lst = ['ant', 'bat', 'cat', 'dog']
for i in range(len(lst))
    print(i, lst[i])
I is for ...

• **Identity**
  • Who are you? Computer Science Student

• **Invariant**
  • Reasoning formally and informally about loops

• **Internet**
  • Network of networks
  • Far more than that!
Lynn Conway

See Wikipedia and lynnconway.com

• Helped invent dynamic scheduling early '60s IBM
• Transgender, fired in ‘68
• IBM apologized in 2020 (52 years later)

• Joined Xerox Parc in 1979
• Revolutionized VLSI design with Carver Mead

• Joined U. Michigan 1985
• NAE '89, IEEE Pioneer ‘09
• Professor and Dean, retired '98

“If you want to change the future, start living as if you are already there.”
Announcements

• Assignment 2 Turtles due Thurs!
• Lab 5 Friday – Prelab coming out Wed or Thur
• Coming, APT-3 out Thursday
• Coming, APT-1 QUIZ (Feb 23-27)
  • Timed APTs, take when you want during these dates
  • Your own work!
• DO NOT discuss Exam 1 until it is handed back
  • Will be handed back on Gradescope
Plan for the Day

• Accumulator Pattern
• Range
• Loop Index
• Loop Tracing
• Files
The Accumulator Pattern

- Pattern you will see with a lot of loops
- Here is the pattern:
  - Initialize a variable
  - loop over a sequence (list or string)
    - Accumulate (add a little more to variable)
  - Do something with variable (result)
Example of Accumulator Pattern

```python
def sumlist(lst):
    total = 0
    for num in lst:
        total += num
    return total
```
Example of Accumulator Pattern

```python
def sumlist(lst):
    total = 0
    for num in lst:
        total += num
    return total

lst = [3, 7, 8, 2, 6]
print(sumlist(lst))
```

Output:
```
Example 2: Accumulator Pattern

```python
def numLetters(word):
    total = 0
    for letter in word:
        total += 1
    return total
```
Example 2: Accumulator Pattern

```python
def numLetters(word):
    total = 0
    for letter in word:
        total += 1
    return total

word = "card"
print(numLetters(word))
```

Output:
REVIEW: Looping over Sequences

• Let’s explore this:
  • Given a sentence:
    • “Duke Computer Science is so much fun!”
  • How do we create this sentence?
    • “Dk Cmprtr Scnc s s mch fn!”
  • Input is sentence. Output has vowels removed
Accumulator Pattern: NoVowels

• “For each character, if it’s not a vowel add it to the output string”

• Accumulator pattern: change a variable in a loop
  • Accumulate a value while iterating through loop

```python
def noVowels(phrase):
    ret = ""
    for ch in phrase:
        if not isVowel(ch):
            ret = ret + ch
    return ret
```
range() Sequence

• Range generates a sequence of values

• `range(y)` – starts at 0 and goes up to but doesn't include y: 0 ... (y-1)
  • y is an integer

• `range(x, y):` x ... (y-1)
  • x and y are integers

• Sequence that provides access to int values

• "up to but not including" sounds familiar? Slicing!
Example

range(5)
list(range(5))
range(5)[0]
range(5)[4]
range(5)[5]
range(5,10)
list(range(5,10))
range(5,10)[3]
for x in range(3):
  print(x)
Range Examples

• Access all the values in a list to print them
  • Use the "for each in sequence" pattern

```python
lst = ["ant", "bat", "cat", "dog"]
for s in lst:
    print(s)
```
Range Examples

• Access all the values in a list to print them
  • Use an index to access $i^{th}$ element

