Miscellaneous Topics

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)

How many have discovered Appendix B in the textbook?

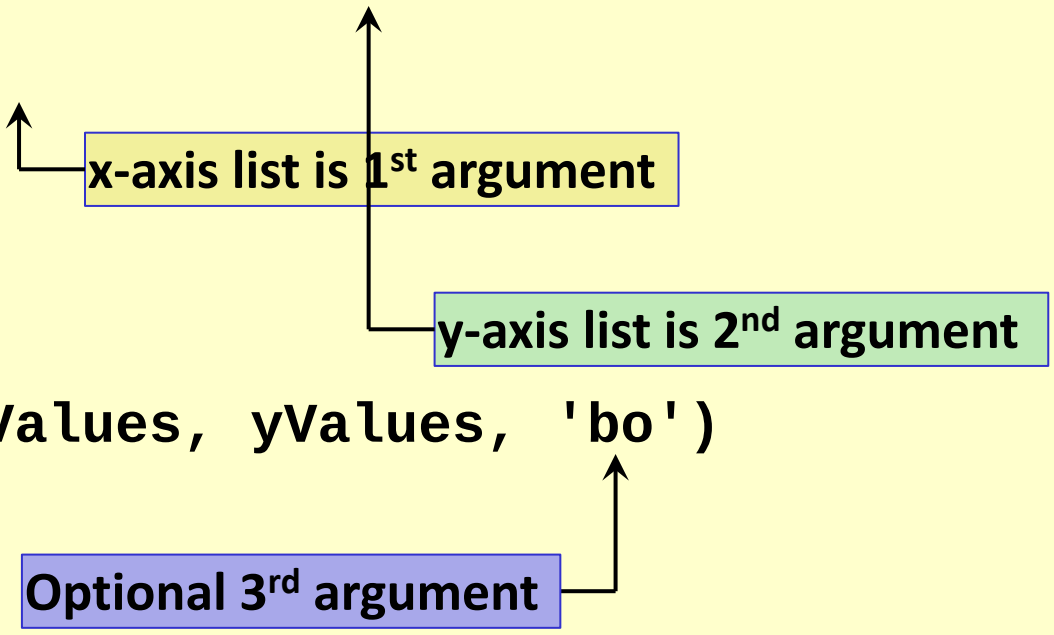
What about Appendix A?

Some useful stuff!

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()
```




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


- **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

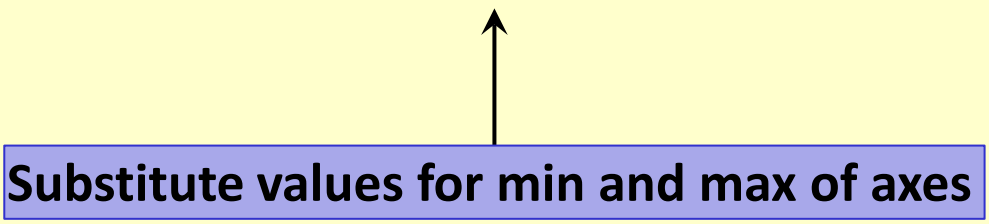
- Plotting with several sets of *x*- and *y*-axes

```
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])
```



- **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?

Assignment (again!)

- **What is the definition of *assignment*?**
  - X = some value
    Y = some computational object
- **A:– "The name on the left of the '=' sign is applied as a label to the value or object on the right!"**
- **Let `L = [5, 4, 3, 2, 1]`**
- **Let `M = L`**
  - What is the value of `M[4]`?
- **Now do `L[4] = 10`**
  - What is the value of `M[4]`?

What is going on?

Assignment (again!) — continued

- **L is a label for the computational object (i.e., list) `[5, 4, 3, 2, 1]`**
- **`M = L` makes M a label for the *same* object!**
  - i.e., the same list
- **Therefore,**

```
L[4] = 10
```

- **… changes the list that L refers to**
- **Since M is *another* label for the same object, M see that element #4 of that object has changed!**
- **Big Deal!**
- **Totally unexpected for students of other languages!**

Fundamental to *Python*

- **Expect same behavior for all objects — e.g.,**
  - Dictionaries
  - Graphic objects —
    - Circles, rectangles, lines, buttons, etc.
  - Pictures
    - E.g., maps
- **Any *classes* that you create in *Python***
- **Any object may have one or more labels —**
  - *Variable*
  - *Parameter*
  - *List element*
  - *Dictionary entry*
  - …
- **If *all* of the labels are forgotten or replaced, particular object can no longer be accessed.**
  - *Python* "garbage collects" it!

Questions?

Reading Assignment

- **Read all of Chapter 3**
- **Partly covered in Week 1**
  - *Elements of a Python Program*

Shortcuts

- **`v = v+1` is a nuisance**
  - Have to write v twice
  - Okay if v is a simple name
  - Not so convenient if v is an element of a list of a list!
- **Same for `-`, `*`, `/`, `%`, etc.**
- ***Python* uses a shortcut first introduced in *C***
  - Later adopted by *C++*, *Java*, and other languages
- **`v += 1` means**
  - Add one to the value referred to by v and assign the result back to v
- **`v -= anyExpression`**
  - Subtract value of right side, assign back to v
- **`v *= expr`**
  `v /= expr`
  `v %= expr`

Shortcuts (continued)

- Example — `TaylorCos(x, n)`

```
sum = 0
for i in range(n):
   sum += (-1)**i * x**(2*i) / fact(2*i)
return sum
```

Shortcuts (continued)

- Strangely enough …

  … these widely used shortcuts are not mentioned in the textbook!

Questions?

Another topic — Scope Rules

- **This topic is introduced on p. 175 of the text**
  - Last term, a number of students bumped into it by this time, so let's talk about it now
- **A variable defined *inside* of a function cannot be seen *outside* that function**
  - It ceases to exist when function returns!
  - A new variable of same name is created when function is called again!
- **Reason:– when working on a team project, don't want to constrain team members' use of names *within* their own responsibilities!**
- **Same rule applies to all modern programming languages**

Scope Rules (continued)

- **Same rule applies to modules**
- **I.e., a variable defined in a module can be seen by all functions of that module …**
- **… but not by functions of any other module**
- **Same reason:–**
  - Software designers need to have the freedom to express and name their own internal data without constraint by outside consideration.
- **Exception:–**
  - If you import a variable from one module by another, can see it in the second module!

Scope Rules (concluded)

- **Note:–**
- **Earlier in course, I mistakenly suggested that a similar rule applies to variable of a for-loop**
- **… and to variables created and used *within* the for-loop**
  - Incorrect
- **A variable created inside a for-loop retains its visibility for the rest of the function or module**
  - Retains value until assigned again

Questions?