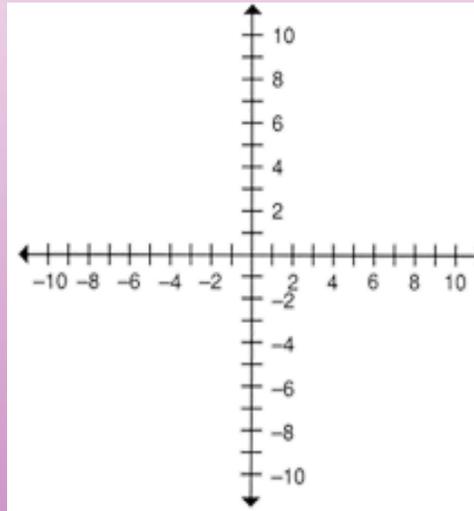


Multimedia

Ohhhhh yeahhhhh!!

The Coordinate Plane

- Have you ever used a coordinate plane in math?
- The type of coordinate planes that you're used to look like this:



- In the above picture, $(0,0)$ is in the center of the coordinate plane
- The coordinate planes used in Python have $(0,0)$ at the top left corner

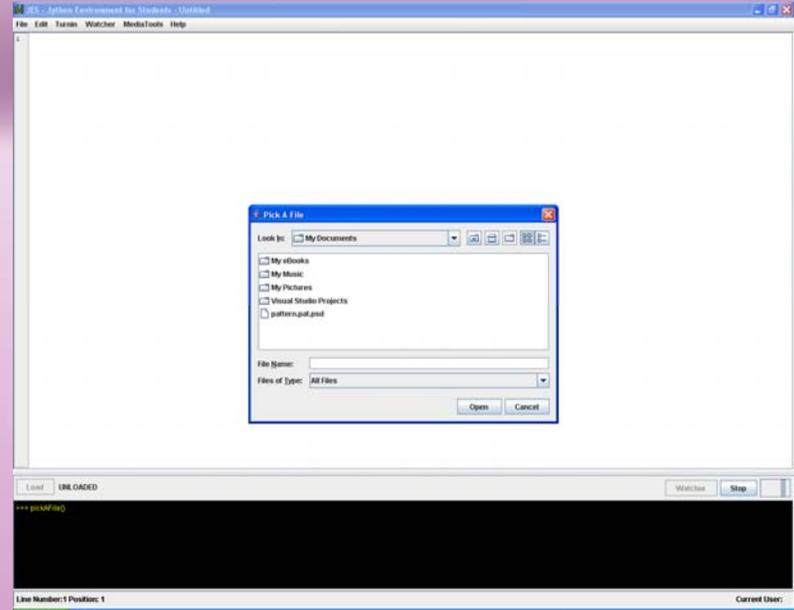
The Coordinate Plane and Images

- The coordinate plane is like a grid
- If there is an image on the coordinate plane, each of the items in the grid is a pixel (picture element)
- Each of the pixels is made up of a combination of three colors (Red, Green and Blue)
- Each of the three colors has a value between 0 and 255
- For example, if a pixel has the color black, the value for each of the colors is:

R: 0 G: 0 B: 0

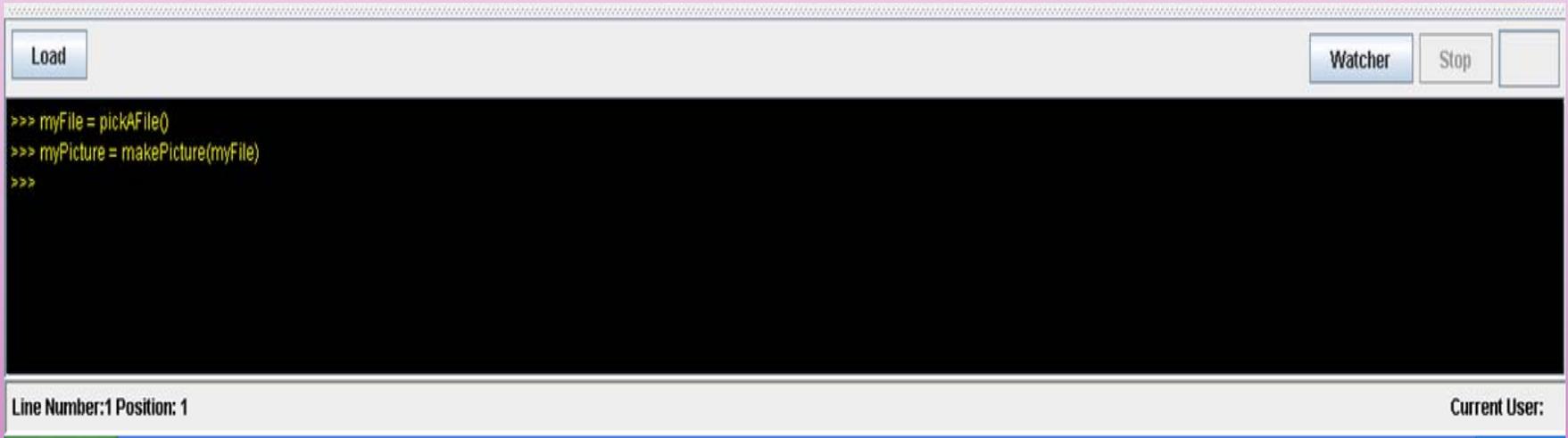
pickAFile()

