Dummy PDF file

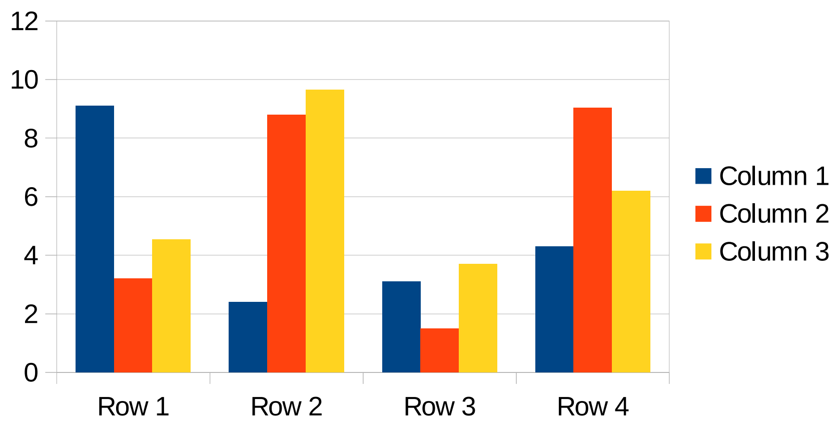
Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac faucibus odio.

Vestibulum neque massa, scelerisque sit amet ligula eu, congue molestie mi. Praesent ut varius sem. Nullam at porttitor arcu, nec lacinia nisi. Ut ac dolor vitae odio interdum condimentum. **Vivamus dapibus sodales ex, vitae malesuada ipsum cursus convallis. Maecenas sed egestas nulla, ac condimentum orci.** Mauris diam felis, vulputate ac suscipit et, iaculis non est. Curabitur semper arcu ac ligula semper, nec luctus nisl blandit. Integer lacinia ante ac libero lobortis imperdiet. *Nullam mollis convallis ipsum, ac accumsan nunc vehicula vitae.* Nulla eget justo in felis tristique fringilla. Morbi sit amet tortor quis risus auctor condimentum. Morbi in ullamcorper elit. Nulla iaculis tellus sit amet mauris tempus fringilla.

Maecenas mauris lectus, lobortis et purus mattis, blandit dictum tellus.

- **Maecenas non lorem quis tellus placerat varius.**
- *Nulla facilisi.*
- Aenean congue fringilla justo ut aliquam.
- Mauris id ex erat. Nunc vulputate neque vitae justo facilisis, non condimentum ante sagittis.
- Morbi viverra semper lorem nec molestie.
- Maecenas tincidunt est efficitur ligula euismod, sit amet ornare est vulputate.



In non mauris justo. Duis vehicula mi vel mi pretium, a viverra erat efficitur. Cras aliquam est ac eros varius, id iaculis dui auctor. Duis pretium neque ligula, et pulvinar mi placerat et. Nulla nec nunc sit amet nunc posuere vestibulum. Ut id neque eget tortor mattis tristique. Donec ante est, blandit sit amet tristique vel, lacinia pulvinar arcu. Pellentesque scelerisque fermentum erat, id posuere justo pulvinar ut. Cras id eros sed enim aliquam lobortis. Sed lobortis nisl ut eros efficitur tincidunt. Cras justo mi, porttitor quis mattis vel, ultricies ut purus. Ut facilisis et lacus eu cursus.

In eleifend velit vitae libero sollicitudin euismod. Fusce vitae vestibulum velit. Pellentesque vulputate lectus quis pellentesque commodo. Aliquam erat volutpat. Vestibulum in egestas velit. Pellentesque fermentum nisl vitae fringilla venenatis. Etiam id mauris vitae orci maximus ultricies.

Cras fringilla ipsum magna, in fringilla dui commodo a.

	Lorem ipsum	Lorem ipsum	Lorem ipsum
1	In eleifend velit vitae libero sollicitudin euismod.	Lorem	
2	Cras fringilla ipsum magna, in fringilla dui commodo a.	Ipsum	
3	Aliquam erat volutpat.	Lorem	
4	Fusce vitae vestibulum velit.	Lorem	
5	Etiam vehicula luctus fermentum.	Ipsum	

Etiam vehicula luctus fermentum. In vel metus congue, pulvinar lectus vel, fermentum dui. Maecenas ante orci, egestas ut aliquet sit amet, sagittis a magna. Aliquam ante quam, pellentesque ut dignissim quis, laoreet eget est. Aliquam erat volutpat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Ut ullamcorper justo sapien, in cursus libero viverra eget. Vivamus auctor imperdiet urna, at pulvinar leo posuere laoreet. Suspendisse neque nisl, fringilla at iaculis scelerisque, ornare vel dolor. Ut et pulvinar nunc. Pellentesque fringilla mollis efficitur. Nullam venenatis commodo imperdiet. Morbi velit neque, semper quis lorem quis, efficitur dignissim ipsum. Ut ac lorem sed turpis imperdiet eleifend sit amet id sapien.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Nunc ac faucibus odio. Vestibulum neque massa, scelerisque sit amet ligula eu, congue molestie mi. Praesent ut varius sem. Nullam at porttitor arcu, nec lacinia nisi. Ut ac dolor vitae odio interdum condimentum. Vivamus dapibus sodales ex, vitae malesuada ipsum cursus convallis. Maecenas sed egestas nulla, ac condimentum orci. Mauris diam felis, vulputate ac suscipit et, iaculis non est. Curabitur semper arcu ac ligula semper, nec luctus nisl blandit. Integer lacinia ante ac libero lobortis imperdiet. Nullam mollis convallis ipsum, ac accumsan nunc vehicula vitae. Nulla eget justo in felis tristique fringilla. Morbi sit amet tortor quis risus auctor condimentum. Morbi in ullamcorper elit. Nulla iaculis tellus sit amet mauris tempus fringilla.

Maecenas mauris lectus, lobortis et purus mattis, blandit dictum tellus.

Maecenas non lorem quis tellus placerat varius. Nulla facilisi. Aenean congue fringilla justo ut aliquam. Mauris id ex erat. Nunc vulputate neque vitae justo facilisis, non condimentum ante sagittis. Morbi viverra semper lorem nec molestie. Maecenas tincidunt est efficitur ligula euismod, sit amet ornare est vulputate.

In non mauris justo. Duis vehicula mi vel mi pretium, a viverra erat efficitur. Cras aliquam est ac eros varius, id iaculis dui auctor. Duis pretium neque ligula, et pulvinar mi placerat et. Nulla nec nunc sit amet nunc posuere vestibulum. Ut id neque eget tortor mattis tristique. Donec ante est, blandit sit amet tristique vel, lacinia pulvinar arcu. Pellentesque scelerisque fermentum erat, id posuere justo pulvinar ut. Cras id eros sed enim aliquam lobortis. Sed lobortis nisl ut eros efficitur tincidunt. Cras justo mi, porttitor quis mattis vel, ultricies ut purus. Ut facilisis et lacus eu cursus.

In eleifend velit vitae libero sollicitudin euismod.

Fusce vitae vestibulum velit. Pellentesque vulputate lectus quis pellentesque commodo. Aliquam erat volutpat. Vestibulum in egestas velit. Pellentesque fermentum nisl vitae fringilla venenatis. Etiam id mauris vitae orci maximus ultricies. Cras fringilla ipsum magna, in fringilla dui commodo a.

Etiam vehicula luctus fermentum. In vel metus congue, pulvinar lectus vel, fermentum dui. Maecenas ante orci, egestas ut aliquet sit amet, sagittis a magna. Aliquam ante quam, pellentesque ut dignissim quis, laoreet eget est. Aliquam erat volutpat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Ut ullamcorper justo sapien, in cursus libero viverra eget. Vivamus auctor imperdiet urna, at pulvinar leo posuere laoreet. Suspendisse neque nisl, fringilla at iaculis scelerisque, ornare vel dolor. Ut et pulvinar nunc. Pellentesque fringilla mollis efficitur. Nullam venenatis commodo imperdiet. Morbi velit neque, semper quis lorem quis, efficitur dignissim ipsum. Ut ac lorem sed turpis imperdiet eleifend sit amet id sapien.

python

Languages

- Solve problems using a computer, give the computer instructions.
- Remember our diaper-changing exercise?

Talk the talk

- Speak its language
  - High-level: Python, C++, Java
  - Low-level: machine language, computers can only execute these
  - High-level languages have to be processed into low-level before the computer can run them
  - But high-level languages can run on different kinds of computers and are easier for humans to write and read, so most programs are written in high-level

Translation

- How does high-level get translated into low-level?
  - Interpreters and compilers!
  - Interpreter processes the program a little bit at a time and runs it
  - Compiler translates everything before running it

What is python ?

Python is a programming language that lets you work more quickly and integrate your systems more effectively.

(in other words, magic!)

http://www.python.org/

Python Vocab

- ***Program*** – A larger file of code that may contain one or more functions.
- ***Variable*** - names that you can assign values to, allowing you to reuse them later on.

  E.g.: `x = 1` or `msg = “Hi, I’m a message!”`
- ***Comments*** – These are notes ignored by the computer. In Python, comments start with a hash mark (#) and end at the end of the line.

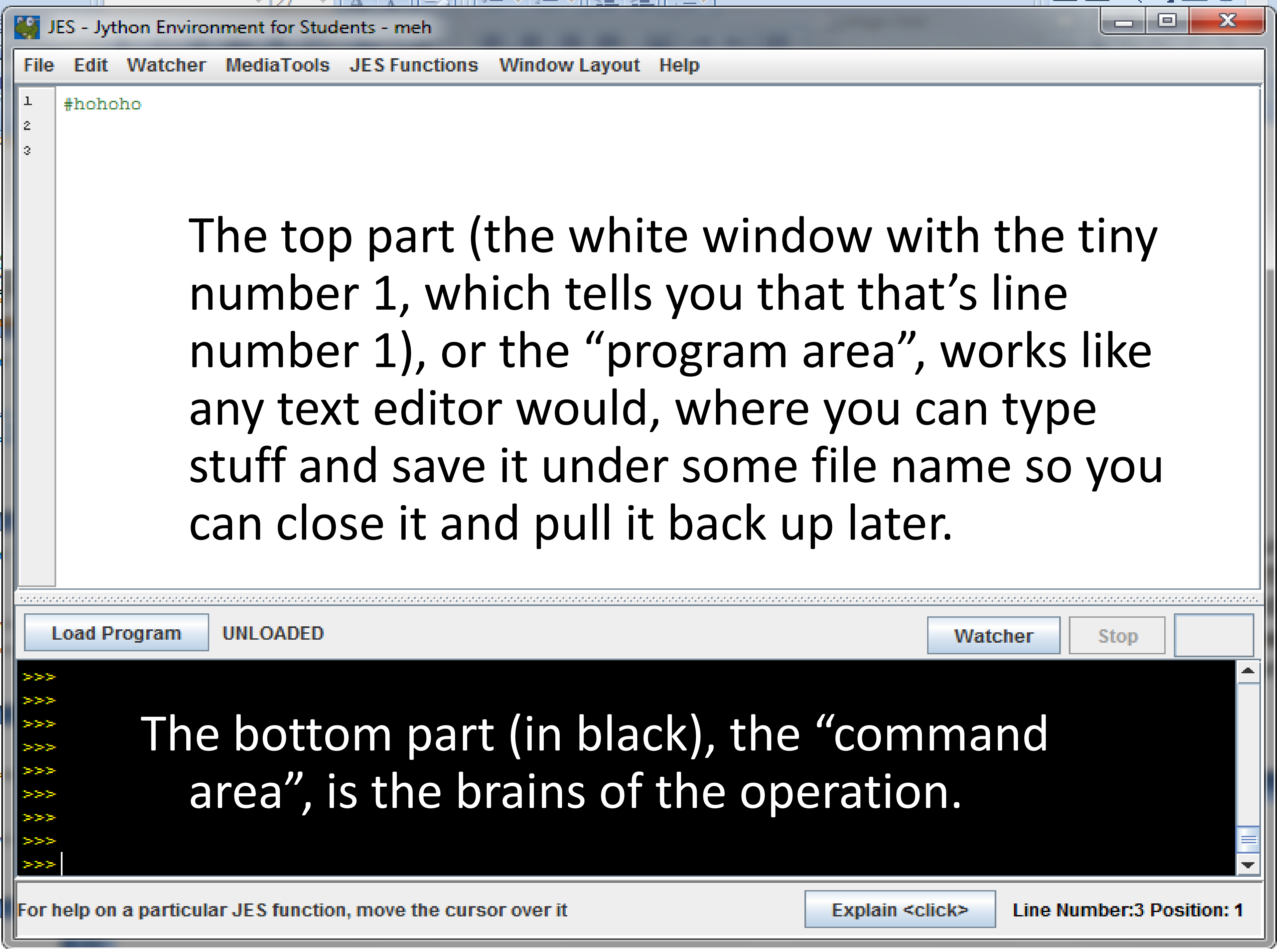
  E.g.: `>>> x + y #both variables store user input`
- ***Operators*** – Mathematical symbols, like +, -, *, and /, but also ** (for exponents).

