Computing to the max

The not-so-subtle art of singling out the best (and worst) of anything...

Computing with language

- What's in a Writ1 paper, anyway?
- Battle-tested ciphers & how to break them...

Last hw?

N-step sleepwalking?

Turtle graphics??

Artistic renderings!!!

This week!

Hw #3 due next Monday...

pr0: Are we The Matrix?
pr1: Lab: sounds good...
pr2: Sorting + Caesar!
ex cr: Add'l UIOLI fun'!
A recipe for life?
max

A recipe for life?
and python already has it for us...

The real problem is knowing **what** we want to maximize!
A recipe for life?
and python already has it for us...

The real problem is knowing what we want to maximize!

... or minimize, with min
to the **max**

Want the highest price?

\[
\text{max}( [475.5, 458.0, 441.3, 470.8, 532.8, 520.9] )
\]

What if the months are in there, as well?

\[
\text{max}( [ [470.8,'may'], [532.8,'jul'], [520.9,'sep'] ] )
\]

\[
\text{max}( [ ['may',470.8], ['jul',532.8], ['sep',520.9] ] )
\]
to the $\max$

Want the highest price?

$$\max( \ [475.5, 458.0, 441.3, 470.8, 532.8, 520.9] \ )$$

What if the months are in there, as well?

$$\max( \ [[470.8,'\text{may}'], [532.8,'\text{jul}'], [520.9,'\text{sep}']] \ )$$

$$\max( \ [['\text{may}',470.8], ['\text{jul}',532.8], ['\text{sep}',520.9]] \ )$$

Mudd's $\max$?

$$L = [\text{'Harvey'}, \text{'Mudd'}, \text{'College'}, \text{'seeks'}, \text{'to'}, \text{'educate'}, \text{'engineers'}, \text{'scientists'}, \text{'and'}, \text{'mathematicians'}, \text{'well-versed'}, \text{'in'}, \text{'all'}, \text{'of'}, \text{'these'}, \text{'areas'}, \text{'and'}, \text{'in'}, \text{'the'}, \text{'humanities'}, \text{'and'}, \text{'the'}, \text{'social'}, \text{'sciences'}, \text{'so'}, \text{'that'}, \text{'they'}, \text{'may'}, \text{'assume'}, \text{'leadership'}, \text{'in'}, \text{'their'}, \text{'fields'}, \text{'with'}, \text{'a'}, \text{'clear'}, \text{'understanding'}, \text{'of'}, \text{'the'}, \text{'impact'}, \text{'of'}, \text{'their'}, \text{'work'}, \text{'on'}, \text{'society'}]$$

max(MSt)

Or Mudd's $\min$?

$$\min(\text{MSt})$$
Is 42 really better than 47?

Michelle Timmins

to me

Hi Prof. Dodds--

I noticed some things in this week's article that I think are very interesting! The article on Watson's Jeopardy success has the following lines:

"The final tally was $77,147..." (2).
"It had wagered $947 on its result" (2).

I didn't notice any 42s....just saying...

-Michelle Timmins (Pomona student and 47 supporter)

42? 47? Others?
min + max?  Thanks, Eli!

---

Eli Fujita

to me

Hi Prof. Dodds!

"Pitzer College produces engaged, socially responsible citizens of the world through an academically rigorous, interdisciplinary liberal arts education emphasizing social justice, intercultural understanding and environmental sensitivity. The meaningful participation of students, faculty and staff in college governance and academic program design is a Pitzer core value. Our community thrives within the mutually supportive framework of The Claremont Colleges, which provide an unsurpassed breadth of academic, athletic and social opportunities."

---

CMC 40's #: 46

<table>
<thead>
<tr>
<th>Claremont McKenna College / Founded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1946</td>
</tr>
</tbody>
</table>
def max( L ):
    """ returns the max element from L
    input: L, a nonempty list
    """
    if len(L) < 2:  return L[0]  # only 1 elem.
        maxOfRest = max(L[1:])  # max of the rest
        if L[0] > maxOfRest :
            return L[0]  # either L[0]
            else:
                return maxOfRest  # or maxOfRest!
def maxSS(L):
    """ returns L's highest scrabble-scoring element (input: L, a nonempty list) """
    if len(L) < 2:
        return L[0]  # only 1 elem.
    maxOfRest = maxSS(L[1:])  # rest's max
    if L[0] > maxOfRest:
        return L[0]  # either L[0]
    else:
        return maxOfRest  # or maxOfRest!
```python
def maxSS(L):
    """ returns L's highest scrabble-scoring element (input: L, a nonempty list) """
    if len(L) < 2: return L[0] # only 1 elem.
    maxOfRest = maxSS(L[1:]) # rest's max
    if sScore(L[0]) > sScore(maxOfRest):
        return L[0] # either L[0]
    else:
        return maxOfRest # or maxOfRest!
```

