

Outline

Quiz

Regular expressions

- using python
- Regex artists

Homework review

Cool media art pieces

Prof. Angela Chang

Lecture 10: Quiz & Regular Expressions 2

Fall 2017. Oct 11

CODE, CULTURE, AND PRACTICE

Quiz (30 minutes)

Helpful hints:

Python code is designed to be readable. Read each line using your finger so you can focus carefully on the words.

What happens in the function, stays in the function!

Range and Slices handle up to three arguments:

sequence [start_index : end_index : step_index]
range (start_index , end_index , step_index)

Leonard Richardson

Swapping the dialogue between two different texts

ALICE'S ADVENTURES IN THE WHALE

Leonard Richardson

Alice's Adventures in the Whale is a generated novel created by Leonard Richardson. His script ([accessible here](#) for the code-savvy and adventurous) replaces all dialogue in one novel with dialogue from another — in this case, *Alice's Adventures in Wonderland* and *Moby-Dick; or, The Whale*. We present this excerpt for your enjoyment.

CHAPTER V. “Advice from a Caterpillar”

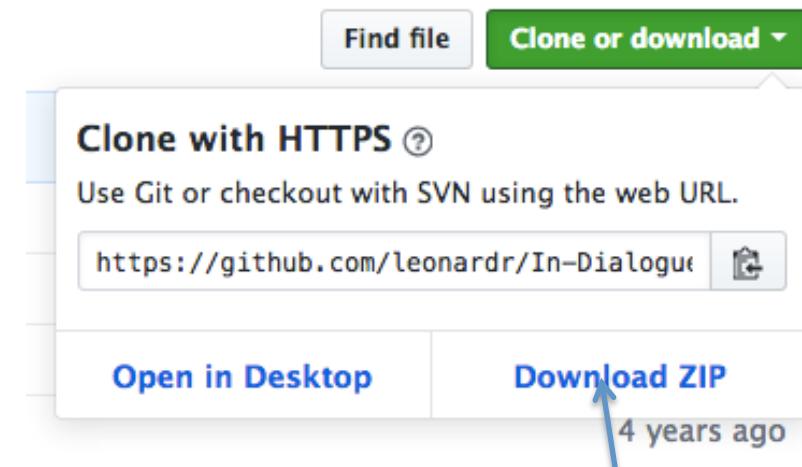
The Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. “I say, pull like god-dam,” said the Caterpillar.

This was not an encouraging opening for a conversation. Alice replied, rather shyly, “There she slides, now! Hurrah for the white-ash breeze! Down with the Yarman! Sail over him!”

<http://wagsrevue.com/20/fiction/richardson1>

Download the source here:

<https://github.com/leonardr/In-Dialogue/>



Lecture10_RegexAndImages.ipynb

Once the source is downloaded to your directory...

Read the instructions on the webpage for how to run it

- 1.Download source
- 2.Look at readme
- 3.Download source text from gutenberg
4. Run scripts to get output.

In Dialogue

a 2013 NaNoGenMo project

by Leonard Richardson

"In Dialogue" extracts the dialogue from Project Gutenberg ebooks and provides a tool for replacing all the dialogue from one text with dialogue from another.

The NaNoGenMo entry itself is in two parts found in the entry/ directory, and can also be [read on my web site](#).

To generate your own texts, run `0-extract-dialogue.py`, then `1-generate-book.py`. The `1-generate-book.py` script will show you which texts are available.

You can add new texts by downloading Project Gutenberg files into the raw/ directory. For example, these commands will add "Oliver Twist" to the list of available texts:

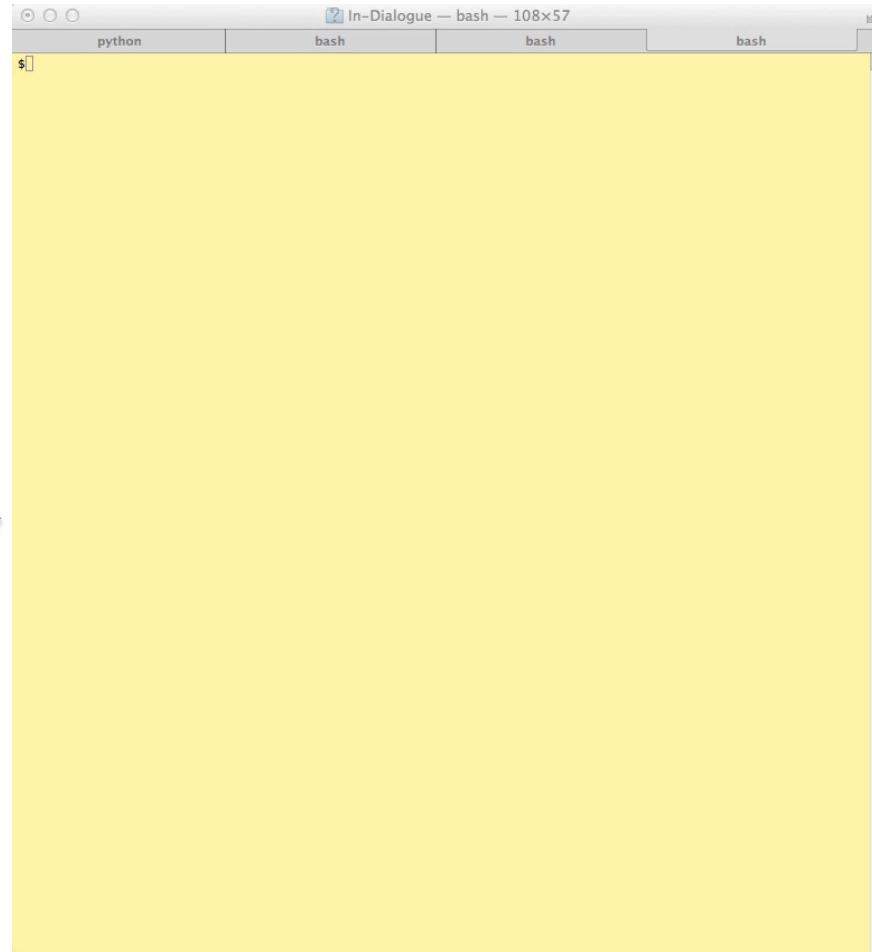
```
$ wget http://www.gutenberg.org/ebooks/730.txt.utf-8 -O raw/730.txt.utf-8 $ ./0-extract-dialogue.py
```

