Defining and Using Functions

- Zelle: Chapter 6
- Lutz: read once you now the basics

Monday, February 2, 2009

Hints and reminders ...

- Problem Set 3
  1) Write two functions operating on strings
     - Multiple solutions exist, depending on what string functions you use
  2) Write three functions computing spatial information among points in 2-d
     - Think about the type of loop structures needed; write code in order the functions are listed and test individually
     Read the description carefully!

- Exam 1: Thursday, February 19, 6:30-7:30pm
  - Basic Python operations, statements, and functions
  - Can use Appendix A of Zelle, but otherwise closed book

Monday, February 2, 2009
Clicker Question
from string import *
mystring = "we can split a string into what using the split function?"
L = mystring.split("a")
print L[0], "\nlength of list: ", len(L)

Prints:
A. "we can"
   length of list: 4
B. we c length of list: 4
C. we c length of list: 1
D. we c
   length of list: 4

Why functions?

- Functions allow decomposition of programs into manageable, reusable, components
- Functions are similar to mathematical functions, but also differ:
  - parameter “passing”
  - side effects
The function of functions

- What you’ve already seen:
  - Programs having a “main” function
  - Built-in Python functions: abs, min, max,…
    - [http://docs.python.org/lib/built-in-funcs.html](http://docs.python.org/lib/built-in-funcs.html)
  - Library functions: math.sqrt or sqrt
  - Functions capture “canned” computations that can be reused

- Functions avoid drawbacks of duplicating similar/identical code in different places:
  - writing same code twice
  - maintaining separate copies
  - Functions make code more readable
Functions, informally

- A function is like a subprogram: a small program inside a larger program
- A function gives a name to that subprogram
- The subprogram can be executed by referring to the name

Functions, informally

- Functions are first defined
- The definition can then be used: “called”/“invoked”
- Functions be placed anywhere in the program, but before they are invoked
def fun1 (x, y):
    x = 25
    print x+y

def fun2(a,b):
    a = a + 1
    print a, b

def fun3(r,q):
    return r+q

def main():
    s, t = 1, "test"
    x, y = 20, 100
    fun1(s, x)
    fun2(s,t)
    a = fun3(s, y)

main()
Functions, informally

```python
def main():
    print "Happy birthday to you!"
    print "Happy birthday to you!"
    print "Happy birthday, dear Fred..."
    print "Happy birthday to you!"
```

Functions, informally

```python
def happy():
    print "Happy birthday to you!"

def main():
    happy()
    happy()
    happy()
    print "Happy birthday, dear Fred..."
    happy()
```
def happy():
    print "Happy birthday to you!"
def singfred():
    happy()
    happy()
    print "Happy birthday, dear Fred..."
    happy()

def singlucy():
    happy()
    happy()
    print "Happy birthday, dear Lucy..."
    happy()
Functions, informally

```python
def sing(person):
    happy()
    happy()
    print "Happy birthday dear",
    person, "...
    happy()

def main():
    sing("Fred")
    print
    sing("Lucy")
```

Functions, informally

- A parameter is a variable that is initialized to the value of the corresponding argument when the function is called
- Python has a number of additional useful rules/conventions for parameters
Functions, informally

- Functions use “bound” variables and “unbound” variables
- In function f, variable x is bound to f (x in f is different from x outside f)
- In function g, x is unbound and it refers to the x existing outside of g
- This can get tricky and can have unexpected side effects!

```
def f(x):
    return x+1

def g():
    return x+1
```

x=10

Functions & parameters: the details

- Variables are bound to “scopes”
- Different scopes can use the same name for different variables
- Variables can be “globally” scoped
- Variables can be scoped “locally” within a function
Functions & parameters: the details

```python
def f(x):
    x = x + 1
    return x
def g():
    y = f(x)
    print x, y
x = 12
g()
```
Functions & parameters: the details

- A function is called by using its name followed by a list of arguments, one for each parameter:
  \texttt{name(arguments)}

- A function is defined with a possibly empty list of parameters:
  \texttt{def name(parameters):
    body}

Functions & parameters: the details

- At a call:
  - Execution of the caller is suspended
  - The arguments to the call are assigned to the parameters of the called function
  - The body of the called function is executed
  - Execution of the caller is resumed

Monday, February 2, 2009
```python
def main():
    sing("Fred")
    print
    sing("Lucy")
```
def main():
    sing("Fred")
    person = "Fred"
    print
    sing("Lucy")

def sing(person):
    happy()
    happy()
    print "Happy birthday, dear", person,
    "..."
    happy()
Functions & parameters: the details

def main():
    person = "Fred"
    sing("Fred")
    print
    sing("Lucy")

    person = "Fred"
    def sing(person):
        happy()
        happy()
        print "Happy birthday, dear", person,
        "..."
        happy()
    
    def happy():
        print "Happy birthday to you!"

Monday, February 2, 2009
def main():
    sing("Fred")
    person = "Fred"
    print
    sing("Lucy")
    def sing(person):
        happy()
        happy()
        print "Happy birthday, dear", person,
        "...
        happy()

def happy():
    print "Happy birthday to you!"

Monday, February 2, 2009
def main():
    sing("Fred")
    print
    sing("Lucy")

def sing(person):
    happy()
    happy()
    print "Happy birthday, dear", person, "...
    happy()
Functions & parameters: the details

def main():
    sing("Fred")
    print
    sing("Lucy")

Monday, February 2, 2009
def main():
    sing("Fred")
    print
    sing("Lucy")

def sing(person):
    happy()
    happy()
    print "Happy birthday, dear", person,
"...
    happy()

Monday, February 2, 2009
```python
def main():
    sing("Fred")
    print
    sing("Lucy")

def sing(person):
    happy()
    happy()
    print "Happy birthday, dear", person,
    "..."
    happy()
```

Monday, February 2, 2009
Functions & parameters: the details

- Multiple parameters are matched up by position: arguments are assigned one by one to their parameters in the order they appear.
- Consider:
  ```python
  def f(x,y):
      return x + y
  ```
  
f(1,2) assigns 1 to x and 2 to y
  
f(a,b) assigns the current value of a to x and the current value of b to y

Getting results from functions

- Arguments are inputs to functions, initializing their parameters.
- Calling the same function with different arguments can produce different results.
- We’ve already seen use of the `return` statement to provide a result.
Clicker question

```python
def print_twice(bruce):
    print bruce
    print bruce

def cat_twice(part1, part2):
    cat = part1 + part2
    print_twice(cat)

st1 = "Bing 
st2 = "Bang 

cat_twice(st1, st2)
```

What is printed?

A. Bing Bang
Bing Bang

B. bruce
bruce

C. part1part2
part1part2

D. Bing Bang Bing Bang

Monday, February 2, 2009

Clicker question

```python
def f(x):
    x = x+1
    return x

x = 11

def g():
    y = f(x)
    print x, y

    x = 12
    g()
```

What is printed?

A. 11  12
B. 12  13
C. 13  13
D. 12  11

Monday, February 2, 2009
Clicker question

def fun1(x, y):
    x = 25
    print x+y

def fun2(a,b):
    a = a + 1
    return a+b

if __name__ == '__main__':
    s = 1
    x, y = 20, 100

    fun1(s, x)
    t = fun2(s, y)
    print s, t, x

What is printed?

A. 120
   1  102  20

B. 45
   1  102  20

C. 45
   2  102  20

D. 125
   1  102  20