L = ['aliens', 'zap', 'hazy', 'code']
max with scrabble-score

def maxSS( L ):  # returns L's highest scrabble-scoring element (input: L, a nonempty list)
    if len(L) < 2:
        # only 1 elem.
        return L[0]
    maxOfRest = maxSS(L[1:])  # rest's max
    if sScore(L[0]) > sScore(maxOfRest):
        return L[0]  # either L[0]
    else:
        return maxOfRest  # or maxOfRest!

L = ['aliens', 'zap', 'hazy', 'code']

Let's see if we can simplify this process... just for LoLs!

which element has the highest scrabble score?
A more comprehensive solution: \texttt{LoL}

```
L = [ 'aliens', 'zap', 'hzy', 'code' ]

def \texttt{maxSS}( L ):
    """ returns L's max-scrabble-score word """

    \texttt{LoL} = [ [\texttt{sScore}(w), w] \texttt{for} w \texttt{in} L ]

    \texttt{bestpair} = \texttt{max}( \texttt{LoL} )

    \texttt{return} \texttt{bestpair}[1]
```

This \textit{does} look funny!
A more comprehensive solution

```python
L = ['aliens', 'zap', 'hazy', 'code']

def maxSS( L ):
    """ returns L's max-scrabble-score word """

    LoL = [ [sScore(w), w] for w in L ]
```

This does look funny!

Karen Carlson <kcarlson48@gmail.com>
to me ✅

Thanks for the email. I'll write you soon. Glad you made it home safely. Lol, mom
A more comprehensive solution

def maxSS(L):
    """ returns L's max-scrabble-score word
    """
    LoL = [ [sScore(w), w] for w in L ]
    bestpair = max( LoL )
    return bestpair[1]

This does look funny!

L = ['aliens', 'zap', 'hazy', 'code']
A more **comprehensive** solution

```python
def maxSS(L):
    """ returns L's max-scrabble-score word """
    LoL = [[sScore(w), w] for w in L]
    bestpair = max(LoL)
    return bestpair[1]
```
A more comprehensive solution

```python
L = ['aliens', 'zap', 'hazy', 'code']

def maxSS(L):
    """ returns L's max-scrabble-score word """

    LoL = [ [sScore(w), w] for w in L ]

    bestpair = max( LoL )

    return bestpair[1]

I loathe hazy code!
```

L = ['aliens', 'zap', 'hazy', 'code']

LoL = [ [6, 'aliens'], [14, 'zap'], [19, 'hazy'], [7, 'code'] ]

bestpair = [19, 'hazy']

return bestpair[1]

'hazy'
Everything ... is a max problem?

```python
def lastrest(L):
    """ another example – what's returned? """
    LoL = [ [w[1:], w] for w in L ]
    bestpair = max( LoL )
    return bestpair[1]

L = [ 'aliens', 'zap', 'hazy', 'code' ]
```

I know the best word here... but does Python?
Everything ... is a max problem?

```python
def lastrest(L):
    """ another example – what's returned? """

LoL = [ [w[1:], w] for w in L ]

bestpair = max( LoL )

return bestpair[1]
```

L = ['aliens', 'zap', 'hazy', 'code']

I know the best word here... but does Python?
Everything ... is a max problem?

```python
def lastrest(L):
    """ another example – what's returned? """
    LoL = [ [w[1:], w] for w in L ]
    bestpair = max(LoL)
    return bestpair[1]
```

I know the best word here... but does Python?

```
L = ['aliens', 'zap', 'hazy', 'code']
LoL = [ ['liens', 'aliens'], ['ap', 'zap'], ['azy', 'hazy'], ['ode', 'code'] ]
bestpair = ['ode', 'code']
```

Everything ... is a max problem?

```python
def lastrest(L):
    """ another example -- what's returned? """
    LoL = [ [w[1:], w] for w in L ]
    bestpair = max(LoL)
    return bestpair[1]
```