```python
lst = ["ant", "bat", "cat", "dog"]
for i in range(len(lst))
    print(i, lst[i])
```
Repetition with Range

- Sometimes rather than looping over a sequence of values you want to repeat # times
  - Do this 4 times
  - Do that 250 times

- Can do this with the Python range function!
  - If don’t care about the value in the range (e.g. “Do this four times”), can do:
    ```python
    for _ in range(4):
        CODE
    ```
Code-Tracing a Loop

1. Find the changing variables/expressions

2. Create table, columns are variables/expressions
   1. First column is loop variable
   2. Add columns to help track everything else

3. Each row is an iteration of the loop
   1. *Before* execute code block, copy down each variable’s value
   2. Execute code block, update a value in the row as it changes
Code-Tracing a Loop

1. Find the changing variables/expressions
2. Create table, columns are variables/expressions
   1. First column is loop variable
   2. Add columns to help track everything else

```python
def mystery(lst):
    idxMax = 0
    for i in range(len(lst)):
        if lst[idxMax] < lst[i]:
            idxMax = i
    return idxMax
```

What should be the table’s columns?
## Fill in table

1. Before execute code block, copy down each variable's value
2. Execute code block, update a value in the row as it changes

```python
def mystery(lst):
    idxMax = 0
    for i in range(len(lst)):
        if lst[idxMax] < lst[i]:
            idxMax = i
    return idxMax
```

mystery([2, 12, 4, 15, 15])

<table>
<thead>
<tr>
<th>i</th>
<th>idxMax</th>
<th>lst[idxMax]</th>
<th>lst[i]</th>
<th>lst[idxMax] &lt; lst[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Before execute code block, copy down each variable’s value

2. Execute code block, update a value in the row as it changes

```python
def mystery(lst):
    idxMax = 0
    for i in range(len(lst)):
        if lst[idxMax] < lst[i]:
            idxMax = i

    return idxMax
```

mystery([2, 12, 4, 15, 15])

<table>
<thead>
<tr>
<th>i</th>
<th>idxMax</th>
<th>lst[idxMax]</th>
<th>lst[i]</th>
<th>lst[idxMax] &lt; lst[i]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WOTO-2 Loop Tracing

• Remember the steps
• (1) Find the changing variable/expressions,
• (2) Create the table with these as the column
• (3) Each row is an iteration of the loop
Examples of Processing Data

• Lecture 1: count letters in Bible

• Another example: Google Ngram viewer
  • Ngram informs how words evolve
  • Shows number of times phrases occur in books over the years
  • [https://books.google.com/ngrams](https://books.google.com/ngrams)

• Funny video on irregular words
  • [https://www.youtube.com/watch?v=tFW7orQsBuo](https://www.youtube.com/watch?v=tFW7orQsBuo)
Studying Language Evolution

- friend vs enemy
Processing Data

• How do we find the longest word in .. Any text?
• How do we find the word that occurs the most?
• How is this related to how Google Search works?

• Text files can be viewed as sequences
  • Sequences of lines
  • Each line is a string
  • Some clean-up because of ‘\n’
File Pattern: One line at a time

• Simplest and reasonably efficient Python pattern
  • Open, loop, close, return/process
  • LineCounter.py

• File as sequence
  • One line at-a-time

```python
def lineCount(fname):
    """
    return # lines in file fname
    """
    f = open(fname)
    lc = 0
    for line in f:
        lc = lc + 1
    f.close()
    return lc
```
def lineCount(fname):
    
    
    return # lines in file fname
    
    f = open(fname)
    
    lc = 0
    
    for line in f:
        lc = lc + 1
        
        f.close()
        
    return lc
altCount function

```python
def altCount(fname):
    
    return # lines in file fname
    
    f = open(fname)
    lc = len(f.readlines())
    f.close()
    return lc
```
```python
if __name__ == "__main__":
    name = "data/poe.txt"
    pc = lineCount(name)
    print("# lines:", pc)
    pc2 = altCount(name)
    print("# lines:", pc2)
```
File Objects

• A file is an object, like a string
  • Functions applied to object: `len("word")`
  • To get file object use `open("data.txt")`
  • What is returned? Integer value, file object

• Often methods (aka function) applied to object
  • `f.readlines(), f.read(), f.close()`
  • Just like: `st.lower(), st.count("e")`
WOTO-3 Files