Topics for this week

- Strings (Zelle 4, Lutz 7))
- Lists (Zelle 4.3, 11.1+11.2, Lutz 8)

Problem Set 2

- Switching signs – many solutions exist
- If you know VPython, you may want to visualize the points generated in problem 2

Academic honest expectations in programming

- What you submit must be your own code
- Do not share your code
- Can discuss assignments in initial stages (often very helpful)

Two-way decisions

if <condition>:
    <statements1>
else:
    <statements2>

Multi-way decisions

if <condition1>:
    <statements1>
elif <condition2>:
    <statements2>
else:
    <statements3>
# quadratic_with_NestedIf.py
# A program that computes the real roots of a quadratic equation.
# Illustrates use of a multi-way decision

import math

def main():
    print("This program finds the real solutions to a quadratic\n")
    a, b, c = input("Please enter the coefficients a, b, c, separated by ', ': ")

    discrim = b * b - 4 * a * c
    if discrim < 0:
        print("The equation has no real roots!"
    elif discrim == 0:
        root = -b / (2 * a)
        print("There is a double root at", root)
    else:
        discRoot = math.sqrt(b * b - 4 * a * c)
        root1 = (-b + discRoot) / (2 * a)
        root2 = (-b - discRoot) / (2 * a)
        print("The solutions are:\", root1, root2)

main()
Strings

- Sequence of characters
- Literals delimited in programs by " or single '
  - Use " to include '
  - Use ' to include "
  - Use backslash (\) to “escape” either
    print "Alice said, "Hello, 'Bob'"
- See Lutz, page 126 for other \ options
- Multiline string literals delimited by '''
  x = '''line one
  line two
  last line'''
  % useful for comments

Basic string operations

txt1 = "green ham"
txt2 = "and eggs"

txt = txt1 + txt2  + is concatenation of two strings
txt = txt1 + " " + txt2

txt[0], txt[3]  indexing into the string — returns a character
txt[16]        string index out of range
txt[1:4]       slicing returns a section of the string
 txt*3          creating repetitions
len(txt)       returns the length of the string
More string operations

txt1 = "green ham"
txt2 = "and eggs"
txt = txt1 + " " + txt2 (= “green ham and eggs”)

for i in range (len(txt)):  iterate through the string
  print txt[i]

for i in txt:  also iterates through the string
  print i

Slice Examples

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Helpful to think of slice positions being between characters

- S[0:2] = “AB”
- S[4:10] = “EFGHIJ”
- S[10]  results in index out of range
- S[2:12] = “CDEFGHIJ”
- S[-5:-1] = “FGHI”
- S[3:]  from element at 4 to the end of the string
Library Operations on Strings (import string)

capitalize("hello there")   -> 'Hello there'
capwords("hello there")     -> 'Hello There'
center("hello there", 20)   -> '    hello there     '
count("x y z", " ")         -> '2'
find("hello", "l")          -> '2'
join(["a", "b", "c"]       -> 'a b c'
ljust("hello", 10)          -> 'hello    '
lower("HELLO")              -> 'hello'
lstrip("   hello")          -> 'hello'
replace("hello", "l", "w")  -> 'hewwo'
rfind("hello", "l")         -> '3'
rjust("hello", 10)          -> '    hello'
rstrip("hello   ", " ")     -> 'hello'
split("a b c d", " ")       -> ['a', 'b', 'c', 'd']
upper("hello")              -> 'HELLO'

Clicker Question

s = “JanFebMar”

What operation does not result in “FebFeb”

B. 2*s[3:6]
D. s[-6:-3]
E. s[3:6]*2
Lists

- Sequence of arbitrary values of arbitrary types
- Very similar operations as on strings
  - Indexing and slicing
  - Concatenation (L1+L2)
  - repetition (L* 3)
  - len(L)
  - for i in L: … iterating over the elements in a list
- Other operations, e.g.,
  - L.append("new last")
- And the big difference compared to strings … the way we change them

Use the shell to execute list operations

L1 = ['a', 'b', 'c', 'd', 'f']
L2 = [0, 1, [2, 2.5], 5, [3.3, 3.5, 3.95]]
   L2 is a list of length 5
   L2[2] = [2, 2.5]
L3 = ["green", "red", "blue"]
L4 = [15, 16, 17]

matrix = [[1, 2, 3], [2, 3, 4], [4, 5, 6] ]
Matrix[0] = [1, 2, 3]
Problem

- From http://www.imdb.com/chart/top, get the top 250 movies (file top250.txt)
- Write a program that finds the top movies for a given year
- Tools we need
  - Open file, read lines
  - Split lines into lists of elements
  - Access list elements
  - Format a string

The input file ....

Top 250 movies as voted by our users

<table>
<thead>
<tr>
<th>Rank</th>
<th>Rating</th>
<th>Title</th>
<th>Year</th>
<th>Votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>9.1</td>
<td>The Shawshank Redemption &lt;/title/tt0111161/&gt; (1994)</td>
<td></td>
<td>401,139</td>
</tr>
<tr>
<td>2.</td>
<td>9.1</td>
<td>The Godfather &lt;/title/tt0068646/&gt; (1972)</td>
<td></td>
<td>336,865</td>
</tr>
<tr>
<td>4.</td>
<td>8.9</td>
<td>Buono, il brutto, il cattivo, Il &lt;/title/tt0060196/&gt; (1966)</td>
<td></td>
<td>116,436</td>
</tr>
<tr>
<td>5.</td>
<td>8.9</td>
<td>The Dark Knight &lt;/title/tt0468569/&gt; (2008)</td>
<td></td>
<td>335,624</td>
</tr>
<tr>
<td>6.</td>
<td>8.9</td>
<td>Pulp Fiction &lt;/title/tt0110912/&gt; (1994)</td>
<td></td>
<td>331,249</td>
</tr>
<tr>
<td>7.</td>
<td>8.8</td>
<td>Schindler's List &lt;/title/tt0108052/&gt; (1993)</td>
<td></td>
<td>219,237</td>
</tr>
<tr>
<td>8.</td>
<td>8.8</td>
<td>One Flew Over the Cuckoo's Nest &lt;/title/tt0073486/&gt; (1975)</td>
<td></td>
<td>167,823</td>
</tr>
</tbody>
</table>
import string

def main():
    infile = open("top250.txt")
    infile.readline()  # ignore first three lines in file
    infile.readline()
    infile.readline()

    for line in infile:
        fields = line.split(" ")

Finding the year in a line

for line in infile:
    fields = line.split(" ")

    length = len(fields)
    lastField = fields[length-1]
    year = lastField[1:5]
import string
searchstring = "2008"

def main():
    infile = open("top250.txt")
    infile.readline()
    infile.readline()
    infile.readline()
    total = 0
    for line in infile:
        fields = line.split(" ")
        length = len(fields)
        lastField = fields[length-1]
        year = lastField[1:5]

        if year == searchstring:
            total = total + 1
            print fields

    print "Total number of", searchstring, "top movies", total

main()