I know the best word here... but does Python?

```
L = [ 'aliens', 'zap', 'hazy', 'code' ]
LoL = [ [ 'aliens', 'aliens'], [ 'ap', 'zap'], [ 'azy', 'hazy'], [ 'ode', 'code'] ]
bestpair = [ 'ode', 'code']
return bestpair[1]
'code'
```
Everything ... is a max problem?

```python
def lastrevved(L):
    """ another example – what's returned? """
    LoL = [ [w[::-1], w] for w in L ]
    bestpair = max( LoL )
    return bestpair[1]
```

I know the best word here... but does Python?
Everything ... is a max problem?

```python
L = ['aliens', 'zap', 'hazy', 'code']

def lastrevved(L):
    """ another example – what's returned? """
    LoL = [ [w[::-1], w] for w in L ]
    bestpair = max(LoL)
    return bestpair[1]
```

I know the best word here... but does Python?

L = ['aliens', 'zap', 'hazy', 'code']

LoL = [ ['sneila', 'aliens'], ['paz', 'zap'], ['yzah', 'hazy'], ['edoc', 'code'] ]

bestpair = max(LoL)

return bestpair[1]
def lastrevved( L ):
    """ another example – what's returned? """
    LoL = [ [w[::-1], w] for w in L ]
    bestpair = max( LoL )
    return bestpair[1]
Everything ... is a max problem?

```python
def lastrevved(L):
    """
    another example – what's returned?
    """
    LoL = [ [w[::-1], w] for w in L ]
    bestpair = max( LoL )
    return bestpair[1]
```

```
L = [ 'aliens', 'zap', 'hazy', 'code' ]
LoL = [ [ 'sneila', 'aliens' ], [ 'paz', 'zap' ], [ 'yzah', 'hazy' ], [ 'edoc', 'code' ] ]
bestpair = [ 'yzah', 'hazy' ]
return bestpair[1]  
'hazy'
```
Other examples...

What is `bestnumb`?  
What is `mostnumb`?

```python
>>> bestnumb( [10,20,30,40,50,60,70] )
40

>>> bestnumb( [100,200,300,400] )
100

>>> bestnumb( [1,2,3,4,5,6,7,8,7] )
8

>>> mostnumb( [1,2,3,4,5,6,7,8,7] )
7
```
def maxlen(L):
    LoL = [ [len(s),s] for s in L ]
    bstpr = max( LoL )
    return bstpr[1]

def bestnumb(L):
    """ returns the # in L closest to 42 """
    LoL = [ [6,'aliens'], [3,'zap'],  ___________,   __________  ]
    bstpr = __________
    return bstpr[1]

def mostnumb(L):
    """ returns the item most often in L """
    LoL = [  ]
    bstpr = __________
    return bstpr[1]

L = [ 'aliens', 'zap', 'hazy', 'code' ]

1. What is LoL? here is a start: LoL is [ [6,'aliens'], [3,'zap'],  __________,   __________  ]
2. What is bstpr?
3. What is returned?

L = [ 3,4,5,7,6,7 ]

Use the LoL method to write these two functions

1. Extra! Change exactly three characters in this code so that 3 is returned.

   Here is a start: LoL is [  [6,'aliens'], [3,'zap'],  __________,   __________  ]

   bstpr = __________
   return bstpr[1]

L = [ 30, 40, 50 ]

Use the LoL method to write these two functions

1. Extra! Change exactly three characters in this code so that 3 is returned.

L = [ 3,4,5,7,6,7 ]

Use the LoL method to write these two functions

L = [ 30, 40, 50 ]

Use the LoL method to write these two functions

L = [ 30, 40, 50 ]

Use the LoL method to write these two functions

Extra! Change exactly three characters in this code so that 3 is returned.
Quiz

```
L = ['aliens', 'zap', 'hazy', 'code']

def maxlen(L):
    LoL = [ [len(s),s] for s in L ]

    bstpr = max( LoL )  # 2. What is bstpr? [6, 'aliens']

    return bstpr[1]  # 3. What is returned? 'aliens'

L = [ 'aliens', 'zap', 'hazy', 'code' ]
```