Python Vocab

- ***Keyword*** – Words with meaning/purpose in Python. E.g. “`and`”, “`print`”, and “`if`”.
- ***Function*** – A chunk of code that performs an action. Functions have names and are reusable. Kinds: built-in ones, and “user-defined” ones.
- ***Expression*** – Statements that produce values. Examples include 3 + 5, "Hello world!“.
- ***Error*** – When your program has a problem, the command area will provide an error message.

What is JES?

Jython Environment for Students allows you to program and experiment with python.

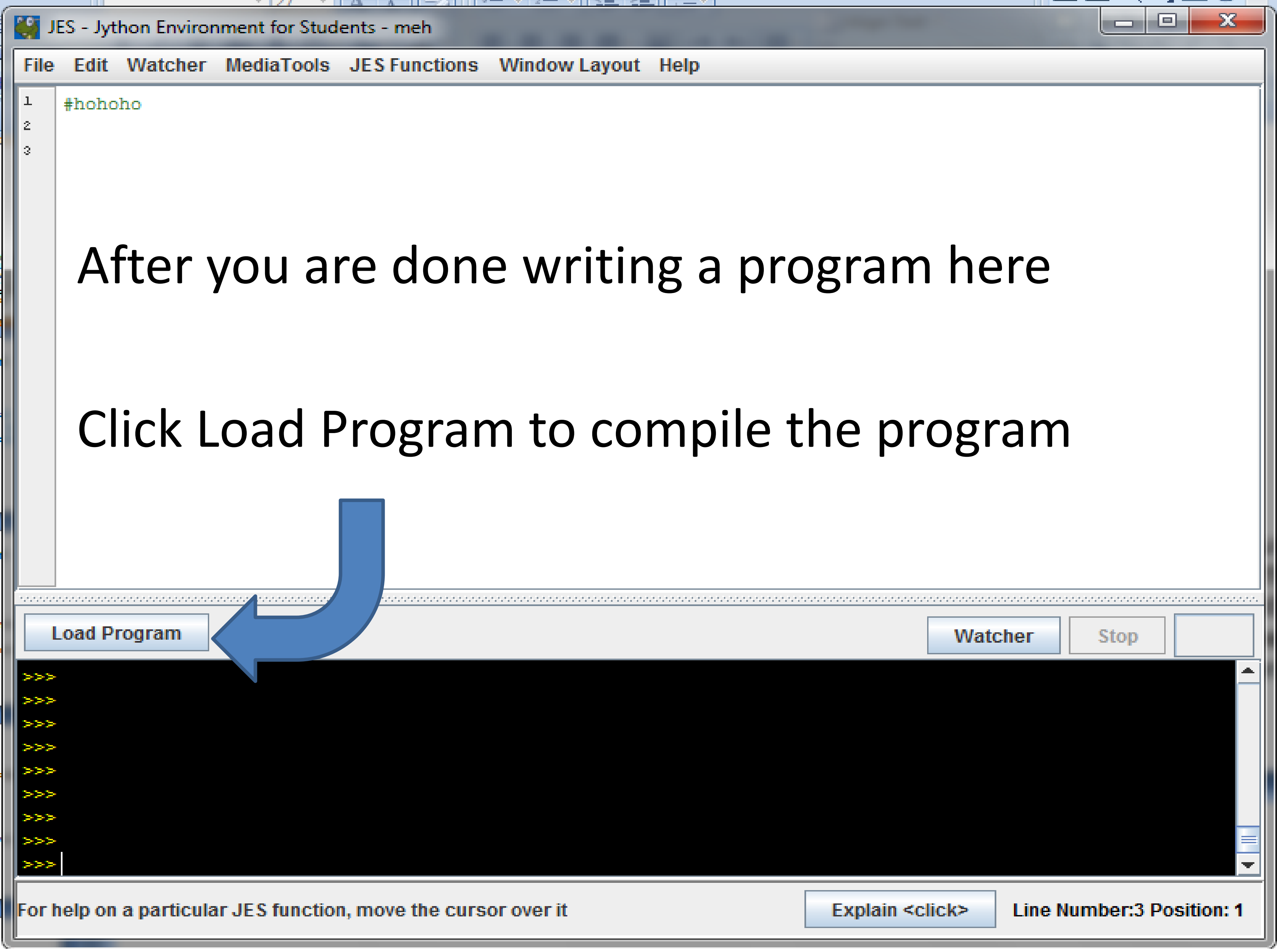

JES - Jython Environment for Students - meh
File
Edit
Watcher
MediaTools
JES Functions
Window Layout
Help
Load Program
UNLOADED
Watcher
Stop
Explain <click>
Line Number:3 Position: 1
For help on a particular JES function, move the cursor over it

```
#hohoho
```

```
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

The top part (the white window with the tiny number 1, which tells you that that's line number 1), or the "program area", works like any text editor would, where you can type stuff and save it under some file name so you can close it and pull it back up later.

The bottom part (in black), the "command area", is the brains of the operation.



```
#hohoho
```

After you are done writing a program here

Click Load Program to compile the program

```
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

For help on a particular JES function, move the cursor over it

Line Number:3 Position: 1

Writing Your Program

Header Comments:

```
# file name: circ.py
# author: Durrah Almansour
# description: a program to
calculate the area of a circle.
```

Indentation

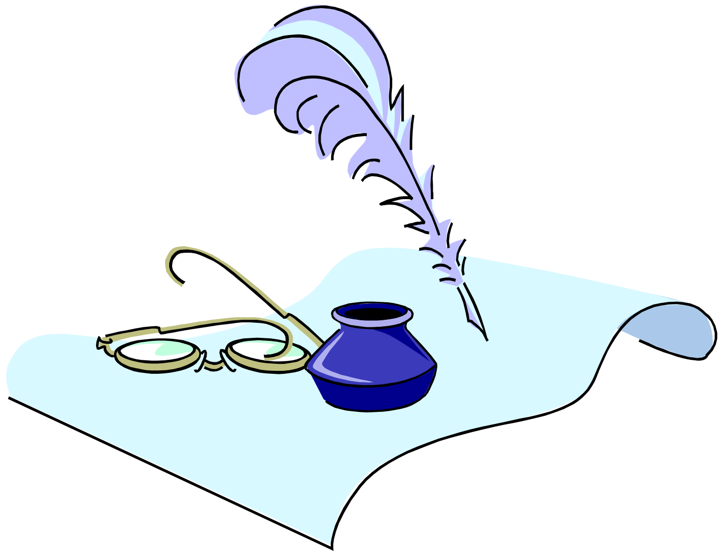
- Not an Option, but a Requirement!
- In Python, indenting specifies the “scope” of different chunks of your code.

```
def main():
    print "Hello world!"
```

The print line will be executed when you invoke main().

Writing Your Program

- Always plan what you want your program to do (pseudocode).
- Divide it into parts, it will make it easier to debug and update later.



Writing Your Program

Defining Functions:

```
def main():
    print "Hello world!"
```

- Name the function for the purpose you wrote it for.
- Don’t forget to indent the instructions that go inside the function.

Argument

A value passed to a function or method, assigned to a named local variable in the function body.

Ex.
```
def addUp(a,b):
        print("this is a+b: ", a+b)
```

#a and b are the *parameters*, 1 and 3 are the arguments we passed in.

#The function outputs : this is 1+3: 4

http://docs.python.org/glossary.html#glossary

Calling Functions

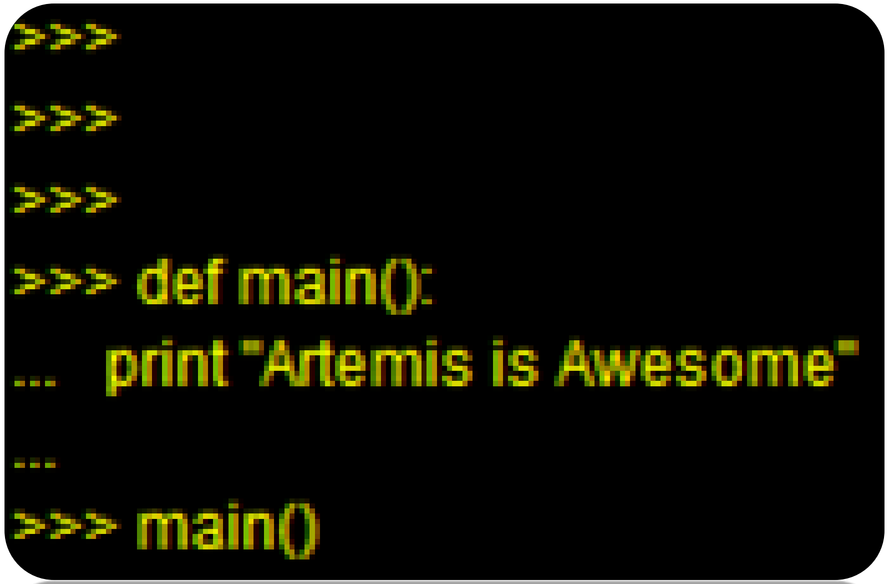
```
main()
```

will give the output

Hello world!

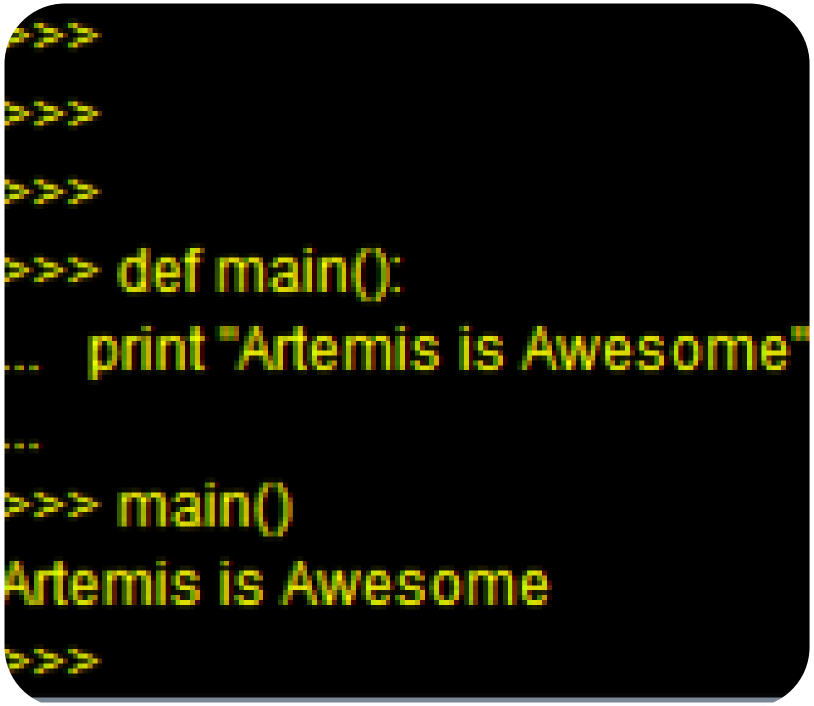
- There is no colon when calling a function!
- Use colons with “def”.

