Sentiment Analysis

So, how does that make you feel?

Today's Plan:

- Sentiment Analysis
    - What even is this thing?
    - Why do I care?
    - What am I doing again?
- Python
    - Variables, strings etc
    - If/Else statements, Loops
    - Lists and Dictionaries
    - Reading Files
    - Other resources

Sentiment Analysis

What even is this thing?

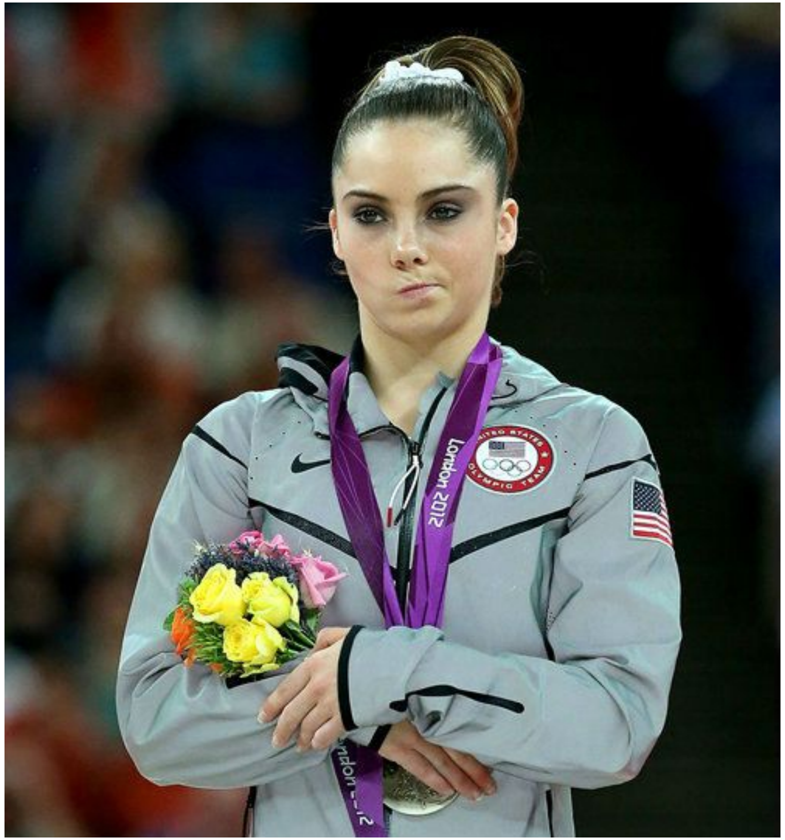
- "The use of text analysis to identify and extract subjective information in source materials" (thanks Wikipedia)
- Essentially, searching for subjective things in a text
- e.g. Is this product review positive or negative?
  e.g. Is this book romantic, and should I give it to my romance-obsessed friend?

Turns out, words carry meaning beyond just their dictionary definition.

(Source: http://spotfire.tibco.com/blog/wp-content/uploads/Sentiment-Analysis-300x199.jpg)

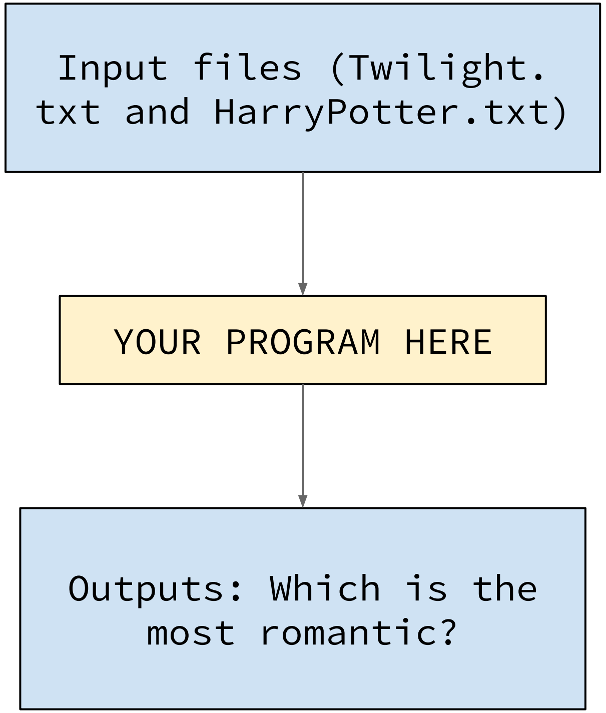
Why do I care?