- In order to pick a file to store as an image, you use the predefined method, pickAFile().
- If you just call the method, a screen pops up allowing you to browse your folders so that you can select a file
- However, if you would actually like to do something with your file, it is recommended that you store it as a variable:



makePicture()

- After you have picked your file and stored it, if your file is a picture, you have to actually make a picture using the file and pass in the file as a parameter into makePicture():



The screenshot shows a code editor window with a light green header bar containing three buttons: "Load", "Watcher", and "Stop". The main area is a black console with yellow text showing the following commands:

```
>>> myFile = pickAFile()
>>> myPicture = makePicture(myFile)
>>>
```

At the bottom of the window, there is a status bar with "Line Number:1 Position: 1" on the left and "Current User:" on the right.

- As you can see above, the picture version of the file has also been stored as a variable

show()

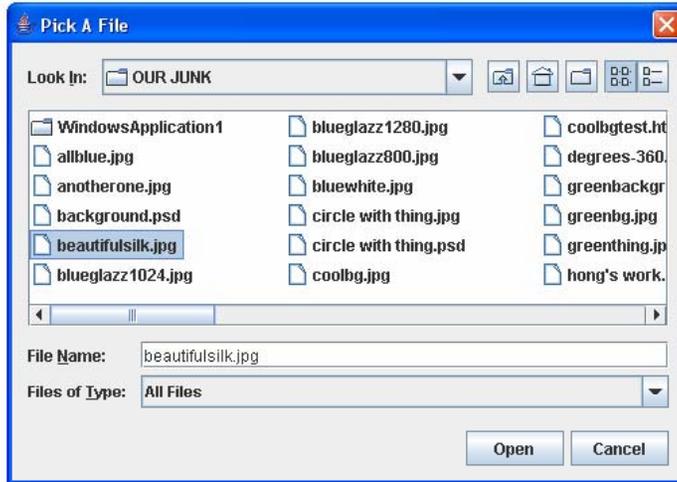
- To show a picture, call the method, show() passing in the picture as a parameter:



Putting it all together

- Instead of picking a file, converting it into a picture and then showing it in the black screen over and over again, every time that you want to show a picture, you can write a method that will do everything at once and you can call that method from the black screen (don't forget to save the method and then import it!).

```
1 def pickAndShow():
2     myfile = pickAFile()
3     mypict = makePicture(myfile)
4     show(mypict)
```



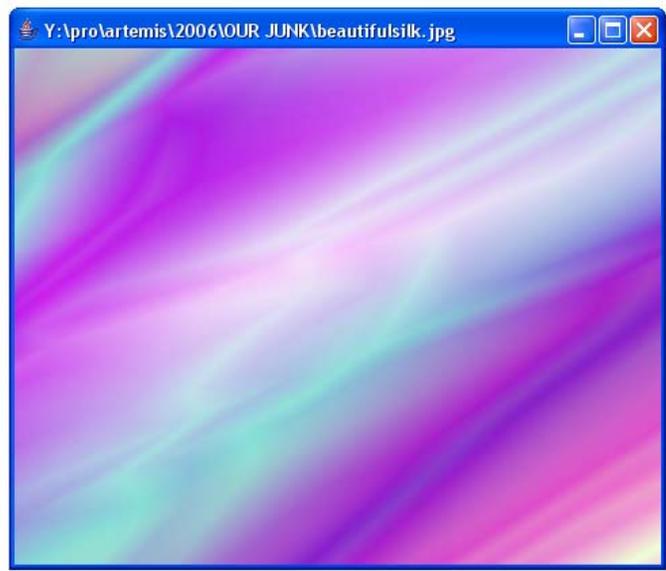
Load

Watcher

Stop

>>> pickAndShow()

```
1 def pickAndShow():  
2     myfile = pickAFile()  
3     mypict = makePicture(myfile)  
4     show(mypict)
```