- Try this yourself



```
>>>
>>>
>>>
>>> def main():
...    print "Artemis is Awesome"
...
>>> main()
```

DON'T FORGET TO INDENT!



```
>>>
>>>
>>>
>>> def main():
...    print "Artemis is Awesome"
...
>>> main()
Artemis is Awesome
>>>
```

Output

E.g. Print:

```
print "Hello world!"
```

will give the output

```
Hello world!
```

Similar formatting, different output

- `print "Hello", "world", "!"`

  will output

  Hello world !

- `print "Hello" + "world" + "!"`

  will output

  Helloworld!

Similar formatting, different output

-
  ```
  print "Hello"
  print "world"
  print "!"
  ```
- will give the output

  Hello

  world

  !

-
  ```
  print "Hello",
  print "world",
  print "!"
  ```
- will give the output

  Hello world !

Data Types

The data type of an object determines the values it can hold, and the operations which can be performed on it.

- Numeric Data

Numeric data comes in 2 main flavors:

– Integers (whole numbers) 2, 5, -7, etc.

– Floating Point Numbers (non-integers) 0.2, 5.125, etc.

Data Types

- Non-numeric Data Types:

  Those include strings (text), lists, dictionaries, etc..

  Basically anything you can not add up using a simple plus sign (+).

Not a String? Not a Problem!

You can also format outputting *variables* you've defined:

-
  ```
  x = 42
  print" The value of x is", x, "."
  ```
- will give the output

  The value of x is 42 .

Not a String? Not a Problem!

-
  ```
  x = 42
  print "$" + x
  ```
  causes an error.
- So what do we do?

  ```
  x = 42
  print "$" + str(x)
  ```
  will give the output
  ```
  $42
  ```

Defining Variables

Rules for naming variables:

- Have to start with a letter or underscore (_)
- Can contain letters, numbers, and underscores
- Can’t contain spaces, punctuation, etc.
- Can’t be Python keywords
- Are case sensitive

Defining Variables

Things that aren’t rules, but are worth considering:

- You should give your variables sensible names (“price”, “pixelColor, or “samplingRate” instead of “x”)
- Just because you technically can start your variable names with underscores doesn’t mean you should.

Defining Variables

- For multi-word variable names, two options:
  - start capitalizing each word after the first “myCar”
  - separate words with underscores. For instance, a variable for “Ford Focus” could be “my_car”.
- Abbreviating is common for longer words. So, a variable for “average price” could be “avgPrice” or even “avg”.

Variables

- Variables can hold all kinds of values, including strings, different types of numbers, and user input.
- To assign a string value to a variable, you have to wrap the string in quotes (like usual).

```
firstName = "John"
lastName = "Doe"
mathProblem = "5 + 5"
print lastName, ",", firstName, ";",
mathProblem
```

will give the output

```
Doe , John ; 5 + 5
```

Variables

- Variables can also be assigned new values that are relative to their old values. For example:

```
total = 10
print "Original total:", total
total = total + 4
print "New total:", total
```

will give the output

```
Original total: 10
New total: 14
```

Variables

- Remember: A variable has to have been defined on a previous line before it can be used on the right-hand side of an equation, so:

-
  ```
  total = total + 4
  print "Total:", total
  ```
  causes an error, since there was no mention of the value of “total” before the line trying to redefine it.

Numeric Operators

- Python built-in numeric operators:
- + addition
- - subtraction
- * multiplication
- / division
- ** exponentiation
- % remainder (modulo)

Python Arithmetic

- Try writing the following code in your program area and see what it outputs

```
Def main():
    a = 12
    b = 2
    c = 16
    d = 3
    e = 2.5

    print "the value of a is", a
    print (a / b) * 5
    print a + b * d
    print (a + b) * d
    print b ** d
    print c - e
    a = a + b
    print "the value of a is", a
```

Python Arithmetic

- Is this what you got?

```
the value of a is 12
30
18
42
8
13.5
the value of a is 14
```

Exercise Time!

- Write a program that takes in a birthday (dd, mm, yy) and returns:
  - The age
  - Number of days until next birthday

Taking User Input

Sometimes, instead of passing in arguments, you can ask for them after calling the function.

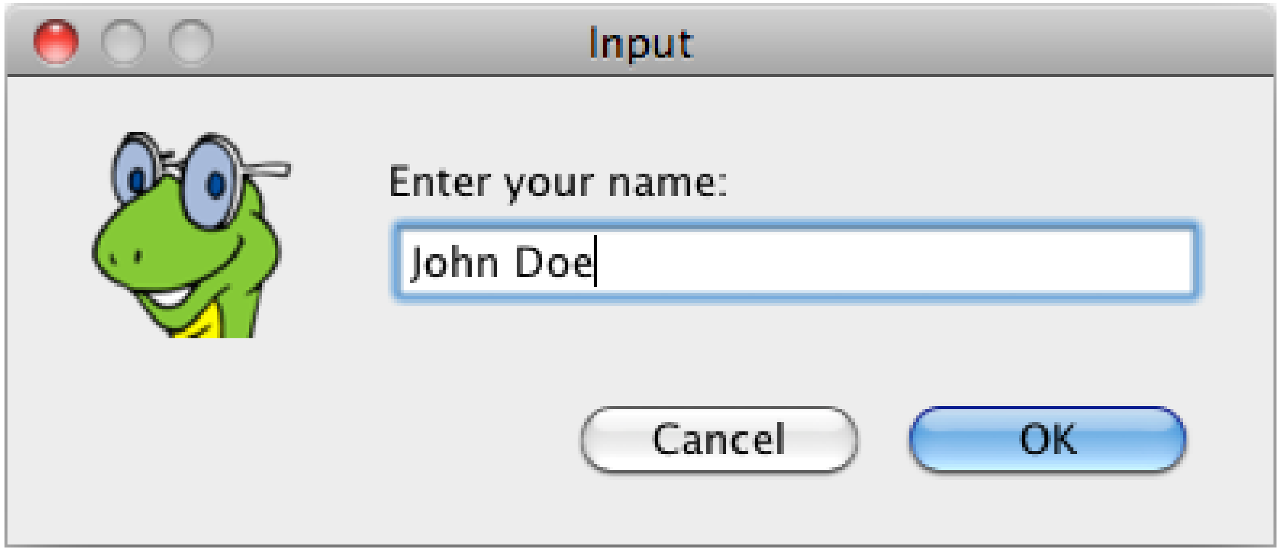
Taking User Input

- *requestNumber()*
- *requestInteger()*
- *requestIntegerInRange()*
- *requestString()*

Taking User Input

```
name = requestString("Enter your name:")
print name
```

first pops up a dialog box (where you can enter a name, say 'John Doe'):



Input

Enter your name:

John Doe

then outputs

```
John Doe
```

Ex. Try it with Numbers!

```
num = requestNumber("Enter a number:")
print "Your number:", num
print "Your number squared:", num*num
```

This is where you put the message you want to appear with your input box!

Ex. Try Inputting a String

- Make JES print “<input> is awesome!”

```
name = requestString("Enter your name:")
    print name, “is awesome!”
```

The For Loop

- Also known as the “definite loop”.
- Allows you to specify a list of items (numbers, words, letters, etc.), and specify action(s) to be performed on each one.
- The official form for the for loop is this:

```
for <var> in <sequence>:
    <body>
```

(Note that the body is indented to in the loop)

The Kittens Need Your Help!

You are working at an animals shelter, you are asked to take a group of kittens, bathe, dry, and feed each one individually.

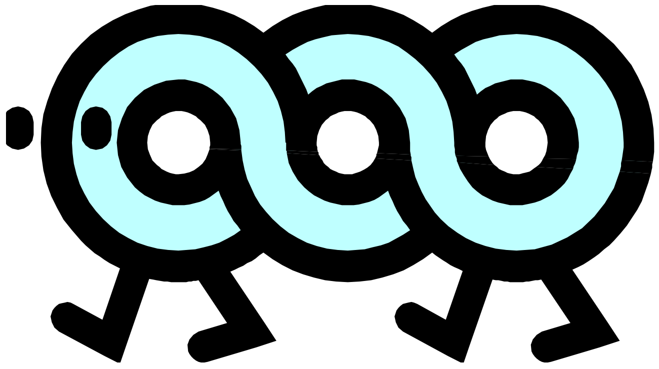
The Kittens Need a Loop!

Using a for-loop type notation, your instructions would look like this:

```
Kittens = [kitty #1, kitty#2,
kitty#3, ...]
for kitty in Kittens:
    bathe kitty
    dry kitty
    feed kitty
```

Ex. For Loop

-
  ```
  phrase = "Hello world!"
  for letter in phrase:
  print "the next letter is:", letter
  ```
  will give the output
  ```
  the next letter is: H
  the next letter is: e
  the next letter is: l
  the next letter is: l
  the next letter is: o
  the next letter is: 
  the next letter is: w
  the next letter is: o
  the next letter is: r
  the next letter is: l
  the next letter is: d
  the next letter is: !
  ```



What Just Happened?

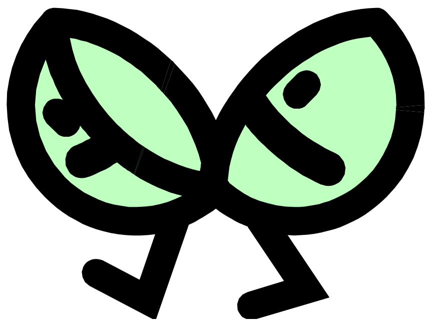
- What Python did was that it went through the line one character at a time, treating the line like a sequence.
- That means that the line can be split into its components (the characters).

split()

Want to work with units of a phrase that aren't characters?

Put something in the <sequence> position that isn't just a string.

The result is a list of all the items in the phrase that are separated by spaces.



split()

```
phrase = "Hello beautiful world!"
for word in phrase.split():
    print "the next word is:", word
```

will give the output

```
the next word is: Hello
the next word is: beautiful
the next word is: world!
```

split()

In fact, if you printed phrase.split() to see what it looked like, you’d get ['Hello', 'beautiful', 'world!'], a list containing each “word”

(You will learn about lists tomorrow ☺)

Accumulator Variables

When you're using a for loop, sometimes you want to keep a running total of numbers you're calculating, or re-combine bits of a string.

Accumulator Variables

Steps:

1. Define it for the first time before the for loop starts.
2. Redefine it as itself plus some operation in the body of the for loop.

```
total = 0
  for num in [1,2,4,10,20]:
    total = total + num
  print "Total:", total
```

will give the output

```
Total: 37
```

Accumulator Variables

What is the point of accumulator variables?

-Counting.

-keeping score (affects program does).

-debugging.

Conditional Statements

Equals: ==

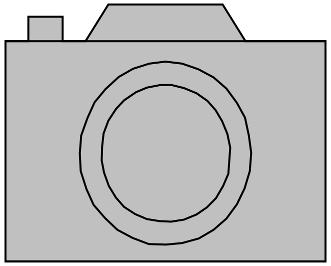
Does not equal: !=

Try this:

```
x = 1
If( x != 2)
   print "Artemis rocks"
```

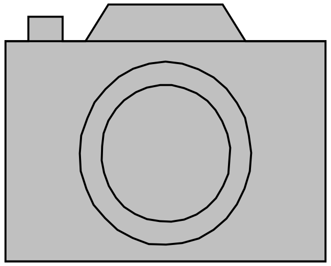
Pictures

File Functions



- pictures live within files.
- You must get your program to go find and read a file that's somewhere else on your computer.

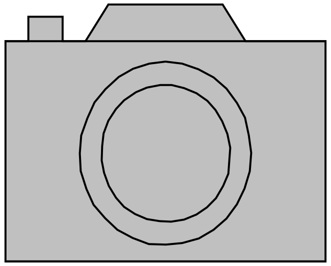
File Functions



Steps:

1. Store the “address” for the file you want as a variable
2. Use functions to read, display, or modify the file at that location.

