Plan for October 19-23

- **Review Catchup and Midterm and Future**
  - ➢ **Make sure everyone understand options**

- **Review Assignment 5, Word Games**
  - ➢ **APIs, Global Variables, Interactive Games**

- **Images, tuples, RGB color model**
  - ➢ **Ready for lab, next assignment, and next set of APTs**

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**

Jumble Review from Last Week

http://www.jumble.com

Use this problem to think about word games

● **Human approach**
  ➢ **What do you do?**

● **Computational method?**
  ➢ **Cheating or insight?**

JUMBLE

Unscramble these four Jumbles,
one letter to each square,
to form four ordinary words.

©2010 Tribune Media Services, Inc.
All Rights Reserved.

SCAMK

CARPH

TECTAL

SLIRGY

THAT SCRAMBLED WORD GAME
by Mike Argirion and Jeff Knurek

C'mon seven

WHEN HE JOINED
THE DICE GAME, HIS
CHANCES WERE ---

Now arrange the circled letters
to form the surprise answer, as
suggested by the above cartoon.

Print your answer here: " ◯◯◯◯◯ "

(Answers tomorrow)

Yesterday's | Jumbles: SKUNK   TWEAK   GAMBOL   ADJOIN
Answer: The plastic surgeon sought the clerk's help
because she had a — "KNOWS" JOB

Review Jumble Programming Concepts

- **When you run the program it starts in __main__, see Jumble.py for details**
  - ➢ **This is how Python works, boilerplate code**
  - ➢ **Global variables accessed in this section**

- **What's the variable *words* at beginning?**
  - ➢ **Global variable. Accessible in *every function* in the module (global required for modifying)**
  - ➢ **Used sparingly often useful in a small module**
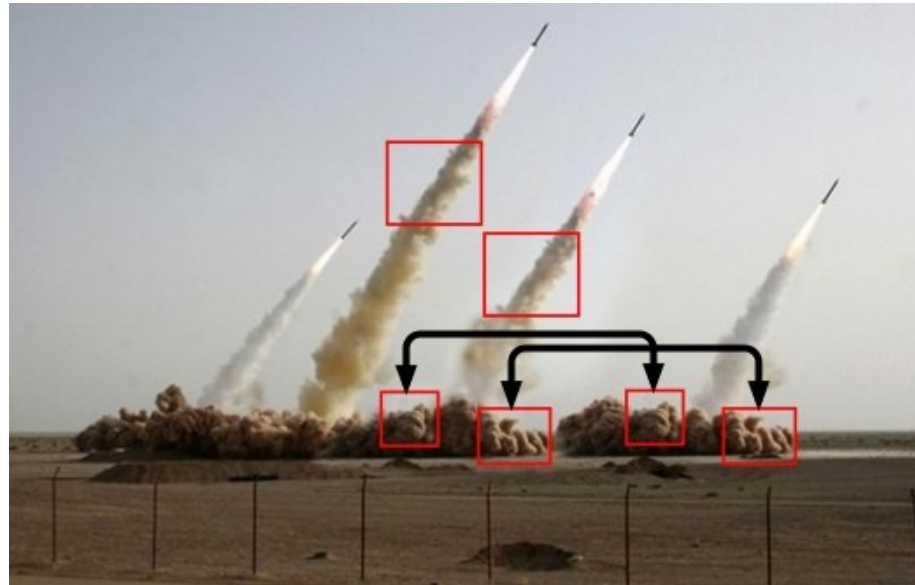 - ➢ **Abused, can lead to hard to maintain code**

Questions About Assignment 5

http://bit.ly/101fall15-oct20-1

After this: image processing

Image Processing

- ## *What's real, what's Photoshopped*
  - ➢ http://bit.ly/1Kj0Kn6 from 2008
  - ➢ Learn more at 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

Example: convert blue to green

Process each pixel
Convert blue ones to green

Is this like red-eye removal?

Need new concepts and Image library

- **Red, Green, Blue color model**
  - ➤ **Triples of (R,G,B) are processed as Python tuples.**
  - ➤ *Let's study tuples!*

- **Images can be very big, what's 4K display?**
  - ➤ **4,096 x 2,160 = 8,847,360 pixels, 8Mb at least**
  - ➤ **Creating huge lists takes up memory**
  - ➤ **Sometimes only need one pixel at-a-time**
  - ➤ *Let's study generators!*

Need new concepts and Image library

- **Red, Green, Blue color model**
  - Additive model, each pixel specified by (r,g,b) triple, values of each between 0-255
  - https://en.wikipedia.org/wiki/RGB_color_model
  - White is (255,255,255) and Black is (0,0,0)
- **Images stored as sequence of (r,g,b) tuples, typically with more data/information too**
  - 256 values, represented as 8 bits, $2^8 = 256$
  - 32 bits per pixel (with alpha channel)
  - In Python we can largely ignore these details!

Image library: Two ways to get pixels

- **Each pixel is a *tuple* in both models**
  - ➤ **Like a list, indexable, but *immutable***
  - ➤ `pix = (255,0,0)`
    - What is `pix`?, `pix[0]`? What is `pix[5]`?
- **Invert a pixel: by subscript or named tuple**
  - ➤ **Access by assignment to variables!**

```
npx = (255-pix[0],255-pix[1],255-pix[2])
```

```
(r,g,b) = pix
npx = (255-r,255-g,255-b)
```

Let's look at GrayScale.py

- **Key features we see**
  - ➤ **Import Image library, use API by example**
  - ➤ **Image.open creates an image object**
- **Image functions for Image object im**
  - ➤ `im.show()`, displays image on screen
  - ➤ `im.save("xy")`, saves with filename
  - ➤ `im.copy()`, returns image that's a copy
  - ➤ `im.load()`, [x,y] indexable pixel collection
  - ➤ `im.getdata()`, iterable pixel collection
- **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!**

Alternate : Still 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**

```
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**

    ```
    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!**

    ```
    [2*n for n in range(10000)]
    (2*n for n in range(10000))
    ```

- **See this in PyDev console**

Questions about Image Code

http://bit.ly/101fall15-oct20-2

`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)
```