Then, this command prints the text of "A Christmas Carol", with all the dialogue replaced with dialogue from "Oliver Twist":

```
$ ./1-generate-book.py "A Christmas Carol" 46 730
```

Trying it out on the command line

- 1.Download source
- 2.Look at readme
- 3.[Download source text from gutenberg](#)
4. Run scripts to get output.



A screenshot of a terminal window titled "In-Dialogue — bash — 108x57". The window has four tabs: "python", "bash", "bash", and "bash". The active tab is the first "bash" tab. It contains a single line of text starting with a dollar sign (\$) and a small square icon. The background of the terminal is yellow.

Used wget to download source

Errors:

Needed a source directory
Misunderstood how to run it.

[Youtube walkthrough for this step](#)

Go to Project Gutenberg and get more texts*, save to raw directory (see video in next slide):

Project Gutenberg's Prufrock and Other Observations, by T. S. Eliot

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org

Title: Prufrock and Other Observations

Author: T. S. Eliot

Posting Date: August 27, 2008 [EBook #1459]

Release Date: September, 1998

Language: English

Character set encoding: ASCII

*** START OF THIS PROJECT GUTENBERG EBOOK PRUFROCK AND OTHER OBSERVATIONS ***

Produced by Bill Brewer

PRUFROCK AND OTHER OBSERVATIONS

By T. S. Eliot

To Jean Verdenal 1889-1915

Certain of these poems appeared first in "Poetry" and "Others"

Contents

- The Love Song of J. Alfred Prufrock
- Portrait of a Lady
- Preludes
- Rhapsody on a Windy Night
- Morning at the Window
- The Boston Evening Transcript
- Aunt Helen
- Cousin Nancy
- Mr. Apollinaire
- Hysteria

You use the wget command to get the texts off Gutenberg
wget (Gutenberg URL) -O (filename in raw/ directory)
wget <http://www.gutenberg.org/files/1459/1459.txt> -O raw/1459.txt.utf-8

*make sure you use the same naming scheme

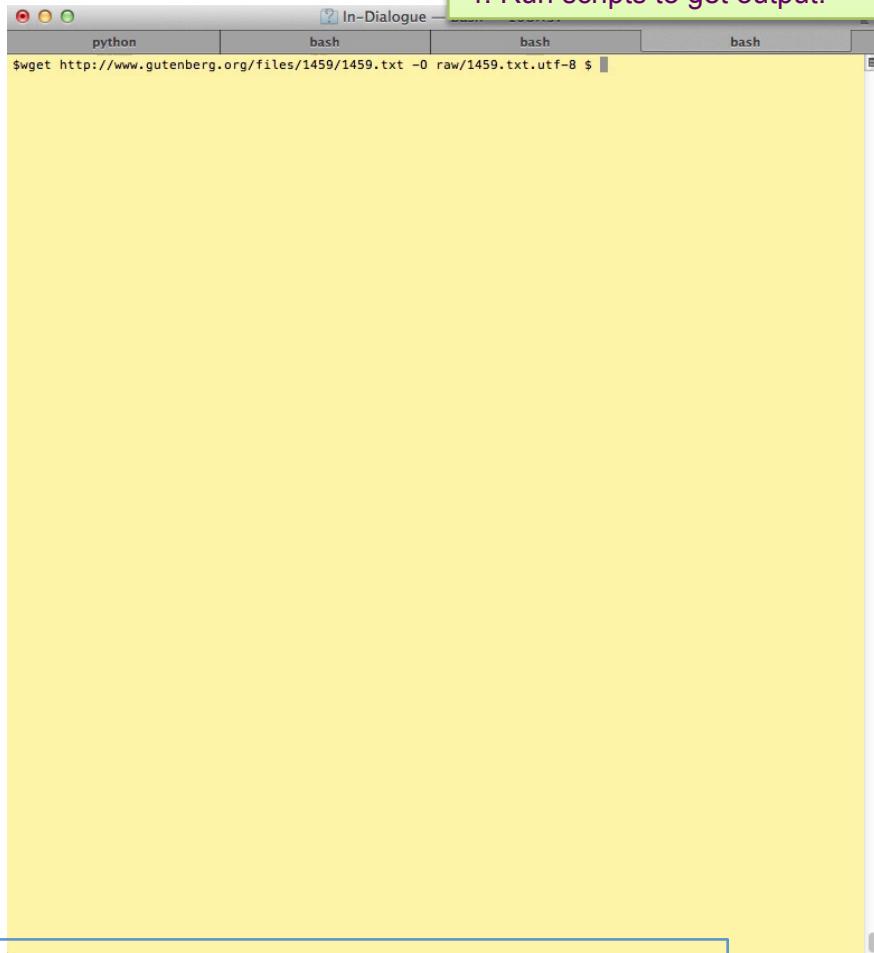
You will then use this command to extract the dialogue:
.0-extract-dialogue.py