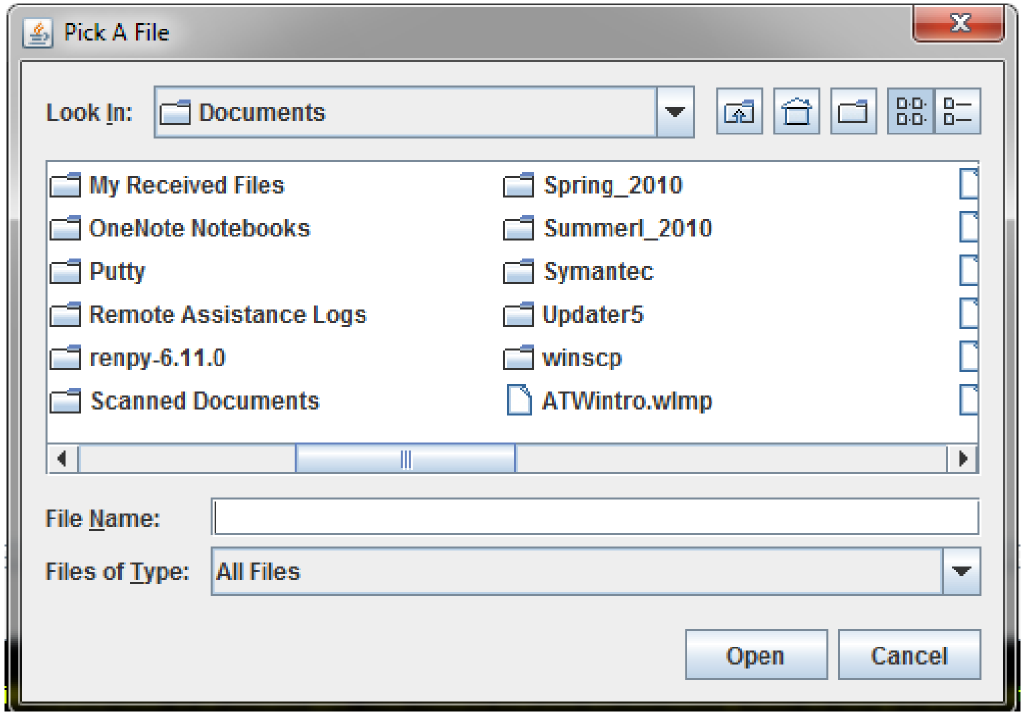
File Functions



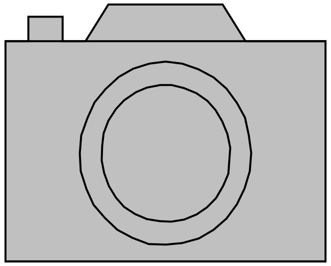
```
myFile = pickAFile()
```

This function notes the 'path name' (as a string) of your file, i.e. the "address" of that file on your computer.

It brings up the 'file selector dialog'.



File Functions

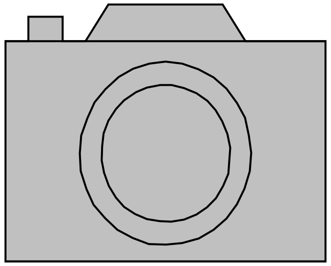


If it is a picture, and you wish to treat it as such:

```
myPic = makePicture(myFile)
```

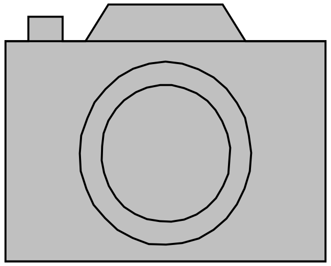
These functions will not make anything appear.
So far, things are just stored in the computer's memory, invisible to the user.

Pictures



After modifying your sound/picture you may want to save it as a new file, since ‘repainting’ a picture or ‘playing’ a sound will simply show you your work, not save it anywhere.

Pictures



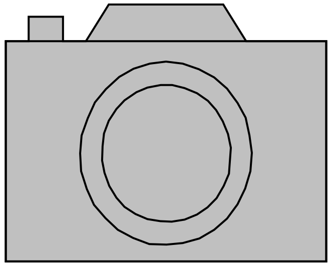
How?

Use *`writePictureTo`* after you specify the path.

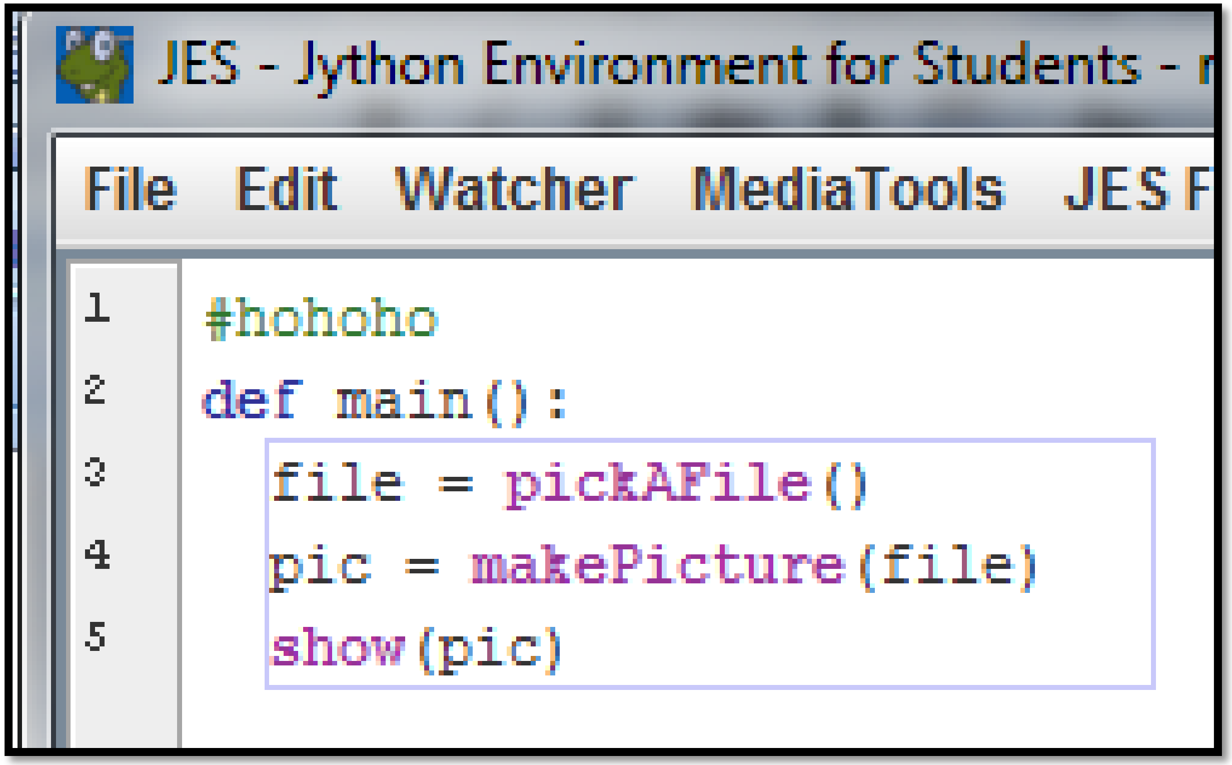
Try this in the command area

```
path ="/Users/Durrah/Documents/stuff/editedpic.jpg"
   writePictureTo(pic,path)
```

Pictures



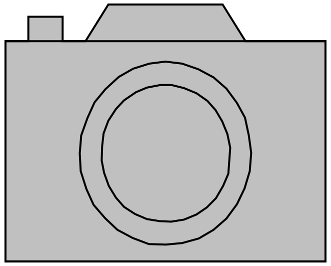
Try writing this simple function


JES - Jython Environment for Students - r
File Edit Watcher MediaTools JES F

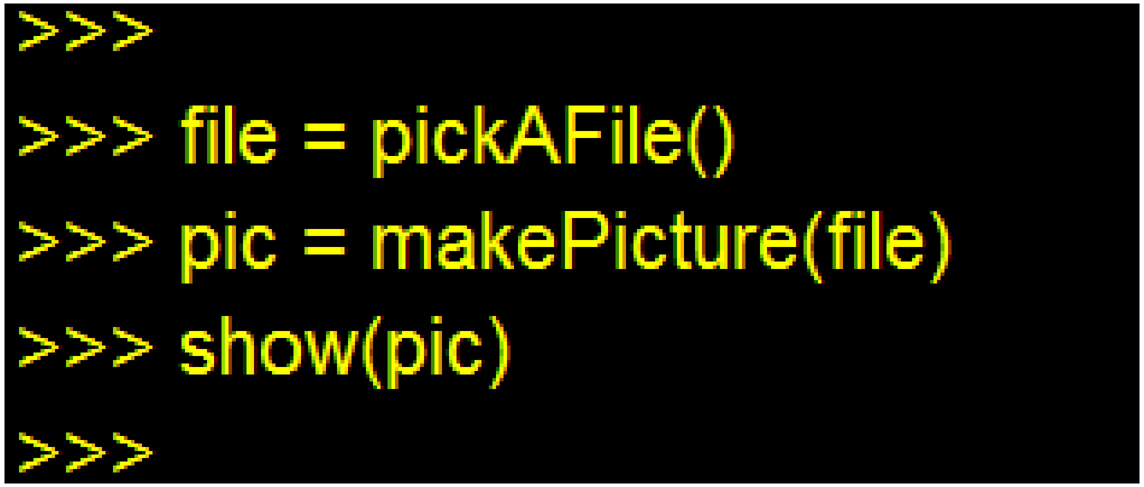
```
#hohoho
def main():
    file = pickAFile()
    pic = makePicture(file)
    show(pic)
```

This shows a window containing the picture.

Pictures

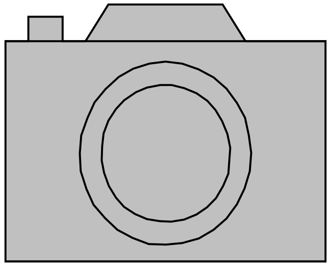


- Find a small picture (use Google)
- Save it in Documents.
- Enter the following

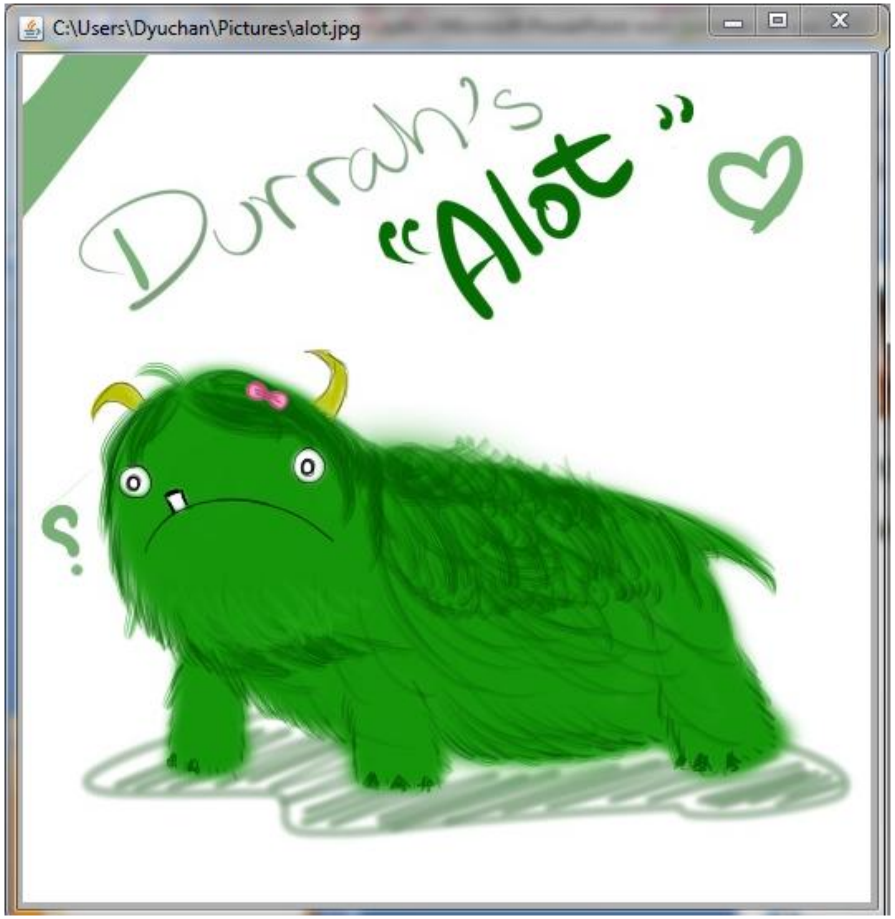


```
>>>
>>> file = pickAFile()
>>> pic = makePicture(file)
>>> show(pic)
>>>
```