Load

Watcher

Stop

```
>>> pickAndShow()  
>>>
```

Changing Colors

- As mentioned before, each image is made up of pixels
- Each pixel stores a value for the colors Red, Blue and Green
- What if you wish to change one of the values of the colors?
- That would change the color of the pixel.

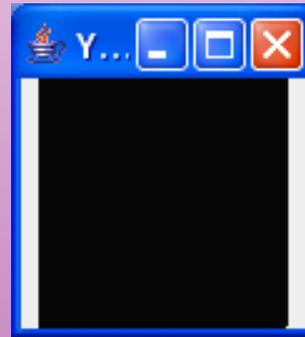
Changing Colors: One by One

- In order to change the color of one pixel, first you have to know which pixel exactly you want to change (the pixel's coordinates), you need to be able to get that actual pixel so that you can make the change, and then you need to set its new color

ORIGINAL PICTURE:



RESULT:

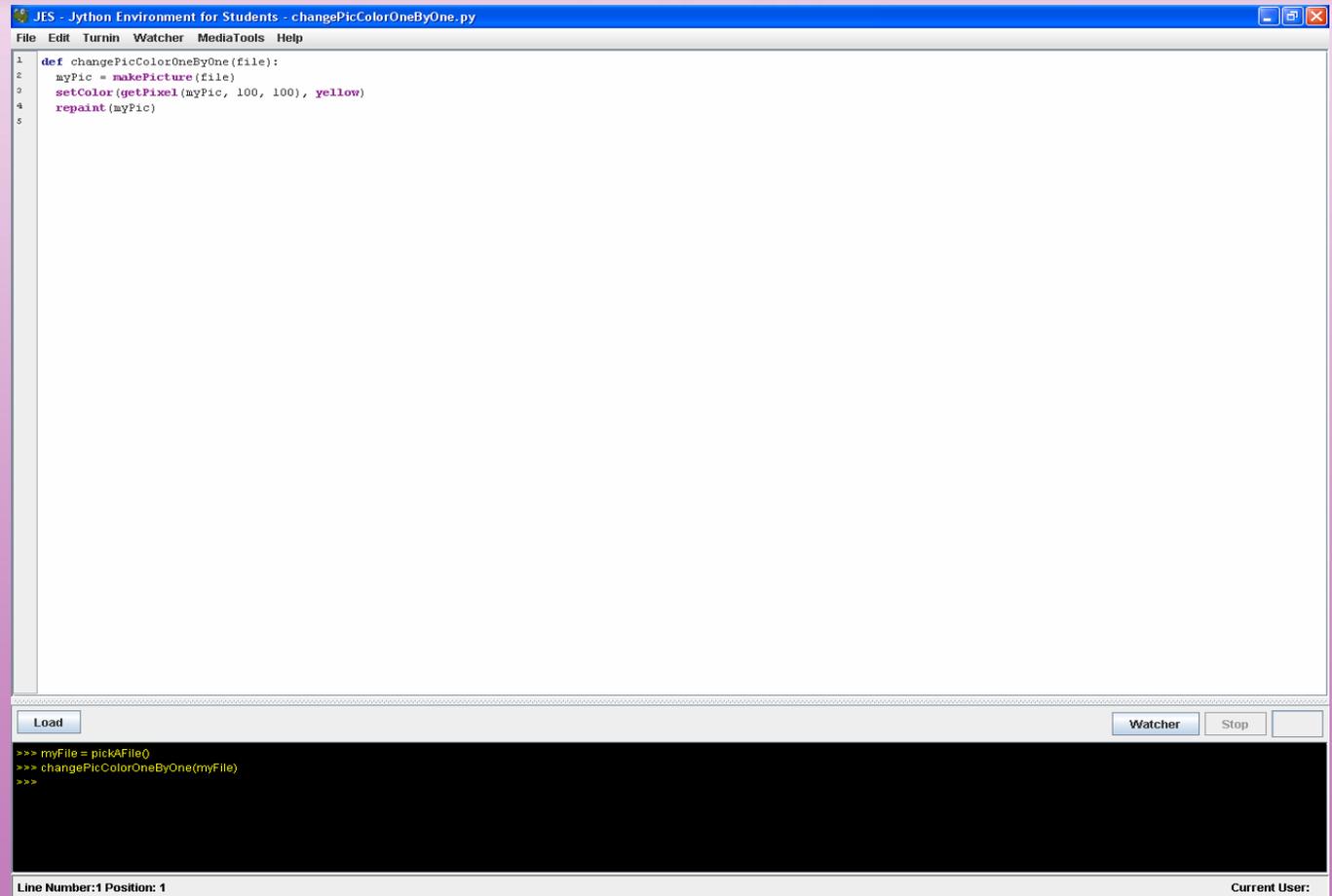


Well of course you can't see the change. Why? Because you just changed one pixel! Changing one pixel won't make a difference because pixels are so small.

How'd ya do it?

- To change one pixel...

