CSE 503

Introduction to Computer Science for Non-Majors

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 19
Reading and Writing Files

Announcements

- Project has been released…I hope
  - Don't worry too much about it yet, we have much more to learn
  - Keep an eye out for certain concepts
  - Labs 4 and 5 will also be geared towards project concepts

Recap

- Opening and reading from text files in Python…
- Started looking at opening and reading from CSV files in Python…

Opening Files in Python

The **`open(...)`** function is usually used with a **`with...as`** statement:

```
with open("test1.txt","r") as f:
    # do something with the file...
```

`f` is a variable. It refers to a file object.

The **`with...as`** statement ensures that the file is automatically closed at the end of the suite of statements, no matter what happens.

File Objects and Iteration

File objects support iteration so we can use a for loop to iterate over each line in a file:

```
with open("test1.txt","r") as f:
    for line in f:
        # do something with each line...
        print(line)
```

CSV Files

Comma-separated values

In computing, a **comma-separated values** (**CSV**) file is a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

Excerpt from https://en.wikipedia.org/wiki/Comma-separated_values

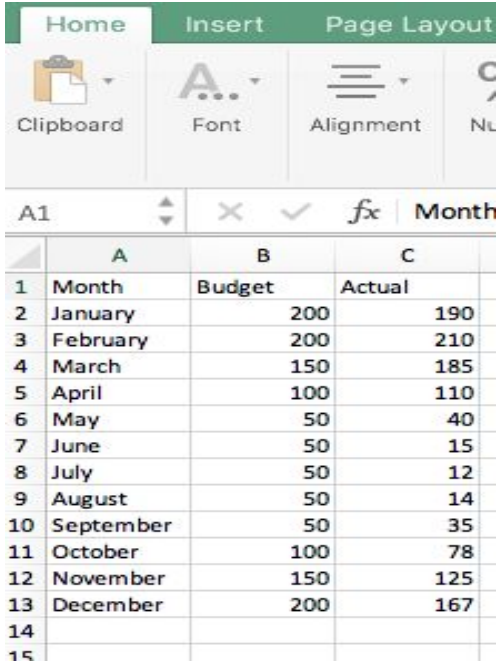
CSV Files

Example

```
Month,Budget,Actual
January,200,190
February,200,210
March,150,185
April,100,110
May,50,40
June,50,15
July,50,12
August,50,14
September,50,35
October,100,78
November,150,125
December,200,167
```

CSV Files

Can be imported/exported with spreadsheet programs like excel/numbers/Google sheets



	A	B	C
1	Month	Budget	Actual
2	January	200	190
3	February	200	210
4	March	150	185
5	April	100	110
6	May	50	40
7	June	50	15
8	July	50	12
9	August	50	14
10	September	50	35
11	October	100	78
12	November	150	125
13	December	200	167
14			
15			

```
Month,Budget,Actual
January,200,190
February,200,210
March,150,185
April,100,110
May,50,40
June,50,15
July,50,12
August,50,14
September,50,35
October,100,78
November,150,125
December,200,167
```

Heating

Month	Budget	Actual
January	200	190
February	200	210
March	150	185
April	100	110
May	50	40
June	50	15
July	50	12
August	50	14
September	50	35
October	100	78
November	150	125
December	200	167

Reading CSV Files

Let's write a program to read the data in our csv file into a dictionary. We'll use the month as a key, and put the rest of the data into a list.

For example:

```
{ 'Month': ['Budget', 'Actual'],
  'January': ['200', '190'],  'February': ['200', '210'],
  'March': ['150', '185'],    'April': ['100', '110'],
  'May': ['50', '40'],        'June': ['50', '15'],
  'July': ['50', '12'],       'August': ['50', '14'],
  'September': ['50', '35'],  'October': ['100', '78'],
  'November': ['150', '125'], 'December': ['200', '167'] }
```

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    # Read data from file...





    return budget
```

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    # Read data from file...





    return budget
```

Import library for reading csv files

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    with open(filename, "r", newline='') as f:
        reader = csv.reader(f)




    return budget
```

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    with open(filename, "r", newline='') as f:
        reader = csv.reader(f)




    return budget
```

Open the file *mostly* the same as before (newline is required for CSV file reading)

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    with open(filename, "r", newline='') as f:
        reader = csv.reader(f)
```

Create a **`csv.reader`** object from our file

```
    return budget
```

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    with open(filename, "r", newline='') as f:
        reader = csv.reader(f)
        for line in reader:
            # Do something with each line


    return budget
```

We can iterate over the lines in the file just like we did with textfiles.