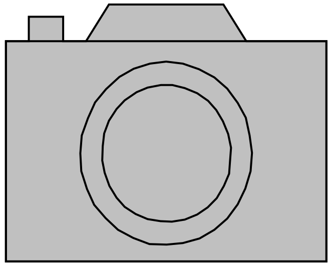
Pictures



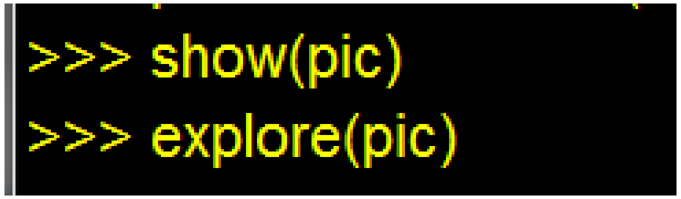
Your picture should appear



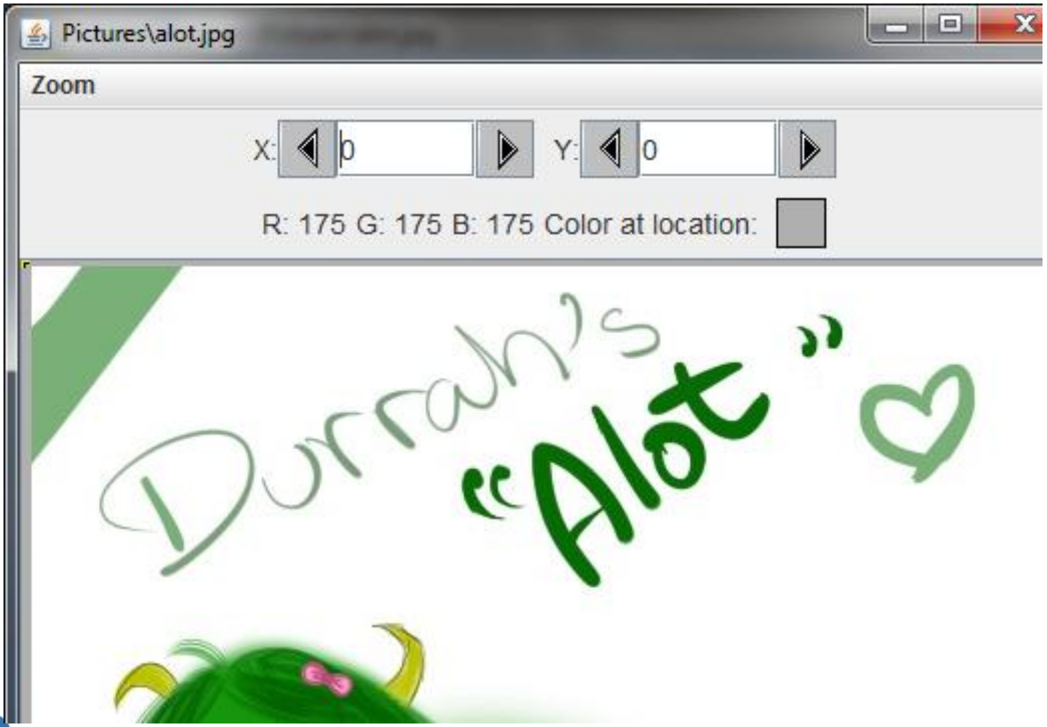
Pictures



Now enter explore(pic)

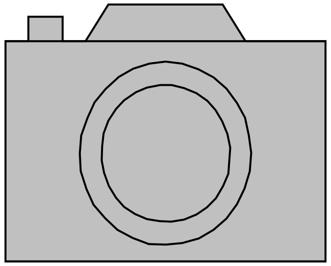


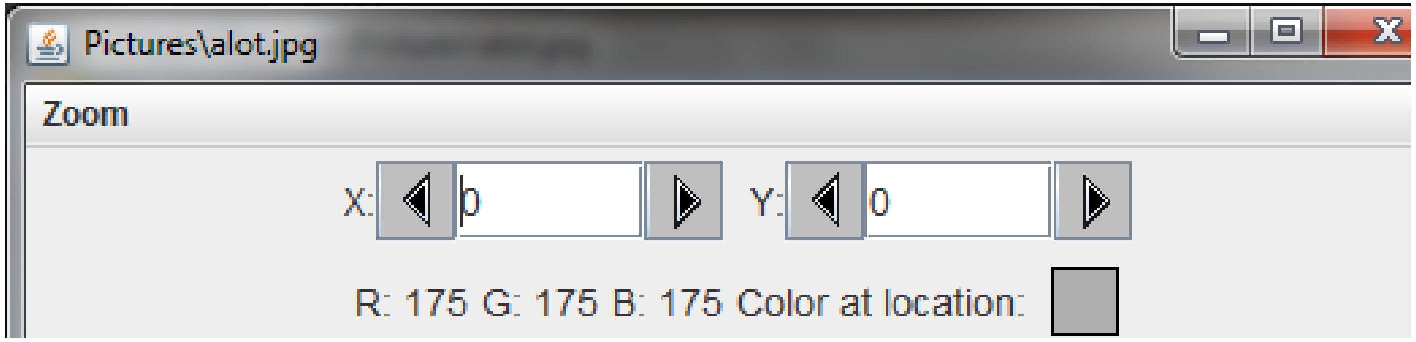
```
>>> show(pic)
>>> explore(pic)
```



Click anywhere in the picture that just popped up and you'll see the X & Y coordinates of the pixel.

Pictures

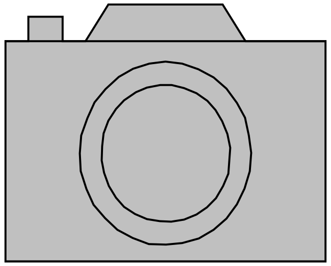




R: 175 G: 175 B: 175 Color at location:

This function is useful when you are trying to locate a certain pixel that you want to play with. It gives you the Red, Green and Blue values of that pixel. This is the stuff of which Photoshop is made ☺!

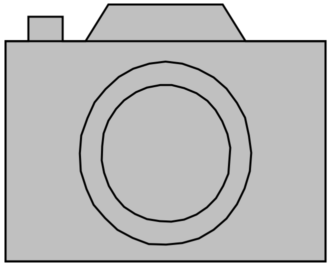
Pixel Functions



So how do you make changes to your picture?

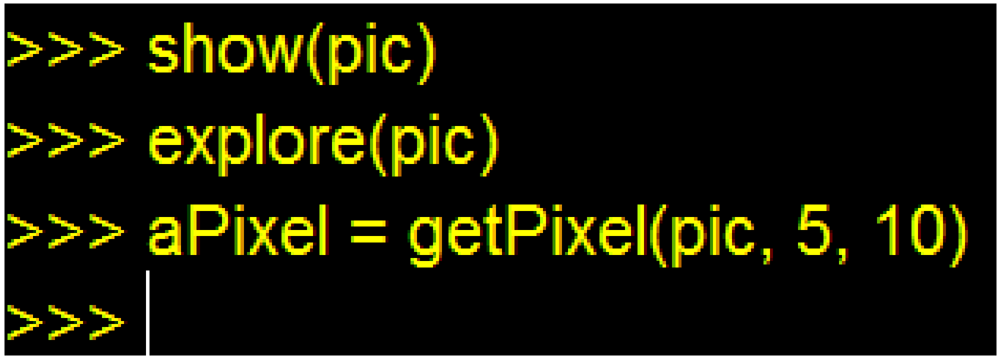
- Target one pixel
- Target all pixel

Getting a Pixel



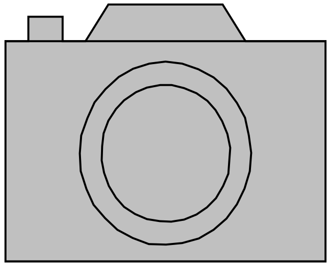
`getPixel` takes three parameters, the picture to take the pixel info from, the x-value of the desired pixel, and the y-value of the pixel, in the form `getPixel(pic, x, y)`

Add this to your command area



```
>>> show(pic)
>>> explore(pic)
>>> aPixel = getPixel(pic, 5, 10)
>>>
```

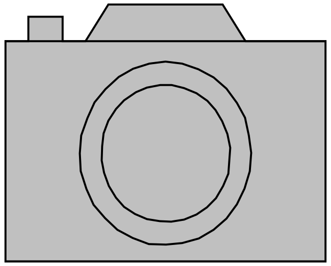
Pixel Functions



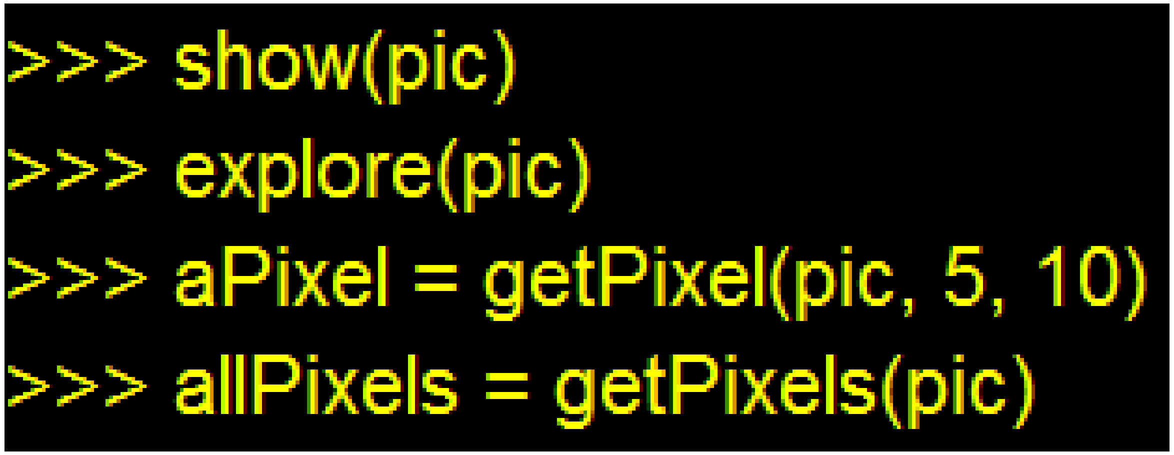
`aPixel = getPixel(pic, x, y)` stores the color information of the pixel located at (5, 10) in the picture `pic` in the variable `aPixel`.

But what if we want ALL the pixels?

Get Those Pixels!

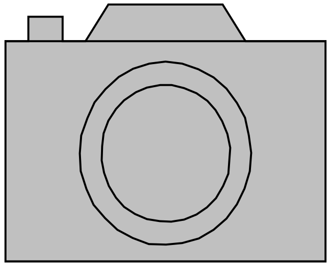


`getPixels(pic)`



```
>>> show(pic)
>>> explore(pic)
>>> aPixel = getPixel(pic, 5, 10)
>>> allPixels = getPixels(pic)
```

PIXELIZATION!

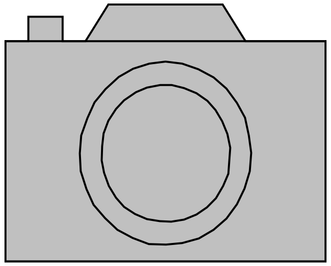


Now we want to change every single pixel… How will we ever do that?

