Lab 9: CSV files

Feb 22, 2018

Main Event

1. CSV dragons

Download dragons.csv from the class website. This CSV file has a header that outlines what the columns actually mean.

Many of these exercises can be accomplished by assuming certain values for each dragon attribute and coding to those values. For example, examining the CSV file to see exactly what breeds exist, and hard-coding those breed names into your script. You should not do this. Instead, develop an understanding of the different attributes-value possibilities in your code. In the end, if given a dragon file with a different set of breeds or ages, your code should still work *without alteration*.

2. Dragon stats

Write a Python script that calculates the following:

- The average length of a dragon name. Recall the question from the previous lab that asked you to use a dictionary to count characters in a string. The logic to solve this question is equivalent.
- The minimum, maximum, and mode of the dragon ages. Python has builtin functions for each:

```
>>> x = [1, 2, 2, 3]
>>> min(x)
1
>>> max(x)
3
>>> import statistics
>>> statistics.mode(x)
2
```

Solution:

```
import statistics

names = [] # list of name lengths
ages = []  # list of ages

fp = open('dragons.csv')
header = fp.readline().strip().split(',')
for line in fp:
    row = line.strip().split(',')

    d = {}
    for i in range(len(header)):
        d[header[i]] = row[i]

    names.append(len(d['name']))
    ages.append(int(d['age']))
fp.close()

print('Average name length:', round(statistics.mean(names), 2))

indent = ' ' * 2
print('Age stats:')
for f in [ min, max, statistics.mode ]:
    print(indent, f.__name__, f(ages))
```

3. Breed stats

Write a Python script that calculates the following:

- The number of dragons in each breed.
- The profile (name, age, size, and color) of the largest dragon in each breed. If there are ties, identifying a single dragon within the maximal set sufficient.

Solution:

```
import statistics

number_of_dragons = {} # dragons in each breed
dragons_per_breed = {} # dragons per breed
```

```
fp = open('dragons.csv')
header = fp.readline().strip().split(',')
for line in fp:
    row = line.strip().split(',')

    d = {}
    for i in range(len(header)):
        d[header[i]] = row[i]
    breed = d['breed']

    # breed counts
    if breed not in number_of_dragons:
        number_of_dragons[breed] = 0
    number_of_dragons[breed] += 1

    # largest per breed
    if breed not in dragons_per_breed:
        # if this breed hasn't been seen before, then it's definitely
        # the largest...
        dragons_per_breed[breed] = d
    else:
        # ... otherwise compare it to what as been seen before
        seen = dragons_per_breed[breed]
        previous = int(seen['size'])
        current = int(d['size'])
        if current > previous:
            dragons_per_breed[breed] = d
fp.close()

indent = ' ' * 2

print('Total dragons:')
for breed in number_of_dragons:
    print(indent, breed, number_of_dragons[breed])

print('Largest dragons:')
for dragon in dragons_per_breed.values():
    info = []
    for i in ['name', 'age', 'size', 'color']:
        info.append(dragon[i])
    print(indent, dragon['breed'] + ':', ' '.join(info))
```

4. Group-by breed

Notice that each dragon is part of a particular breed. Rewrite the file—generate a new CSV file—such that it is *grouped-by* breed.

Grouping a file by a particular attribute means that neighboring lines all have the same value for a particular attribute. Consider the following list of people:

Name	Age	Size
Bob	21	medium
Tom	12	small
Meghna	56	medium
Xi	32	large
Alex	23	medium
Maria	45	large

If such a list were grouped-by size, it would be written:

Name	Age	Size
Tom	12	small
Bob	21	medium
Meghna	56	medium
Alex	23	medium
Xi	32	large
Maria	45	large

The order in which the grouped categories appear does not matter. That they appear together does.

For this, and the next, exercise, you will not only need to read in the entire dragon file, but you will need to structure the data in such a way that you can easily manipulate it later. Remember, lists can hold lots of values, and dictionaries can group various attributes—think through how these might come together to solve your problem(s) in this case. It might help if you draw out what's going on, and how you want to organize the data.

