Lists, For-loops, and Pyplot

Professor Hugh C. Lauer
CS-1004 — Introduction to Programming for Non-Majors

(Slides include materials from *Python Programming: An Introduction to Computer Science*, 2nd edition, by John Zelle and copyright notes by Prof. George Heineman of Worcester Polytechnic Institute)

Lists — Review

- **An *ordered* collection of values or objects**
  - Enclosed in square brackets
  - Separated by commas
- **E.g.,**
  - [1, 2, 3, 5, 7, 11, 13, 17, 19, 23]
  - [“Listen”, “my”, “children”, “and” , “you”, “shall”, “hear”]

Note:–The brackets and commas are *not* part of list, but merely for display.

- **May be all the same type or of different types**
  - Examples
- **The *empty list***
  - []
- **May be assigned to variables**
  - May be passed as arguments to functions
  - May be returned from functions as results

Lists (continued)

- **Accessing elements of a list**
  - X[0], X[1], …, X[i+j], …
  - X[-1] is the *last* element of the list
    X[-2] is the *second last* element of the list
- **Adding to end of list**
  - X.append(newElement)

Note "dot" notation
- **append() is a *method* of list object**

- **Updating elements of a list**
  - X[0] = 5
    X[-1] = X[0] + X[1]
- **Length of a list**
  - len(X) # returns number of elements in list
    # always non-negative

Questions?

For-Loop

- What does a for-loop look like? (Lab #1)
- How would you explain it to a friend not yet in this course?

```
for var in <something>:
   body statement1
   body statement2
   …
   body statementk
```

This is a new *variable* name!

…y not be …rently in use

Each continuation line is indented one "unit"— i.e., tab

End of for-loop denoted by reve… to previous indentation level

For use only in loop body

For-Loop

- **What does a for-loop look like? (Lab #1)**
- **How would you explain it to a friend not yet in this course?**

```
for var in <something>:
   body statement1
   body statement2
   …
   body statementk
```

- **Meaning:–**
  - Go thru (i.e., enumerate) **`<something>`**
  - For each item in enumeration …
  - … assign **`var = that_item`**
  - … execute the body statements using **`var`**
  - Repeat with next item of enumeration, etc.
  - Loop stops when enumeration is exhausted

What can we enumerate?

- **More or less anything!**

- **For now, we will enumerate integers:–**
  - E.g., range(10)

- **Meaning:–**
  - Each time around loop, call **`range()`** to emit the next value
  - Stop when **`range()`** has emitted all that it is going to emit!

- **`range()` is a special kind of *Python* function …**
  - … called a *generator!*
  - Remembers what it did last
  - Each time, it returns the next item
  - …
  - … until the end, when it emits a special code to tell loop to stop

For-Loop (continued)

- **Explain `range(10)`**
  - i.e., what numbers are generated?
- **Can we see a "range"?**
  - Yes, use the `list()` function
- **Another form of range?**
  - `range(start, stop)`
    `range(start, stop, step)`

Includes start but *not* stop!

Other kinds of enumerations

```
for item in List:
   body statement1
   body statement2
   …
   body statementk
```

Applies entire loop body separately to each item in List

```
for char in String:
   body statement1
   body statement2
   …
   body statementk
```

Applies entire loop body separately to each character in String

- Even more kinds of enumerations later in course

Questions?

Notes on matplotlib and pyplot

pyplot

- *pyplot:* a module inside of *matplotlib*
  - Installed at start of course
- Collection of functions that make *matplotlib* work (somewhat) like MATLAB
- Getting started:–

```
import matplotlib.pyplot as plt
```

"as" clause is optional

Allows shorthand naming!

```
someList = [1, 1/2, 1/3, 1/4, 1/5, 1/6]
plt.plot(someList)
plt.show()
```

Brings up a graph window

plot adds it own x-axis

pyplot (continued)

- **Plotting with *x*- and *y*-axes**

```
yValues = [1, 1/2, 1/3, 1/4, 1/5, 1/6]
xValues = [1,  2,  3,  4,  5,  6]

plt.plot(xValues, yValues)
plt.show()
```

x-axis list is 1st argument

y-axis list is 2nd argument

```
plt.plot(xValues, yValues, 'bo')
```

Optional 3rd argument

- **3rd argument indicates format of points, etc.**
  - `'bo'` — blue circles
  - `'g^'` — green triangles
  - `'r-'` — red line
  - …

Multiple plots

- Plotting with several sets of *x*- and *y*-axes

```
y1Values = [1, 1/2, 1/3, 1/4, 1/5, 1/6]
y2Values = [1, 2*2, 3*3, 4*4, 5*5, 6*6]
xValues = [1, 2, 3, 4, 5, 6]

plt.plot(xValues, y1Values)
plt.plot(xValues, y2Values)
plt.show()
```

```
plt.plot(xValues, y1Values, 'bo')
plt.plot(xValues, y2Values, 'r^')
plt.show()
```

Alternative for Multiple plots

- Plotting with several sets of *x*- and *y*-axes

```
y1Values = [1, 1/2, 1/3, 1/4, 1/5, 1/6]
y2Values = [1, 2*2, 3*3, 4*4, 5*5, 6*6]
xValues = [1, 2, 3, 4, 5, 6]

plt.plot(xValues, y1Values,
         xValues, y2Values)
plt.show()
```

```
plt.plot(xValues, y1Values, 'bo',
         xValues, y2Values, 'r^')
plt.show()
```

Other pyplot functions

```
plt.ylabel('some text')
plt.xlabel('some other text')

plt.axis([xMin, xMax, yMin, yMax])
```

Substitute values for min and max of axes

- **Many, many options and controls**
  - More than can be covered in this course
  - More than you will need in near future

- **http://matplotlib.org/users/pyplot_tutorial.html**

  - Read thru this. Very basic, easy to understand.

Questions?