Functions!

28 Sep 2009
CMPT140
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Outline for today

- for loops and range()
- Procedures (functions, subroutines)
  - No parameters
  - With parameters
  - Scope
  - Global variables (why not to use them)
- Functions (return a value)
- Predicates: pre-conditions, post-conditions
- Call-by-value vs call-by-reference

for loops

- Many loops do counting: the for loop is an easy construct that prevents many of these errors
- Syntax:
  - **for *target* in *expression list* :**
    - ***Statement sequence***
- Example:
  - **for counter in (0, 1, 2, 3, 4):**
    - **print counter,**
  - Output: **0 1 2 3 4**
- for loops can also take an else sequence, like while loops

range()

- The built-in function range() produces a list suitable for use in a for loop:
  - **range(10)** → **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**
  - Note 0-based, and omits end of range
- Specify starting value:
  - **range(1, 10)** → **[1, 2, 3, 4, 5, 6, 7, 8, 9]**
- Specify increment:
  - **range(10, 0, -2)** → **[10, 8, 6, 4, 2]**

- *Technically, range() returns a list (mutable), rather than a tuple (immutable). More on this later.*

for loop examples

- Print squares from 1^2 up to 10^2:
  - **for counter in range(1, 11):**
    - **print counter * counter,**
- for loops can iterate over other lists:
  - **for appleVariety in (“Fuji”, “Braeburn”, “Gala”):**
    - **print “I like”, appleVariety, “apples!”**

- *Technically, the for loop uses an iterator to get the next item to loop over. Iterators are beyond the scope of CMPT140.*

Procedures

- Fourth program structure/flow abstraction is composition
- This is implemented in Python using procedures
  - Also called functions, subroutines
- A procedure is a chunk of code doing a sub-task
  - Written once, can be used many times
- We've already been using procedures:
  - print, input, raw_input, etc. (not if or while)

Procedure input and output

- Procedures can do the same thing every time:
  - **print # prints a new line**
- Or can change behaviour depending on input parameters (arguments):
  - **print(“Hello!”) # prints string param**
    - List of parameters goes in parentheses
      - (print is special and doesn't always need parens) (until Python3.x)
- Procedures can also return a value for use in an expression:
  - **numApples = input(**
    “How many apples? ”)

Example: no parameters

- Procedure to print program usage info:

```
def print_usage():
    """Display a short help text to the user."""
    print "This program calculates the volume",
    print "of a sphere, given its radius."

...
if string.upper(userInput) == "H":
    print_usage()
```

docstring

Example: with parameters

- Calculate volume of a sphere:

```
from math import pi

def print_sphere_volume(radius):
    """Calculate and print the volume of a sphere
    given its radius.
    """
    print "Sphere Volume = %.2f" % (4/3)*pi*(radius**3)

print_sphere_volume(3.5)
```

formal parameter

actual parameter

Scope

- Procedures inherit declarations from enclosing procedures/modules:
  - Declarations:
    - import (e.g., math.pi)
    - variables
    - Other procedures

- Items declared within the procedure are local: not visible outside that procedure
- The scope of a variable is where that variable is visible

Example: scope

```
from math import pi

def print_sphere_volume(radius):
    """Calculate and print the volume of a sphere
    given its radius.
    """
    vol = (4/3) * pi * (radius**3)
    print "Sphere Volume = %.2f" % vol

myRadius = 3.5
print_sphere_volume(myRadius)
```

radius, vol, pi, myRadius

myRadius, pi, print_sphere_volume()

- What variables are visible in print_sphere_volume()?
- What variables are visible outside the procedure?

Avoid global variables

```
from math import pi
def print_sphere_volume(radius):
    """Calculate and print the volume
       of a sphere given its radius.
    """

    myVolume = (4/3)*pi*(radius**3)
    print "Sphere Volume = %.2f" % myVolume

myVolume = 10
print_sphere_volume(3.5)
```

Note assignment to global var

What is the value of myVolume here?

Functions

- Functions (function procedures, “fruitful” functions) are procedures which return a value:
  - string.upper('g') returns 'G'
  - **def double_this(x):**
    - **"""Multiply by two."""**
    - **return x * 2**
- Statically-typed languages require function definition to declare a return type
- Multiple return statements allowed; first one encountered ends execution of the function

Functions in Python

- It turns out that in Python, every procedure returns a value: even....
  - **def print_usage():**
    - **"""Print a brief help text."""**
    - **print "To use this program, type ...."**
- If no explicit return statement, or plain return, then the special None value is returned
- Must use parentheses when invoking procedures
  - Even those without params: **print_usage()**
  - Otherwise you get the function object