Test 2 Review Compsci 101

Owen Astrachan

November 7, 2012

PROBLEM 1 : (*What would Watson Say? (22 points)*)

Consider a list of artists as shown stored in `alist`:

```
alist = ["van Gogh", "Rembrandt", "de Kooning", "Picasso", \
         "Rembrandt", "Miro", "Miro", "Miro", "Picasso"]
```

Part A' (4 points)

Write Python code (an expression) to store in variable `uniq` the number of unique/different values in `alist`. In the example above this is 5.

Part A" (4 points)

Write code to store in variable `sp` a list of the strings in verb!alist! that contain a space. In the example above this is the list `["van Gogh", "de Kooning"]`, but the expression you write will work no matter what strings are stored in `alist`.

Part A (4 points)

Write code to store in `many` a list of those artist/names in `alist` that occur more than once. Your code should work even when the strings in `alist` change. The order of the strings in `many` doesn't matter.

Part B (4 points)

Write code to store in `big` the *number of times* the most frequently occurring name in `alist` occurs (this is `"Miro"` for the values shown above, your code should work with any values stored in `alist`.)

Part C (4 points)

Write code to store in `least` the *artist/name* that occurs the fewest times in `alist`. If more than one name occurs the fewest times your code can store any such name in `least`

Part D (6 points)

Write code to store in `gallery` a list of names, the unique strings in `alist`, sorted from least-frequently occurring to most frequently occuring. Break ties alphabetically. For the values shown above the list returned should be

```
["de Kooning", "van Gogh", "Picasso", "Rembrandt", "Miro"]
```

PROBLEM 2 : (*clip, ship, strip, whip, (24 points)*)

In this problem you'll analyze and write Python code that processes a logfile of websites visited by people using the Duke network. The log-file has the format shown below where each line contains a netid and the IP address of a website visited by the person with that netid, separated by a space. Such a log file represents one day of network traffic at Duke.

```
ola 152.3.215.24
rcd 68.71.216.171
ola 68.71.216.171
fpg 199.239.136.200
fpg 199.239.136.200
fpg 98.142.99.33
rbo 208.179.31.37
ola 152.3.215.24
rbo 208.179.31.37
rcd 152.3.215.24
rbo 68.71.216.171
ola 152.3.215.24
rbo 152.3.215.24
```

The data above shows that the person with netid *ola* visited websites four times, three of the webpages have the IP address 152.3.215.24 and one has the address 68.71.216.171. The person with netid *rbo* also visited a website four times, three of these were different and the site 208.179.31.37 was visited twice by *rbo*.

The function `visits` below returns a list of tuples in the format `(ipaddress,visits)` so that the list returned for the data file above is:

```
[('152.3.215.24', 5), ('68.71.216.171', 3),
 ('199.239.136.200', 2), ('208.179.31.37', 2), ('98.142.99.33', 1)]
```

You can use `visits` as a model for code you write, you'll also be asked to explain some of the code shown in `visits`.

```
def visits(logfile):
   visited = {}                                                          #1
   f = open(logfile)                                                     #2
   for line in f:                                                        #3
       parts = line.strip().split(" ")                                   #4
       ip = parts[1]                                                     #5
       if not ip in visited:                                             #6
           visited[ip] = 0                                               #7
       visited[ip] += 1                                                  #8

   tups = sorted(visited.items(),key=operator.itemgetter(1),reverse=True) #9

   f.close()                                                             #10
   return tups                                                           #11
```

(questions follow)

Part A (3 points)

Explain in one or two sentences and at a high level the purpose of the lines labeled 6-8.

Part B (3 points)

Explain why line 9 results in the list of tuples being returned, in particular why the tuples appear in the order illustrated by the return value shown. Be brief.

Part C (3 points)

If the variable `v` is initialzed via `v = visits("weblogs.txt")` then describe in English what the value `len(v)` represents in terms of the web traffic represented by the log file.

Part D (3 points)

For the same value of `v = visits("weblogs.txt")` in Part C, describe in English the value of this Python expression:

```
sum([x[1] for x in v])
```

Part E (6 points)

Write a Python function `user_data` whose parameter is the name of a logfile such as the one shown above and that returns a dictionary in which the keys are each unique netid stored in the file and for which the associated value is a set of unique website/ip-addresses visited by the person with that netid. For example, for the key `ola` the associated value is the set `set(['152.3.215.24','68.71,216.171'])` since although `ola` visited a website four times, three of the visits were to the same website.

For the data file shown above, the returned dictionary is printed below.

```
{'rbo': set(['152.3.215.24', '68.71.216.171', '208.179.31.37']),
 'ola': set(['152.3.215.24', '68.71.216.171']),
 'fpg': set(['199.239.136.200', '98.142.99.33']),
 'rcd2': set(['152.3.215.24', '68.71.216.171'])}
```

Complete the function here:

```
def user_data(filename)
```

Part F (6 points)

Write function `freq_visit` whose parameter is the name of a logfile. The function returns a dictionary in which the keys are ip-addresses of each unique site in the logfile. The corresponding value for each key is a list of tuples of the form `(netid,visits)` where `visits` is the (int) number of times the user with `netid` (string) has visited the site whose ip-address is the corresponding key. The list of tuples should be sorted such that the first tuple has the highest number of visits to the website, and the last tuple corresponds to the netid of the fewest visits. Breaking ties doesn't matter. For example, for the logfile above the dictionary returned could be:

```
{'152.3.215.24': [('ola', 3), ('rbo', 1), ('rcd2', 1)],
 '68.71.216.171': [('rbo', 1), ('ola', 1), ('rcd2', 1)],
 '199.239.136.200': [('fpg', 2)],
 '208.179.31.37': [('rbo', 2)],
 '98.142.99.33': [('fpg', 1)]}
```

Hint: one easy way to create the list of tuples associated with each ip-address/key is to first create a dictionary associated with each ip-address key. This dictionary has `(netid,visits)` as the `(key,value)` tuples, i.e., each time a line is processed the netid is updated in the dictionary that's the value corresponding to the ip-address/key. After reading the file, the dictionary of `(ip-address,dictionary)` tuples can be changed to `(ip-address,sorted-list)` tuples by calling `sorted` on the items of the dictionary associated with each ip-address key.

```
def freq_visit(filename):
```