while loops and modules

Prof. 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
Comments

- Use comments to make your code clearer
- Always comment any line you had to think hard about
- Don’t comment the obvious:
  - `foo = foo + 1 # add one to foo`
- Use comments and whitespace to make your program’s structure clear
- Three years from now, will you be able to read your own code?
Show fasta2matrix.py
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?

```python
sum = 0
count = 1
while (count < 10):
    sum = sum + count
    count = count + 1
print count
print sum
```
What does this program do?

```python
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.
Modules

- Most Python programs are one main file and several modules
- Modules are additional files containing things your program can use
- We have already used the `sys` module
- import sys

- programe = sys.argv[0]

- firstarg = sys.argv[1]
import

- import allows your program to use a module
- names in the module can be referred to as modulename.variablename

- module sys has a variable named argv

- when you import it, this becomes sys.argv
FASTA database format

>identifier1  comment comment comment
AAOSIUBOASIUETOAISOBUAOSIDUGOAIBUOABOIUAS
AOSIUDTOAISUETOIGLKBJLZXCOITLJLBIULEIJI
>identifier2  comment comment
TXDIGSIDJOIJE0ITJOSIOIGJSOIEJTSOE

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

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

Solutions on the web page (they are too long to fit on a screen).
FASTA database format

>identifier1 comment comment comment
AAOSIUBOA SIUETO AISOBUA OSIUDUGOAIBUOABOIUAS
AOSIUDTOAI SUETOIGLKBJLZXCOITLJLBIEIJLIJ

>identifier2 comment comment
TXDIGSIDJOIJE0ITJOSIJ0IGJSOIEJTSOE

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:])`
FASTA database format

>identifier1 comment comment comment
AAOSIUBOASIUETOAIOSOBUAOSIDUGOAIBUOABOIUAS
AOSIUDTOAISUETOIGLKBJLZXCOITLJLBUEIJLIJ
>identifier2 comment comment
TXDIGSIDJ0IJE0ITJOSIJOIGJS0IEJTS0E

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

Hint: use floating point numbers, not integers!
FASTA database format

```python
>>> 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-fasta.py large.txt
350.192
```