1. Write the method to change one pixel (shown to the right)
2. Save it
3. Load it
4. Pick a file and pass it in as a parameter!



The screenshot shows a window titled "JyES - Jython Environment for Students - changePicColorOneByOne.py". The window contains a Python script with the following code:

```
1 def changePicColorOneByOne(file):
2   myPic = makePicture(file)
3   setColor(getPixel(myPic, 100, 100), yellow)
4   repaint(myPic)
5
```

Below the code editor, there is a "Load" button and a "Watcher" button. The execution output shows the following commands and results:

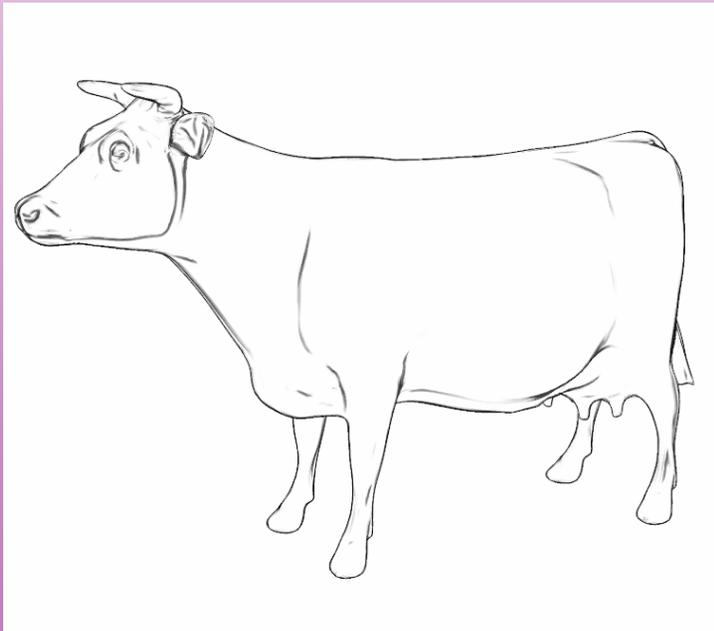
```
>>> myFile = pickAFile()
>>> changePicColorOneByOne(myFile)
>>>
```

At the bottom of the window, it displays "Line Number: 1 Position: 1" and "Current User:".

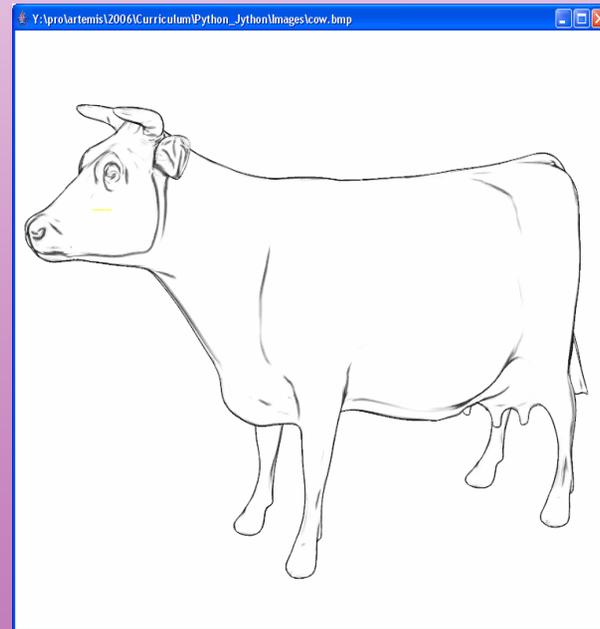
Changing Colors: A Complete Row

- If you want to see results, change more than one pixel. Change a whole row!