- Because you’ll win points for your team AND learn cool things
- Because it’s an interesting problem (or at least I think so)
- It’s something that machines are bad at, but humans are good at
- It’s used a lot in real life
  - Automatically checking for poor reviews of a product on the internet
  - Assessing suitability of videos/webpages for children



What Am I Doing Again?

- Goal: To make a program that
  - Takes in two files (the text of Twilight and Harry Potter)
  - Decides which is the most romantic (swoon)
- Bonus points awarded for:
  - Returns percentage romanticism along with which is the most romantic
  - Analyses for sentiments other than romanticism
  - Analyses more than two texts
  - ??? (You decide and tell me)


Input files (Twilight.txt and HarryPotter.txt)
YOUR PROGRAM HERE
Outputs: Which is the most romantic?

Any hints?

- I’d recommend structuring your program like so:
    a. Decide which words indicate romanticism, store them somewhere
    b. Read in files
    c. Count presence of keywords that indicate romanticism
    d. Check which has more of these
    e. Print the name of the most romantic text
- Ultimately it’s up to you though!

Other hints:

- Make sure to look at the Python notes in the rest of the booklet for info and syntax things. Everything you need to make this program is discussed there.
- Don't be afraid to ask for help; we don't bite students!

Python Basics (Variables, Strings, and other such things)

Variables

- Place for storing values for later use
- We essentially “name” a value
- This helps avoid repetition!
- Reassigning a variable changes its value to something new
  - Comment: Python will let you change x=4 to x=”hi”, even though “hi” is not a number.
  - Many other languages consider that not-okay!

Examples:

```
x=10

y=1.45

z=“Hello World!”

x=“Goodbye.”

z=45
```

Question: What are x, y and z at the end of this block?

Arithmetic

- All the normal arithmetic operations (+,-,*,/) can be used with numbers
    - This includes both constants and variables
    - The RHS gets evaluated, then stored in the LHS variable
    - Using these operations with non-number variables may either not work or give you nonsense!

Examples:

```
x=10+2

y=1.45*2.89

z=3*4

x=z/6

z=18-z
```

Question: What are x, y and z at the end of this block?

Strings

- Surrounded by “” or ‘’
- Contain letters, numbers, punctuation, spaces, etc
- The individual letters, digits, symbols and spaces are called *characters*
- The word string is short for *string of characters*.

Examples:

```
print('abc ABC 123 @!?.#')

print("This message contains
'single quotes'.")
```

Joining Strings

- To join strings, use + (but note that this prints with no space between the strings!)
- You can also join variables and strings to print them!
  - Note: if these variables are not strings themselves, you'll need to coerce them to one (see the last example)

Examples:

```
print('Harry' + ' ' + 'Potter')

name = 'Vernon Dursley'
print(name + ' is a muggle!')


number = 4
print(“Cats? I have ‘ + str
(number))
```

Question: What will these print?

If/Else statements & Loops

Comparisons

- Comparisons are the basis of if/else statements, and most loops
- They allow us to set conditions, and thus alter what the program does based on inputs!
- They are subtly different to assigning a value to a variable

Example:

```
x = “Lumos”
x == “Lumos”
```

- A single = is used for *assignment*.
  - This is what we do to set variables. The program to the left is setting the variable x to the value "Lumos".
- A double == is used for *comparison*.
  - This is what we do to check whether two things are equal. The second line of the program to the left is checking whether x is equal to "Lumos" using a double equals sign. (it does, so it will return True)

Comparing part 2

- Comparisons work on both strings and numbers
- You can compare strings using ‘<’ and ‘>’! What it does is assess the “value” of each letter, based on dictionary order (so ‘a’ > ‘b’ will return false, and ‘abc’ < ‘acb’ will return true)
    - What do you think ‘a’ < ‘A’ will return?

How do we compare things?

Operation	**Symbol**
equal to	==
not equal to	!=
less than	<
less than or equal to	<=
greater than	>
greater than or equal to	>=

Strings and Case

- Strings are special when it comes to comparisons, because they have case
- Do you think ‘a’ == ‘A’ will return true?
  - I mean, they are different characters…
- So for comparisons to work how we expect them to, we need to make sure that all our characters are in the same case
  - Thankfully, Python has something that does this for us!

