1 Administrative Topics

- Return graded quizzes
- On Wed night we will be having a social activity of some sort - movie night, I think. My guess is there will be a movie in our Bob’s classroom at 7pm. Details to follow.
- Project 5 is due Friday night. Come to lab this week if you want assistance. There are currently no TA’s scheduled for Thursday night. The TA schedule is the same as usual. Plan accordingly.
- What to turn in when you turn in code. We do not need to be able to discern your programming process from your code files. Get everything neatly organized and turn in only that code which is necessary for the project part of the assignment.

2 Organizing Code

There are two important types of decisions that need to be made when organizing your code – (1) which functions to write, and (2) where to put them. For the first half of the semester, we are using a hierarchical design for our functions.
2.1 Hierarchical Design

The design of this project is top-down, or hierarchical. We break each task into smaller tasks.

We did this with our turtle projects. We had functions that drew scenes, which called functions that drew complex shapes, which called functions that drew simple shapes.

2.2 Where to put the functions

To keep code neat, it is also important to put the functions into modules that have clear purposes.

For our turtle projects, we put all basic shape functions into one module named shapes. We put the complex shapes in that module as well. Why? Because they all accomplished similar tasks - they drew shapes. Then we put the scene functions into different files. How we chose to design that part was more subjective. I think one good way to do it was to write a module called colby_scenes.py and put the scene files in it.

For main programs, it makes sense to have just one in a file.

2.3 Guidelines

- Each function should perform a specific task. That task should be described in a comment. Also, comments should describe its parameters (i.e. the types of data we need for each parameter value and how it is going to be used) and the output (what is returned).

- Functions that perform similar tasks should be placed together in a module. And within that module, functions should be orders in an organized way. This means that if function $A$ calls function $B$, then function $B$ must be first. It also means that if function $B$ and function $C$ are similar, and function $D$ is slightly less similar, then functions $B$ and $C$ should be next to each other and function $D$ should be before or after them. In your filter.py file, this means you should put your putPixmap functions together and all of your filters together and all of your test
functions together. Since test functions call the other functions, they should be last.

- I think of functions lower in the hierarchy as being more flexible and as being there to do the bidding of the functions higher in the hierarchy. So,

- Data should flow down the hierarchy. E.g. command line arguments should be handled initially by the main program, and any information needed by other functions should be passed to them via their parameters. Also, if there is some data that is hard-coded (e.g. an image file is named, which happens in project 5 in the main collage-building program for task 4), that should happen in the main program.

So when you are turning in files, make sure

- Organize your code so that the whole thing works as one coherent project. There may be multiple main programs, but they should be using the same basic set of functions. This means, for example, that there should never be more than one copy of the same function in your turn-in folder.

- The only .py files you turn in are necessary for the project part of the assignment (i.e. we don’t want to see lab3.py).

- The only image files you turn in are necessary to run the project code (i.e. we don’t need your screen captures or collage images).

- All functions in these files should have comments.

3 Proj 5

Because this is a Pixmap-manipulation project, we organize our break-down around tasks involving Pixmaps.

Let’s use as our example testbuildcollage.py – a program that puts two images into a collage:
1. Build the collage list
   
   (a) Build the initial list (i.e. use 0’s for the offsets and None’s for the images)
   
   (b) Read in the images

2. Build the collage Pixmap
   
   (a) Determine how large the collage must be
   
   (b) Create an empty Pixmap to contain the collage
   
   (c) For each image in the collage list:
       
       i. Apply the appropriate filter
       
       ii. Copy the image into the collage

3. Save the collage

Many of these tasks are accomplished by functions. Others are short enough that they don’t warrant a separate function.

3.1 Building Collages

Let’s go over the design of buildCollage.

1. It assumes that the input will be formatted in a certain way. This is just the way we do things in Python. It is a good idea to write a comment at the top of the function indicating the expected format and meaning of the input. That way, when you return to your code after a few days, you will have a nice reminder about how to use the function. Also, when I am grading it, I will know what the function is intended to do.

2. It determines how large the background needs to be, and makes a blank background of that size.

3. It takes in a list of lists, loops through them, and uses the information to place each image in the right place with the right effect.

4. It returns the new collage. Why is it important to return the collage?
We wrote the code together in class, but I am not posting it because it is part of your project.