It's
processin
g the
dialogues

```
anjchangmacbookpro:In-Dialogue anjchang$ ./0-extract-dialogue.py
Processing 11.txt.utf-8
EBOOK ALICE'S ADVENTURES IN WONDERLAND ***
Single quotes. (533 single, 2 double)
ALICE'S ADVENTU
Processing 12.txt.utf-8
EBOOK THROUGH THE LOOKING-GLASS ***
Single quotes. (633 single, 3 double)
THROUGH THE LOOKING-GL
Processing 1342.txt.utf-8
EBOOK PRIDE AND PREJUDICE ***
Produced by Anonymous Volunteers
Double quotes. (0 single, 1249 double)
Produced by Bill Brewer
Processing 1459.txt.utf-8
EBOOK PRUFROCK AND OTHER OBSERVATIONS ***
Produced by Bill Brewer
Double quotes. (0 single, 0 double)
Produced by Jim Tinsley
Processing 1727.txt.utf-8
EBOOK THE ODYSSEY ***
Produced by Jim Tinsley
Double quotes. (7 single, 466 double)
THE O
Processing 2701.txt.utf-8
EBOOK MOBY DICK; OR THE WHALE ***
Produced by Daniel Lazarus and Jones
Double quotes. (6 single, 969 double)
Produced by Jose Menendez
Processing 46.txt.utf-8
EBOOK A CHRISTMAS CAROL ***
Produced by Jose Menendez
A CHRIS
Double quotes. (1 single, 434 double)
Processing 730.txt.utf-8
EBOOK OLIVER TWIST ***
Produced by Peggy Gaugy and Leigh Little.
HTML
Single quotes. (2625 single, 5 double)
anjchangmacbookpro:In-Dialogue anjchang$
```

Go to Project Gutenberg and get more texts*, save to raw directory (see video):

1. Download source
2. Look at readme
3. Download source text from gutenberg
4. Run scripts to get output.



```
python           bash           bash           bash
$wget http://www.gutenberg.org/files/1459/1459.txt -O raw/1459.txt.utf-8 $
```

Helpful commands:

(command) > file.txt

redirects output from the command line to a file

cd – change to directory

mkdir – make a new directory

ls – list files on Mac

dir – list files on PC

pwd – print working directory (Mac)

open . -- open finder on Mac

explorer . -- open explorer on Windows

[Youtube walkthrough video for this step](#)

Merge the two books in your command line:

./1-generate-book.py "A Christmas Twist" 46 730 > ChristmasTwist.html

./1-generate-book.py "Pride Twist" 1342 730 > Pride_Twist.html

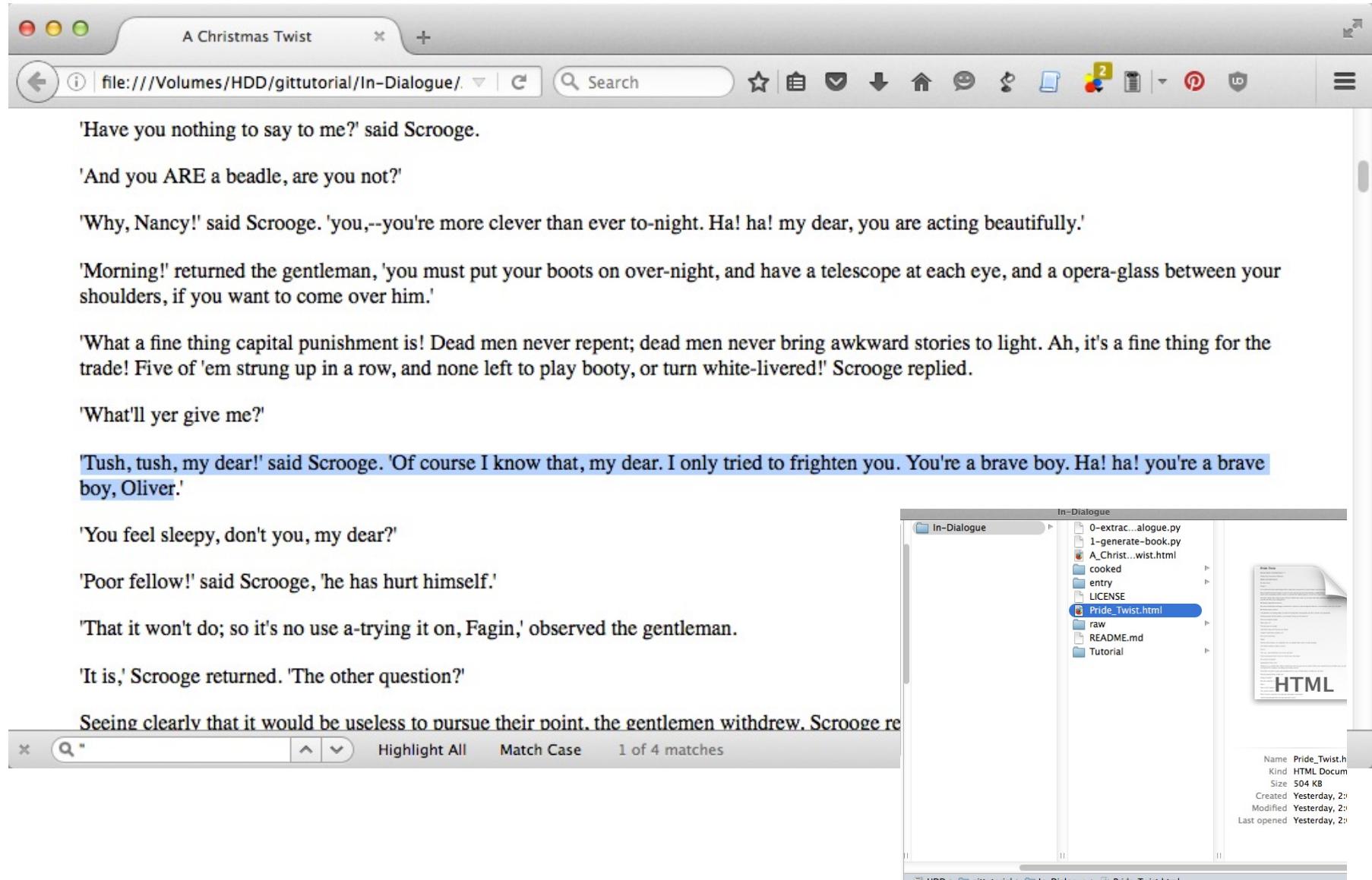
You decide the name

Number of the text source

Number of the dialogue source

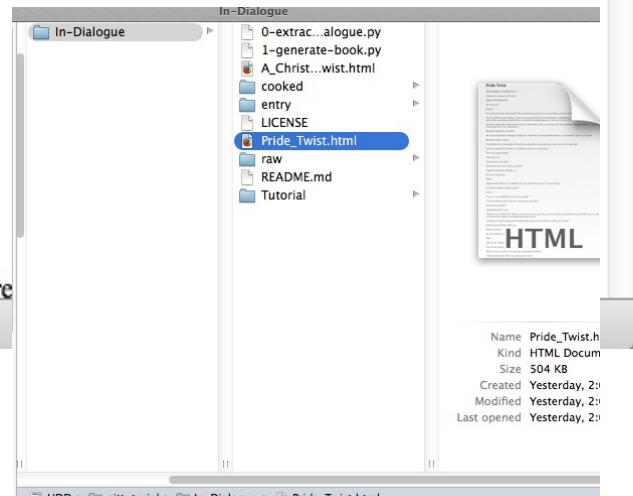
Name the output of the html file

Check the output:



A screenshot of a web browser window titled "A Christmas Twist". The address bar shows the URL "file:///Volumes/HDD/gittutorial/In-Dialogue/". The page content is a transcript of a dialogue between Scrooge and Nancy.

'Have you nothing to say to me?' said Scrooge.
'And you ARE a beadle, are you not?'
'Why, Nancy!' said Scrooge. 'you,--you're more clever than ever to-night. Ha! ha! my dear, you are acting beautifully.'
'Morning!' returned the gentleman, 'you must put your boots on over-night, and have a telescope at each eye, and a opera-glass between your shoulders, if you want to come over him.'
'What a fine thing capital punishment is! Dead men never repent; dead men never bring awkward stories to light. Ah, it's a fine thing for the trade! Five of 'em strung up in a row, and none left to play booty, or turn white-livered!' Scrooge replied.
'What'll yer give me?'
'Tush, tush, my dear!' said Scrooge. 'Of course I know that, my dear. I only tried to frighten you. You're a brave boy. Ha! ha! you're a brave boy, Oliver.'
'You feel sleepy, don't you, my dear?'
'Poor fellow!' said Scrooge, 'he has hurt himself.'
'That it won't do; so it's no use a-trying it on, Fagin,' observed the gentleman.
'It is,' Scrooge returned. 'The other question?'
Seeing clearly that it would be useless to pursue their point, the gentlemen withdrew. Scrooge re



