Genome 559

`for` loops revisited
`while` loops
increment operator

Mary Kuhner

Hints on variable names

- Pick names carefully–they matter to human readers
- Change a name if it confuses you
- Give names to intermediate values for clarity

Hints on variable names

- Distinguish between:
  - file name (a string)
  - file handle (a file object)
  - contents of the file (string or list of strings, depending on how you read them)
- Give names that help keep these distinctions clear
- If a variable will contain a single word, don't call it "words". If it will contain multiple words, don't call it "word".
- If you are unsure which words are reserved in Python, use a prefix: myword, the_line, a_student

for loop review

```
for <item> in <container>:
  first statement
  second statement
  ...
  last statement
```

- <item> can be a newly created variable; it is used as a placeholder for each element of the container as we come to it
- <container> must be an existing thing (a list, a string, a dictionary, a tuple) whose contents we want to deal with
- If we don't have a pre-existing container we can make one on the fly:

  ```
  for number in range(42,47):
    print number
  ```

for loop review–examples

```
for line in myresume:
for student in gs559:
for number in range(20,30):
for letter in mywords[0]:
   # mywords is a list of words
   # mywords[0] is a word (a string)
```

for loop review: counting

- Python for loops don't naturally provide a counter

```
for student in gs559:
```

- What student are we currently processing? We don't know.

- If we want to know, we can count them as we go:

```
counter = 0
for student in gs559:
   counter = counter + 1
   print counter, student
```

for loop review: counting

```
# program 1
counter = 0
for student in gs559:
   counter = counter + 1
   print counter, student

...

# program 2
counter = 0
for student in gs559:
   print counter, student
   counter = counter + 1
```

What is the practical difference between these two programs?

for loop review: counting

```
# program 1
counter = 0
for student in gs559:
   counter = counter + 1
   print counter, student
```

```
1 Mary
2 Jon
3 Chuck
```

for loop review: counting

```
# program 2
counter = 0
for student in gs559:
   print counter, student
   counter = counter + 1
```

```
0 Mary
1 Jon
2 Chuck
```

while loop

```
while (conditional test):
  statement 1
  statement 2
```

While some logical test is true, continue executing the block of statements. If the test is not true skip over them and go on.

What does this program do?

```
sum = 0
count = 1
while (count < 10) :
  sum = sum + count
  count = count + 1
print count
print sum
```

What does this program do?

```
sum = 0
count = 1
while (count < 10) :
  sum = sum + count
  count = count + 1
print count                    # should be 10
print sum                      # should be 45
```

for versus while

- `for` is the most common loop in Python
- `for` is used to loop through a list or over a range
- `while` is used to repeat something until a condition is met

for examples

- `for base in sequence:`
- `for sequence in database:`
- `for base in ["a","c","g","t"]:`
- `for index in range(5,200):`

while examples

- `while (error > 0.05):`
- `while (score <= 25):`

short form of increment operator

`x += y`
is the same as
`x = x + y`

This is a common idiom in Python (and other languages). It's never necessary, but people use it frequently. Also works with other math operators.

FASTA database format

```
>identifier1 comment comment commment
AAOSIUBOASIUETOAISOBUAOSIDUGOAIBUOABOIUAS
AOSIUDTOAISUETOIGLKBJLZXCOITLJLBIULEIJLIJ
>identifier2 comment comment
TXDIGSIDJOIJEOITJOSIJOIGJSOIEJTSOE
```

Problem 1 (count-fasta.py): count the sequences in a FASTA file

Two sample FASTA files are on the web page: small.txt and large.txt. Make sure your program works for both!

```
import sys

# Make sure we got a filename on the command line.
if (len(sys.argv) != 2):
    print("USAGE: count-fasta.py <file>")
    sys.exit(1)

# Open the file for reading.
fasta_file = open(sys.argv[1], "r")

num_seqs = 0
for line in fasta_file:
    # Increment if this is a new sequence.
    if (line[0] == ">"):
        num_seqs += 1
print(num_seqs)
fasta_file.close()
```

FASTA database format

```
>identifier1 comment comment commment
AAOSIUBOASIUETOAISOBUAOSIDUGOAIBUOABOIUAS
AOSIUDTOAISUETOIGLKBJLZXCOITLJLBIULEIJLIJ
>identifier2 comment comment
TXDIGSIDJOIJEOITJOSIJOIGJSOIEJTSOE
```

Problem 2 (get-fasta-ids.py): list the sequence IDs from a FASTA file

FASTA database format

Hints:

- `words = line.split()`
- `first_word = words[0]`
- `print(first_word[1:])`

```
import sys

# Make sure we got a filename on the command line.
if (len(sys.argv) != 2):
    print("USAGE: count-fasta.py <file>")
    sys.exit(1)

# Open the file for reading.
fasta_file = open(sys.argv[1], "r")

num_seqs = 0
for line in fasta_file:
    # Print ID if this line starts a new sequence.
    if (line[0] == ">"):
        words = line.split()
        first_word = words[0]
        print(first_word[1:])
fasta_file.close()
```

FASTA database format

```
>identifier1 comment comment commment
AAOSIUBOASIUETOAISOBUAOSIDUGOAIBUOABOIUAS
AOSIUDTOAISUETOIGLKBJLZXCOITLJLBIULEIJLIJ
>identifier2 comment comment
TXDIGSIDJOIJEOITJOSIJOIGJSOIEJTSOE
```

Problem 3 (compute-average-fasta.py): compute the average sequence length in a FASTA file

Hint: use floating point numbers, not integers!

```
# first 8 lines same as previous program:  get filename, open file
# as "fasta_file"

num_seqs = 0
total_chars = 0.0
for line in fasta_file:
    # Increment the sequence count or the character count.
    if (line[0] == ">"):
        num_seqs += 1
    else:
        # Error check: Make sure we don’t have sequence before ID.
        if (num_seqs == 0):
            print("Invalid FASTA format.")
            sys.exit(1)
        # Subtract one for the end-of-line character.
        total_chars += len(line) - 1
print( total_chars / num_seqs)
fasta_file.close()
```

FASTA database format

```
>>> python count-fasta.py small.txt
5
>>>python count-fasta.py large.txt
125
>>> python get-fasta-ids.py small.txt
104K_THEPA
10KD_VIGUN
10KS_HUMAN
10KS_RAT
110K_PLAKN
>>> python compute-average-fasta.py small.txt
300.6
>>> python compute-average-fast.py large.txt
350.192
```