CSC 108H: Introduction to Computer Programming

Summer 2011

Marek Janicki

Administration

- The exam is **Wednesday Aug 17th, 7 to 10.**
- Office hours next week will be Monday, instead of Tuesday.
- Autotesting results for A2 have been uploaded to Markus.
  - I don't know if you guys can see them, let me know if you can't, and I'll also e-mail them to your cdf accounts.
  - They should be accessible from the screen where you can see your source code. You may need to load them.

Administration

- Clarifications to assignment 3 posted on the discussion board.
- Office hours tomorrow will be 12-2 rather than 2-4.

Storing Data

- Often we want to store data between runs of a program.
- One way to do that is to write it to a file and then load the file.
- This involves a lot of work.
- The file returns everything as a string.
  - So you need write code to cast everything to a string.
  - And more code to take the file and recreate the data that you've lost.

Storing Data

- This is a large bit of extra code to write.
- Also the type of code that can take a long time to debug.
- To avoid this, python has a built in module for storing data.
- It is called Pickle. It allows you to 'pickle' your data to store it in between runs of the program.
- Takes only a single line of code to store and load objects.

Using Pickle

- First we import cpickle.
  - cpickle is a faster version of pickle.
- Then, we a need a file that we'll be either reading from or writing to.
- To store an object we use

```
cpickle.dump(object_name, file_name)
```

- To get an object back we use

```
object_name = cpickle.load(file_name)
```

Interacting with a program.

- So far we've had very little interaction with programs.
  - Really only raw_input statements.
  - Mostly to modify the way a program has been run, we've modifed the parameters we pass it.
- And certainly we've never seen any for the user to decide they want the program to do anything.
- So really all we have real control over is starting programs.

User Interfaces

- UIs are how a user can interact with a program.
- raw input is an example of a command line user interfact (clui).
- We can also pass arguments to a program at the command line.
- To do this we `import sys`
  - The arguments are stored in a list `sys.argv`
  - The first element of the list is always the filename

Break, the first.

User Interfaces

- CLUI (command line user interfaces) used to be quite common.
- Due in part to the low availability of processing power.
- But they haven't been common in 30 years.
- That's why one has windows instead of dos, or why os X doesn't load from the terminal.
- But now we interact with the mouse and on the screen.

Graphical User Interfaces

- Now GUIs are accepted way of doing things.
- We're going to be covering a very basic GUI called easyGui.
- It's not very pretty, but is useful for teaching.
- The books covers a more powerful one called Tkinter.
- easygui does not come built in with python.
  - get it at: http://easygui.sourceforge.net/

GUIs

- GUIs are built out of widgets.
  - 'window gadget'
- Very roughly, there are widgets that display information for the user, and widgets that gather information from the user.
- In easygui we'll be dealing with single widgets at a time.
- But complicated GUIs are built out of lots of widgets.

Display widgets

- To display things in easygui we will use `msgbox`.
- use help(easygui.msgbox) to get all parameters.
- The parameters are optional.
- The first two are the most important ones, referring to the message, and title of the window.

Input widgets.

- Here we have a bit more types of widget to play with.
- We can have inputs where the user clicks a button.
- We can have inputs where the user chooses from a list.
- We can have inputs where the user enters some text.

Button widgets.

- Two main types, buttonbox and indexbox.
  - buttonbox returns the string of the button that was clicked.
  - indexbox returns the index of the button that was clicked.
- The first parameter is the message, the second is the title, and the third is the list or tuple of choices.
- use help(easygui.buttonbox) to get the full list of possible parameters.

List Widgets.

- Here we have choicebox.
- The parameters are the same as for buttonbox and indexbox.
- The difference is that the they are displayed vertically in a list, which makes it easier to fit long lists of choices.

Input Widgets.

- We have entrybox and integerbox.
- enterbox allows the user to input text, intergerbox allows the user to input integers.
  - Both allow you to set default text.
  - Integerbox allows you to set bounds.
- They return the value of the entry field.

Break, the second.

Building programs with easygui

- First decide on the actions you want to provide.
- Then build a widget that allows the user to choose the action.
- For each action, design a function that performs that action.
  - Design means write the parameters and docstring.
- Map each choice to each action.
- Finally, fill in the functions, noting that they may all need their own widgets.

Why is easygui easy?

- Each widget is its own window.
  - This makes easygui great for becoming comfortable with individual widgets.
  - But you don't worry about the layout at all.
- The outputs for all the widgets are predefined and rather simple.
  - So handling user actions is much simpler.
  - Also, since there's only one window at a time, it's a lot easier to design programs.

Event-Driven programming.

- Even in our simple guis, the program doesn't do anything until the user clicks somewhere.
- Mouse clicks and keyboard presses are known as events.
- Because the program waits on them to do anything, this is an example of event-driven programming.
- Contrast with our earlier programs, that just run code until we reach the end of a file.

Event-Driven Programming.

- Most programs that you leave running are event driven.
- More complicated guis have you write your own code that responds to events.
- Generally when making guis we think of three main things:
  - Views: What we show to the user.
  - Models: How we keep the data.
  - Controllers: What reacts to user actions and updates the models. This often then triggers an update in the view.