The file explorer shows the directory structure "In-Dialogue" containing files like "0-extrac...logue.py", "1-generate-book.py", "A_Christ...wist.html", "cooked", "entry", "LICENSE", "Pride_Twist.html" (which is selected), "raw", "README.md", and "Tutorial". A preview pane shows the generated HTML file "Pride_Twist.html". The file details are listed on the right:

Name	Pride_Twist.h
Kind	HTML Docum
Size	504 KB
Created	Yesterday, 2:29
Modified	Yesterday, 2:29
Last opened	Yesterday, 2:29

Path: HDD > gittutorial > In-Dialogue > Pride_Twist.html

Check the output:

A screenshot of a Mac OS X desktop environment. At the top, there is a Dock with several icons. Above the Dock, the system menu bar shows the date ('Tuesday, 10 May 2011') and other system status indicators. Below the menu bar, a browser window is open with two tabs: 'A Christmas Twist' and 'Pride Twist'. The 'Pride Twist' tab is active. The address bar shows the URL 'file:///Volumes/HDD/gittutorial/In-Dialogue/'. The main content area of the browser displays a series of dialogue exchanges from a text. A search bar at the bottom of the browser window contains the word 'oliver'.

res. Leaning on your arm.

'I think I know that,' said Charlotte. 'It's a the--; you're one, are you not?'

'Yes, there's some half-dozen of 'em gone in, that I knows. I don't think your friend's there.'

'Not here,' said Jane. 'I am afraid to speak to you here. Come away--out of the public road--down the steps yonder!'

'What now?'

'It was quite true,' said Jane, 'that they must know them before long, but it might be at a better time than the present, and it could not be at a worse.'

'I hope not, Oliver. I have been very happy with her for some years: too happy, perhaps. It may be time that I should meet with some misfortune; but I hope it is not this.'

'You will see him soon,' said Miss Lucas, 'You shall tell him how happy you are, and how rich you have grown, and that in all your happiness you have none so great as the coming back to make him happy too.'

'You ought to be dead; positively dead with the fright,' said her mother, 'Why didn't you send? Bless me, my man should have come in a minute; and so would I; and my assistant would have been delighted; or anybody, I'm sure, under such circumstances. Dear, dear! So unexpected! In the silence of the night, too!'

'This is young Oliver Twist, whom we were speaking about,'

'Every word!' said Miss Lucas, 'every word that has passed between you and this detested villain, is known to me. Shadows on the wall have

oliver

Highlight All Match Case 1 of 36 matches

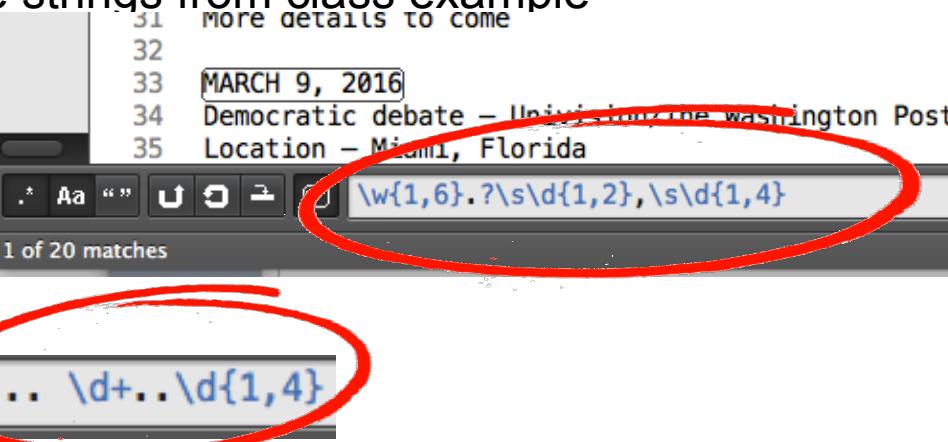
Regex Group Exercises

10 minutes, if we have time...

Grab a generated book mash-up from a friend. Write a regular expression that will find some the occurrences of different parts: numbers and/or dates (days of the week or months or years) but not other parts of the text. Try out your regular expression first in a text editor to ensure that it works. The right answer will depend on the date format used in your particular data.

Share with the class what expression you used.

Date strings from class example



The screenshot shows a text editor interface with several lines of text and two red circles highlighting specific regular expression patterns in the search bar.

