CIS 211
Spring 2014

Project 4: Building GUIs with Tk

Due Monday April 28
Upload via Blackboard by 11:00 P.M.

Goals

This project is an introduction to graphical user interfaces and Python's `tkinter` library.

Reading

Introduction to Computing Using Python, Ch 9, plus resources available on the class web site.

Programming Projects

1. Write a simple "hello, world" program that displays a text message in a window. You don't need to have any controls in the window -- the program will terminate when the user closes the window. Save your program in a file named `hello.py`.

2. Write a program named `payment.py` that computes the monthly payment on a loan. The payment is a function of the loan amount, the annual rate, and the number of years to pay off the loan. To calculate the payment, first compute a value r as

   $$r = \mathrm{rate}/100/12$$

   and another value p as

   $$p = 12 \times \mathrm{years}$$

   Then the payment is

   $$\mathrm{payment} = \frac{r \times \mathrm{amount}}{1 - (1+r)^{-p}}$$

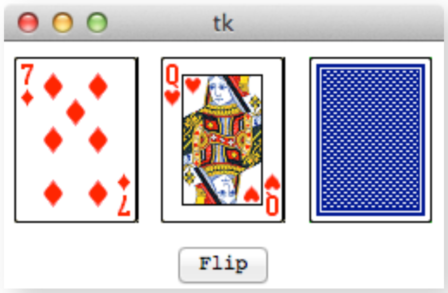
   For example, if you take out a 30-year loan for $150,000 at an annual rate of 4.5% your payment would be $760.03 per month.

   Important*: In order for us to test your program, name your top level window* `root`*, and name the four Entry objects* `amount`*,* `rate`*,* `years`*, and* `payment`*.*

   The GUI for this application should have text entry fields (and labels) for the loan amount, the interest rate, and the number of years to repay the loan. It should have a button labeled "payment". When the user clicks this button, your program should read the values of the three parameters, use the equations shown above to calculate the monthly payment, and display the payment amount. Display the payment in a fourth text entry, but this field should be read-only, *i.e.* users should not be able to change the text in this box once you display it.

3. Write a program named `flipper.py` that will display three playing card images and a button labeled “flip”. Each time the user clicks the button one of the card images should be updated.

   When the window is first opened the user should see the backs of three cards. The first button click should turn over the first card, and the next two clicks should turn over the other two cards. Here is a window after two clicks:

   When you display the front of a card you can pick any card at random.

   The fourth click should appear to remove a card, but in fact your program should change the image to a white square the size of a card. The next two clicks will remove the other two cards.

   The next three clicks should appear to “deal” three new cards face down. Your program just needs to replace the blank images with the image for the back side of a card.

CardLabel Class

The class web sites have two files you can use for the `flipper` project.

`CardLabel.py` defines a class named CardLabel that is derived from Tk’s Label class. The label displays an image that shows the front of a card, the back of a card, or an all-white image that is the same size as a card (to make it appear the card is not there). Once you make a CardLabel object you can call a method named `display` to change the image, e.g. to make it appear a card is being turned over by changing the image from the back side to a front side.

You can also find a compressed file named `cardimages` that has a collection of 55 images used by the CardLabel class. There are two versions, one called `cardimages.tar` and one called `cardimages.zip`. Download whichever one you prefer.

When you uncompress and expand the file you will have a new folder named `cardimages`. There will be 52 images, labeled `card0.gif` through `card51.gif`, with the fronts of the playing cards. The numbers are the same card IDs we used for the last project, *i.e.* the numbers 0 to 51 correspond to 2♣ up through A♠.

The images for the backs of the cards will be in `back-red.gif` and `back-blue.gif`. An all-white blank card will be in `blank.gif`.

When your program starts is should a class method named `load_images` to get the images from the folder and make them available to be used as labels.

Extra Credit Ideas

- Play around with different sizes, shapes, colors, and fonts in your “hello, world” program. Add a quit button. Add buttons with the names of different languages so that when the user clicks a button the message changes to “hello” in that language. This might be a good chance to play with Unicode, both in the message text and in the string displayed on a button.
- You should be able to find formulas that will compute any one of the loan attributes given values for the other three. For example, given the interest rate, loan amount, and payment amount you should be able to calculate the number of years before the loan is paid off. Modify your program so it fills in any one of the boxes if the other three have been filled in.

Testing

We will test your `hello` and `flipper` programs by running them and verifying the buttons trigger the correct actions. We will use a script to run your `payment` program and check the value stored in the output field.

What to Turn In

Documentation Write a short description (two or three paragraphs total) of what you did for this project. This is your chance to tell the graders about anything you think you did well. The documentation should be in a file named `writeup` with an extension that identifies the file format (.doc for Microsoft Word, .pdf for Adobe PDF, .txt for plain text, .rtf for rich text format).

Code For this project you need to submit three different Python programs, named `hello.py, payment.py,` and `flipper.py`.

Create a package (tar or zip format) that includes your writeup and three programs and upload the package via Blackboard.