Before returning to Eliza…
Looking for String Patterns
The Problem

User types: "I had a HUGE fight with my brother"

Program knows: ["mother", "father", "brother", "sister"]
"I had a HUGE fight with my brother"

```
split()
[
    "I",
    "had",
    "a",
    "huge",
    "fight",
    "with",
    "my",
    "brother"
]
in
["mother", "father", "brother", "sister"]
"I had a HUGE fight with my brother"
"I had a HUGE fight with my brother"
"I had a HUGE fight with my brother"

Option 1

```javascript
['I', 'had', 'a', 'huge', 'fight', 'with', 'my', 'brother']
```
"I had a HUGE fight with my brother"

family = ["mother", "father", "brother", "sister"]

userInput = input("> "
words = userInput.lower().split()

familyMatter = False
for word in words:
    if word in family:
        familyMatter = True
        break

if familyMatter == True:
doSomething()
"I had a HUGE fight with my brother"

```
[ "mother",
  "father",
  "brother",
  "sister"
]
```
"I had a HUGE fight with my brother"

```javascript
.find
[
  "mother",
  "father",
  "brother",
  "sister"
]
```
"I had a HUGE fight with my brother"

```javascript
[find
  "mother",
  "father",
  "brother",
  "sister"
]
```
"I had a HUGE fight with my brother"

Option 2

```javascript
[ "mother",
  "father",
  "brother",
  "sister"
]
```
"I had a HUGE fight with my brother"

family = ["mother", "father", "brother", "sister"]

userInput = input( "> " ).lower()

familyMatter = False
for word in family:
    if userInput.find( word ) != -1:
        familyMatter = True
        break

if familyMatter == True:
    doSomething()
Strangely Buggy Code
family = ["mother", "father", "brother", "sister"]

userInput = input( "> " ).lower()

familyMatter = False
for word in family:
    if userInput.find( word ) != -1:
        familyMatter = True
    else:
        familyMatter = False

if familyMatter == True:
    doSomething()
Back to Eliza…

http://cs.smith.edu/dftwiki/index.php/Turing_Test_and_Eliza
For-Loops

Items: [ dog, cat, horse, hen, pig ]
We stopped here last time...
Reviewing For-Loops

Applications

While Loops for Robustness

Break & Continue
For-Loops

Items: [ dog, cat, horse, hen, pig ]
For-Loops

Items: [ dog, cat, horse, hen, pig ]

list = [ dog, cat, horse, hen, pig ]
for x in list:
    process( x )
For-Loops

Items: [ dog, cat, horse, hen, pig ]
For-Loops

Items: [ dog, cat, horse, hen, pig ]
For-Loops

```
list = [ dog, cat, horse, hen, pig ]
for i in range( 1, len(list), 3):
    x = list[i]
    process( x )
```
For loops in context

Applications

While Loops for Robustness

Break & Continue
Applications
Count Matching Items

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, cat, pigeon, hen, sheep ]
Count Matching Items

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, cat, pig, hen, sheep ]

Exact Place Matching
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig, hen, sheep ]

count = 0
for x in items1 and y in items2:
    if x == y:
        count += 1
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig, hen, sheep ]

count = 0
for x in items1 and y in items2:
    if x == y:
        count += 1

Why?
#          0    1    2      3    4
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig,   hen, sheep ]

count = 0
for i in range( len( items1 ) ):
    if items1[i]==items2[i]:
        count += 1
What if the lists do not have the same length?

```python
#          0   1   2   3   4
items1 = [ dog, cat, horse, hen, pig ]
items2 = [ dog, cat, pig ]

count = 0
for ???:
    if items1[i] == items2[i]:
        count += 1
```
What is the risk?
What could go wrong?
What kind of error?
What is the risk?
What could go wrong?
What kind of error?
What if the lists do not have the same length?