Lines of text:

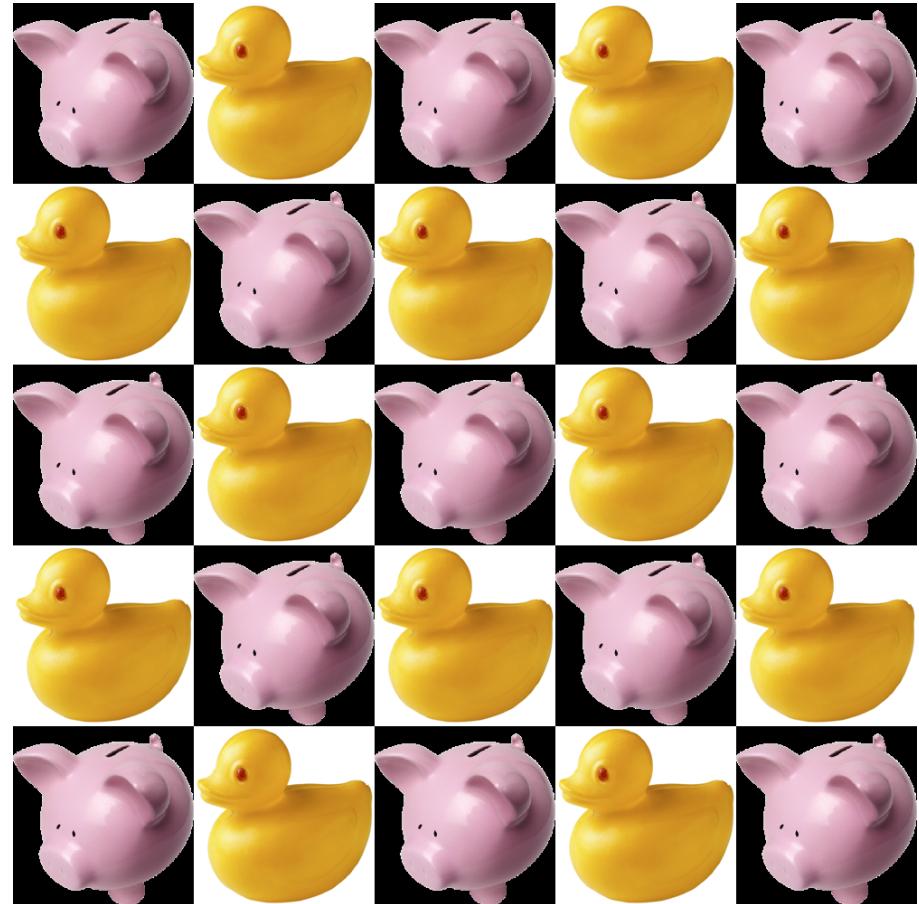
- 31 more details to come
- 32
- 33 MARCH 9, 2016
- 34 Democratic debate – Univision/the washington Post
- 35 Location – Miami, Florida

Search bar (top): `\w{1,6}.\?s\d{1,2},\s\d{1,4}`

Search bar (bottom): `.... \d+..\d{1,4}`

Ps Zip and upload your class files for credit,
and name your collaborators on the documentation.





IMAGES IN PYTHON

Learning about libraries and docstrings

To get a library, you use `import` statement with `library:name`

```
import re
```

To list its the functions:

```
dir(re)
```

```
dir([])
```

Doc strings = “documentation strings”, messages about functions

To read a docstring

```
Object.__doc__
```

note the double underscores

```
print [].count.__doc__
```

```
print len.__doc__
```

To create a docstring when you write a function, just it on a newline after `def`

```
#write a docstring
def tax(subtotal):
    'returns the tax on a meal in Massachusetts'
    return subtotal * 0.0625
```

Python libraries

- Using libraries gives us special functions.

PIL
python imaging library

other libraries we have seen

webbrowser re random math

From 1D to 2D

Extending from our work with numbers and lists...

Number
single entity

20 3.14159 -13

Numbers ... -3,-2,-1,0,1,2,3...
Lists, sequences

Points on a plane
 $(0,0),(0,1),(1,0),(1,1)$

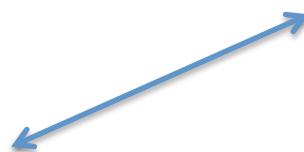


Point



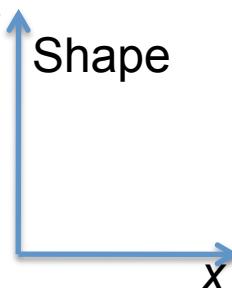
“No length, no width”

Line



“length or width”

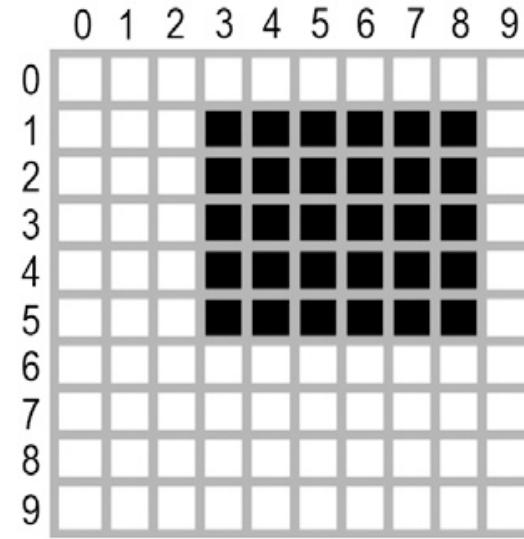
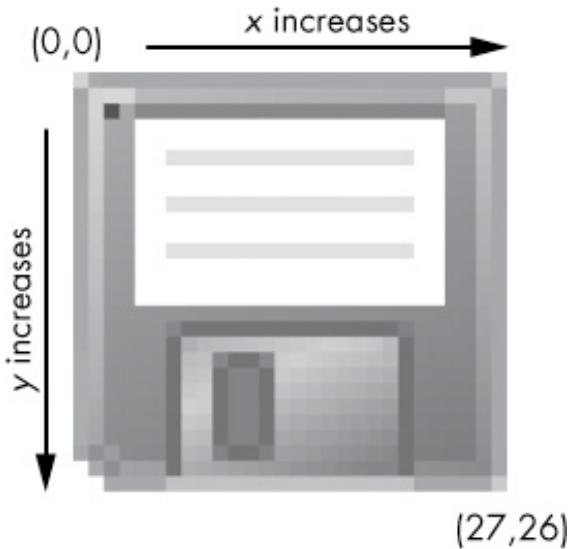
Shape



