Plan for October 22

● Images, tuples, RGB color model
  - Image processing by understanding API
  - Image processing with tuples and generators
  - Image processing with pixels and filters

● Review problem-solving with sets, list comprehensions, and thinking
  - Toward reveling in APT quests and adventures
Near-term Administrivia and Due Dates

- **Midterm regrade:**
  - Review rubric, ask Prof in your section
- **Mastery APTs for mid-term catchup**
  - October 23 and October 30
- **Programming Assignments: Four left**
  - 10/29, 11/5, 11/19, 12/3
- **APTs and APT Quizzes**
  - Quizzes: 11/2, 11/16, 11/30 (moved by one week)
- **Midterm exam and final**
  - November 12, December 9 and 13
Image Processing

● What's real, what's Photoshopped
  - Learn more at [http://bit.ly/1Psi0hG](http://bit.ly/1Psi0hG), we'll do very basic stuff in class and lab, next assignment too!
Example: convert color to gray scale

Process each pixel
Convert to gray
Let's look at GrayScale.py

● Key features we see
  Ø Import Image library, use API by example
  Ø Image.open creates an image object

● Image functions for Image object im
  Ø im.show(), displays image on screen
  Ø im.save("xy"), saves with filename
  Ø im.copy(), returns image that's a copy
  Ø im.load(), [x,y] indexable pixel collection
  Ø im.getdata(), iterable pixel generator

● Let's look at two ways to process pixels!
Image Library: open, modify, save

- **Image.open** can open most image files
  - .png, .jpg, .gif, and more
  - Returns an image object, so store in variable of type Image instance
  - Get pixels with `im.getdata()` or `im.load()`

- **Image.new** can create a new image, specify color model "RGB" and size of image
  - Add pixels with `im.putdata()`

- **These belong to Image package**
im.getdata(), accessing pixels

● **Returns something *like* a list**
  - Use: `for pix in im.getdata():`
  - Generates pixels on-the-fly, can't slice or index unless you use `list(im.getdata())`
  - Structure is called a Python generator!
  - Saves on storing all pixels in memory if only accessed one-at-a-time

● **See usage in GrayScale.py, note how used in list comprehension, like a list!**
Generator: Tuples and Pixels

● The `im.getdata()` function returns list-like iterable

  ➢ Can use in list comprehension, see code
  ➢ Use `.putdata()` to store again in image

```python
pixels = [makeGray(pix) for pix in im.getdata()]

def makeGray(pixel):
    r, g, b = pixel
    gray = (r+g+b)/3
    return (gray, gray, gray)
```
Making Tuples and Generators

- **Overuse and abuse of parentheses**
  - To create a tuple, use parentheses
    
    ```python
    for pix in im.getdata():
        (r,g,b) = pix
        npx = (255-r,255-g,255-b)
    ```
  - To create a generator use parentheses as though creating a list comprehension!
    
    ```python
    [2*n for n in range(10000)]
    (2*n for n in range(10000))
    ```

- **See this in PyDev console**
Questions about Image Code

Example: convert blue to green

Process each pixel
Convert blue ones to green

Is this like red-eye removal?
Making blue things green

● How do we identify blue pixels?
  ➢ In the blue devil image it's easy, not white
  ➢ (R,G,B) triple for white? (255,255,255). So not white?

● Not sure about B value, but perhaps R value is low, at least lower than 255
  ➢ Let's try changing based on R < 200
  ➢ See Colorme.py
  ➢ http://www.rapidtables.com/web/color/RGB_Color.htm

● Tuples are immutable, so examine tuple
  ➢ Return a new tuple, function makeGreen
Code in GrayScale.py and Colorme.py

- **Very similar! Loop over pixels, change each**
  - Capture the similarity by parameterizing what changes, common way to solve-problems
  - Pass in the function makeGray or makeGreen
  - This is what happens in lab this week! See ColorAnyway.py

- **What's a green-screen technique?**
im.load(), accessing pixels

- Returns something that can be indexed \([x,y]\)
  - Only useful for accessing pixels by \(x,y\) coords
- Object returned by `im.load()` is ...
  - Use `pix[x,y]` to read and write pixel values
- Note: this is NOT a generator

```
pix = im.load()
tup = pix[0,0]
pix[1,1] = (255,255,0)
```
Set, List, Join, and APT Review

● Sets don't contain duplicates
  ➢ Simple to create from a list, .add for more
  ➢ Not accessible by index, can iterate over elts
  ➢ Very, very fast: \( x \ in \ SET \), compare list

● Look at WordCompositionGame APT
  ➢ How to think about solving this?
Can you solve this with paper/pencil?

- Conceptually, in words, how to find words worth 3 points for listA player?
  - Describe how you determine this (English, not Python)
  - What about three points for player listB, listC?

- How do you find words that score 2?
  - Can you express in terms of set operations? Like the previous example?
APT WordCompositionGame

- Using sets and set operations can help
  - Set intersection and set union
  - Other set operations
- A | B, set union, A & B, set intersection
- B – A, set difference, B ^ A, symmetric diff
Answer Questions

Code smells you start understanding

● If you wrote code to score for player listA
  ➢ How to use code for player listB and listC?
  ➢ Would the code fragments be similar?

● Capture differences via parameters when there's lots of duplication in code
  ➢ See the example in GrayScale and Colorme
APT AnagramFree

How do you know "spear" and "pears" are anagrams?

- Sort the words and see if sorted form the same
- What is returned by sorted("spear")?
- What type is ''.join(sorted("spear"))
- Can we use '' or ' ' or ':' or '|'?

How do you know whether there are many words that are anagrams? Can sets help?