1. What is LoL?

Extra!

```
L = [3,4,5,7,6,7]  # Extra!
```

```
L = [30, 40, 50]  # Extra!
```

```
L = [3,4,5,7,6,7]  # Extra!
```

```
def bestnumb(L):
    """ returns the # in L closest to 42 """

    LoL = [ [abs(x-42),x] for x in L ]  # Hint: Python has abs(x) built-in

    bstpr = min( LoL )

    return bstpr[1]
```

```
L = [ 30, 40, 50 ]
```

```
L = [30, 40, 50]
```

```
L = [3,4,5,7,6,7]
```

```
def mostnumb(L):
    """ returns the item most often in L """

    LoL = [ [count(e,L),e] for e in L ]  # Hint: Use this helper function!

    bstpr = max( LoL )

    return bstpr[1]
```

```
def count(e,L):
    """ return # of e's in L """

    LC = [1 for x in L if x == e]

    return sum(LC)
```

```
L = [ 3, 4, 5, 7, 6, 7 ]
```
def maxlen(L):
    LoL = [ [len(s),s] for s in L ]

    bstpr = max( LoL )

    return bstpr[1]

L = [ 'aliens', 'zap', 'hazy', 'code' ]

1. What is LoL?
   [ [6,'aliens'], [3,'zap'], [4,'hazy'], [4,'code'] ]

2. What is bstpr?
   [6,'aliens']

3. What is returned?
   'aliens'

Extra! Change exactly three characters in this code so that 3 is returned.
def bestnumb( L ):
    """ returns the # closest to 42 in L """

    LoL = [ [abs(x-42),x] for x in L ]

    bstpr = min( LoL )

    return bstpr[1]
```
def count( e, L ):
    """ returns the # of e's in L """
    LC = [ 1 for x in L if x==e ]
    return sum( LC )

def mostnumb( L ):
    """ returns the item most often in L """
    LoL = [ [count(e,L),e] for e in L ]
    bstpr = max( LoL )
    return bstpr[1]
```

Could you use x here instead of e?
def count( e, L ):
    """ returns the # of e's in L """
    LC = [ 1 for x in L if x==e ]
    return sum( LC )

def mostnumb( L ):
    """ return the item most often in L """
    LoL = [ [count(e,L),e] for e in L ]
    bstpr = max( LoL )
    return bstpr[1]
Today's lab: *big data?*

Any guesses as to what *kind* of data this is?

I find your lack of faith in this data disturbing.
Today's lab: *sound* data!

What are the vertical and horizontal axes here?
Lab3 ~ Sound

Continuous variation of air pressure vs. time

Samples taken every 1/22050th of a second (or some sampling rate)

Each sample is measured on a loudness scale from -32,768 to 32,767. (This fits into 2 bytes.)

These two bytes are called a frame. Raw audio data - such as what is written to the surface of a CD - is simply a list of these frames.

Pulse code modulation = PCM data

Some examples...

```python
play('swnotry.wav')  # run demo()
flipflop('swnotry.wav')
play('swfaith.wav')
changeSpeed('swfaith.wav', 44100)
reverse('swfaith.wav')
play('spam.wav')
reverse('spam.wav')
```
Lab 3's *key challenge*...

```python
def flipflop(filename):
    """flipflop swaps the halves of an audio file
    input: filename, the name of the original file
    output: no return value, but
    this creates the sound file 'out.wav'
    and plays it"
    
    print("Playing the original sound...")
    play(filename)

    print("Reading in the sound data...")
    sound_data = [0,0]
    read_wav(filename,sound_data)
    samps = sound_data[0]
    sr = sound_data[1]

    print("Computing new sound...")
    # this gets the midpoint and calls it x
    x = len(samps)//2
    newsamps = samps[x:] + samps[:x]
    newsr = sr
    new_sound_data = [ newsamps, newsr ]

    print("Writing out the new sound data...")
    write_wav( new_sound_data, "out.wav" ) # write data to out.wav

    print("Playing new sound...")
    play( 'out.wav' )
```

*intro stuff – not important*

*important stuff*

"outro" stuff
Computing with *language*

- ideas / meaning
- language / words / phrases
- strings
- numbers / bits

Python strings are here.

"alphabetic processions"
Computing with language

ideas / meaning

language / words / phrases

strings

numbers / bits

open questions in AI ...

Eliza, Siri, Tay ... trouble?