“length and width”

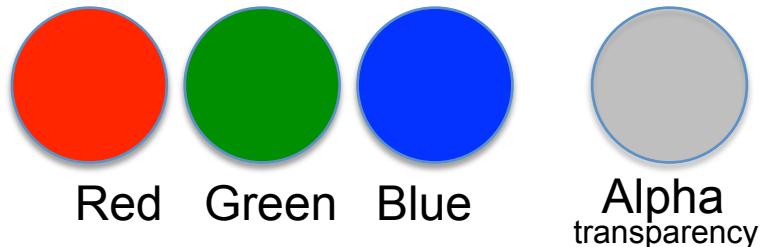
Representations in different domains along increasing dimensions

Images



Area is represented by the box tuple (3,1,9,6)

Each pixel consists three (or four channels):



Each channel has a value:

0 1 2 3 4 5 6 7 (2^3) or 8 bits = 1 byte
[Eight empty boxes] 0-255 (0-FF hex)

Hex #FFFFFF is white
-Full red, green, and blue

Tuples

pairs, triples, quadruples, and so on.

Specified by rounded parenthesis:

(tuple)

Puts things together, but is immutable (once created it does not change).

Create some tuples in iPython notebook:

```
twod = (42,17) #note parenthesis for tuples
```

```
threed = (7,6,14)
```

```
twod
```

```
threed
```

Like lists, but you can't change them.

Tuples vs Lists

```
twod[-1]
```

```
threed[-1]
```

```
threed[0] = 9
```

Guess what's
going to
happen before
you press
enter!

```
alist = [1,2,3]
```

```
alist[0]=15
```

```
alist
```

Does
change
happen?

```
for i in range(5): #can make a lot of tuples by iterating
    print (i,2,3)
```

Tuples are more computationally efficient.
Useful for very large numbers of numbers....
like in an image.

A new type also helps us prevent errors.

```
def to_f(c):
    return (9/5)*c + 32
```

```
to_f("hello") #helpful error
```

But, can we iterate over it?

Double

```
#double double from earlier
def double(sequence):
    result = []
    for element in sequence:
        result = result + [element*2]
    return result
```

```
double(alist)
```

```
[2, 4, 6]
```

```
double((2,3,4))
```

iterating over a tuple here!

```
[4, 6, 8]
```

So we can use it to iterate computations---
we can work on bitmap images this way,
just need 2 iterators to go in each x and y direction.

Get a PIL and create an image

Python Imaging Library

```
#import the PIL Library  
from PIL import Image  
  
ourimage = Image.new('RGB',(100,100), 'white')  
#mode = 'RGB'  
#size = (100,100)  
#color = 'white'
```

Did you get an error?

NO

```
#save the images  
ourimage.save("allwhite.png")
```

Switch to your directory and admire your handiwork. Try changing values.

YES



NameError: name 'Image' is not defined

If you see 'Image' not defined – that means python doesn't have the Image library. Import it again using

```
from PIL import Image
```

Then try again.

[Image link](#)

Image creation

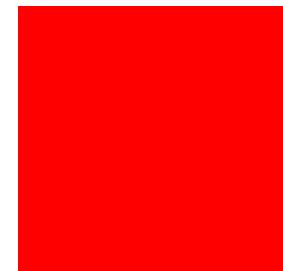
```
ourimage = Image.new('RGB', (100,100), 'white')
```

```
mode = 'RGB' #the format could also take  
size = (100,100)  
color = 'red'
```

```
ourimage2 = Image.new(mode,size,color)
```

```
ourimage2.save('allred.png')
```

Make some images.
Admire your handiwork.



Which number is width, which is height in size?

Try using tuples for the color.

```
ourimage = Image.new('RGB', (150,100), 'blue')
```

```
ourimage.save('allblue_notsquare.png')
```

```
ourimage = Image.new('RGB', (100,100), (127,127,127) )
```

```
ourimage.save('allgray.png')
```

```
ourimage = Image.new('RGB', (100,100), (200,127,127) )
```

```
ourimage.save('allrose.png')
```

Changing images

```
allblack = Image.new('RGB', (100,100), (0,0,0))
```

```
allblack.save("allblack.png")
```

```
allblack.putpixel((50,50), (255,255,255))
```

```
allblack.save("almostall.png")
```

```
#try drawing a line
```

```
for i in range(100):
    allblack.putpixel((i,25), (255,255,255))
```

```
allblack.save('oneline.png')
```

Can you draw a line in the other direction?

