Homework 1
CS-4395

Create a Python program that performs the tasks listed below. Upload your completed Python program file using the file name `hw01_⟨your_net_id⟩.py`

For example:

```
hw01_cid021000.py
```

Your Python file should have a header using the following format.

```
################################################################################
#
#         FILE:
#           filename.py
#       AUTHOR:
#           Your Name
#           Your NetID
#  DESCRIPTION:
#           Homework 1
#           Description of your program, what it does
# DEPENDENCIES:
#           Created with ActiveState Python 3.10.11 (Python version)
#           Any dependencies, i.e. extra libraries required to run (like
#           datetime, NLTK, re, etc.)
#
################################################################################
```

Extra help references:

- Python reference documenntation – https://docs.python.org/3/index.html
- Python strings – https://docs.python.org/3/library/string.html#module-string
- Python data structures – https://docs.python.org/3/tutorial/datastructures.html

Submit a single file Python program that does the following:

1. Manually edit a natural language text file (Treasure_Island.txt) to remove the beginning extraneous information—copyright, table of contents, etc. Begin on line 134, the one whose content is "TREASURE ISLAND". Save the edited version as a separate file, "ti.txt".

2. Process your new natural language text file (ti.txt) and print (i.e. display) information to the screen when instructed.

3. Word tokenize the file so that all word tokens are atomic. That is, no words should be in the same string with other words or punctuation other than apostrophes in contraction such as "don'". Punctuation should be considered as word tokens themselves. The result should be a Pytyhon *list* (i.e. `<class 'list'>`) of word tokens.

   3.1. First separate punctuation which are adjacent to text by inserting spaces

   3.2. Split the text into a list of word strings which exclude spaces, newlines, and carriage returns. Note that in doing so, you will encounter multiple spaces and multiple newlines.

4. Use your word list of the entire text to count the number of instances of each word. You may do this using a Python dictionary data structure, (i.e. `<class 'dict'>`). Be careful to count instances of words ignoring case. That is, “the”, “The”, and “THE” should all be counted as instances of the same word.

5. Sort the dictionary by frequency, having the most frequent word token first.

6. For the top ten entries, display the following information to the screen. Exclude entries for punctuation.

    6.1. The word token, normalized to all lowercase.

    6.2. The word count

    6.3. The word frequency in the text as a floating point number in the range (0.0, 1.0). That is, the word count divided by the total number of words.

7. Then ***for each word***:

    7.1. Calculate the word count and frequency percentage of each word / unigram.

    7.2. Sort the words by frequency.

    7.3. For the top 20 most frequent words / unigrams, on each line display/print the word form (lexeme), its count (integer number of instances in the text), and frequency (float to exactly five decimal places).