ORIGINAL PICTURE:



RESULT:



How'd ya do it?

The screenshot displays the JES application window titled "JES - Jython Environment for Students - changePicColorOneByOne.py". The window features a menu bar with "File", "Edit", "Turnin", "Watcher", "MediaTools", and "Help". The main area contains a Python script with the following code:

```
1 def changePicColorOneByOne(file):
2     myPic = makePicture(file)
3     setColor(getPixel(myPic, 100, 200), yellow)
4     setColor(getPixel(myPic, 101, 200), yellow)
5     setColor(getPixel(myPic, 102, 200), yellow)
6     setColor(getPixel(myPic, 103, 200), yellow)
7     setColor(getPixel(myPic, 104, 200), yellow)
8     setColor(getPixel(myPic, 105, 200), yellow)
9     setColor(getPixel(myPic, 106, 200), yellow)
10    setColor(getPixel(myPic, 107, 200), yellow)
11    setColor(getPixel(myPic, 108, 200), yellow)
12    setColor(getPixel(myPic, 109, 200), yellow)
13    setColor(getPixel(myPic, 110, 200), yellow)
14    setColor(getPixel(myPic, 111, 200), yellow)
15    setColor(getPixel(myPic, 112, 200), yellow)
16    setColor(getPixel(myPic, 113, 200), yellow)
17    setColor(getPixel(myPic, 114, 200), yellow)
18    setColor(getPixel(myPic, 115, 200), yellow)
19    setColor(getPixel(myPic, 116, 200), yellow)
20    setColor(getPixel(myPic, 117, 200), yellow)
21    setColor(getPixel(myPic, 118, 200), yellow)
22    setColor(getPixel(myPic, 119, 200), yellow)
23    setColor(getPixel(myPic, 120, 200), yellow)
24    setColor(getPixel(myPic, 121, 200), yellow)
25    setColor(getPixel(myPic, 122, 200), yellow)
26    setColor(getPixel(myPic, 123, 200), yellow)
27    repaint(myPic)
28
```

Below the code editor, there are three buttons: "Load", "Watcher", and "Stop". The "Load" button is currently selected. The bottom section of the window is a black console area with the following text:

```
>>> myFile = pickAFile()
>>> changePicColorOneByOne(myFile)
>>> |
```

Annoying?

- How tedious is it to have to change *each and every single* pixel
- How can we write one line and change a whole row?
- Use a for loop!

Now *this* is better!

The image shows a screenshot of the Jython Environment for Students (JES) interface. The window title is "JES - Jython Environment for Students - addStripe.py". The menu bar includes "File", "Edit", "Turnin", "Watcher", "MediaTools", and "Help". The main text area contains the following Python code:

```
1 def addStripe(file):
2     myPic = makePicture(file)
3     for x in range(100, 123):
4         setColor(getPixel(myPic, x, 200), yellow)
5     repaint(myPic)
6
```

Below the code editor, there are three buttons: "Load", "Watcher", and "Stop". The "Load" button is highlighted. The console area at the bottom shows the following execution output:

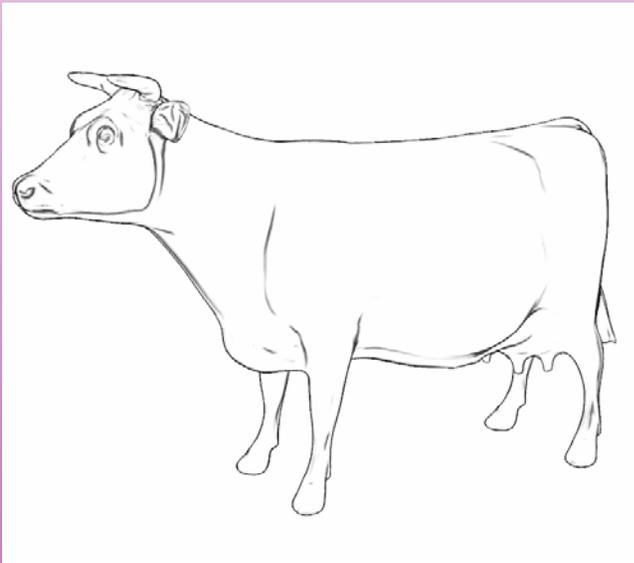
```
>>> myFile = pickAFile()
>>> addStripe(myFile)
>>>
```

At the bottom of the window, the status bar displays "Line Number: 1 Position: 1" on the left and "Current User:" on the right.

