

Compsci 101

Accumulator Pattern, Loop Tracing, Files

Susan Rodger
February 8, 2022

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

2/8/22

Compsci 101, Spring 2022 1

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!

2/8/22

Compsci 101, Spring 2022

2

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

- APT-2 due Thursday!
 - Remember you get 24-hour grace period, can't turn in after that!
- Assignment 2 Turtles out – due Tues Feb 15
- Lab 5 Friday – Prelab out now

2/8/22

Compsci 101, Spring 2022 4

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

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

Example of Accumulator Pattern

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

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

Output:

Example 2: Accumulator Pattern

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

Example 2: Accumulator Pattern

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

```
word = "giraffe"  
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 Cmptr 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

```
20 def noVowels(phrase):  
21     ret = ""  
22     for ch in phrase:  
23         if not isVowel1(ch):  
24             ret = ret + ch  
25     return ret
```

range() Sequence

- **range(y)** – starts at 0 and goes up to but doesn't include y: 0 ... (y-1)
- **range(x,y):** x ... (y-1)
- Sounds familiar? Slicing!
- Sequence that provides access to int values

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

```
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

```
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: `for _ in range(4):`
CODE

WOTO-1 – Accumulator, Range <http://bit.ly/101s22-0208-1>

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

```
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

```
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])
```

i	idxMax	lst[idxMax]	lst[i]	lst[idxMax] < lst[i]

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

```
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])
```

#1

i	idxMax	lst[idxMax]	lst[i]	lst[idxMax] < lst[i]
0	0			

WOTO-2 Loop Tracing

<http://bit.ly/101s22-0208-2>

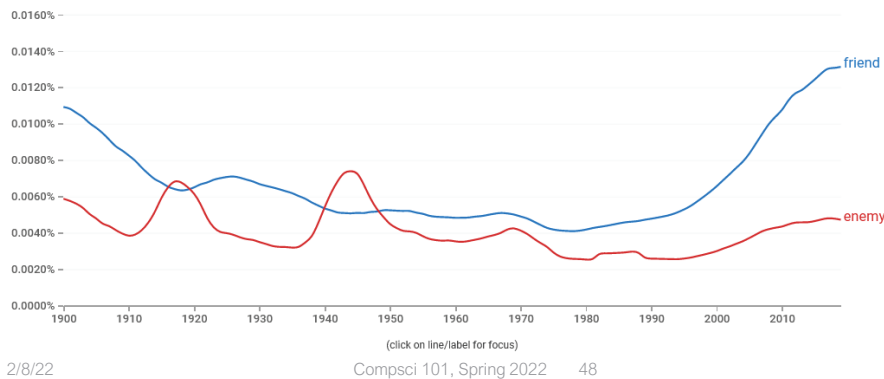
- Remember the steps
- (1) Find the changing variable/expressions,
- (2) Create the table with these as the column
- (3) Each row is an iteration of the loop

Examples of Processing Data

- Lecture 1: count letters in Bible
- Another example: Google Ngram viewer
 - <https://books.google.com/ngrams>

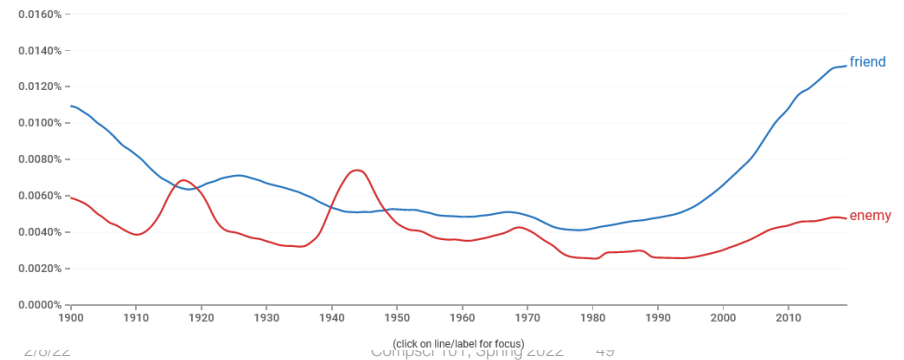
Studying Language Evolution

- Ngram informs how words evolve
- From friend vs enemy
- <https://www.youtube.com/watch?v=tFW7orQsBuo>




Sequences, Repetition

- Parameters? What are they to this query?
 - https://books.google.com/ngrams/graph?content=friend%2Cenemy&year_start=1900&year_end=2019&corpus=26&smoothing=3&direct_url=t1%3B%2Cfriend%3B%



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
- Asymmetry in Open vs Close steps

```

7  def lineCount(fname):
8      """
9      return # lines in file fname
10     """
11     f = open(fname)
12     lc = 0
13     for line in f:
14         lc = lc + 1
15
16     f.close()
17     return lc

```

lineCount function

```
7 def lineCount(fname):
8     """
9     return # lines in file fname
10    """
11    f = open(fname)
12    lc = 0
13    for line in f:
14        lc = lc + 1
15
16    f.close()
17    return lc
```

altCount function

```
19 def altCount(fname):
20     """
21     return # lines in file fname
22    """
23    f = open(fname)
24    lc = len(f.readlines())
25    f.close()
26    return lc
```

main

```
28 if __name__ == "__main__":
29     name = "data/poe.txt"
30     pc = lineCount(name)
31     print("# lines:", pc)
32     pc2 = altCount(name)
33     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")`

Text File Processing Pattern

- See module `FileStuff.py`
 - If newline `'\n'` is read, call `.strip()`
 - If want to break line into “words”, call `.split()`
- Process the list returned by `.split()`
 - May need to convert strings to int or float or ...
- The `for line in f:` pattern is efficient
 - Contrast list returned by `f.readlines()`

FileStuff.py: avgWord

```
def avgWord(fname):  
    f = open(fname, encoding="utf-8")  
    totalWords = 0  
    totalLen = 0  
    for line in f:  
        line = line.strip() #remove newline  
        data = line.split()  
        for word in data:  
            totalWords = totalWords + 1  
            totalLen = totalLen + len(word)  
    f.close()  
    return totalLen/totalWords
```

WOTO-3 Files

<http://bit.ly/101s22-0208-3>