Week 6 Assignment - W200 Introduction to Data Science Programming, UC Berkeley MIDS

Write code in this Jupyter Notebook to solve each of the following problems. Each problem should have its solution in a separate cell. Please upload this **notebook**, your **scrabble.py** file, the **sowpods.txt** file, and your **score_word** module to your GitHub repository in your SUBMISSIONS/week_06 folder by 11:59PM PST the night before class.

Objectives:

- Read and understand PEP 8 standards
- Use all of your previously gained knowledge together on a single program
- Demonstrate how to import a user made module and function into python from another .py file
- Demonstrate how to input command line arguments into a .py file

6-1. PEP 8 Style Guide (reading and response) (10 points)

Your first task for this week is to write a **250 word reading response** to the article referenced below. In addition, please list **3 questions** that you have from the article (the questions do not count towards the 250 word requirement). Please write your response in the markdown cell below.

The writing response is a free response, so you may write about your reactions, such as an interesting thing that you saw in the article, something that really stuck out to you, etc.

Article: The PEP 8 Style Guide (https://www.python.org/dev/peps/pep-0008). This document is really important for Python coders because it describes best practices and customs.

YOUR ANSWER HERE

6-2. Cheating at Scrabble (90 points + 10 Extra Credit Points)

Write a Python program that takes a Scrabble rack as a command-line argument and prints all "valid Scrabble English" words that can be constructed from that rack, along with their Scrabble scores, sorted by score. "valid Scrabble English" words are provided in the data source below. A Scrabble rack is made up of 2 to 7 characters.

Below are the requirements for the program:

- This needs to be able to be run as a command line tool as shown below (not an input statement!)
- Name the python file: `scrabble.py`
- Make a separate module named `wordscore` which contains at a minimum a function called `score word`. This `score word` function will take each word and return the score (scoring

dictionary is described below). Import this function into your main `scrabble.py` program.

- Allow anywhere from 2-7 character tiles (letters A-Z) to be inputted.
- Do not restrict the number of same tiles (e.g., a user is allowed to input ZZZZZQQ).
- Output the **total** list of valid Scrabble words that can be constructed from the rack as (score, word) tuples, sorted by the score and then by the word alphabetically as shown in the first example below.
- Then output 'Total number of words:' and the total number.
- You need to handle input errors from the user and suggest what that error might be caused by and how to fix it (i.e., a helpful error message)
- Implement wildcards as either `*` or `?`. There can be a total of **only** two wild cards in any user input (that is, one of each character: one `*` and one `?`). Only use the `*` and `?` as wildcard characters. A wildcard character can take any value A-Z. Replace the wildcard symbol with the letter in your answer (see the second example below).
- Wildcard characters are scored as 0 points, just like in the real Scrabble game. A two wildcard word can be made, should be outputted and scored as 0 points.
  - In a wildcard case where the same word can be made with or without the wildcard, display the highest score. For example: given the input 'I?F', the word 'if' can be made with the wildcard '?F' as well as the letters 'IF'. Since using the letters 'IF' scores higher, display that score.
- For partial credit, your program should take less than one minute to run with 2 wildcards in the input. For full credit, the program needs to run with 2 wildcards in less than 30 seconds.
- Write docstrings for the functions and puts comments in your code.
- You may only use the Python standard library in this assignment. However, any function in the standard library is allowed.

An example invocation and output:

```
$ python scrabble.py "ZAEFIEE"
(17, feaze)
(17, feeze)
(16, faze)
(15, fez)
(15, fiz)
(12, zea)
(12, zee)
(11, za)
(6, fae)
(6, fee)
(6, fie)
(5, ef)
(5, fa)
(5, fe)
(5, if)
(2, ae)
(2, ai)
(2, ea)
(2, ee)
Total number of words: 19
```

An example wildcard invocation and output:

An example wildcard invocation and output:

```
$ python scrabble.py "?F"
(4, ef)
(4, fa)
(4, fe)
(4, fy)
(4, if)
(4, of)
Total number of words: 6
```

Extra Credit (+10 points):

Requirements:

- Allow a user to specify that a certain letter has to be at a certain location. For the extra credit, locations of certain letters must be specified at the command line, and may not be some sort of user prompt. How you do this is up to you!
- This needs to be included and called from your regular scrabble.py file and work with the base requirements above. That is, your program must work with or without the extra credit portion and the extra credit cannot be in a different .py file.
- Please put comments, any assumptions you made, and a sample of how to run your extra credit in the extra credit cell of this notebook below - it is the last cell. If there is not an example of how to run the extra credit in this cell we will assume that you did not do the extra credit part!

The Data

The file: http://courses.cms.caltech.edu/cs11/material/advjava/lab1/sowpods.zip (http://courses.cms.caltech.edu/cs11/material/advjava/lab1/sowpods.zip) or https://drive.google.com/file/d/1ewUiZL_4HanCDsaYB5pcKEgqjMFVgGnh/view?usp=sharing (https://drive.google.com/file/d/1ewUiZL_4HanCDsaYB5pcKEgqjMFVgGnh/view?usp=sharing) contains all "valid Scrabble English" words in the official words list, one word per line. You should download the word file and keep it in your repository so that the program is standalone (instead of accessing it over the web from Python).

