry with key k from d
d.pop(k)      # remove entry with key k but get value
for k in d: # iterate over keys in d
```

Example: Word frequencies

Given an input string, display
i) each word and
ii) how many times it occurs in the string

```
# WordFrequencies.py
# Get the string:
test_string = '''The programmer, who needs clarity, who must talk all day to
    a machine that demands declarations, hunkers down into a low-grade annoyance.
    It is here that the stereotype of the programmer,
    sitting in a dim room, growling from behind Coke cans, has its origins.
    The disorder of the desk, the floor; the yellow Post-It notes everywhere;
    the whiteboards covered with scrawl:
    all this is the outward manifestation of the messiness of human thought.
    The messiness cannot go into the program;
    it piles up around the programmer.
    ~ Ellen Ullman
'''
```

Example: Word frequencies

Given an input string, display each word and how many times it occurs in the string

Pseudocode

```
get the string
break the string into a list of words
initialize a dictionary of counters
loop through the list of words
    if the word is in the dictionary
        increment its counter by 1
    otherwise
        set its counter to 1
loop through the entries in the dictionary
    for each entry display the key (the word) and the value
```

Example: Word frequencies

```
# WordFrequencies.py
# Get the string:
test_string = '''The programmer, who needs clarity, who must talk all day to
    a machine that demands declarations, hunkers down into a low-grade annoyance.
    It is here that the stereotype of the programmer,
    sitting in a dim room, growling from behind Coke cans, has its origins.
    The disorder of the desk, the floor; the yellow Post-It notes everywhere;
    the whiteboards covered with scrawl:
    all this is the outward manifestation of the messiness of human thought.
    The messiness cannot go into the program;
    it piles up around the programmer.
    ~ Ellen Ullman
'''

# Break the string into a list of words:
words = test_string.split()
# Initialize a dictionary of counters:
word_counts = {}

# Loop through the list of words:
for word in words:
    # If the word is in the dictionary:
    if word in word_counts:
        # Increment its counter by 1:
        word_counts[word] = word_counts[word] + 1
    # Otherwise:
    else:
        # Set its counter to 1:
        word_counts[word] = 1

# Loop through the entries in the dictionary:
for word in word_counts:
    # Display the key value pair:
    print(word, ':', word_counts[word])
```

[demo]

Example: Word frequencies improvements

problems:

```
The : 3
floor; : 1
annoyance. : 1
the : 10
~ : 1
```

What string methods can we use to fix these?

Example: Word frequencies improvements

problems:

```
The : 3
floor; : 1
annoyance. : 1
the : 10
~ : 1
```

```
37 for word in words:
38     w = word.lower()
39     w = w.strip('.,;:\'"?!()') # Notice the \ to es
40     if w.isalpha():
41         if w in word_counts:
42             word_counts[w] = word_counts[w] + 1
43         else:
44             word_counts[w] = 1
45
```

Example: Scrabble scoring

Write a function taking a word as input, and returning the total Scrabble value.

Use a dictionary to map letters to their scrabble value

```
A=1 B=3 C=3 D=2  E=1 F=4 G=2 H=4 I=1 J=8 K=5 L=1 M=3
N=1 O=1 P=3 Q=10 R=1 S=1 T=1 U=1 V=4 W=4 X=8 Y=4 Z=1
```

Pseudocode ?

Example: Scrabble scoring

Write a function taking a word as input, and returning the total Scrabble value.

Use a dictionary to map letters to their scrabble value

```
A=1 B=3 C=3 D=2  E=1 F=4 G=2 H=4 I=1 J=8 K=5 L=1 M=3
N=1 O=1 P=3 Q=10 R=1 S=1 T=1 U=1 V=4 W=4 X=8 Y=4 Z=1
```

Pseudocode

```
def ScrabbleValue( s ):
    Initialize the total value to 0
    Loop through the word a letter at a time
        Look up the letter's value
        Increment the total value by this letter's value
    Return the total value
```

Example: Scrabble scoring

```
1 # ScrabbleScoring.py
2 LETTER_VALUES = {'A':1, 'B':3, 'C':3, 'D':2, 'E':1, 'F':4, 'G':2,
3                  'H':4, 'I':1, 'J':8, 'K':5, 'L':1, 'M':3, 'N':1,
4                  'O':1, 'P':3, 'Q':10, 'R':1, 'S':1, 'T':1, 'U':1,
5                  'V':4, 'W':4, 'X':8, 'Y':4, 'Z':1}
6
7 def scrabble_value(s):
8     total_value = 0
9     for letter in s:
10        total_value = total_value + LETTER_VALUES[letter]
11    return total_value
12
13 if __name__ == '__main__':
14    print('The value of the word HERE is', scrabble_value('HERE'))
```

Example: Book database

- This is an example where values are non-numerical

- A small database of book information stored in a dictionary

- Key: book title, value: authors

- There can be more than one author, so we're using lists

- Write a function listing titles of the books by a given author!