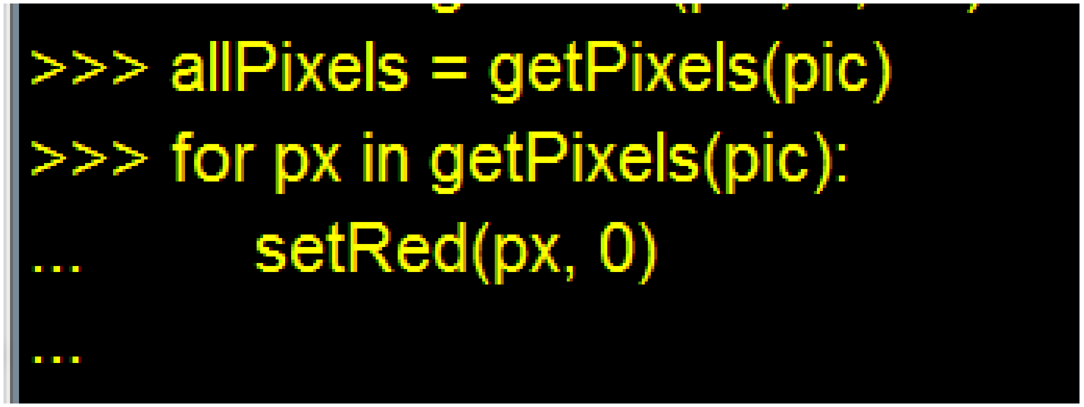
Why of course!

The LOOP to rescue!

There are New Pixels in Town

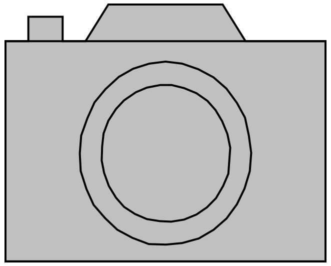


Don’t forget to indent!



```
>>> allPixels = getPixels(pic)
>>> for px in getPixels(pic):
...         setRed(px, 0)
...
```

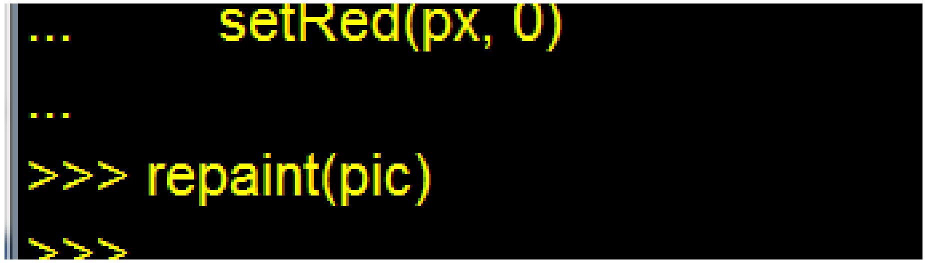
Pictures



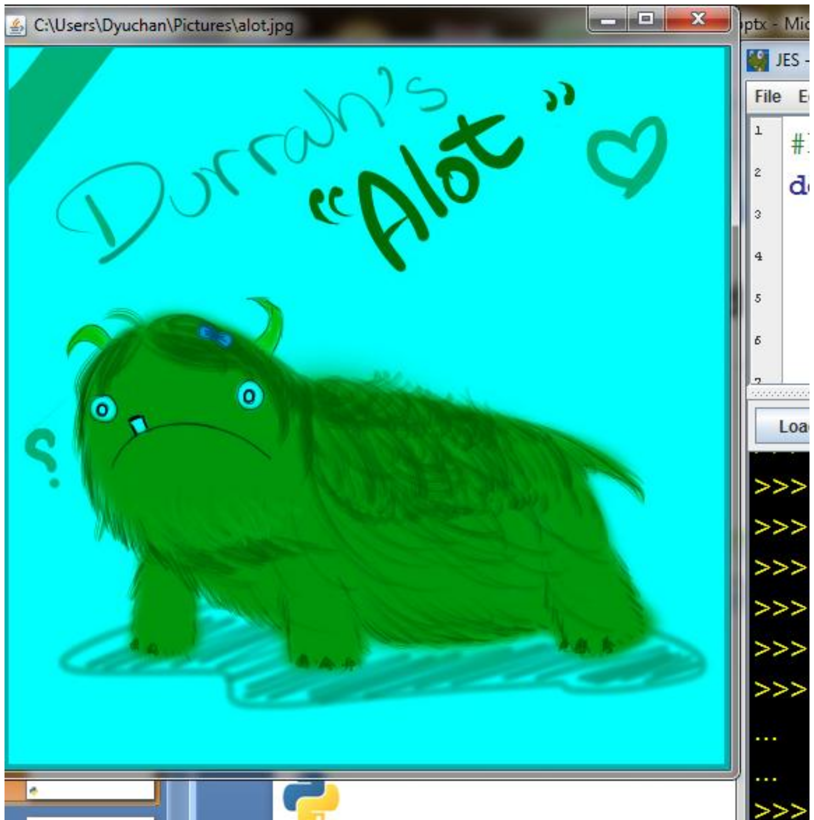
Your picture has not changed yet? That is because you haven't applied the changes to it!

Enter:

`repaint(pic)`

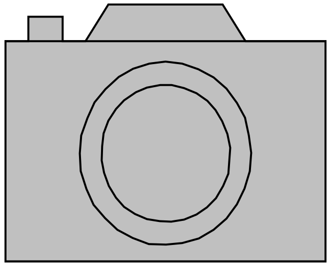


```
...       setRed(px, 0)
...
>>> repaint(pic)
>>>
```



C:\Users\Dyuchan\Pictures\alot.jpg

THE CHALLENGE

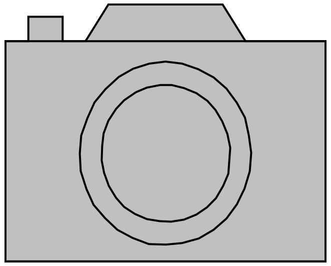


Can you write a program (in the program area) that changes the color of the picture the way we did?

You have all the functions in your command area.

I will give you the pseudocode and you have to code it however you see fit.

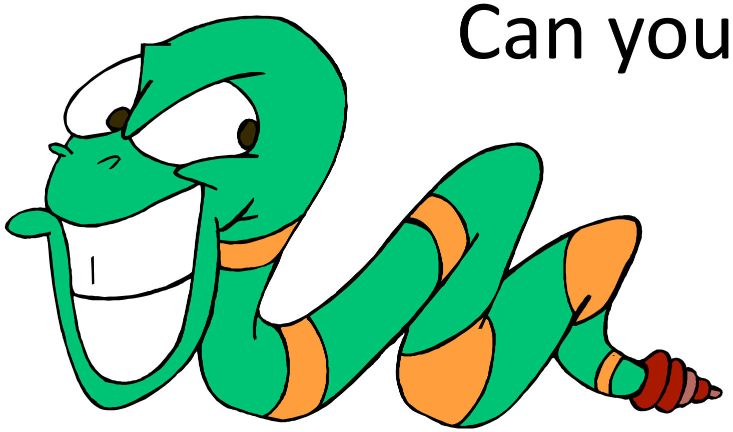
THE CHALLENGE



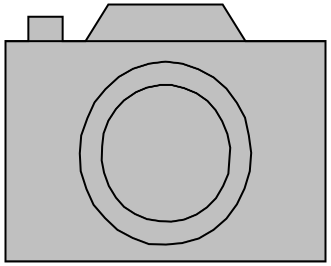
Remember to start with def `main():` and to indent everything to be inside it!

The program should have 9 lines of code. I will show one line of code every TWO minutes

Can you beat the PowerPoint?



