Benford's Law; file I/O in Python

Ben Bolker

22:27 02 February 2017

Benford's law

Benford's law describes the (surprising) distribution of first digits of many different sets of numbers. Read it about it on Wikipedia or MathWorld

We'll write a Python function `benford_count` that tabulates the occurrence of digits from a set of numbers

- But where do we get our numbers from?

file I/O

A bunch of ways to read a file

Reference

- `f = open(filename,mode)` (mode = `r` for reading, `w` for writing, `a` for appending) - text by default; returns a **file object**

```
f = open("test.txt")
print(f)
```

- you can use a `for` loop: `for L in f: do_something_with(L)`. This is probably the most common way to process a file.

```
f = open("test.txt")
for L in f:
    print(L)
    x = float(L)
    print(x**2)
```

- `f.read(size)`; reads `size` characters (`f.read()` reads the whole file), unless we are at the end of the file; then returns `''`

```
f = open("test.txt")
print(f.read())
```

- includes **new lines** (`\n`)
- or we can read a line at a time, or read lines into a list:

```
f = open("test.txt")
L = f.readline() ## read one line
print(L)
print(L,end="")  ## print normally *appends* a newline
print(repr(L))   ## repr() prints a *representation* of the line
L2 = f.readlines()
print(L2)
```

- the file object keeps track of how much has been read

`next()`, and more flow control

- what if we want to a little more control? **`next()`** function, works on any **iterable** (tuples, lists, ranges, files . . .)

```
f = open("test.txt")
L = next(f) ## read one line
print(L)
```

- when you get to the end of a file (or a list or whatever) and try to use `next()` you get a `StopIteration` error; use `try`/`except` to handle it safely

```
f = open("test.txt")
finished = False
while not finished:
   try:
      L = next(f)
   except StopIteration:
      finished = True
```

- `try` is a *general* way to handle errors safely
- a standard idiom for doing something until it works uses `break`:

```
while True:
   try:
      x = int(input("enter a number: "))
      break
   except ValueError:
      print("Try again!")
```

More I/O details

- getting rid of pesky newlines: `.strip()` method for strings (gets rid of leading and trailing *whitespace*)

```
f = open("test.txt")
L = f.readline() ## read one line
print(repr(L))
print(repr(L.strip()))
```

- breaking lines into words: `.split()` method for strings

```
f = open("test.txt")
L = f.readline() ## read one line
LL = L.strip().split(" ")
print(LL)
```

And even more

- `import os` in order to find out working directory (`os.getcwd()`), or set the directory `os.chdir(newpath)`; use full path or use (e.g.) `..` to go up one level
- opening a URL from the web `import urllib.request as ur`; `ur.urlopen(url)`
- to read a text file: `io.TextIOWrapper()`
- to read a CSV file: `import csv`, use `csv.reader()`

CodeLab Qs: (files) 51182, 51356 , (loops and strings) 51005