Solution:

```
comma = ','
by_breed = {}
```

```
fp = open('dragons.csv')
header = fp.readline().strip().split(comma)

for line in fp:
    row = line.strip().split(comma)

    d = {}
    for i in range(len(header)):
        d[header[i]] = row[i]
    breed = d['breed']

    if breed not in by_breed:
        by_breed[breed] = []
    by_breed[breed].append(row)
fp.close()

fp = open('by-breed.csv', mode='w')
print(comma.join(header), file=fp) # header
for dragons in by_breed.values():
    for d in dragons:
        print(comma.join(row), file=fp)
fp.close()
```

5. Sort-by age

Rewrite dragons.csv such that it is ordered by age. Remember, to sort a list in Python:

```
>>> x = [6, 2, 3, 8, 2]
>>> x.sort()
>>> print(x)
[2, 2, 3, 6, 8]
```

Solution:

```
comma = ','

by_age = {} # dragons per age

fp = open('dragons.csv')
header = fp.readline().strip().split(comma)
```

```
for line in fp:
    row = line.strip().split(comma)

    d = {}
    for i in range(len(header)):
        d[header[i]] = row[i]
    age = int(d['age'])

    if age not in by_age:
        by_age[age] = []
    by_age[age].append(row)
fp.close()

fp = open('by-age.csv', mode='w')
print(comma.join(header), file=fp) # header

ages = list(by_age.keys())
ages.sort()
for i in ages:
    for dragons in by_age[i]:
        print(comma.join(dragons), file=fp)

fp.close()
```

Additional Practice

1. Group stats

Use your understanding of groupings to write Python scripts that do the following:

- Calculates the average age for each breed.
- Displays the profile (name, age, size, and color) of *all* of the largest dragons in each breed. Whereas before ignoring ties was fine, this time they should be considered: the final list for each breed should include *all* dragons that are of the maximal age.
- Displays the number of dragons for each color, for each breed. Example output might be as follows:

```
Norwegian Ridgeback:
  red: 12
```

```
  blue: 10
  black: 3
Chinese Fireball:
  white: 4
  blue: 6
```

Solution:

```
from statistics import mean
from collections import defaultdict

by_breed = defaultdict(list) # dictionaries that handle existence chec
casts = ['age', 'size']      # columns that require integer casts

# First, organize all of the dragons by their breed. This dictionary
# will then be used for all subsequent questions.
with open('dragons.csv') as fp: # context manager
    header = fp.readline().strip().split(',')
    for line in fp:
        row = line.strip().split(',')
        d = dict(zip(header, row)) # create the dictionary without a l

        for i in cases:
            d[i] = int(d[i])
        breed = d['breed']

        by_breed['breed'].append(d)

indent = ' ' * 2

print('Average age:')
for (breed, dragons) in by_breed.items():
    ages = []
    for i in dragons:
        ages.append(i['age'])
    print(indent, breed, round(mean(ages), 2))

print('Largest dragons:')
for (breed, dragons) in by_breed.items():
    largest = []
    for i in dragons:
        if not largest:
            largest.append(i)
```

```
            else:
                previous = largest[0]['age']
                current = i['age']

                if current > previous:
                    largest.clear()
                if current >= previous:
                    largest.append(i)

        print(indent, breed + ':')
        for i in largest:
            info = []
            for i in ['name', 'age', 'size', 'color']:
                info.append(dragon[i])
            print(indent * 2, ' '.join(info))

print('Demographics:')
for (breed, dragons) in by_breed.items():
    print(indent, breed)

    colors = defaultdict(int)
    for d in dragons:
        colors[d['color']] += 1

    for (i, j) in colors.items():
        print(indent * 2, i + ':', j)
```

Introduction to Computer Science

Introduction to Computer Science
jerome.white@nyu.edu

jerome-white

Learning computer science concepts through practice.