```
books = { "The C Programming Language": ['Brian W. Kernighan',
                                         'Dennis M. Ritchie'],
          "Harry Potter and the Philosopher's Stone" : ['J. K. Rowling'],
          "The AWK programming language" : ['Alfred V. Aho',
                                            'Brian W. Kernighan',
                                            'Peter J. Weinberger'],
          "The practice of programming": ['Brian W. Kernighan', 'Rob Pike']
          "The cat in the hat": ['Dr. Seuss'],
          "The UNIX Programming Environment": ['Brian W. Kernighan',
                                               'Rob Pike'],
        }
```

Example: Book database

- Write a function listing titles of the books by a given author!

Pseudocode

```
def search_by_author( database, author ):
    Initialize book_list (the list of books by this author)
    For each key-value pair in the database
        If the value contains author
            Add the key to the book_list
    Return book_list
```

```
books = { "The C Programming Language": ['Brian W. Kernighan',
                                         'Dennis M. Ritchie'],
          "Harry Potter and the Philosopher's Stone" : ['J. K. Rowling'],
          "The AWK programming language" : ['Alfred V. Aho',
                                            'Brian W. Kernighan',
                                            'Peter J. Weinberger'],
          "The practice of programming": ['Brian W. Kernighan', 'Rob Pike']
          "The cat in the hat": ['Dr. Seuss'],
          "The UNIX Programming Environment": ['Brian W. Kernighan',
                                               'Rob Pike'],
        }
```

Example: Book database

- Write a function listing titles of the books by a given author!

```
# BooksDict.py
def search_by_author(database, author):
    book_list = []
    for key in database:
        if author in database[key]:
            book_list.append( key )
    return book_list

if __name__ == '__main__':
    books = { "Harry Potter and the Philosopher's Stone" : ['J. K. Rowling'],
              "The cat in the hat": ['Dr. Seuss'],
              "The C Programming Language": ['Brian W. Kernighan',
                                             'Dennis M. Ritchie'],
              "The UNIX Programming Environment": ['Brian W. Kernighan',
                                                   'Rob Pike'],
              "The AWK programming language" : ['Alfred V. Aho',
                                                'Brian W. Kernighan',
                                                'Peter J. Weinberger'],
              "The practice of programming": ['Brian W. Kernighan', 'Rob Pike']
            }
    print('J. K. Rowling wrote:', search_by_author(books, 'J. K. Rowling'))
    print('Brian W. Kernighan wrote:',search_by_author(books, 'Brian W. Kernighan'))
    print('Sinan Bulut wrote', search_by_author(books, 'Tim Topper'))
```

Example: List of Dictionaries

- Just as other types can occur in a dictionary, dictionaries can occur in other types
- the fact that we can combine different types is a powerful programming concept

```
data = [ {'id':4721, 'sex':'F', 'age':31},
         {'id':1828, 'sex':'M', 'age':56},
         {'id':7816, 'sex':'M', 'age':72},
         #. . . lots more records . . .
         {'id':3286, 'sex':'M', 'age':29},
         {'id':5063, 'sex':'F', 'age':22}
       ]
```

Problem 1: How can display the entry of the oldest individual?

Example: List of Dictionaries problem 1

Problem 1: How can display the entry of the oldest individual?

Pseudocode

```
We have to start somewhere so let's begin with the first entry
and set it to be the oldest record (after all it's the oldest
we have seen so far!)
Consider each item in the database from the second to the end
    If the age of this entry is older than our current oldest
        Update our oldest record
Display the oldest record
```

```
data = [ {'id':4721, 'sex':'F', 'age':31},
         {'id':1828, 'sex':'M', 'age':56},
         {'id':7816, 'sex':'M', 'age':72},
         #. . . lots more records . . .
         {'id':3286, 'sex':'M', 'age':29},
         {'id':5063, 'sex':'F', 'age':22}
       ]
```

Example: List of Dictionaries problem 1

Problem 1: How can display the entry of the oldest individual?

```
1 # ListOfDicts.py
2 data = [ {'id':4721, 'sex':'F', 'age':31},
3          {'id':1828, 'sex':'M', 'age':56},
4          {'id':7816, 'sex':'M', 'age':72},
5          {'id':3286, 'sex':'M', 'age':29},
6          {'id':5063, 'sex':'F', 'age':22}
7        ]
8
9 oldest = data[0]
10 for entry in data[1:]:
11     if entry['age'] > oldest['age']:
12         oldest = entry
13 print('The oldest person is:', oldest)
```

Example: List of Dictionaries problem 1

Problem 2: output number of male and female records

```
data = [ {'id':4721, 'sex':'F', 'age':31},
         {'id':1828, 'sex':'M', 'age':56},
         {'id':7816, 'sex':'M', 'age':72},
         #. . . lots more records . . .
         {'id':3286, 'sex':'M', 'age':29},
         {'id':5063, 'sex':'F', 'age':22}
       ]
```

Example: List of Dictionaries problem 2

Problem 2: output number of male and female records

```
data = [ {'id':4721, 'sex':'F', 'age':31},
         {'id':1828, 'sex':'M', 'age':56},
         {'id':7816, 'sex':'M', 'age':72},
         #. . . lots more records . . .
         {'id':3286, 'sex':'M', 'age':29},
         {'id':5063, 'sex':'F', 'age':22}
       ]
```

Example: List of Dictionaries problem 2

Problem 2: output number of male and female records

Pseudocode

```
Set counter of males to 0
Set counter of females to 0
Consider each item in the database
    If the value for the key 'sex' is 'M'
        Increment the counter of males
    Elif the value of the key 'sex' is 'F'
        Increment the counter of females
Display the male and female counters
```

Example: List of Dictionaries problem 2

Problem 2: output number of male and female records

```
18 nmales = 0
19 nfemales = 0
20 for entry in data:
21     if entry['sex'] == 'M':
22         nmales = nmales + 1
23     elif entry['sex'] == 'F':
24         nfemales = nfemales + 1
25 print('There are', nmales, 'males and', nfemales, 'females.')
```

Aside: it is a common practice to name counter variables with ‘n’ prefix

like **nfemales**, **nmales**

as a short from of phrase “**n**umber of **males**” etc.