Examples:

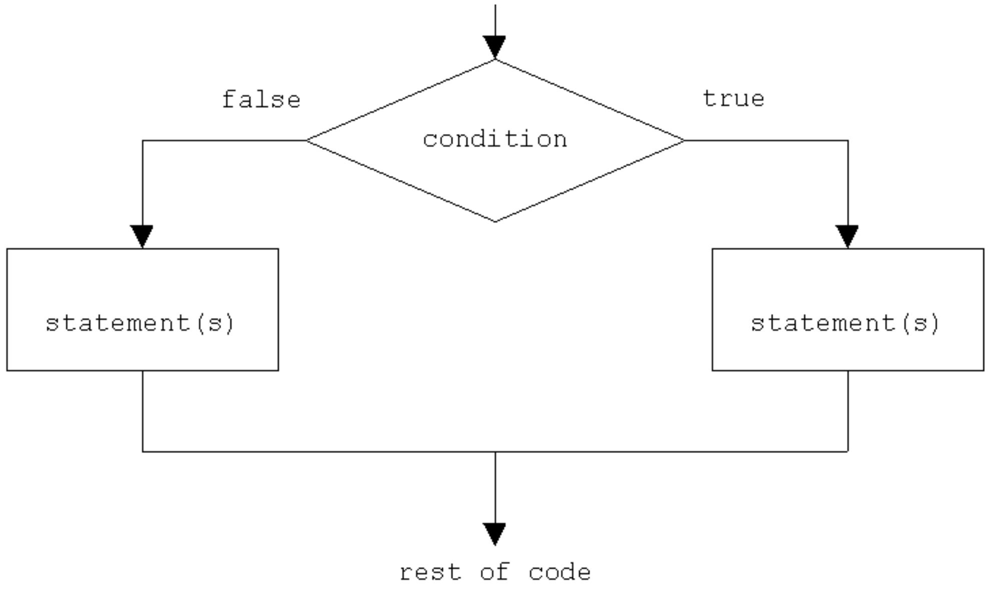
```
words = “I like Pie!”
print(words.lower())
```

```
words = “I like Pie!”
print(words.upper())
```

Question: What will these print?

If/Else Statements

- If/Else statements allow us to make decisions (just like in real life)
- These are important, as they allow us to “skip” steps that aren’t relevant to our situation
    - e.g. if it isn’t raining, I don't need to get an umbrella



Source: http://dotprogramming.blogspot.com/2013/11/if-ifelse-statement-example-turbo-c.html

Constructing if/Else Statements

- If/Else statements (and other control structures* later on), control a chunk of code called a block
- Python shows these blocks via indentation
    - Note: Indentation needs to be the same for every block in the program, otherwise you'll get errors!

*Control structures are things like if/else statements or loops which control the direction the program takes.

Example:

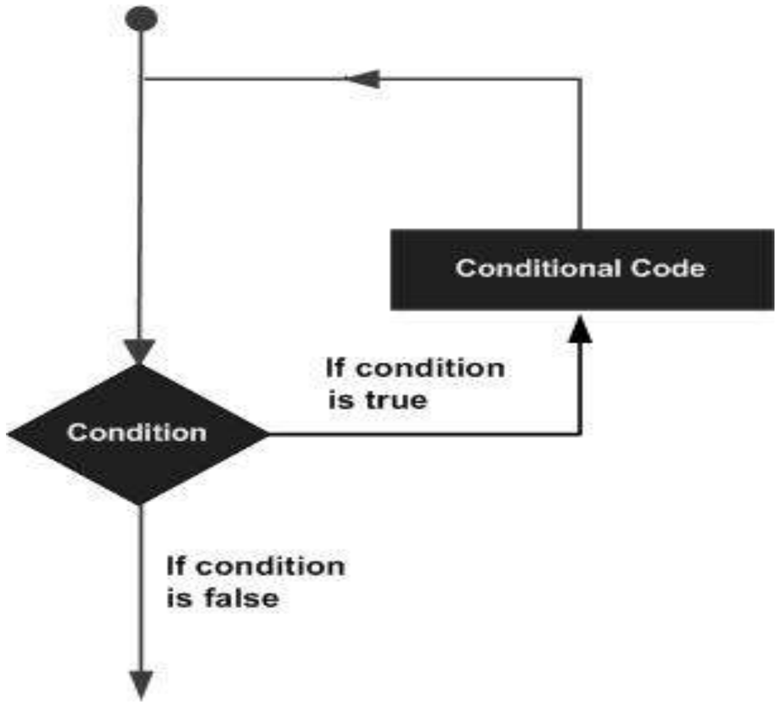
```
name = ‘Gerald’

if name == ‘Gerald’
    print(‘That’s my name too!’)
else:
    print(‘Pleased to meet you.’)
```

Question: What happens if name isn’t Gerald?