```python
#           0    1    2      3    4
text1 = [ dog, cat, horse, hen, pig ]
text2 = [ dog, cat, pig ]

count = 0
for i in range( len( text1 ) ):
    if text1[i]==text2[i]:
        count += 1
```
items1 may not be the longest list

```python
#           0    1    2    3    4
items1 = [ dog, cat, horse]
items2 = [ dog, cat, pig, hen, pig ]

count = 0
for i in range( len(items1) ):
    if items1[i]==items2[i]:
        count += 1
```

items1 may not be the longest list
## 2. Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here:

<table>
<thead>
<tr>
<th>Function</th>
<th>Built-in Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(x)</td>
<td>help()</td>
</tr>
<tr>
<td>all()</td>
<td>min()</td>
</tr>
<tr>
<td>any()</td>
<td>setattr()</td>
</tr>
<tr>
<td>ascii()</td>
<td>divmod()</td>
</tr>
<tr>
<td>bin()</td>
<td>enumerate()</td>
</tr>
<tr>
<td>bool()</td>
<td>eval()</td>
</tr>
<tr>
<td>bytearray()</td>
<td>exec()</td>
</tr>
<tr>
<td>bytes()</td>
<td>filter()</td>
</tr>
<tr>
<td>callable()</td>
<td>float()</td>
</tr>
<tr>
<td>chr()</td>
<td>frozenset()</td>
</tr>
<tr>
<td>classmethod()</td>
<td>getattr()</td>
</tr>
<tr>
<td>compile()</td>
<td>globals()</td>
</tr>
<tr>
<td>complex()</td>
<td>hash()</td>
</tr>
<tr>
<td>delattr()</td>
<td>hash()</td>
</tr>
<tr>
<td>help()</td>
<td>int()</td>
</tr>
<tr>
<td>hex()</td>
<td>isinstance()</td>
</tr>
<tr>
<td>id()</td>
<td>issubclass()</td>
</tr>
<tr>
<td>input()</td>
<td>iter()</td>
</tr>
<tr>
<td>oct()</td>
<td>issubclass()</td>
</tr>
<tr>
<td>object()</td>
<td>join()</td>
</tr>
<tr>
<td>ordered()</td>
<td>join()</td>
</tr>
<tr>
<td>open()</td>
<td>join()</td>
</tr>
<tr>
<td>ord()</td>
<td>join()</td>
</tr>
<tr>
<td>pow()</td>
<td>join()</td>
</tr>
<tr>
<td>property()</td>
<td>joins()</td>
</tr>
<tr>
<td>reversed()</td>
<td>keywords()</td>
</tr>
<tr>
<td>round()</td>
<td>locals()</td>
</tr>
<tr>
<td>set()</td>
<td>map()</td>
</tr>
<tr>
<td>slice()</td>
<td>max()</td>
</tr>
<tr>
<td>sorted()</td>
<td>memoryview()</td>
</tr>
<tr>
<td>staticmethod()</td>
<td>min()</td>
</tr>
<tr>
<td>str()</td>
<td>next()</td>
</tr>
<tr>
<td>super()</td>
<td>open()</td>
</tr>
<tr>
<td>sum()</td>
<td>open()</td>
</tr>
<tr>
<td>tuple()</td>
<td>open()</td>
</tr>
<tr>
<td>type()</td>
<td>open()</td>
</tr>
<tr>
<td>vars()</td>
<td>open()</td>
</tr>
<tr>
<td>zip()</td>
<td>open()</td>
</tr>
</tbody>
</table>

[https://docs.python.org/3.4/library/functions.html](https://docs.python.org/3.4/library/functions.html)
items1 = ['dog', 'cat', 'horse']
items2 = ['dog', 'cat', 'pig', 'hen', 'pig']

count = 0
len1 = len(items1)
len2 = len(items2)
for i in range(min(len1, len2)):
    if items1[i] == items2[i]:
        count += 1
Applications, #2
Count Matching
Misplaced Items

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ cat, pig, pigeon, hen, dog ]
Count Matching Misplaced Items

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]
Algorithm

0  1  2  3  4
Items1: [ dog, cat, horse, hen, pig ]

0  1  2  3  4
Items2: [ dog, pig, pigeon, hen, cat ]
Algorithm

Items1: [dog, cat, horse, hen, pig ]

Items2: [dog, pig, pigeon, hen, cat ]
Algorithm

\[ \begin{array}{c}
\text{Items1: [ dog, cat, horse, hen, pig ]} \\
\text{Items2: [ dog, pig, pigeon, hen, cat ]} \\
\text{count: 1}
\end{array} \]
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

\[ \text{Items1: [ dog, cat, horse, hen, pig ]} \]

\[ \text{Items2: [ dog, pig, pigeon, hen, cat ]} \]

count: 1
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

\[ \begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 \\
\hline
0 & 1 & 2 & 3 & 4 \\
\end{array} \]

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

\[
\begin{array}{ccccccc}
& 0 & 1 & 2 & 3 & 4 \\
\text{Items1: [ dog, \textcolor{red}{cat}, \text{horse}, \text{hen}, \text{pig} ]} & & & & & & \\
\end{array}
\]

\[
\begin{array}{ccccccc}
& 0 & 1 & 2 & 3 & 4 \\
\text{Items2: [ dog, pig, pigeon, hen, cat ]} & & & & & & \\
\end{array}
\]

count: 1
Algorithm

\[
\begin{array}{cccccc}
 & 0 & 1 & 2 & 3 & 4 \\
0 & & & & & \\
1 & & & & & \\
\end{array}
\]

Items1: [ dog, cat, horse, hen, pig ]

\[
\begin{array}{cccccc}
 & 0 & 1 & 2 & 3 & 4 \\
0 & & & & & \\
1 & & & & & \\
\end{array}
\]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

0 1 2 3 4

Items1: [ dog, cat, horse, hen, pig ]

0 1 2 3 4

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: 1
Algorithm

Items1: [ dog, cat, horse, hen, pig ]

Items2: [ dog, pig, pigeon, hen, cat ]

count: \( \not\equiv 2 \)
items1 = [ dog, cat, horse, hen, pig]
items2 = [ dog, pig, pigeon, hen, cat]

count = 0

for i in range( len( items1 ) ):
    for j in range( len( items2 ) ):
        if items1[i] == items2[j]:
            count += 1
For loops in context

Applications

**While Loops for Robustness**

Break & Continue
Robust Input
With While Loops

- Example 1: While quantity not valid
- Example 2: While item not in list
# get a positive number from user
x = int( input( "> " ) )

...
What if user enters negative number?

