```
## "infile" - the csv file used for homework
##
## Site,Observations,Species,Expenditure
## Lake_Creek,4,12,180
## Los_Alamos,8,340
## Big_Bend,a,6,280
## McDonald,5,20,280
## Balmorrhea,3,3,174



line_list = []

with open(infile) as f:
        for line in f:
                line = line.strip().split(',')
                line_list.append(line)

expenses = {}

## Try tinkering a bit to answer the questions below

for line in line_list[1:]: # Why skip the first item?
        expenses[line[0]] = line[-1] # Why use negative indexing?

for site in expenses:
        print "Spent %s at %s" % (expenses[site],site)

## Be able to slice and dice data

for line in line_list[1:]:
        print line[2] # Verbally, what does this loop do? Why must we write th
e code this way
```

Table of ways to speed up your code

Instead of ...	Do	Why	When to use the slower code
readlines()	Loop over the file	readlines reads in the whole file at once, which takes a lot of memory. When you loop, you can start processing the data *immediately*, which saves time, and allows you to save smaller subsets of the file in memory, if you wish	When you really quick want to visualize the file at the interpreter
declaring and making a list	list comprehension	These run in the C language, which makes them faster	For readability. If your code will be read by novices, it might be best to keep it simple
open	with open	this is not a speed-up per se, but prevents us from having to remember to close our loops	Again, readability.