Bonus: What happens if name = ‘gerald’

Loops

- Loops are another control structure, that saves you from hitting “copy, paste” over and over to get something done
- As long as the condition is met, they’ll keep doing the same block of code over and over and over and over…



Source: http://www.tutorialspoint.com/computer_programming/computer_programming_loops.htm

Loop Implementation

- Loop implementation differs depending on whether we're using numbers or not
- If we know how many times we want something done, we can use range
  - Note that the range goes from start to end-1, not start to end like you may expect!
- If we have a string, we can use a for loop to go through each letter in the string!

Example:

```
for n in range(5, 8):
    print(n)

word = “expelliarmus”
for letter in word:
    print(letter)
```

Question: What will these print?

Lists

Lists

- Lists allow us to store multiple input elements in one place!
  - This can be very useful when we have a sentence and we want to look at each word individually
  - Or if we have a set of words/numbers we want to keep together
- We can either make lists from a string, or construct them ourselves
- To cycle through the elements in our list, we use a for loop, similar to how we go through a string (see example on the right)

Example:

```
data = “english maths geography”
subjects = data.split()
for subject in subjects:
    print(subject)

odds = [1, 3, 5, 7, 9]

sadWords = [‘sad’, ‘gloomy’, ‘cry’]
```

Reading (and using) Files

Reading From Files

- Python makes reading from files very simple (yay!)
- Before a program can read data from a file, it must tell the operating system that it wants to access that file.
- Files sitting in the same directory as the running program can be referred to just using the *filename*, e.g. test.txt.
    - This is the setup we will use here.

Example:

```
fileone = open('fileone.txt')
```

- Note: If the file doesn't exist, you'll get an error message saying the file wasn't found
    - This could be because the file isn't there at all, or because it's in a different folder
    - It's easiest to make sure the program and file are together!

Looping Over File Contents

- You can treat the file lines just like a list in a for loop!
- The biggest difference is removing whitespace
  - We do this using two functions, strip() and split()
  - strip(): removes newlines (“enter” at the end of a line) from our string
  - split(): splits the line into a list based on whitespace (so we can look at each word individually)

Example:

```
f = open('words.txt')
for line in f:
  print(line.strip())
  words = line.split()
  print(words)
```

Question: What will this code print, if words.txt has only one line: “Yer a wizard, Harry.”

Extra Info and Other things

Other resources for Today

- Our workspace for the day: http://www.tutorialspoint.com/ipython_terminal_online.php
- Our glorious texts:
    - Twilight: http://insearchofspoons.com/static/Twilight.txt
    - Harry Potter: http://insearchofspoons.com/static/HarryPotter.txt
- Python documentation: https://www.python.org/doc/

Other Things (kinda) like This:

Things on the internets:

- Rosalind: platform for learning bioinformatics and programming through problem solving http://rosalind.info/
- Australian Informatics Olympiad: Annual competition, teaching both coding and algorithms. Very self directed, but very rewarding http://orac.amt.edu.au/
- NCSS Challenge: 6-week online Python programming competition that teaches you Python as you compete in it. http://www.groklearning.com/challenge
- Codeacademy: An online learn-to-code website that teaches HTML, Javascript and Ruby on Rails. http://www.codecademy.com/

Things IRL:

- Honeywell Engineering Camp: 6-day summer campthat introduces students to the university degrees and careers available to professional engineers in industry. http://www.engineersaustralia.org.au/sydney-division/honeywell-summer-school
- Hour of code: 1-hour introduction to computer science, designed to demystify code and show anybody can learn the basics. https://hourofcode.com/au
- RoboGals: simple, fun and FREE NXT LEGO robotics workshops for school girls either at UNSW or school http://sydney.robogals.org.au/
- Rails Girls: Community events by women, for women, learning Ruby. http://railsgirls.com/
- NCSS: 10-day summer camp where students complete an intensive course of programming and web dev. http://www.ncss.edu.au/