```python
# get a positive number from user
x = int(input("> ") )

...
Solution: keep on asking until input is ok

```python
# get a positive number from user
x = int(input("> " ))
while x < 0:
    x = int(input("Invalid number\n> " ))
```
Solution: keep on asking until input is ok

```python
# get a positive number from user
x = int( input( "> " ) )
while x < 0:
    x = int( input( "Invalid number\n> " ) )

# or

while True:
    x = int( input( "> " ) )
    if x > 0:
        break
    print( "Invalid input" )
```
Write Robust Functions That Prompt for Quantities
Solution 1

```python
# get a positive number from user
def getPositiveInt():
    x = int( input( "> " ) )
    while x < 0:
        x = int( input( "Invalid number\n> " ) )
    return x

x = getPositiveInt()
```
def getPositiveInt():
    while True:
        x = int(input("> "))
        if x > 0:
            return x
        print("Invalid number")

x = getPositiveInt()
# get a positive number from user

def getPositiveInt():
    while True:
        x = int(input(" > "))
        if x > 0:
            return x
        print("Invalid number")

x = getPositiveInt()
Robust Input
With While Loops

• Example 1: While quantity not valid

• Example 2: While item not in list
# get a YES/NO answer from user

def getAnswerYesNo():
    x = input( "Continue (Yes/No)? " )
    while ???:
        print( "Invalid input, must be YES or NO" )
        x = input( "Continue (Yes/No)? " )
    return x