Pictures



Get the file

Make it a picture

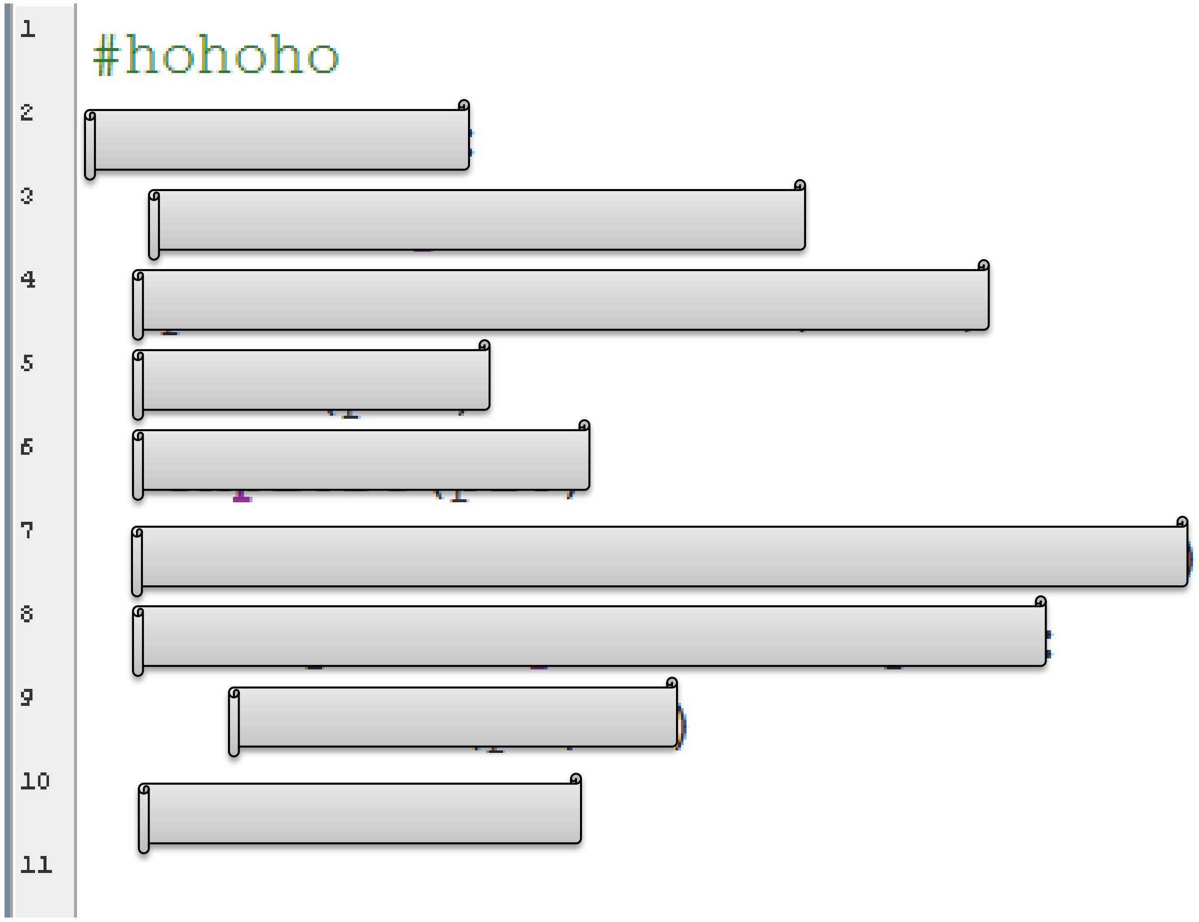
Display it

Explore it

Get one pixel

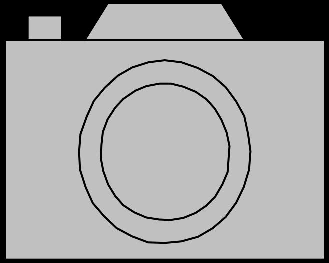
Change the color
of the picture

Apply the changes



```
#hohoho
```

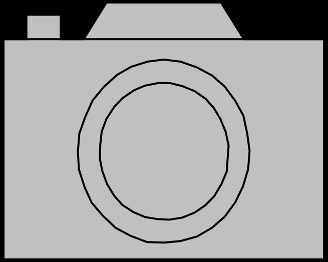
Negative



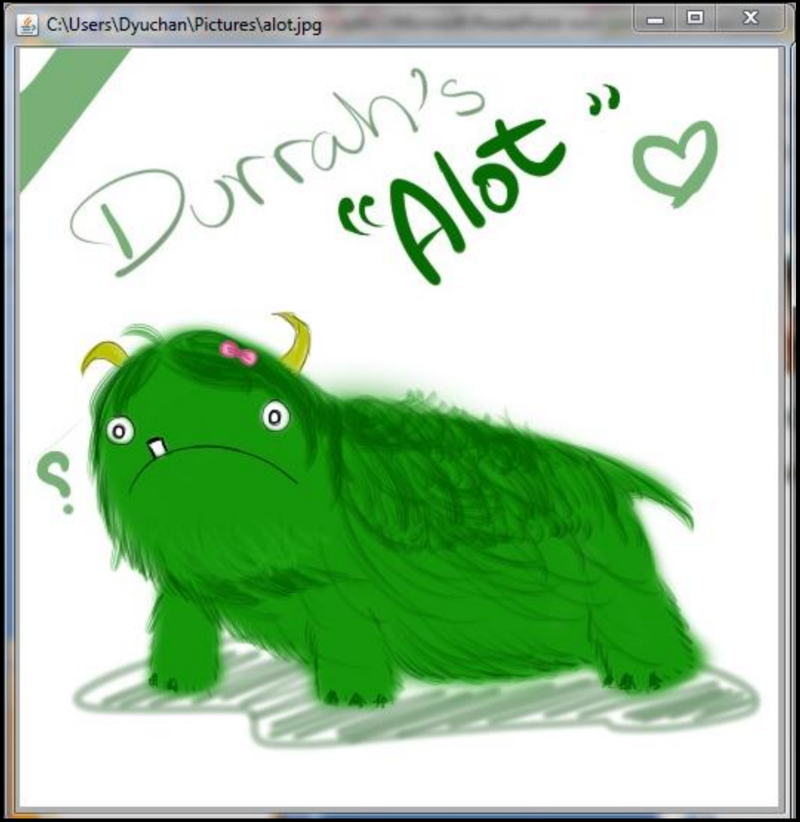
When creating a negative, we want the opposite of each of the current values for red, green, and blue. If we have a red component of 0, we want 255 instead, If we have 255, we want 0.

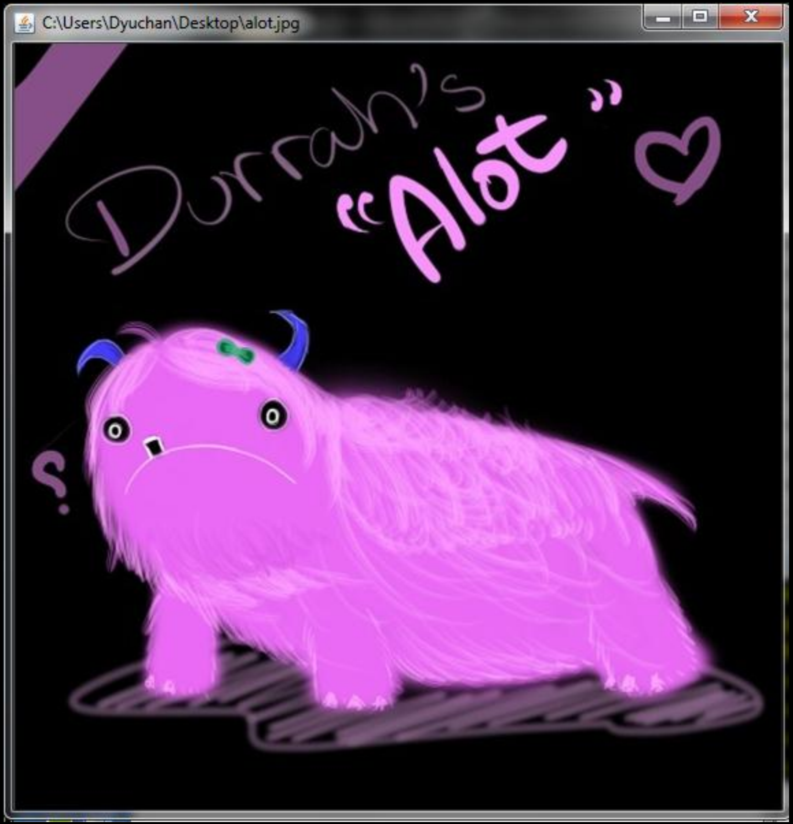
In other words, we are getting |255-current| for every pixel in the picture.

Negative



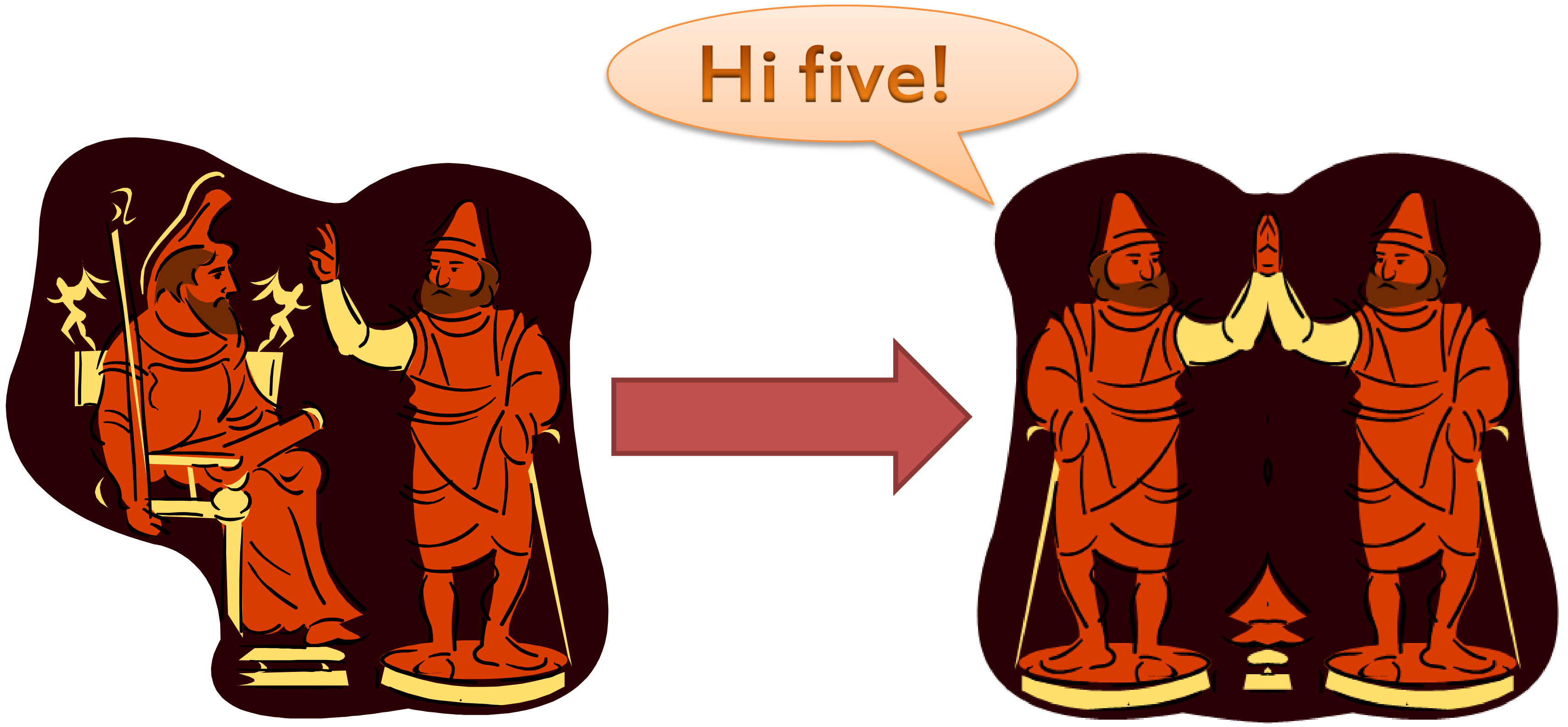
```
Def negative():
  file = pickAFile()
  pic = makePicture(file)
  for px in getPixels(pic):
      red = getRed(px)
      green = getGreen(px)
      Blue = getBlue(px)
      negColor = makeColor(255-red, 255-green, 255-blue)
      setColor(px, negColor)
  repaint(pic)
  show(pic)
```

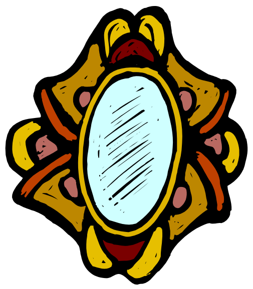

C:\Users\Dyuchan\Pictures\alot.jpg
Durrah's
"Alot"


C:\Users\Dyuchan\Desktop\alot.jpg
Durrah's
"Alot"

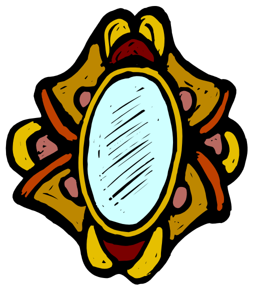
Mirror, Mirror on the Wall

- We can use Python to manipulate more than the colors of the picture. We can do this:





Code, Code on the Screen



```
#Starting with pseudocode
mirrorVertical():
    Get a picture
    Identify its middle.
    in every row
        replace each column with the at the same
        distance from the middle until you reach the
        middle
    apply changes
    show the picture
```

Code, Code on the Screen

```
def mirrorVertical():
  file = pickAFile()
  pic = makePicture(file)
  mirrorpoint = getWidth(pic)/2
  for y in range(1, getHeight(pic)):
    for xOffset in range(1, mirrorpoint):
      pright = getPixel(pic, mirrorpoint+xOffset, y)
      pleft = getPixel(pic, mirrorpoint-xOffset,y)
      c = getColor(pleft)
      setColor(pright, c)
  repaint(pic)
  show(pic)
```

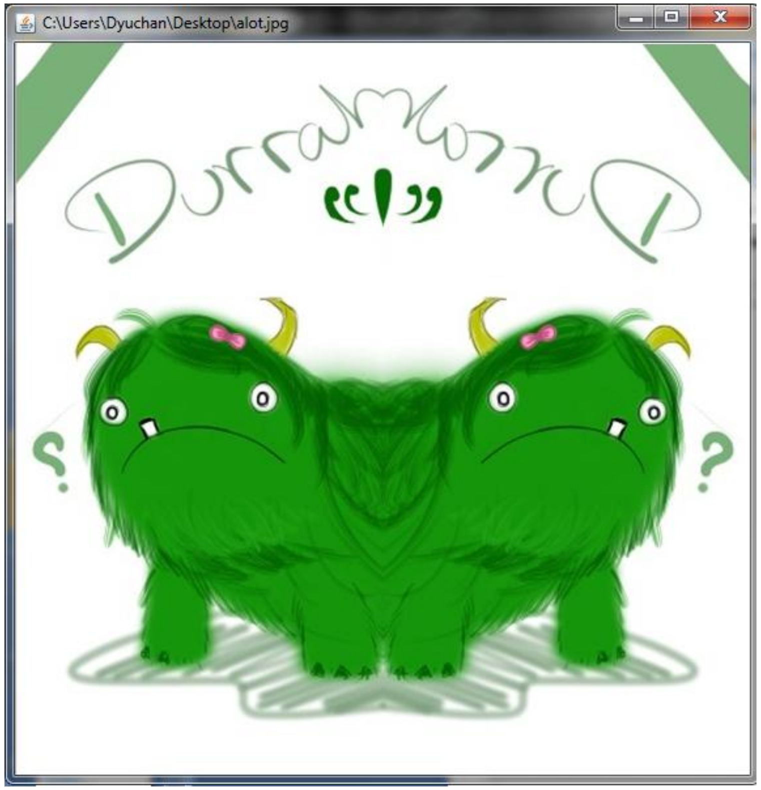
Extra Challenge

- As you can see, the right side mirrored the left side (and that's why the creature has two heads).
- Can you change your code

So it does the opposite?