This week...

processing language – how English-y is it?

how strings are represented and stored

Next week...
Caesar Cipher: \textbf{encipher} + \textit{decipher}

\textbf{encipher}(s, n)

\begin{align*}
\text{encipher( 'I <3 Latin' , 0 )} & \quad \text{returns} \quad 'I <3 Latin' \\
\text{encipher( 'I <3 Latin' , 1 )} & \quad \text{returns} \quad 'J <3 Mbujo' \\
\text{encipher( 'I <3 Latin' , 2 )} & \quad \text{returns} \quad 'K <3 Ncvkp' \\
\text{encipher( 'I <3 Latin' , 3 )} & \quad \text{returns} \quad 'L <3 Odwlq' \\
\text{encipher( 'I <3 Latin' , 4 )} & \quad \text{returns} \quad 'M <3 Pexmr' \\
\text{encipher( 'I <3 Latin' , 5 )} & \quad \text{returns} \quad 'N <3 Qfyns' \\
\vdots \quad \vdots \\
\text{encipher( 'I <3 Latin' , 25 )} & \quad \text{returns} \quad 'H <3 Kzshm'
\end{align*}
Caesar Cipher: **encipher** + **decipher**

`encipher(s,n)` should return the string `s` with each *alphabetic* character shifted/wrapped by `n` places in the alphabet.

| `encipher('I <3 Latin', 0)` | returns | 'I <3 Latin' |
| `encipher('I <3 Latin', 1)` | returns | 'J <3 Mbujo' |
| `encipher('I <3 Latin', 2)` | returns | 'K <3 Ncvkp' |
| `encipher('I <3 Latin', 3)` | returns | 'L <3 Odwlq' |
| `encipher('I <3 Latin', 4)` | returns | 'M <3 Pexmr' |
| `encipher('I <3 Latin', 5)` | returns | 'N <3 Qfyns' |
| ... | ... | ... |
| `encipher('I <3 Latin', 25)` | returns | 'H <3 Kzshm' |
'Caesar Cipher: encipher

>>> encipher('Bzdrzq bhogdq? H oqdedq Bzdrzq rzkzc.',25)
'Aycqyp agnfcp? G npcdcp Aycqyp qyjyb.'

>>> encipher('Bzdrzq bhogdq? H oqdedq Bzdrzq rzkzc.',15)
'Qosgof qwdvsf? W dfstsf Qosgof gozor.'

>>> encipher('Bzdrzq bhogdq? H oqdedq Bzdrzq rzkzc.',4)
'Fdhvdu flskhu? L suhihu Fdhvdu vdodg.'

>>> encipher('Bzdrzq bhogdq? H oqdedq Bzdrzq rzkzc.',1)
'Caesar cipher? I prefer Caesar salad.'

>>> encipher('Hu lkbjhapvu pz doha ylthpuz hmaly dl mvynla \lclyfaopun dl ohcl slhyulk.',19)
'An education is what remains after we forget everything we have learned.'
Caesar Cipher: \textbf{decipher}

```python
>>> decipher('Bzdrzq bhogdq? H oqdedq Bzdrzq rzkzc."
'Caesar cipher? I prefer Caesar salad.'

>>> decipher('Hu lkbjhapvu pz doha ylthpuz hmaly dl mvynla "
  lclyfaopun dl ohcl slhyulk.')
'An education is what remains after we forget everything we have learned.'

>>> decipher('Uifz xpsl ju pvu xjui b qfodjm!')

>>> decipher('gv vw dtwvg')
```

How!? Which is more difficult computationally?
### ASCII

American Standard Code for Information Interchange

<table>
<thead>
<tr>
<th>Binary</th>
<th>Dec</th>
<th>Hex</th>
<th>Glyph</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010 1111</td>
<td>47</td>
<td>2F</td>
<td>/</td>
</tr>
<tr>
<td>0011 0000</td>
<td>48</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0011 0001</td>
<td>49</td>
<td>31</td>
<td>1</td>
</tr>
</tbody>
</table>

1 byte
8 bits

The SAME bits represent an integer or a string, depending on type: int or str
The SAME bits represent an integer or a string, depending on type: \texttt{int} or \texttt{str}

1 byte

8 bits

The types determine how to interpret the bits; the names don't matter at all...
The SAME bits represent an integer or a string, depending on type: int or str

In Python, `chr` and `ord` convert to/from Unicode + ASCII

<table>
<thead>
<tr>
<th>Binary</th>
<th>Dec</th>
<th>Hex</th>
<th>Glyph</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010 1111</td>
<td>47</td>
<td>2F</td>
<td>/</td>
</tr>
<tr>
<td>0011 0000</td>
<td>48</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0011 0001</td>
<td>49</td>
<td>31</td>
<td>1</td>
</tr>
</tbody>
</table>
ASCII ⊂ Unicode

This is why 'CS' < 'clear'!
Some fun characters...

