ECE12: Introduction to Programming
Lecture 10

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 10: Overview

- Sequences
  - Operations on sequences
  - Interactive examples
  - List example
    - **`histogram.py`**
  - List methods
- Dictionaries
  - Dictionary operations
  - Dictionary methods
  - Dictionary example
    - **`dictionary.py`**

Sequences

- Types of sequences
  - String `s = “This is a string.”`
  - List `l = [6, 3, 2, 4, 5]`
  - Tuple `t = (3, 2, 0)`
    - Lists are mutable, strings and tuples are immutable!
- Operations on sequences
  - Length
    - `len(s) = 17` `len(l) = 5` `len(t) = 3`
  - Element access (by position)
    - from the front
      - `s[0] = “T”` `l[1] = 3` `t[2] = 0`
    - from the end
      - `s[-1] = “.”` `l[-2] = 4` `t[-3] = 3`
  - Concatenation and extension
    - **+**, **+=** operators
      - `s + “XYZ” = “This is a string.XYZ”`

Sequences

- Operations on sequences (continued)
  - Iteration over sequence
    - ```
      sequence = [23, 45, 67]
      for item in sequence:
          print item
      ```
    - Remember: **`range()`** returns a sequence of integers!
  - Sequence packing and unpacking
    - ```
      vector = (42, 7, 99)
      x,y,z = vector
      ```
    - ```
      a = 10 ; b = 20
      a,b = b,a
      ```
  - Slicing
    - **`[start:end]`** operator
      - **`s = “This is a test string.”`**
      - **`s[0:4] = “This”`**
      - **`s[-7:-1] = “string”`**
      - **`s[:4] = “This”`**
      - **`s[-12:] = “test string.”`**
      ```

List Example

- Program **`histogram.py`**:

```
# histogram.py: print a histogram for a list of numbers
#
# author: Rainer Doemer
#
# modifications:
# 02/04/04 RD    initial version (similar to fig05_05.py)

# initialize
values = []

# input
while 1:
    s = raw_input("Enter a number or type 'q' to quit: ")
    if s == 'q':
        break;
    i = int(s)
    values += [i]

# compute and output
print "Histogram for %d values:" % len(values)
for v in values:
    print "%3d" % v, "*" * v
```

List Methods

- Additional operations on lists are available as methods
  - **`append(item)`** inserts **`item`** at the end of the list
  - **`count(elem)`** returns the number of **`elem`** in the list
  - **`extend(list)`** concatenates **`list`** to the list
  - **`index(elem)`** returns the position of the first **`elem`**
  - **`insert(index,item)`** inserts **`item`** at position **`index`**
  - **`pop()`** removes and returns last element
  - **`pop(index)`** removes and returns element at **`index`**
  - **`remove(elem)`** removes the first **`elem`** from the list
  - **`reverse()`** reverses the list contents (in place)
  - **`sort()`** sorts the list contents (in place)
- Example: **`s=[3,2,6]; s.append(4); s.sort()`**

Dictionaries

- Dictionary
  - built-in data type in Python
  - aka. *hash table* or *associative array*
  - (unordered) set of key-value pairs
    - Key: immutable data type (such as integer, string, tuple)
    - Value: any data type (basic or composite)
- Dictionary operations
  - Example: Grade book
    - **`gb = { “John”:66, “Jane”:77, “Joe”:87 }`**
    - **`print gb[“Jane”]`** (read access operation)
    - **`gb[“Jane”] = 75`** (write access operation)
    - **`gb[“Jack”] = 79`** (insertion operation)
    - **`del gb[“John”]`** (deletion operation)

Dictionary Methods

- Additional operations on dictionaries are available as methods
  - **`clear()`** deletes all items in the dictionary
  - **`copy()`** creates a (shallow) copy of the dictionary
  - **`get(key)`** returns the value associated with **`key`**
  - **`has_key(key)`** returns **`1`** if **`key`** is in the dictionary, otherwise **`0`**
  - **`keys()`** returns a list of the keys in the dictionary
  - **`values()`** returns a list of the values in the dictionary
  - **`items()`** returns a list of tuples of key-value pairs
  - **`update(dict)`** adds all key-value pairs of **`dict`** to the dictionary (overwriting same entries)
  - etc.

Dictionary Example

- Program **`dictionary.py`**:

```
# dictionary.py: simple English-German dictionary
# author: Rainer Doemer
#
# 02/05/04 RD  initial version

# initialize
dict = {"one":"eins", "two":"zwei", "three":"drei",
        "four":"vier", "five":"fuenf", "six":"sechs",
        "seven":"sieben", "eight":"acht", "nine":"neun"}

# input, compute, output
while 1:
    s = raw_input("Enter an English word or type 'q' to quit: ")
    if s == 'q':
        break;
    if (dict.has_key(s)):
        print "'%s' in English is '%s' in German." % (s,dict[s])
    else:
        print "No data for '%s'!" % s
```