ans = getAnswerYesNo()
# get a YES/NO answer from user

def getAnswerYesNo():
    x = input("Continue (Yes/No)? " ).upper()

    while ( x in [ "YES", "NO" ] ) == False:
        print("Invalid input, must be YES or NO")
        x = input("Continue (Yes/No)? " ).upper()

    return x

ans = getAnswerYesNo()
# get a YES/NO answer from user

def getAnswerYesNo():
    x = input("Continue (Yes/No)? ") .upper()

    while not ( x in [ "YES", "NO" ] ):
        print("Invalid input, must be YES or NO")
        x = input("Continue (Yes/No)? ") .upper()

    return x

ans = getAnswerYesNo()
For loops in context

Applications

While Loops for Robustness

Break & Continue
Break and Continue
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Monopoly = loop
break = ?
continue = ?
Example
Count number of ints in range (50,75) and (75,∞)
Stop when getting -1

# initialize counters
count1 = 0
count2 = 0

# assume user enters numbers between 0 and 100,
# and enters -1 to stop
while True:
    x = int(input(" > "))
    if x == -1:
        break
    if x < 50:
        continue
    if 50 <= x <= 75:
        count1 += 1
    else:
        count2 += 1
print(count1, count2)
# initialize counters
count1 = 0
count2 = 0

# assume user enters numbers between 0 and 100,
while True:
    x = int(input( "> " ) )
    if x == -1:
        break
    if x < 50:
        continue
    if 50 <= x <= 75:
        count1 += 1
    else:
        count2 += 1
print(count1, count2)
Count number of ints in range (50,75) and (75,100)  
Stop when getting -1

```python
# initialize counters
count1 = 0
count2 = 0

# assume user enters numbers between 0 and 100,
while True:
    x = int( input( "> " ) )
    if x == -1:
        break
    if x < 50:
        continue
    if 50 <= x <= 75:
        count1 += 1
    else:
        count2 += 1

print( count1, count2 )
```

The order is important! Why?
We stopped here last time...
Review Lab 8: Programming Patterns

Exercise with While Loops

“Man in the Hole” Stories
your pretty mouth and talk while I look around for some liquor! I know you must have some liquor on the place! Where could it be, I wonder? Oh, I spy, I spy! \nSTELLA: Blanche, you sit down and let me pour the drinks. I don't know what we've got to mix with. Maybe a coke's in the icebox. Look'n see, honey, while I'm--\nBLANCHE: No coke, honey, not with my nerves tonight! Where--where--where is--?\nSTELLA: Stanley? Bowling! He loves it. They're having a--found some soda!--tournament...\nBLANCHE: Just water, baby, to chase it! Now don't get worried, your sister hasn't turned into a drunkard, she's just all shaken up and hot and tired and dirty! You sit down, now, and explain this place to me! What are you doing in a place like this?\nSTELLA: Now, Blanche--\nBLANCHE: Oh, I'm not going to be hypocritical, I'm going to be honestly critical about it! Never, never, never in my worst dreams could I picture--Only Poe! Only Mr. Edgar Allan Poe!--could do it justice! Out there I suppose is the ghoul-haunted woodland of Weir?\nSTELLA: No, honey, those are the L & N tracks.\nBLANCHE: No, now seriously, putting joking aside. Why didn't you tell me, why didn't you write me, honey, why didn't you let me know?\nSTELLA: Tell you what, Blanche?\nBLANCHE: Why, that you had to live in these conditions!\nSTELLA: Aren't you being a little intense about it? It's not that bad at all! New Orleans isn't like other cities.\nBLANCHE: This has got nothing to do with New Orleans. You might as well say--forgive me, blessed baby! The subject is closed!\nSTELLA: Thanks. \n
>>>
Congratulations Computer Science Graduates!
$\frac{1}{2} + \frac{1}{4}$
\[ \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} \]
\[ \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} \]
$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32}$

