

I/O & Files

Week 2, day 4

Simple input / output

- `raw_input()` for commandline user prompts
- `print()` for commandline output
- Actually `stdin`, `stdout`

```
>>> raw_input (  
    "Enter a number: ")
```

```
Enter a number:
```

```
print a, b, c  
print a, b, c,
```

Mostly files

- Reading files
- Writing files
- Reading and writing file-like things
- `open, close`
- `readline, readlines`
- `write`

open & close

- `filehandle = open (filename, mode)`
- mode is |+ of 'r' (read), 'w' (write), 'a' (append), 'b' (binary)
- `close` flushes output

```
infile = open ('ovis.fasta', 'rb')  
[...]  
infile.close()
```

readline

- returns a single line
- includes end-of-line
- EOF = ""

```
line = infile.readline()
while (line):
    # do something
    line = infile.readline()
```

readlines

- more usual
- returns a list of lines
- can also `filename.read (bytes)`

```
for line in infile.readlines():  
    # do something
```

Files are “iterable”

- Newer construct
- File acts like list of lines

```
infile = open ("fasta.seq")
for line in infile:
    # do something
infile.close()
```

write

- Writes a single line to file
- No implicit end-of-line

```
infile.write ("#NEXUS\n")  
infile.write ("# version=%s\n" % ver)
```

Traps

- Stray whitespace
- Backslashes in DOS filenames
`"c:\Documents\Simulations\July"`
- Errors, format inconsistencies: "be strict in what you produce, be liberal in what you accept"
- Line endings

Line-endings

- Different for different OS
 - Unix `\n`
 - Old Mac `\r`
 - Dos `\r\n`
- Solution: read as text, write as binary
`open (name, 'rb')`

StringIO

- Treat a string like a file
- Can init StringIO with string

```
import StringIO

msg = "Too lazy"

stio = StringIO.StringIO(
    msg)

stio.write (" to write")

print file.getvalue()
```

Other useful modules

- csv
- xlrd
- xml
- Biopython