Computer Science 1 — CSci 1100
Lecture 10 — For Loops

Overview

- Review the basic idea of `for` loops
- Other uses of `for` loops
- Ranges
- Nested for loops and images
- Enumerations

Reading: *Practical Programming*, Section 7.1 and the start of 7.2.

Current Knowledge of For Loops

- Based on the idea of accessing each entry in a list or input file in succession and applying one or more operations to each.
- Example 1 (from Lecture 7):

```
co2_levels = [ 320.03, 322.16, 328.07, 333.91, 341.47, 348.92,
 357.29, 363.77, 371.51, 382.47, 392.95 ]

avg = sum(co2_levels) / len(co2_levels)
for value in co2_levels:
      diff = value - avg
      print 'Difference from average is %.2f' %diff, 'ppm'
```

- Example 2 (from Lab 5):

```
file = open('lab5businesses.txt','r')
for line in file:
    p_line = parse_line(line)
    #  other processing code to be added here
```

What Else Might We Want to Do?

- Output the year associated with each CO2 level — requires knowing the position in the list
- Look for trends in the data: compare value for current year with value for a previous year.
- Access data from multiple lists in the same loop.
- Look at combinations of values or loop over multi-dimensional data.

Ranges

- Generates a list of integers. For example,

```
 >>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- Notice this is up through and **not including** the last value!
- If we want to start with something other than 0, we provide the starting values

```
>>> range(3,8)
[3, 4, 5, 6, 7]
```

- We can create increments:

```
>>> range(4,20,3)
[4, 7, 10, 13, 16, 19]
```

  starts at 4, increments by 3, stops when 20 is reached or surpassed.

- We can create backwards increments

```
>>> range(-1, -10, -1)
[-1, -2, -3, -4, -5, -6, -7, -8, -9]
```

Using Ranges in For Loops

- We can use the `range` as the list of values in a for loop. Our first example is printing the contents of the `planets` list

```
planets = [ 'Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter',
    'Saturn', 'Uranus', 'Neptune', 'Pluto' ]
for i in range(len(planets)):
    print planets[i]
```

- The variable `i` is variously known as the "index" or the "loop index variable" or the "subscript".
- We will modify the loop in class to do the following:
  - Print the indices of the planets
  - Print the planets backward.
  - Print every other planet.

Exercise

- Generate a range for the positive integers less than 100. Use this to calculate the sum of these values, with and without a for loop.
- Use a range and a for loop to print the even numbers less than a given integer `n`.
- The years for the CO2 levels were in parts per million every 5 years starting in 1960. Generate a range based on this and on the length of `co2_levels`. Assign the resulting list to the variable `year`.
- Suppose we want a list of the squares of the digits 0..9. The following does NOT work

  ```
  squares = range(10)
  for s in squares:
      s = s*s
  ```

  Why not? Write a different for loop that uses indexing into the `squares` list to accomplish our goal.

Accessing Two Lists in One Loop: Printing Years and CO2 Levels

- We can use our generated list, `years`, to solve the problem of printing both the year and the CO2 level.

  ```
  for i in range(len(co2_levels)):
      print "Year %d: CO2 level %.2f" %(years[i], co2_levels[i])
  ```

  Notice that we created a second range of indices and used these to index **both** `years` and `co2_levels`.

Loops That Do Not Iterate Over All Indices

- Sometimes the loop index should not go over the entire range of indices, and we need to think about where to stop it "early", as the next two examples show
- Returning to our example from the very first lectures, we will consider the following problem: Given a string, how can we write a function that decides if it has three consecutive double letters? We'll write this function together in class, starting with

  ```
  def has_three_doubles(s):
  ```

- We'd like to count how many years the CO2 level has risen relative to the previous five years. We'll write a simple loop to do this.
- In each case, we have to think carefully about where to start our looping and where to stop!

Nested For Loops

- Some problems require "iterating" over either
  - two dimension of data, or
  - all pairs of values from a list
- As an example, here is code to print all of the products of digits:

```
digits = range(10)
for i in digits:
    for j in digits:
        print "%d x %d = %d" %(i,j,i*j)
```

- How does this work?
  - for each value of i — the variable in the first, or "outer", loop,
  - Python executes the *entire* second, or "inner", loop

Nested vs. Sequential Loops

- We can understand more by looking at the difference between these three sets of loops, two of which involve nested loops and the other two consecutive, or "sequential", loops

```
# Version 1
sum = 0
for i in range(10):
    for j in range(10):
        sum += 1
print sum

# Version 2
sum = 0
for i in range(10):
    for j in range(i+1,10):
        sum += 1
print sum

# Version 3
sum = 0
for i in range(10):
    sum += 1
for j in range(10):
    sum += 1
print sum
```

Image Examples

- We will start with the problem of converting a color image to gray scale as an example of the use of double for loop. This is the code `rgb2gray.py` on the course Piazza site.
- We will modify this to do several things:
  - Flip an image upside down
  - Reflect an image left-right
  - Create a cartooned version of the image

Enumerate

- We can use Python `enumerate` to generate a pairs from a list containing the index and the value in the list at that index.
- For example,

```
planets = [ 'Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter',
    'Saturn', 'Uranus', 'Neptune', 'Pluto' ]
for pr in enumerate(planets):
    print pr[0], pr[1]
```

- We can also use the short-hand:

```
planets = [ 'Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter',
    'Saturn', 'Uranus', 'Neptune', 'Pluto' ]
for i,pl in enumerate(planets):
    print i, pl
```

Multiple Assignments

- The idea behind `enumerate` can be applied to making multiple assignments even without loops.
- Examples we will consider in class include

```
>>> x,y = 5,6
>>> a,b = [ 'hi', 'bye' ]
>>> l0, l1, l2 = 'abc'
```

- Then we'll look at what happens when the number of variables on the left and values on the right do not match.

Summary

- Use ranges to generate for loop indices
- Use indices in a for loop to access and change the contents of a list
- The same indices can be used to access two different lists in the same for loop
- Nested for loops access multidimensional data
- Enumerations allow simultaneous access to indices and values in a list.