SI 330 Fall 2017 Midterm

This midterm test consists of eight (8) questions. You have 75 minutes to complete this test.
Points are allocated according to the following table:

0. Name and uniqname on every page	/1
1. Programming: sorting veggies	/5
2. Programming: sorting cities	/5
3. Programming: names	/5
4. Documentation	/8
5. Regular expressions	/20
6. Fetching web content	/16
7. Profiling and efficiency	/10
8. Coding	/20
Total	/90

Your Name: _____ Uniqname: _____

1. Python Programming [5 points]

Which is the correct output for this code?
(circle one choice at right ->)

```
v = {'carrot': 0.79,
     'green pepper': 1.49,
     'celery': 0.99,
     'mushroom': 1.39}

s = sorted(v)

for k in s:
    print(v[k])
```

(a)
carrot
celery
green pepper
mushroom

(b)
0.79
0.99
1.39
1.49

(c)
0.79
0.99
1.49
1.39

(d)
carrot
green pepper
celery
mushroom

Your Name: _____ Uniqname: _____

2. Python Programming [5 points]

Which is the correct output for this code?
(circle one choice at right ->)

```
c = {"Aberdeen, Scotland":(57,9,'N',2,9 ,'W'),
   "Copenhagen, Denmark":(55,40,'N',12,34 ,'E'),
   "Marseilles, France":(43,20,'N',5,20 ,'E'),
   "Liverpool, England":(53,25,'N',3,0 ,'E')}

s = sorted(c, key=lambda x: (c[x][2],
                             -c[x][0],
                             -c[x][1],
                             c[x][5],
                             -c[x][3],
                             -c[x][4]))

for k in s:
   print(k)
```

(a) Aberdeen, Scotland
Copenhagen, Denmark
Marseilles, France
Liverpool, England

(b) Copenhagen, Denmark
Liverpool, England
Marseilles, France
Aberdeen, Scotland

(c) Marseilles, France
Liverpool, England
Copenhagen, Denmark
Aberdeen, Scotland

(d) Aberdeen, Scotland
Copenhagen, Denmark
Liverpool, England
Marseilles, France

Your Name: _____ Uniqname: _____

3. Python Programming [5 points]

Which is the correct output for this code?
(circle one choice at right ->)

Recall that s.join(b) for a string s and a list of strings b will create a new string consisting of all strings in b joined together using string s as a separator. Thus '+'.join(['1','2','3'] produces the string '1+2+3'

```
import re
n = ['Bob Denier',
     'Alan Hu, Jr.',
     'Natalie Silver',
     'Donna Marie Gallant']

e = []
for k in n:
    f = re.findall('(?<!, )[A-Z]',k)
    e.append('.'.join(f)+'.')

print(e)
```

(a) `['B.D.', 'A.H.', 'N.S.', 'D.M.G.']`

(b) `['B.D.', 'A.H.J.', 'N.S.', 'D.M.G.']`

(c) `['Alan Hu, Jr..']`

(d) None of the above

Your Name: _____ Uniqname: _____

4. Python Documentation [8 points]

Fill in the missing documentation
(0.5 points per blank)

```
# This function takes three parameters:
#
# The  first parameter (a) is of type file and represents the csv file
#
# The second parameter (b) is of type list and represents student ids to skip
#
# The  third parameter (c) is of type string or tuple and represents feild to extract from the row
#
# The function returns a dictionary with keys of type _____ and values of type _____

def mystery_function(a, b, c):

    r = {}                              # Create an empty dictionary
    with open(a,'r') as f:
        rdr = csv.DictReader(f)
        for z in rdr:                   # Variable 'z' is of type dictionary
            stu = z['StudentID']
            if stu not in b:
                print('Skipping', stu)
                continue                # If we do not find 'stu', go to _____
            print('Processing',stu)
            if stu not in r:            # check if the _____ is in the dictionary
                r[stu] = []
            ss = z[c]
            r[stu].append(ss)           # append the _____ to the the list of students to be return

            r[stu] = sorted(r[stu],reverse = True) # sort the values of r[stu] in _____ order
    return r
```

Your Name: _____ Uniqname: _____

5. Regular Expressions [5x4 = 20 points]

Consider the following text passage:

> Two roads diverged in a yellow wood,
> And sorry I could not travel both
> And be one traveler, long I stood
> And looked down one as far as I could
> To where it bent in the undergrowth;

List all substrings, in the order they occur, separated by commas) that will be matched by the given regular expression.