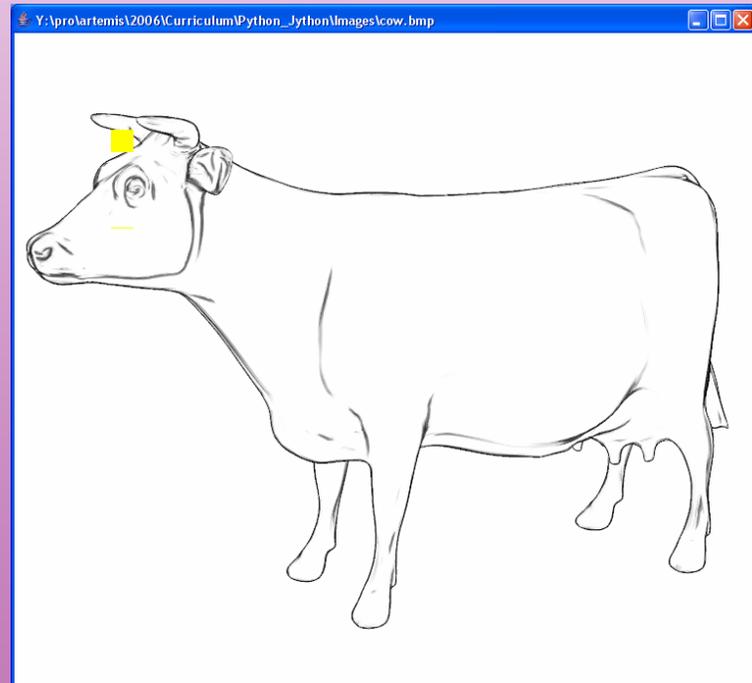
Changing Colors: A Block

- What if instead of a line, you want to change a whole block of the picture?

ORIGINAL PICTURE



RESULT



How'd ya do it?

The screenshot displays the JES application window titled "JES - Jython Environment for Students - yellowBox.py". The window has a menu bar with "File", "Edit", "Turnin", "Watcher", "MediaTools", and "Help". The main area contains a Python script with the following code:

```
1 def yellowBox(file):
2     myPic = makePicture(file)
3     for x in range(100, 123):
4         setColor(getPixel(myPic, x, 200), yellow)
5         for y in range(100, 123):
6             setColor(getPixel(myPic, x, y), yellow)
7     repaint(myPic)
8
```

Below the code editor, there are three buttons: "Load", "Watcher", and "Stop". The "Watcher" button is currently active. The bottom section of the window shows a black console area with the following input and output:

```
>>> myFile = pickAFile()
>>> yellowBox(myFile)
```

At the bottom left, the status bar indicates "Line Number: 1 Position: 14". At the bottom right, it says "Current User:".

Altering Colors Directly

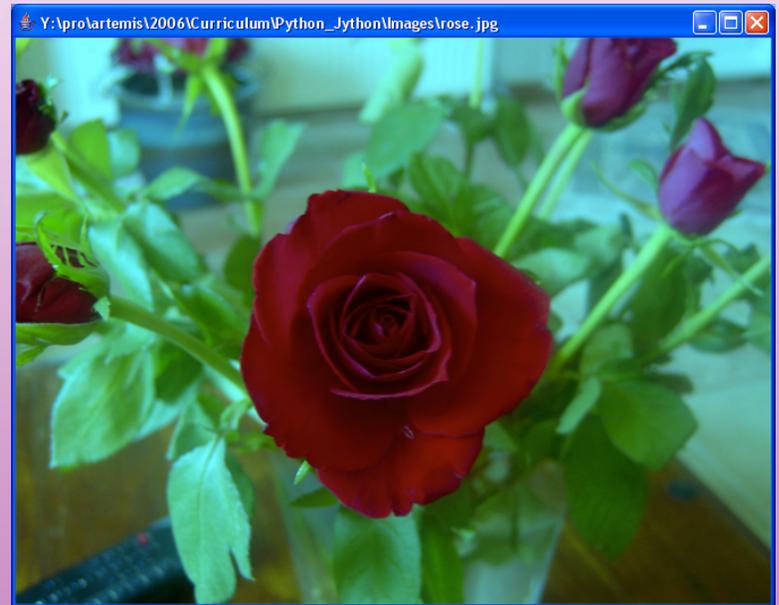
- Notice how changing colors just changes a color to a whole new color
- What if you want to slightly change it?
- You would alter the color by altering each pixel
- You would have to alter each of the RGB values
- Let's say you want to decrease the red in your image
- You would have to loop through each of the pixels in your image and change each red value to a lower amount
- Let's say you want to decrease it by half