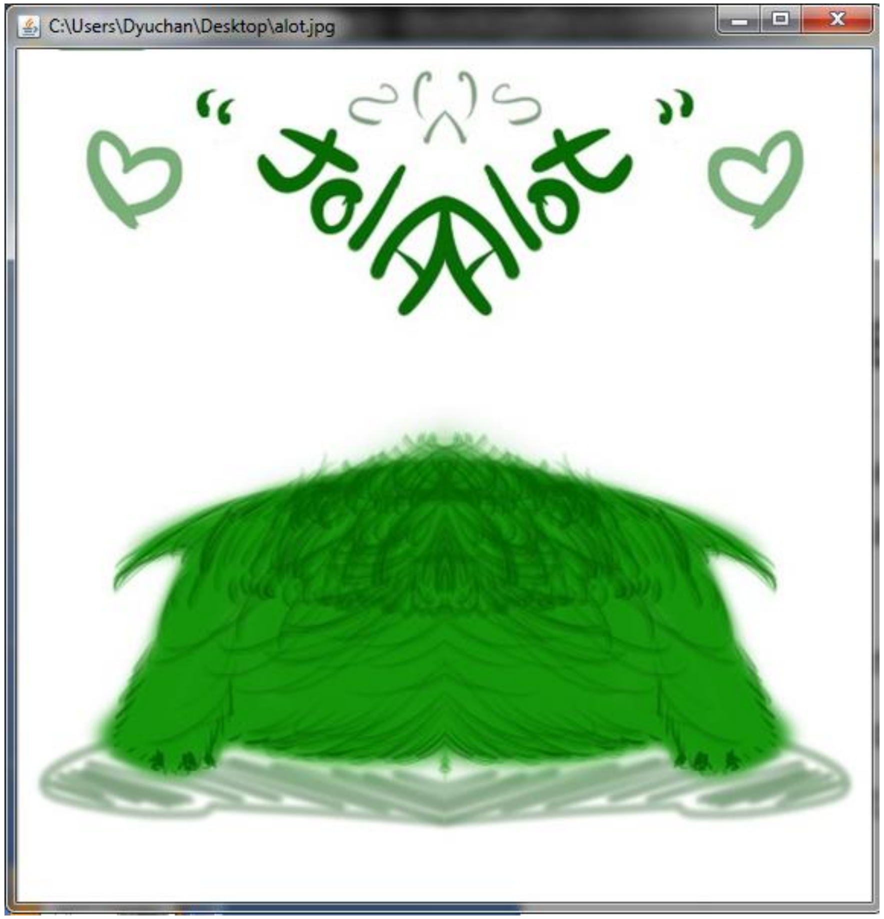
(i.e. let the left side mirror

the right side)

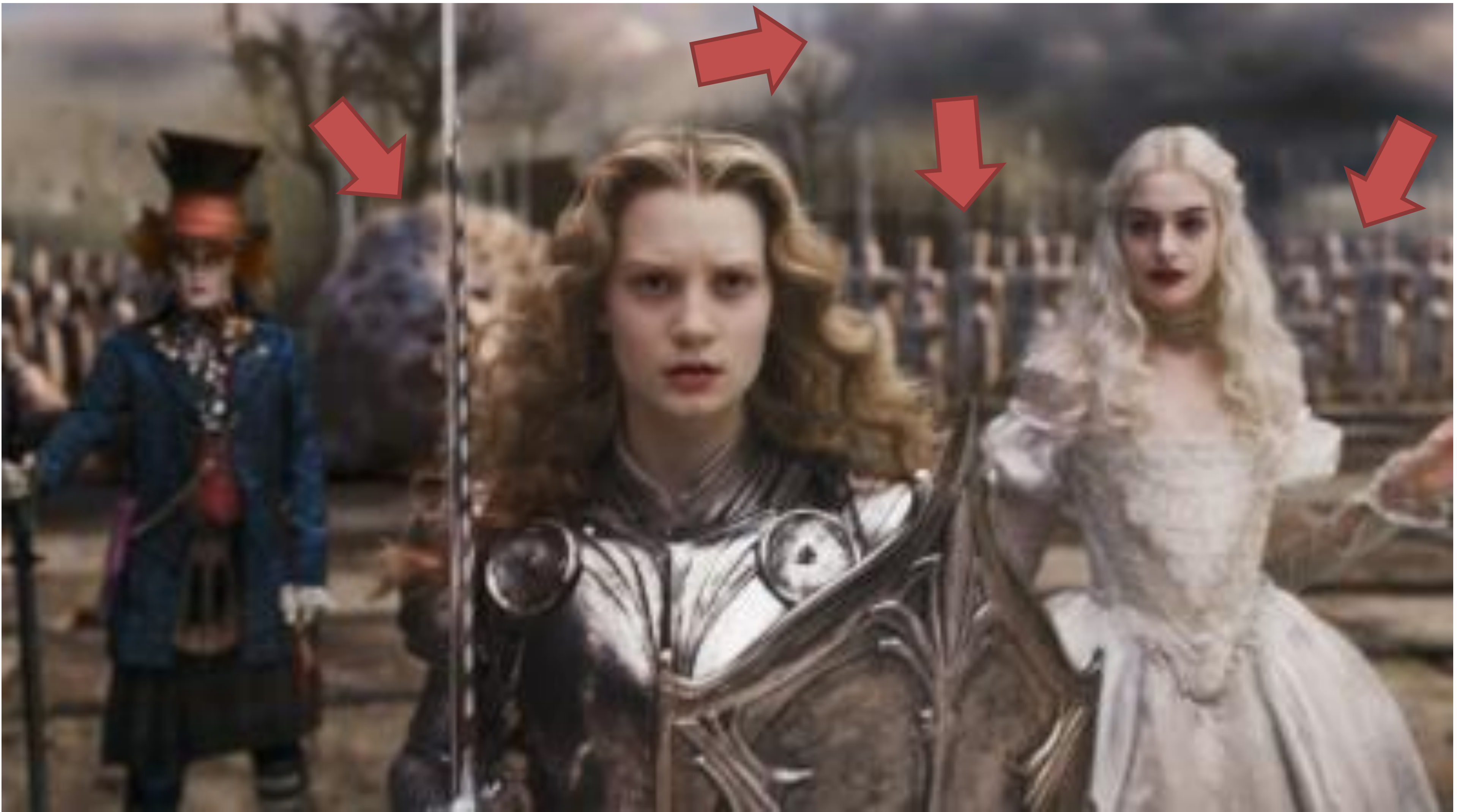


Reversed Mirroring

- This is the result I got:



Are the soldiers, clouds, big creature actually there?



Greet the Green☺

The Green Screen

- Placing an object (foreground) in a background of our choice.

The Green Screen

- Tell the computer look at each pixel, and see if its red and blue values are less than its green value.
- If (red+blue < green) then that pixel is likely to belong to the green screen. Now tell the computer to get the pixel at the same location from the background and paint it on the green screen.

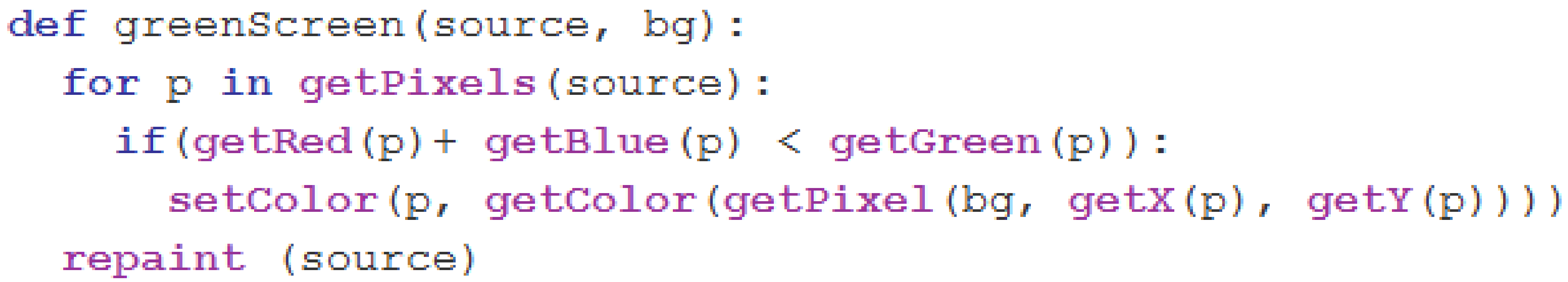
The Green Screen

So in a way, you are actually placing the background on the foreground, not the other way around.

Also, it could be a blue screen. (How will the code change?)

The Green Screen Code

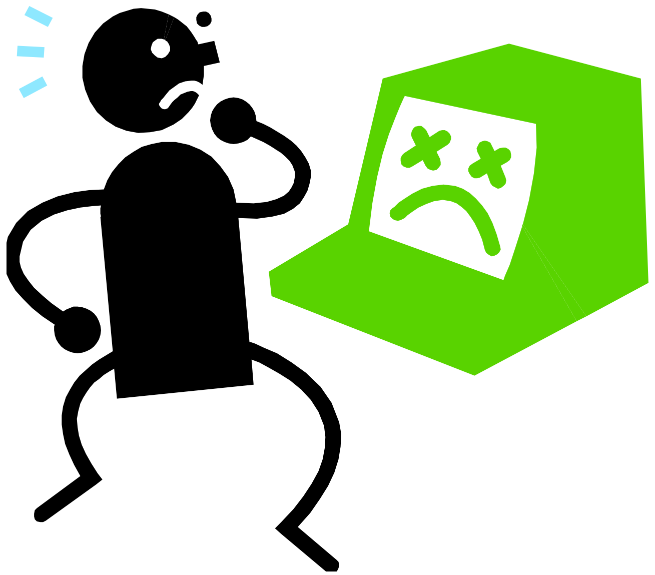
- How do you think the code should look like?

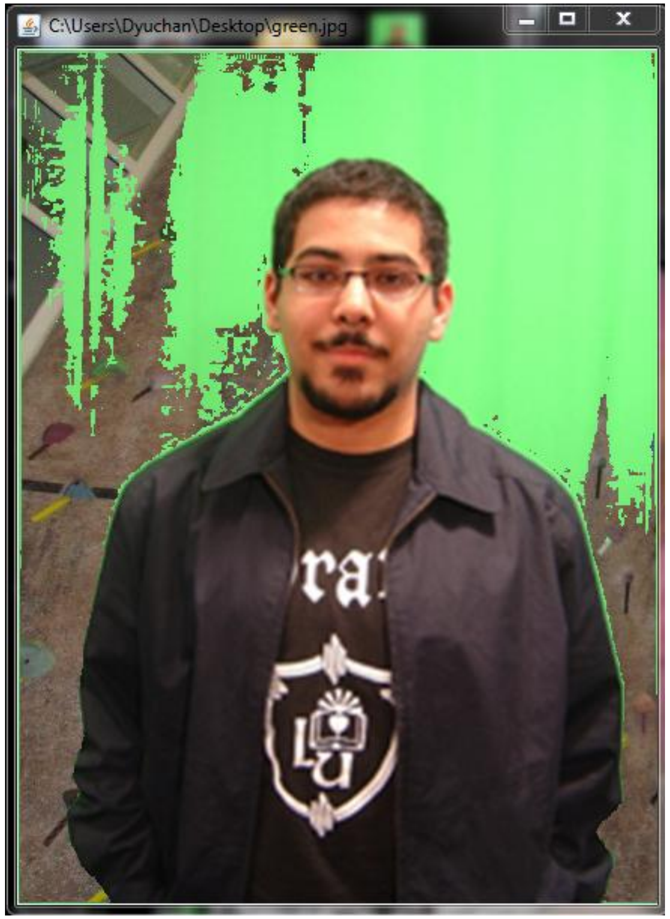


```
def greenScreen(source, bg):
  for p in getPixels(source):
    if(getRed(p)+ getBlue(p) < getGreen(p)):
      setColor(p, getColor(getPixel(bg, getX(p), getY(p))))
  repaint (source)
```

The Green Screen Effect

You do not get perfect results all the time…





Our Journey Ends Here

But yours doesn't need to!

Want to learn more?

Go to:

wiki.**python**.org/moin/BeginnersGuide

Questions?

This presentations was based on the CS101 Guide to Python and JES (http://www.cs.bu.edu/courses/cs101b1/jes)