`chr(156265)`  `chr(9835)`  `chr(9731)`

on Win10: `chcp 65001`

My favorite is `chr(1661)`
We'll build rot13 as a starting point...

rot13('n') should output 'a'
rot13('W') should output 'J'
rot13(' ') should output ' '
rot13('<') should output '<'

Rot13

a useful and illustrative starting point...
rot13's surprising history...
rot13's key ideas...

What is `ord('U') // 2`?

What is `chr(ord('i') + 13)`?

What is `chr(ord('W') + 13)`?

how do we wrap?

<table>
<thead>
<tr>
<th>chr value</th>
<th>abcdefghijklmnopqrstuvwxyz</th>
</tr>
</thead>
<tbody>
<tr>
<td>ord value</td>
<td>97 99 101 103 105 107 109 111 113 115 117 119 122</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>chr value</th>
<th>ABCDEFGHIJKLMNOPQRSTUVWXYZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>ord value</td>
<td>65 67 69 71 73 75 77 79 81 83 85 87 90</td>
</tr>
</tbody>
</table>
def rot13( c ):
    """ rotates c by 13 chars, "wrapping" as needed
    NON-LETTERS don't change!
    """
    if 'a' <= c <= 'z':
        if ord(c)+13 <= ord('z'):
            return chr( ord(c)+13 )
        else:
            return chr(                    )
    elif 'A' <= c <= 'Z':           # upper-case test!
        return chr(                    )
    else:

(0) What do these tests do?
(1) What code will "wrap" to the alphabet's other side?
(2) How will upper case change? Try noting only the code differences...
(3) What if c is not a letter at all?

Extra: How would you rotate n places, instead of 13?
def rot13(c):
    """ rotates c by 13 chars, "wrapping" as needed
    NON-LETTERS don't change!
    """
    if 'a' <= c <= 'z':
        if ord(c)+13 <= ord('z'):
            return chr( ord(c)+13 )
        else:
            return chr( ord(c)+13-26 )
    elif 'A' <= c <= 'Z':  # upper-case test!
        return c
    else:
        # same, but using 'Z'
        return c

(0) What do these tests do?
(1) What code will "wrap" to the alphabet's other side?
(2) How will upper case change? Try noting only the code differences...
(3) What if c is not a letter at all?

Extra: How would you rotate n places, instead of 13?
Caesar Cipher: \textit{decipher}

\begin{verbatim}
>>> decipher('Bzdrzq bhogdq? H oqdedq Bzdrzq rzkzc.')
'Caesar cipher? I prefer Caesar salad.'

>>> decipher('Hu lkbjhapvu pz doha ylthpuz hmaly dl mvynla \ lclyfaopun dl ohcl slhyulk.')
'An education is what remains after we forget everything we have learned.'

>>> decipher('Uifz xpsl ju pvu xjui b qfodjm!')

>>> decipher('gv vw dtwvg')
\end{verbatim}
Decipher?

Strategies?

*Algorithms*?
Decipher?

All possible decipherings

Strategies?

Algorithms?
Decipher?

Algorithms?

Strategies?

decPR(LAT)
decPR2(LAT)
decPR3(LAT)

All possible decipherings

Score them all

quantifying Englishness?
Decipher?

All possible decipherings

gv vw dtwvg
hw wx euxwh
ix xy fvyxi
jy yz gwzyj
kz za hxazk
la ab iybal
mb bc jzcbm
nc cd kadcn
od de lbedo
pe ef mcfep
qf fg ndgfq
rg gh oehgr
sh hi pfihs
ti ij qgjit
uj jk rhkju
vk kl silkv
wl lm tjmlw
xm mn uknmx
yn no vlony
zo op wmpoz
ap pq xnqpa
bq qr yorqb
cr rs zpsrc
ds st aqtsd
et tu brute
fu uv csvuf

max!

[4, 'la ab iybal'],
[0, 'mb bc jzcbm'],
[2, 'nc cd kadcn'],
[4, 'od de lbedo'],
[3, 'pe ef mcfep'],
[0, 'qf fg ndgfq'],
[2, 'rg gh oehgr'],
[2, 'sh hi pfihs'],
[3, 'ti ij qgjit'],
[2, 'uj jk rhkju'],
[1, 'vk kl silkv'],
[0, 'wl lm tjmlw'],
[2, 'yn no vlony'],
[3, 'zo op wmpoz'],
[1, 'ap pq xnqpa'],
[1, 'bq qr yorqb'],
[0, 'cr rs zpsrc'],
[1, 'ds st aqtsd'],
[4, 'et tu brute'],
[3, 'fu uv csvuf']

Score them all

yields the "most English" phrase

Strategies?

