CS 6364 Homework 1

August 16, 2021

Deadline for the first submission: **Sep-6-2021**.
All assignments **MUST** have your name, student ID, course name/number at the beginning of your documents.
Your homework **MUST** be submitted via Blackboard with file format and name convention as follows:
HW#_Name_writeup.**pdf** (for writing part)
HW#_Name_code.**zip** (for coding part)
If you have any questions, please contact me.

You have 11 tasks to finish, please write all codes in a single code file called "**hw1.py**" rather than 11 files.

- **Task 1: Test Python Environment**

  Try to install python3 environment, copy and paste following codes into file "hw1.py"(note that python is indentation sensitive), and run it in the terminal with "python hw1.py" or in your preferred IDE, such as Pycharm or Anaconda.

  ```
  def task1():
      print "hello world"

  if __name__ == '__main__':
      task1()
  ```

  Note, when you copy and paste the code, please be careful with the proper quotation marks.

- **Task 2: Define Object**

  Define a function called 'task2' to assign items= [1, 2, 3, 4, 5] as a list object, and print the list.

- **Task 3: File Reading**

  Create a file called 'task3.data.' and type string '1 2 3 4 5 6 7 8 9 10' in it without quotation marks, and then write a function to read the file and load the string as two list-objects items1 = [1, 2, 3, 4, 5], items2 = [6, 7, 8, 9, 10]. Print the lists finally.

- **Task 4: Data Structure**

  It is important to be familiar with the functions of dictionary: items(), keys(), values(), write a program to use all these functions.

  Notes: Dictionary has two ways to initialize
  data = dict()
  data['school'] = 'UAlbany'
  data['address'] = '1400 Washington Ave, Albany, NY 12222'
  data['phone'] = '(518) 442-3300'
  data = {'school': 'UAlbany', 'address': '1400 Washington Ave, Albany, NY 12222', 'phone': '(518) 442-3300'}
  Print the results as follows by accessing the defined dictionary.
  school: UAlbany
  address: 1400 Washington Ave, Albany, NY 12222
  phone: (518) 442-3300

- **Task 5: Data Serialization**

  Use json to store above dictionary in task-5 and then print item, key and values.
  Notes: This task tells how to store a dictionary object to a file and then load the dictionary object from the file. In python, object of any type can be mostly saved in json format to a txt file.

  You need to be familiar with the following two json functions : json.dumps(object), which dumps an object to a json format (string), json.loads(a json format string), which loads a json format string back to the original object. We do not care about if the original object is list, dictionary or others. json.dumps() can automatically recognize.

  Note, json.load(object) only can only load an object, not a file.

- **Task 6: Data Serialization**

  Store a number of different types objects (e.g., list, dictionary, array) to a file and then load the objects from the file.

  Write a function to dump list object items = [1,2,3,4,5] and above dictionary in task-5 to a file called 'task6.data', and then load them from the same file and print.

- **Task 7: Data Preprocessing**

  Read the tweets from the file "CrimeReport.txt" and print the id for each tweet.

  Here are some functions that you will use in the task: open().readlines(), tweet = json.loads(), print tweet.keys(), you will know the keys of tweet dictionary object, then you can find which key relates to tweet id, and you can then retrieve the id of this specific tweet.

- **Task 8: Data Preprocessing: tweets filtering**

  INPUT: "CrimeReport.txt"
  OUTPUT: a file "task8.data" that stores the 10 most recent tweets
  ****
  Suggestions:
  tweet[$'created - at'$] gives the created time of this tweet. Rank tweets based on the time from the earliest to the most recent. Then we can identify the 10 most recent tweets. Some example lines that are **not** directly runnable

```
import datetime
tweets = []
for line in open().readlines():
    tweet = json.loads(line)
    tweets.append(tweet)
    #datetime.datetime.strptime(item['created-at'], '%a %b %d %H:%M:%S +0000 %Y')
    #converts the string format of a date time to the datetime object
sorted_tweets = sorted(tweets, key = lambda item:
    datetime.datetime.strptime(item['created-at'], '%a %b %d %H:%M:%S +0000 %Y'))
# sorted tweets based on time.
f = open('output.txt', 'w')
for tweet in sorted_tweets[-5:]:
    f.write(json.dumps(tweet) + '\')
f.close()
```

  Note, when you copy and paste the code above, please be careful with the proper indentation and quotation mark.

- **Task 9: File operations**

  INPUT :CrimeReport.txt: in this file, each line is a raw tweet json format.
  output-folder: where new results will be stored
  REQUIREMENT: read tweets and separate these tweets in to groups based on the specific hours (Mon-Day-Year-Hour). The tweets related to a specific hour will be stored in a separate file in the folder "**task9-output**"

with the file name "**Mon-Day-Year-Hour.txt**"
OUTPUT: new files generated and stored in the folder "**task9-output**", in which each file stores the tweets corresponding to a specific hour.

- **Task 10: NLP and Sentiment Analysis**

  Write a function to calculate the sentiment score of each tweet text in the raw tweet file "CrimeReport.txt" and print out the sentiment score for each tweet.
  The next step, is to separete tweets into tweets with positive sentiments and tweets with negative sentiments and store them into two different files: "positive-entiment-tweets.txt" and "negative-sentiment-tweets.txt"

  "https://github.com/clips/pattern"

  search "sentiment", you will find the following information:

  Written text can be broadly categorized into two types: facts and opinions. Opinions carry people's sentiments, appraisals and feelings toward the world. The pattern.en module bundles a lexicon of adjectives (e.g., good, bad, amazing, irritating, ...) that occur frequently in product reviews, annotated with scores for sentiment polarity (positive $\longleftrightarrow$ negative) and subjectivity (objective $\longleftrightarrow$ subjective).
  The sentiment() function returns a (polarity, subjectivity)-tuple for the given sentence, based on the adjectives it contains, where polarity is a value between -1.0 and +1.0 and subjectivity between 0.0 and 1.0. The sentence can be a string, Text, Sentence, Chunk, Word or a Synset (see below).

  The positive() function returns True if the given sentence's polarity is above the threshold. The threshold can be lowered or raised, but overall +0.1 gives the best results for product reviews. Accuracy is about 75% for movie reviews. sentiment(sentence) # Returns a (polarity, subjectivity)-tuple. Note that,function **sentiment(sentence)** will return a tuple, not a value;function **positive(sentence, threshold)** will return a value.

  positive(s, threshold=0.1) # Returns True if polarity $\geq$ threshold.
  ≫ from pattern.en import sentiment
  ≫
  ≫ print sentiment("The movie attempts to be surreal by incorporating various time paradoxes,"
  ≫ "but it's presented in such a ridiculous way it's seriously boring.")
  (-0.34, 1.0)
  In the example above, -0.34 is the average of surreal, various, ridiculous and seriously boring. To retrieve the scores for individual words, use the special assessments property, which yields a list of (words,polarity, subjectivity, label)-tuples.
  ≫ print sentiment('Wonderfully awful! :-)').assessments
  ≫[(['wonderfully', 'awful', '!'], -1.0, 1.0, None), ([':-]'), 0.5, 1.0, 'mood')]