What it looks like...

ORIGINAL PICTURE



RESULT



How'd ya do it?

The screenshot shows the JES (Jython Environment for Students) interface. The title bar reads "JES - Jython Environment for Students - decreaseRed.py". The menu bar includes "File", "Edit", "Turnin", "Watcher", "MediaTools", and "Help". The main editor area contains the following Python code:

```
1 def decreaseRed(file):  
2     myPic = makePicture(file)  
3     for p in getPixels(myPic):  
4         value = getRed(p)  
5         setRed(p, value*0.5)  
6     repaint(myPic)
```

Below the editor, there are three buttons: "Load", "Watcher", and "Stop". The console area at the bottom shows the following execution commands:

```
>>> myFile = pickAFile()  
>>> decreaseRed(myFile)
```

The status bar at the bottom left indicates "Line Number:1 Position: 1" and the bottom right indicates "Current User:".

Altering Colors Indirectly

- Let's say you wanted to darken all the reds
- You would increase each of the values of red in the entire picture by a certain amount
- Let's say this amount is 20
- You would have to loop through each and every single pixel and do this

Altering Colors Indirectly

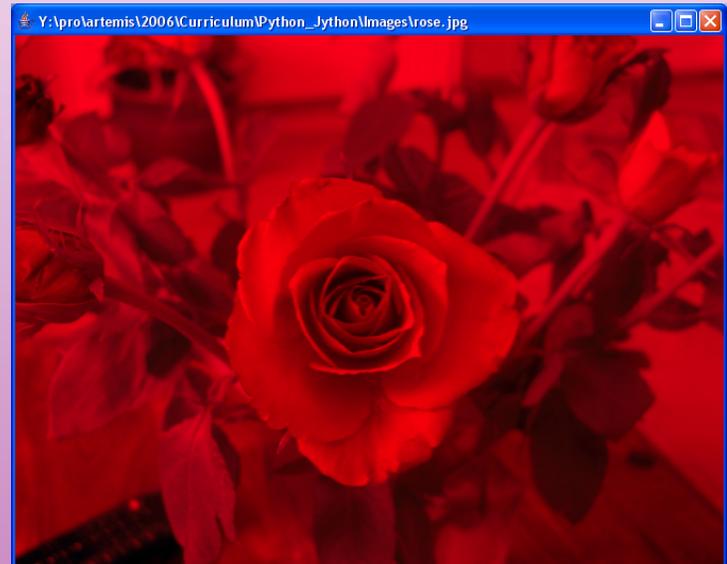
- What if one pixel has an extremely high value of red? Such as 240?
- If you darken it by 20, it would become 260
- But, remember that the values can only be between 0 and 255
- Instead of the value becoming 260, it would wrap around back to 0 and become 5
- How do we solve this problem?
- Instead of darkening the value of the red by increasing it, lighten the values of the green and blue by decreasing them
- This gives the appearance that the amount of red in each pixel was increased!

What it looks like...

ORIGINAL PICTURE



RESULT



How'd ya do it?

The image shows a screenshot of the Jython Environment for Students (JES) interface. The window title is "JES - Jython Environment for Students - increaseRed.py". The menu bar includes "File", "Edit", "Turnin", "Watcher", "MediaTools", and "Help".

The main editor area contains the following Python code:

```
1 def increaseRed(picture):
2     for p in getPixels(picture):
3         value=getBlue(p)
4         setBlue(p,value*.05)
5         value2=getGreen(p)
6         setGreen(p,value2*.05)
7     repaint(picture)
```

Below the editor, there is a "Load" button and a "Watcher" button. The "Stop" button is also present but disabled. The console area shows the following execution commands:

```
>>> myFile = pickAFile()
>>> myPic = makePicture(myFile)
>>> increaseRed(myPic)
```

At the bottom left, the status bar displays "Line Number:6 Position: 1". At the bottom right, the status bar displays "Current User:".

