ECE12: Introduction to Programming
Lecture 12

Rainer Dömer

doemer@uci.edu

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

Lecture 12: Overview

- Functional Programming
  - Introduction
  - Function objects
    - **`apply()`**
    - Example Bubble Sort
  - Functional list operations
    - **`map()`**
    - **`filter()`**
    - **`reduce()`**
    - Examples

Functional Programming

- What is Functional Programming?
  - "Function-oriented" programming
    - thinking of functions as objects
  - Using functions as objects
    - passing function objects around
    - calling function objects
    - anonymous functions
      - **`lambda`** statement (we skip this!)
  - Recursion
    - powerful programming concept
    - divide-and-conquer paradigm

Function Objects

- Interactive Example:
  - a function is an object
    - with a type
    - with a location
    - can be called
  - a function can be assigned to another identifier
  - built-in function **`apply(fct,args)`**
    - calls the function **`fct`**
    - with arguments **`args`**
    - returns return result of **`fct(args)`**

```
% python
>>> def f(a,b):
...     return 3*a + 5*b
...
>>> f(9, 3)
42
>>> type(f)
<type 'function'>
>>> print f
<function f at 0x811d704>

>>> g = f
>>> type(g)
<type 'function'>
>>> g(9, 3)
42

>>> apply(g, (9,3))
42
>>> args = (9,3)
>>> apply(g, args)
42
```

Function Objects

- Passing functions as arguments
- Example: Bubble Sort

Note:

Either function **`CmpGreater`** or **`CmpSmaller`** can be used in function **`BubbleSort`** for the comparison!

```
# bubblesort.py: Bubble Sort algorithm
#
# author: Rainer Doemer
#
# modifications:
# 02/09/04 RD    initial version

# function definitions

def CmpGreater(item1, item2):
    return item1 > item2

def CmpSmaller(item1, item2):
    return item1 < item2

def BubbleSort(list, cmp_fct=CmpGreater):
    for i in range(len(list)):
        for j in range(i+1, len(list)):
            if cmp_fct(list[i], list[j]):
                list[i],list[j] = list[j],list[i]

# initialize
items = []
...
```

Function Objects

- Example Bubble Sort, continued...

```
...
# input the sort order and a list of strings
while 1:
    s = raw_input("Sort order: (a) ascending or (b) descending? ")
    if (s == 'a'):
         comparison = CmpGreater
         break
    elif (s == 'b'):
         comparison = CmpSmaller
         break
while 1:
    s = raw_input("Enter a string or type '.' to quit: ")
    if s == '.':
         break;
    items += [s]

# compute
print "Sorting...",
BubbleSort(items, comparison)
print "Done."

# output the sorted list
print "The sorted list is:"
for item in items:
    print item
```

Functional List Operations

- Functional programming is very useful for processing lists of data
- Python provides three built-in functions
  - **`map(fct, seq)`**
  - **`filter(fct, seq)`**
  - **`reduce(fct, seq)`**
- Each function
  - takes a function **`fct`** as first argument
  - takes a sequence **`seq`** as second argument
  - applies **`fct`** to the elements of **`seq`**
  - returns the result of these operations
    - as a new list (**`map()`**, **`filter()`**)
    - as a single value (**`reduce()`**)

Functional List Operations

- **`map(fct, seq)`**
  - applies **`fct`** to each element of **`seq`**
  - returns a list of the results of these function calls
- Example:

```
% python
>>> def cube(x):
...     return x*x*x
>>> map(cube, [1,2,3,4,5,6])
[1, 8, 27, 64, 125, 216]
```

- Multiple sequences:
  - **`map(fct, seq1, seq2, ...)`**
  - **`fct`** is called with items from each sequence
  - **`fct(item1, item2, ...)`**
- Example:

```
% python
>>> def add(a, b):
...     return a + b
>>> map(add, [1,2,3,4,5], [9,8,7,6,5])
[10, 10, 10, 10, 10]
```

Functional List Operations

- **`filter(fct, seq)`**
  - applies **`fct`** to each element of **`seq`**
  - returns a list of those elements for which **`fct`** returns true
- Example:

```
% python
>>> def is_positive(x):
...     return x > 0
...
>>> def is_negative(x):
...     return x < 0
...
>>> def is_even(x):
...     return x % 2 == 0
...
>>> l = [0,-1,1,-2,2,-3,3,-4,4,-5,5]
>>> filter(is_positive, l)
[1, 2, 3, 4, 5]
>>> filter(is_negative, l)
[-1, -2, -3, -4, -5]
>>> filter(is_even, l)
[0, -2, 2, -4, 4]
```

Functional List Operations

- **`reduce(fct, seq)`**
  - reduces sequence **`seq`** to a single element
  - applies **`fct`** to "pairs" of elements of **`seq`**
    - `tmp = fct(seq[0], seq[1])`
    - `tmp = fct(tmp, seq[2])`
    - `tmp = fct(tmp, seq[3])`
    - `...`
    - `result = fct(tmp, seq[N])`
- Example:

```
% python
>>> def add(a, b):
...     return a + b
>>> def max(a, b):
...     if a > b:
...         return a
...     else:
...         return b
>>> reduce(add, [1,2,3,4,5,6,7,8,9])
45
>>> reduce(max, [3,4,2,6,9,4,3,1])
9
```