Algorithms?

decPR(LAT)
decPR2(LAT)
decPR3(LAT)
Measuring **Englishness**

- **Very English-y**
  - "Call me Ishmael."
  - "Attack at dawn!"
  - "rainbow, table, candle"
  - "Yow! Legally-imposed CULTURE-reduction is CABBAGE-BRAINED!"
  - "quadruplicity drinks procrastination"
  - "Hold the newsreader's nose squarely, waiter, or friendly milk will countermand my trousers."
  - "the gostak distims the doses"
  - "hension, framble, bardle"
  - "jufict, stofwus, lictpub"
  - "itehbs, rsnevtr, khbsota"
  - "epadxo, nojarpn, gdxokpw"
  - "h o q dedqBzdrzqrzkz"  
  - higher scores

- **Not English-y**
  - quantifying **Englishness?**
  - lower scores

All of these sound good to me!
Decipher?

All possible decipherings

gv vw dtwvg
hw wx euxwh
ix xy fvyxi
jy yz gwzyj
kz za hxazk
la ab iybal
mb bc jzcbm
nc cd kadcn
od de lbedo
pe ef mcfep
qf fg ndgfq
rg gh oehgr
sh hi pfihs
ti ij qgjit
uj jk rhkju
vk kl silkv
wl lm tjmlw
xm mn uknmx
yn no vlony
zo op wmpoz
ap pq xnqpa
bq qr yorqb
cr rs zpsrc
ds st aqtsd
et tu brute
fu uv csvuf

max!
[4, 'la ab iybal'],
[0, 'gv vw dtwvg'],
[2, 'hw wx euxwh'],
[2, 'ix xy fvyxi'],
[0, 'jy yz gwzyj'],
[2, 'kz za hxazk'],
[1, 'nc cd kadcn'],
[4, 'od de lbedo'],
[3, 'pe ef mcfep'],
[0, 'qf fg ndgfq'],
[2, 'rg gh oehgr'],
[2, 'sh hi pfihs'],
[3, 'ti ij qgjit'],
[2, 'uj jk rhkju'],
[1, 'vk kl silkv'],
[0, 'wl lm tjmlw'],
[1, 'xm mn uknmx'],
[2, 'yn no vlony'],
[3, 'zo op wmpoz'],
[1, 'ap pq xnqpa'],
[4, 'bq qr yorqb'],
[4, 'et tu brute'],
[3, 'fu uv csvuf']

Score them all

"Englishness" score based on # of vowels

decPR(LAT)
decPR2(LAT)
decPR3(LAT)

Strategies?

Algorithms?
Decipher?

Strategies?

Algorithms?

decPR(LAT)
decPR2(LAT)
decPR3(LAT)

All possible decipherings

[6.9e-05, 'gv vw dtwvg'],
[3.6e-05, 'hw wx euxwh'],
[1.4e-07, 'ix xy fvyxi'],
[8.8e-11, 'jy yz gwzyj'],
[7.2e-10, 'kz za hxazk'],
[0.01503, 'la ab iybal'],
[3.7e-08, 'mb bc jzcbm'],
[0.00524, 'nc cd kadcn'],
[0.29041, 'od de lbedo'],
[0.00874, 'pe ef mcfep'],
[7.3e-07, 'qf fg ndgfq'],
[0.06410, 'rg gh oehgr'],
[0.11955, 'sh hi pfihs'],
[3.1e-06, 'ti ij qgjit'],
[1.1e-08, 'ui rk rhkju'],
[2.6e-09, 'vk kl silkv'],
[0.007, 'wl lm tjmlw'],
[0.00012, 'yn no vlony'],
[3.1e-06, 'xm mn uknmx'],
[0.00024, 'cr rs zpsrc'],
[0.00011, 'fu uv csvuf']

"Englishness" based on letter-probabilities

max!

[0.45555, 'et tu brute'],
[0.00011, 'fu uv csvuf']
Earbuds are helpful for lab - unless you really like Darth Vader!

We'll see you in Lab 3!