`[A-Z][a-z]*`	Two, And, I , And, I, And, I, To
`[a-z]+,`	non greedy, 1 or more char before a comma
`[aeiou]{2}`	
`[\w]{8,}`	words that are 8 char or longer
`\w*[Ii]\w+`	char zero or more times with an upper or lowercase i followed by one or more char

Your Name: _____ Uniqname: _____

6. Fetching Web Content [16 points]

Fill in the missing code to complete the function that scrapes the width and height of each image (‘img’ tag) inside anchors (‘a’ tags).

```
import requests
from bs4 import BeautifulSoup
def find_image_measurements(url):
    response = requests.get(url)
    h = response.text
    e = []
    soup = BeautifulSoup(h)
    for aTag in soup.find_all(a):
        for iTag in aTag.find_all(img):
            width = iTag.adder('width')
            height = iTag.adder('height')
            e.append(width,height)
    return e
```

Example HTML read from URL:

```
<html>
<head><title>Soup</title></head>
<body>
<a href=”bean.html”>
  <img src=”bean.png” width=100 height=100 />Bean</a>
<a href=”pho.html”>
  <img src=”pho.png” width=40 height=40 />Pho</a>
<a href=”tomato.html”>
  <img src=”tomato.png” width=15 height=15 />Pho</a>
</body>
</html>
```

Desired output: a list of tuples, each of which is a pair representing the width and height of each img tag.

```
[(100,100),(40,40),(15,15)]
```

Your Name: _____ Uniqname: _____

7. Profiling and Efficie

Two versions of a pro following profile outputs were obtained.

(a) ***Circle*** the s the **optimized** one.

```
         413322 function calls

Ordered by: standard name

ncalls  tottime  percall  filename:line
 17964    0.047    0.00  00 docdist.py:109(c
 26946    0.062    0.0  000 docdist.py:137(inner
  8982    0.060    0.0  000 docdist.py:151(vector_an
 17964    0.052    0.0  000 docdist.py:93(get_words_fro
     9    0.006    0.0  046 optimize_this.py:102(lookup_sim
     9    0.066    0.0  065 optimize_this.py:143(find_alternat
     1    0.000    0.0  99 optimize_this.py:15(main)
     9    0.000    0.00  0 optimize_this.py:213(find_unique_targets)
    39    0.098    0.003  optimize_this.py:250(get_csv_rows)
    10    0.001    0.000  timize_this.py:262(write_output_file)
    11    0.020    0.002  mize_this.py:39(set_sentence_id)
    11    0.000    0.000  this.py:71(replace_target_with_blank
     1    0.000    0.000  ain())
     0    0.000           0.0  )
```

```
         3831279 function calls

Ordered by: standard name

ncalls  tottime  percall  cumtime  perc  o(function)
 26946    0.826    0.000    1.431    0.0  inner_product)
  8982    0.067    0.000    1.519    0.00  vector_angle)
          2.679    0.000    5.068    0.000  t_words_from_string)
          0.297    0.000    0.474    0.000  unt_frequency)
                   0.001    0.437    0.049  :102(lookup_similar_id)
                   0.010    7.362    0.818  :143(find_alternate_sentence)
                   000    8.102    8.10  :15(main)
                            0.000    0.0  py:213(find_unique_targets)
                            0.907    0  .py:250(get_csv_rows)
                                     s.py:262(write_output_file)
    11                               is.py:39(set_sentence_id)
    11                               this.py:71(replace_target_with_blank)
     1    0.000                      0(main())
     0    0.000                      e:0(profiler)
```

(b) List the functions t the code:

_____

_____

Your Name: _____ Uniqname: _____

8. Coding [20 points]

Consider the following string (assume it is one line):

```
s ='It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it
was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, i
was the spring of hope, it was the winter of despair'
```

One of the things we are interested in from a natural language processing perspective is the analysis of bigrams. Bigrams are pairs of adjacent words, irrespective of punctuation. Thus, in the string above, the first 3 bigrams are: (“It”, “was”), (“was”, “the”), (“the”, “best”). Typically, we want to count unique bigrams, and we also want to convert words to lowercase before creating the bigrams (in which the first bigram in the above example would be (“it”, “was”))..