area < 0.01 = 1%?
How many cuts until less than 1% remaining?
How many cuts until less than 1% remaining?

```python
# howManyCutsUntil1Percent.py
# D. Thiebaut

def main():
    area = 0

    portion = 1.0
    noCuts = 0
    while 1-area >= 0.01:
        portion = portion / 2.0
        noCuts += 1
        area = area + portion
        print( "portion = ", portion,
               "area = ", area,
               "# cuts = ", noCuts )

    input( ">")

main()
```
Man in Hole Stories

(Text processing...)
An Introduction by Kurt Vonnegut
(Visiting Prof. at Smith College, 2001)

https://www.youtube.com/watch?v=oP3c1h8v2ZQ
• Paris Review article: "Man in Hole," 4 Feb 2015

• Motherboard article: "There are only six basic book plots, according to computers," 3 Feb 2015

• Prof. **Matthew Jokers**, U. Nebraska, English Dept., released software tool on github to generate book plots
There are sort of two ways of looking at what “plot” is […]. There’s the way the events unfold in the world of the story, and the way the author reveals events to the reader.

taken from http://www.matthewjockers.net/
http://www.matthewjockers.net/
Tim O’Brien

THE THINGS THEY CARRIED

“A marvel of storytelling . . . a vital, important book — a book that matters not only to the reader interested in Vietnam, but to anyone interested in the craft of writing as well.” — NEW YORK TIMES

http://cs.smith.edu/dftwiki/index.php/The_Things_They_Carried

Digression: Kahn Memory Project
# compute top 100 most frequent word in document
import string

doc = 'TheThingsTheyCarried.txt'
stopwords = 
"a about above across after afterwards again against all almost alone along already also although always am among amongst amount an and another any anyhow anyone anything anyway anywhere are around as at back be became because become becomes becoming been before beforehand behind being below beside besides between beyond bill both bottom but by call can cannot cant co computer con couldnt cry de describe detail do done down due during each eg eight either eleven else elsewhere empty enough etc even ever every everyone everything everywhere except few fifteen fifty fill find fire first five for former formerly forty found four from front full further get give go had has hasnt have he hence her here hereafter hereby herein hereupon hers herse" him himself his how however hundred i ie if in inc indeed interest into is it itself keep last latter latterly least less ltd made many may me meanwhile might mill mine more moreover most mostly move much must my myself name namely neither never nevertheless next nine no nobody none noone nor not nothing now nowhere of off often on once one only onto or other others otherwise our ours ourselves out over own part per perhaps please put rather re same see seem seemed seeming seems serious several she should show side since sincere six sixty so some somehow someone something sometime sometimes somewhere still such system take ten than that the their them themselves then thence there thereafter thereby therefore therein thereupon these they thick thin third this those though three through throughout thru thus to together too top toward towards twelve twenty two un under until up upon us very via was we well were what whatever when whence whenever where whereafter whereas whereby wherein whereupon wherever whichever which while whither who whoever whole whom whose why will with within without would yet you your yours yourself yourselves dont got just did didnt im ***

def main():
    global doc, stopwords
text = open( doc, "r" ).read()

stopwords = set( stopwords.lower().split() )
dico = {}
exclude = set(string.punctuation)
text = ".join(ch for ch in text if ch not in exclude)

for word in text.lower().split():
    if word in stopwords: continue
    try:
        dico[word] += 1
    except:
        dico[word] = 1

list = []
for key in dico.keys():
    list.append( (dico[key], key) )
list.sort()
list.reverse()

words = [k for (n,k) in list]
print( "\n".join( words[0:100] ) )

#displayStops()
main()
Dark waters of the id
Carried war inside a man,
Long years crossed away.
Morning looked for feeling,
Night remembered love.
Stories kept the river moved.

— MJ Wraga, Dept. Psychology
Smith College