Iterating through all the pixels

```
#iterate through the entire image
rectangle = Image.new('RGB', (150,100), (0,0,0))
```

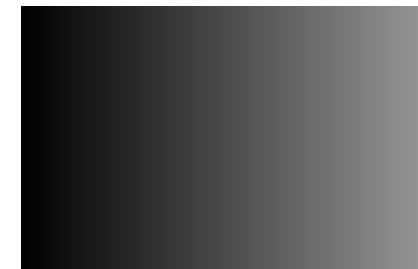
```
for x in range(150):
    for y in range(100):
        rectangle.putpixel((x,y),(127,127,127))
```

```
rectangle.save('rectangle.png')
```

```
#now change the color of each pixel
```

```
for x in range(150):
    for y in range(100):
        rectangle.putpixel((x,y),(x,x,x))
```

```
rectangle.save('gradient.png')
```



Now try a vertical gradient or even a diagonal gradient.

Generalizing image operations

Need to get image properties.



```
#generalize to images of any size  
rectangle.size
```

```
rectangle.size[0]
```

```
rectangle.size[1]
```

```
bigger = Image.new('RGB', (500,600), (0,0,0))
```

```
bigger.size
```

```
bigger.size[0]
```

```
bigger.size[1]
```

```
# bundle grayout function generalized for images of different sizes  
def grayout(pngimage):  
    for x in range(pngimage.size[0]):  
        for y in range(pngimage.size[1]):  
            pngimage.putpixel((x,y),(127,127,127))
```

```
test = Image.new('RGB', (150,100), (0,0,0))
```

```
grayout(test)
```

```
test.save('test1.png')
```

Load and save images

```
#loading an existing image  
ourimage = Image.open('heart.png')
```

```
ourimage.save('ours.png')
```

```
#check that these are the same image  
#now tryout our grayout function  
grayout(ourimage)
```

```
ourimage.save('ournew.png')
```

```
#check that this new image is all grayed out  
#otherwise check your function or restart ipython
```



Now we can write functions that operate on images.

Lighten or darken an image

Need to look at pixels.

```
ourimage.getpixel((0,0))    #get a pixel from the image  
  
transparent = Image.open('heart_trans.png')  
  
transparent.getpixel((0,0))
```

```
#fill in the blanks to generalize the size  
def modify(pngimage):  
    for x in ____:  
        for y in ____:  
            (r,g,b) = pngimage.getpixel((x,y))  
            pngimage.putpixel((x,y),(r,g,b))
```

Not quite modify:

```
def modify(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))
            pngimage.putpixel((x,y),(r,g,b))
```

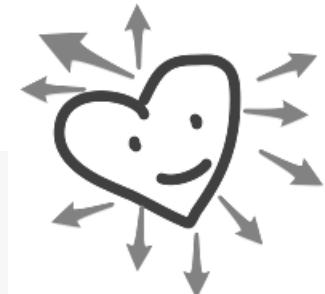
```
ourimage = Image.open('heart.png')
```

```
modify(ourimage)
```

```
#verify it gets saved out
ourimage.save('modheart.png')
#now actually modify the function to change the intensity
#by lightening the image\ procesing\ 1.ipynb
```

Class 6 stopped here! Please try the following things at home!

Lighten the image



```
def modify(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))
            pngimage.putpixel((x,y),(r+64,g+64,b+64))
```

```
modify(ourimage)
ourimage.save('modheart.png')
```

```
#note no errors if r,g,b > 255
#putpixel uses saturation arithmetic
```



```
#generalize so modify works with transparent images too
def modify(pngimage):
    for x in range(pngimage.size[0]):
        for y in range(pngimage.size[1]):
            (r,g,b) = pngimage.getpixel((x,y))[:3] #get colors only
            new_color = ( r+64, g+64, b+64)
            new_color = new_color + pngimage.getpixel((x,y))[3:]#add transparency
            pngimage.putpixel((x,y),new_color)
```

Try it out...

```
modify(ourimage)  
  
ourimage.save('modheart2.png')  
  
modify(transparent)  
  
transparent.save('modtransparent.png')
```



Now try darkening the image....

```
def darken(pngimage):  
    for x in range(pngimage.size[0]):  
        for y in range(pngimage.size[1]):  
            (r,g,b) = pngimage.getpixel((x,y))[:3] #get colors only  
            new_color = ( r-64, g-64, b-64)  
            new_color = new_color + pngimage.getpixel((x,y))[3:]#add transparency  
            pngimage.putpixel((x,y),new_color)
```



Summary of today

- Quiz – hope you feel good about it
- Textual analysis of large files using regex
 - Grabbing source code off the web
 - Running another person's code
 - Using regex expressions
- Introduction to working with PIL library
- Class Participation – Upload a zip file with your Jupyter notebook from today, along with generated text and images
- Homework - Readings on technology and questions