Here’s the scenario: You have been asked to write some Python code to do this, which you did. Unfortunately your co-worker accidentally deleted some of your code while you were getting lunch, and they had to leave before you got back. They left some comments in the code, but you’ll have to re-create the missing code.

Fill in the blanks on the following page with your replacement for the code that was deleted. Two blank pages for your rough work follow.

Your Name: _____ Uniqname: _____

```
# This function takes a string and returns a list of 2-tuples, each of
# which is a pair of words (i.e. a bigram). For example, given a string of
# 'The quick brown fox.', this function will return
# [('the','quick'),('quick','brown'),('brown','fox')]
# NOTE: all words are converted to lower case.
import re
def get_bigrams(s):
    bigrams = []
    words = re.split('\W+',s)
# Sorry -- I think the code here looped through the lines and maybe added them to
# a list…
```

Fill this in

```
    return bigrams


# count_bigrams(list of bigrams)
# This function takes a list of tuples, each of which is a bigram.
# For example: [('The','quick'),('quick','brown'),('brown','fox')]
# It returns a dictionary of bigrams, with the keys being 2-tuples of
# the bigrams and the values being the count.
#
def count_bigrams(bigrams):
    ret = {}
# Sorry (again)... I think this added tuples to a dict….
```

Fill this in

```
    return ret

# sort_bigrams takes a dictionary and returns the keys of the dictionary
# sorted DESCENDING by the VALUES
#
def sort_bigrams(bigramDict):
# Again, sorry... I deleted the sort line….
    sorted_bigrams =
```

Fill this in

```
    return sorted_bigrams

def main():
# The following string is all one line (there are no linebreaks or
# carriage returns)
    s = 'It was the best of times, it was the worst of times, it was the age of
Wisdom, it was the age of foolishness, it was the epoch of belief, it was the
epoch of incredulity, it was the season of Light, it was the season of Darkness,
it was the spring of hope, it was the winter of despair'
    # The first four lines of output should be
    # it-was: 10
    # was-the: 10
    # of-times: 2
    # times-it: 2
    bigrams = get_bigrams(s)
    bigrams_with_counts = count_bigrams(bigrams)
    bigrams_with_counts_sorted = sort_bigrams(bigrams_with_counts)
    for bigram in bigrams_with_counts_sorted :
        print("{0}-{1}: {2}"
              .format(bigram[ 0],bigram[ 1],bigrams_with_counts[bigram]))

if __name__ == "__main__":
    main()
```

Your Name: _____ Uniqname: _____

Thoughts for other questions:

1. I'll give you an AWS Lambda with some parts removed (like the previous example) and the desired output and you fill in the missing part(s)
2. I'll give you an AWS Lambda function and associated API gateway. I'll give you a URL and ask you what the return value is
3. I'll ask you to fill in part of an NLP pipeline to calculate the type-token ratio of some text.
4. I'll ask you how to create a pandas DataFrame
5. I'll ask you how to transform a dataset using ufuncs
6. I'll ask you to interpret a pivot table, perhaps using a fill-in-the-blank question

Your Name: _____ Uniqname: _____

NOTES and REMINDERS:

1. The test will start at 1:15 this Thursday in this room. You will have 80 minutes to complete the test.

2. If you complete the test early, please hand your assignment to a member of the teaching team on your way out. Please respect your colleagues who are still completing their work.

3. If you have submitted an SSD VISA form, please contact me for details about your accommodations

4. This is a closed-book, closed-technology test. You are allowed one, 2-sided 8.5”x11.0” “cheat-sheet” with ANYTHING YOU WANT on it

Your Name: _____ Uniqname: _____