You can read data from a text file with the following code:

```
with open("sowpods.txt","r") as infile:
    raw_input = infile.readlines()
    data = [datum.strip('\n') for datum in raw_input]
```

This will show the first 6 words:

```
print(data[0:6])
```

Please use the dictionary below containing the letters and their Scrabble values:

```
scores = {"a": 1, "c": 3, "b": 3, "e": 1, "d": 2, "g": 2,
          "f": 4, "i": 1, "h": 4, "k": 5, "j": 8, "m": 3,
          "l": 1, "o": 1, "n": 1, "q": 10, "p": 3, "s": 1,
          "r": 1, "u": 1, "t": 1, "w": 4, "v": 4, "y": 4,
```

```
"x": 8, "z": 10}
```

Tips:

- If you don't know what "scrabble" is or the basic background of the game please look it up online!
- We recommend that you try to break down the problem into steps on your own before writing any code. Once you've scoped generally what you want to do, then start writing some code. If you get stuck, go back to thinking about the problem rather than trying to fix lots of errors at the code level.
- If you have questions on getting arguments from the command line, please review async video 6.17 and Drill 6.18.
- If you keep getting stuck, then check out: https://wiki.openhatch.org/wiki/Scrabble_challenge (https://wiki.openhatch.org/wiki/Scrabble_challenge) or https://drive.google.com/file/d/1g3yz5ljkzaAeQ-AgQR1Hofy8ZJ0jo25x/view?usp=sharing (https://drive.google.com/file/d/1g3yz5ljkzaAeQ-AgQR1Hofy8ZJ0jo25x/view?usp=sharing). This is where we got the idea for this assignment and it provides some helpful tips for guiding you along the way. However, we would recommend that you try to implement this first before looking at the hints on the website.

Good luck!

The code below will test your command line implementation of the scrabble.py code. We've made some of these tests available for you to try!

In [ ]:
```
# Code for the testing

import subprocess
from nose.tools import assert_equal
from nose.tools import assert_true
from nose.tools import assert_greater
from nose.tools import assert_less
```

In [ ]:
```
""" Code runs and can produce at least one error message """
# Autograde cell - do not erase/delete

# no rack error
!python scrabble.py
```

In [ ]:
```
""" Does not fail due to trivial mistakes and takes correct wildcard ch
aracters """
# Autograde cell - do not erase/delete

# does not fail due to case
!python scrabble.py "PENguin"
```

In [ ]:
```
# Autograde cell - do not erase/delete

# takes wildcards

!python scrabble.py "PEN*?in"
```

```
!python scrabble.py "PEN" "in"
```

In [ ]:

```
""" Produces a list of all words and scores that matches our expectatio
ns """
# Autograde cell - do not erase/delete

# The way windows and mac end lines is different
# - windows adds a \r\n to each line
# - mac/linux adds just a \n to each line
# The autograder will detect the system platform and use that to determ
ine the 'correct' solution

# The code below is shown here so you know how the autograder works
# Try this block to see if your solution matches
# If your answer looks like it matches but still throws an 'autograder'
error, please to double check your answer BUT
# The purpose of the assignment isn't to make your code match the autog
rader - if you answer is correct it will be graded so
# We will check every answer the autograder deems is incorrect

import platform

# test whether your output matches our expectation
cmd = ['python', 'scrabble.py', 'Penguin']
test = bytes.decode(subprocess.Popen(cmd, stdout=subprocess.PIPE).commu
nicate()[0])

if platform.system() == 'Windows':
    solution = '(10, penguin)\r\n(9, pening)\r\n(8, genip)\r\n(8, unpe
g)\r\n(7, ingenu)\r\n(7, penni)\r\n(7, ping)\r\n(7, pung)\r\n(7, unpen)
\r\n(7, unpin)\r\n(6, gip)\r\n(6, gup)\r\n(6, peg)\r\n(6, pein)\r\n(6,
 peni)\r\n(6, pig)\r\n(6, pine)\r\n(6, pug)\r\n(5, ennui)\r\n(5, genu)
\r\n(5, gien)\r\n(5, ginn)\r\n(5, nep)\r\n(5, nip)\r\n(5, pen)\r\n(5, p
ie)\r\n(5, pin)\r\n(5, piu)\r\n(5, pun)\r\n(4, eng)\r\n(4, gen)\r\n(4,
 gie)\r\n(4, gin)\r\n(4, gnu)\r\n(4, gue)\r\n(4, gun)\r\n(4, neg)\r\n
(4, nine)\r\n(4, pe)\r\n(4, pi)\r\n(4, up)\r\n(3, gi)\r\n(3, gu)\r\n(3,
inn)\r\n(3, nie)\r\n(3, nun)\r\n(3, ug)\r\n(3, uni)\r\n(2, en)\r\n(2, i
n)\r\n(2, ne)\r\n(2, nu)\r\n(2, un)\r\nTotal number of words: 53\r\n'
else:
```