Grayscale

- Grayscale involves changing an image from color to black and white (remember how we did this in Photoshop?)
- As you might remember from Andy's lecture, when each of the RGB values of a pixel are equal, you get the color...
- GRAY!
- We use this important fact when changing an image to grayscale

Say what?

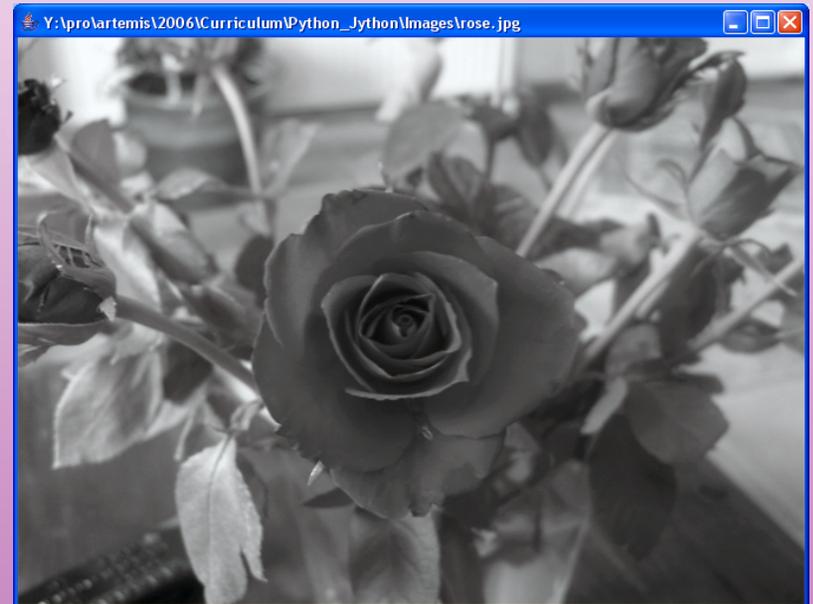
- To change an image to grayscale, take each of the RGB values and find the average
- Reset each of the RGB values to the average of the original values
- Now that they're the same value, you get a shade of gray!

What it looks like...

ORIGINAL PICTURE



RESULT



How'd ya do it?

The screenshot displays the JES environment with a Python script in the editor and a console window showing the execution of the script.

Editor Content:

```
1 def grayScale(file):
2     myPic = makePicture(file)
3     for p in getPixels(myPic):
4         intensity = (getRed(p)+getGreen(p)+getBlue(p))/3
5         setColor(p, makeColor(intensity, intensity, intensity))
6     repaint(myPic)
```

Console Content:

```
>>> myFile = pickAFile()
>>> grayScale(myFile)
```

Interface Elements:

- Window Title: JES - Jython Environment for Students - grayScale.py
- Menu Bar: File Edit Turnin Watcher MediaTools Help
- Buttons: Load, Watcher, Stop
- Status Bar: Line Number: 1 Position: 1, Current User:

Negatives

- To create a negative version of an image using Photoshop, the program would simply invert the colors for you
- You can do this using Jython by inverting each of the RGB values for each and every pixel
- How would you invert the RGB values (get its opposite)?
- Subtract it from 255!

What it looks like...

ORIGINAL PICTURE



RESULT



How'd ya do it?

The image shows a screenshot of the Jython Environment for Students (JES) interface. The window title is "JES - Jython Environment for Students - negative.py". The menu bar includes "File", "Edit", "Turnin", "Watcher", "MediaTools", and "Help". The main editor area contains the following Python code:

```
1 def negative(file):
2     myPic = makePicture(file)
3     for px in getPixels(myPic):
4         red = getRed(px)
5         green = getGreen(px)
6         blue = getBlue(px)
7         negColor = makeColor(255-red, 255-green, 255-blue)
8         setColor(px, negColor)
9     repaint(myPic)
```

Below the editor, there are buttons for "Load", "Watcher", and "Stop". The console area shows the following execution:

```
>>> myFile = pickAFile()
>>> negative(myFile)
```

At the bottom of the window, the status bar displays "Line Number: 1 Position: 1" on the left and "Current User:" on the right.

That's all folks!

- Now, its your turn!