But now, instead of a line of text, **`line`** is a ***list of values***

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    with open(filename, "r", newline='') as f:
        reader = csv.reader(f)
        for line in reader:
            key = line[0]                # The month will be our key
            value = [line[1], line[2]] # [budget,actual] will be our value
            budget[key] = value
    return budget
```

Reading CSV Files

```
import csv

def readBudget(filename):
    budget = {}
    with open(filename, newline='') as f:
        reader = csv.reader(f)
        next(reader) # skip the first line
        for line in reader:
            key = line[0] # The month will be our key
            # [budget,actual] will be our value (convert values to ints)
            value = [int(line[1]), int(line[2])]
            budget[key] = value
    return budget
```

Skip over the header line

Convert values to integers

Exercises

1. Define a function, **`overspent`**, which takes a dictionary like that produced by the **`readBudget`** function, and returns a dictionary of the months in which expenditures were over budget, along with the difference (as a negative value).

2. Define a function, **`underspent`**, which takes a dictionary like that produced by the **`readBudget`** function, and returns a dictionary of the months in which expenditures were under budget, along with the difference (as a positive value).

Exercise - Overspent

```
def overspent(budget):
  dict = {}
  for key in budget.keys():                    # Look at all keys
    diff = budget[key][0] - budget[key][1] # Compute budget - actual
    if diff < 0:                               # If negative, we've overspent
      dict[key] = diff

  return dict
```

Exercise - Underspent

```
def underspent(budget):
  dict = {}
  for key in budget.keys():                    # Look at all keys
    diff = budget[key][0] - budget[key][1] # Compute budget - actual
    if diff > 0:                               # If positive, we've underspent
      dict[key] = diff

  return dict
```

Writing Text Files

To write to a text file, open the file in write mode instead (`"w"`).

The file object has a **`write`** function we can call, passing in the string we want to write:

```
with open("testfile.txt", "w") as f:
    f.write("Text on the first line\n")
    f.write("Text on the second line!\n")
```

Writing Text Files

If we have multiple things to write, we can use a for loop (note the addition of the newline character `'\n'`) :

```
def write(filename, contents):
    with open(filename, "w") as f:
        for item in contents:
            f.write(item + '\n')
```

Writing CSV Files

Writing CSV files uses the **`csv.writer`** object, which write records (sequences) to a single line, with each value separated by a comma.

```
import csv

def writeCSV(filename, dataTable):
   with open(filename, "w", newline='') as f:
        writer = csv.writer(f)
        for record in dataTable:
            writer.writerow(record)
```

Writing CSV Files

What do file1.csv and file2.csv look like with the following calls?

```
dt = [ ['abc', 'def'] , ['ghij', 'klmn'] ]

writeCSV('file1.csv',dt)
writeCSV('file2.csv',dt[0])
```

Writing CSV Files

What do file1.csv and file2.csv look like with the following calls?

```
dt = [ ['abc', 'def'] , ['ghij', 'klmn'] ]

writeCSV('file1.csv',dt)
writeCSV('file2.csv',dt[0])
```

file1.csv

```
abc,def
ghij,klmn
```

file2.csv

```
a,b,c
d,e,f
```

Exercise

Define a function which, given the name of a csv file, reads data from that csv file. Each record in the file has fields f_0 through f_N. Write a new csv file, whose file name is the same as the original but prefixed with 'R', which has the same records but with the fields reversed, from f_N through f_0.