CS116 – List indexing - Part 1

Purpose: If we are processing corresponding items in multiple lists then we are unable to iterate through the lists using the schema that we have been using. Instead we have a mechanism for referring to list items by their position which we call the *index*.

Skills: After completion of this module you should be able to

1. Refer to a list item by using an index

Knowledge: This module will help you become familiar with the following content knowledge:

1. processing corresponding elements in multiple lists

Schema:

```
for index in range(0,length of lists):
      process list1[index]
      ...
      process listn[index]
```

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 30 minutes.

1. In the console create the list `x = [3,4,5]`. Then try the following in the console:

   ```
   x[0]
   x[2]
   i = 1
   x[i+1]
   ```

   What does the expression `mylist[index]` mean?

2. Look at the following Picture method which copies the Picture object and returns the copy.

   ```
   def copy(self):
       result = Picture(self.getWidth(),self.getHeight())
       pixels1 = self.getPixels()
       pixels2 = result.getPixels()
       for i in range(0,len(pixels1)):
           color = pixels1[i].getColor()
           pixels2[i].setColor(color)
       return result
   ```

What are the two lists that are being processed? Why are we processing those lists?

What is the index and what range of values are we using for the index? Why do we want the index to take those values?

What are we doing with pixels1[i] and pixels2[i]?

How does make the Picture `result` become a copy of the original Picture `self`?

3. What if we can "hear" a picture? Well we can! The following Picture method converts a Picture into a Sound:

```
def toSound(self):
     pixels = self.getPixels()
     sound = Sound(len(pixels))
     samples = sound.getSamples()
     for i in range(0,len(pixels)):
         if pixels[i].getRed() > 200:
             samples[i].setValue(1000)
         elif pixels[i].getBlue() > 200:
             samples[i].setValue(-1000)
         else:
             samples[i].setValue(0)
     return sound
```

What are the two lists that are being processed?

What range of values are we using for the index? Why do we want the index to take those values?

What are we doing with the corresponding elements of the two lists?

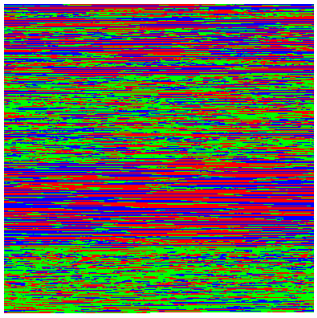
Complete each of the following assignments to be submitted for grading. Each should be done individually but you can consult with a classmate to discuss your strategies or if you get an error message that you do not understand.
For list indexing problems it will help if you ask yourself the following questions:

1. What are the multiple lists that are being processed?
2. What range is needed for the index?
3. How should corresponding elements in the lists be processed?

Assignment 1:
We want to "see" a sound by turning a Sound into a Picture. For a Sound of length n we will create a Picture with roughly n pixels by making
`picture = Picture(int(math.sqrt(length))+1,int(math.sqrt(length))+1)`

Assign corresponding items in the two lists as follows:
Pixels will be red for sound samples greater than 1000.
Pixels will be blue for sound samples less than -1000
All other pixels will be green.

Criteria of Success: Here is the picture for the ducks-thames.wav sound:



Assignment 2:
Write a Picture method authenticate(self,anotherPic) which checks to see if the two pictures are identical. This method should return a new picture which is the highlighting the differences in the pictures. For each pixel position you will need to compare if the colors of the two pictures are the same. If they are the same then that pixel in the resulting picture should be white. If they are different then that pixel in the resulting picture should be the same color as the original picture.

Criteria of Success: Here are the differences of `barbaraOrig.png` and `barbaraManip.png`:

Assignment 3:
We want to generate a sequence of pictures which makes it appear that the starting picture is morphing in the final picture. In other words each subsequent picture will systematically appear more and more like the final picture.

Suppose that you want to generate n intermediate pictures. Consider the k^{th} picture in the sequence. We can compute the pixel color of each pixel by computing a weighted average of each of the red, green, and blue values. Suppose that *startRed* and *endRed* are the red values at corresponding pixels in the original and ending pictures. We would compute the red value of the corresponding pixel in the k^{th} picture as follows:

$$newRed = startRed + ((endRed - startRed)/n) * k$$

We would need to do the same for the green and blue values as well.

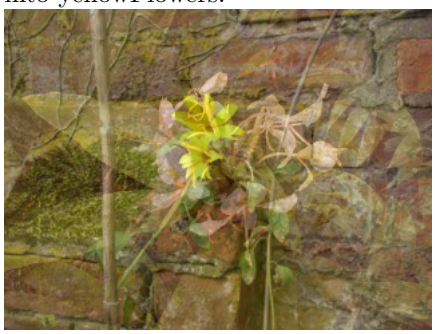
Write a method `kthMorph(self,end,n,k)` that returns the k^{th} picture in a sequence of n pictures.

The subgoals are:

1. Create a new empty picture which is the same size as your given pictures.
2. Get the pixel lists of all three of your pictures.
3. For each of the set of three corresponding pixels, compute the weighted values of the red, green, and blue components.
4. Set the new color of your resulting picture.
5. Return your resulting picture after all the pixel colors have been set.

You will need to test this with start and end pictures which are exactly the same size such as the whiteFlower.jpg, yellowFlowers.jpg

Criteria of Success: Your method should work for any positive n and any positive k value less than n. Here is the third picture out of five morphing whiteFlower into yellowFlowers:



Assignment 4:

Write a method `morphTo(self,end,n)` which displays the n pictures which morphs the original to ending pictures. Since we are not using multiple lists here, you will want to use `kthMorph` and the schema to loop through a list to accomplish this method.

Criteria for Success: Your function should display n pictures using `kthMorph` where each successive picture morphs the original picture more and more into the ending picture.

Submit your python files with your methods in Canvas for grading.