MSc in Artificial Intelligence

COMP70053
Python Programming

Mock Timed Remote Assessment

Duration: 1 hours 15 minutes
(Includes 10 minutes for access and reading
and 5 minutes for submission)

Answer all questions

Thursday, 2nd of December 2021

Instructions

- There are 4 questions in the lab test.
- You may answer the questions in any order. The difficulty increases with each question.
- You can assume that the scripts will be executed from inside each subdirectory `qN/`.
- You are provided with files to aid you with testing (these are named `test_qN.py`). You may freely edit these for your own testing purposes.
- The Python Standard Library is sufficient to complete the questions. You are also allowed to use `numpy`, `matplotlib`, `scikit-learn` and `pandas`, although these are unnecessary.
- The examiner will manually mark your code, guided by the results of some automated internal tests. Partial marks may be awarded for any attempts that are not correctly implemented or that do not pass the internal tests.

Downloading and extracting the skeleton code

- Download the ZIP file from `https://www.doc.ic.ac.uk/~jwang4/mock/skeleton.zip`
- Extract the ZIP file to your hard drive.

Submission

- Please submit your answers via AnswerBook as a single ZIP file.
- Keep the directory structure as in the original skeleton ZIP file. That is, your ZIP file should contain the subdirectories `q1/`, `q2/`, etc. at its top level.

Tips

- Please read all questions before you start and plan your time accordingly. The 10 minutes included in the duration is for this purpose.
- You are also given an additional 5 minutes at the end of the exam to zip up and submit your code.
- Please make sure that your scripts run without any syntax errors. The examiner may penalise you otherwise.

Question 1 (5 marks)

In `q1/q1.py`, please complete the function `odd_even_swap(word)` to convert a `str` such that each character at an odd-numbered position n is swapped with the next even numbered character at position $n + 1$. If the string is of an odd length, the last character will remain where it is.

The function should return a `str` where the characters in `word` have been swapped as described.

Sample input and output

Input: `"abcdefghi"`
Output: `"badcfehgi"`

Input: `"Python Programming"`
Output: `"yPhtnoP orrgmaimgn"`

Question 2 (10 marks)

In `q2/q2.py`, please complete the function `closest_pair(coord_list)` which takes a set of coordinates in K-dimensional space and returns the pair of points that are closest to each other as measured using Euclidean distance.

`coord_list` is a `list` of `tuples`, where each `tuple` represents the coordinate of a point in K-dimensional space. Assume $K \geq 1$.

The function should return a `set` of two `tuples`, representing the pair of closest points.

The Euclidean distance D of two points a and b in K-dimensional space is defined as $D(a, b) = \sqrt{\sum_k^K (x_{a,k} - x_{b,k})^2}$.

Sample input and output

Input: `[(1, 2), (3, 5), (5, 4), (2, 1), (5, 2)]`
Output: `{(1, 2), (2, 1)}`

Input: `[(1, 1, 1), (2, 3, 1), (3, 3, 2), (1, 3, 3), (4, 1, 1)]`
Output: `{(2, 3, 1), (3, 3, 2)}`

Question 3 (15 marks)

In `q3/q3.py`, please complete the function `top5_bigram_frequency(filename)` to return a dictionary containing the frequency count of the top 5 bigrams in `filename`.

An bigram is a sequence of two consecutive words. For example, in the sentence `"my term was very intense"`, the set of valid bigrams are `"my term"`, `"term was"`, `"was very"`, `"very intense"`.

Your function should:

- first convert the text contained in the file `filename` to be all **lowercase**.
- then return a `dict` of the top 5 most frequent bigrams and their frequency.

You can assume that each word in the file is separated by a space.

You may also assume that each line in `filename` is independent; there is no need to consider the last word from the previous line to be part of the bigram. For example, for the following two lines, `"buns one"` is not a bigram.

```
Line 1: hot cross buns hot cross buns
Line 2: one a penny two a penny hot cross buns
```

Sample input and output

Using `"baby.txt"` provided in `q3/`, `top5_bigram_frequency("baby.txt")` should return:

```
{'baby baby': 36, 'like baby': 14, 'yeah yeah': 12, 'baby oh': 9, "i 'm": 9}
```

Using `"frankenstein.txt"`, `top5_bigram_frequency("frankenstein.txt")` should return:

```
{'of the': 527, 'of my': 272, 'in the': 263, 'i was': 228, 'i had': 218}
```

Question 4 (20 marks)

In `q4/q4.py`, please complete the function `get_winner(board)` to compute the winner of a game of Tic-Tac-Toe given the state of a `board`.

`board` is a two-dimensional `list` representing an $N \times N$ grid, where N can be an arbitrary number between 3 and 50 (both inclusive). Each cell can be filled with either `"x"`, `"o"` or `" "` (a blank space).

The function returns one of the following: `"x"`, `"o"`, `"draw"` or `None`.

The winner of the game is where either `"x"` or `"o"` has occupied all grid cells in a single row, column or diagonal. If all cells are occupied without a winner, return `"draw"`. Otherwise, return `None` to indicate that the game has not yet finished.

Sample input and output

```
board = [
    ['x','o',' ','o','x'],
    ['x','x','o',' ',' '],
    ['o','o','x',' ',' '],
    [' ','o','o','x','o'],
    [' ',' ',' ',' ','x']
]
```

Output: `"x"`

```
board = [
    ['o','x','o','x'],
    ['o','o','o','x'],
    ['x','x','x','o'],
    ['x','o','x','o']
]
```

Output: `"draw"`

```
board = [
    [' ','x',' '],
    ['o','o','o'],
    ['x','x',' ']
]
```

Output: `"o"`

```
board = [
    [' ',' ',' ','o',' ',' ','o'],
    [' ',' ',' ',' ',' ',' ',' '],
    [' ',' ',' ',' ',' ',' ',' '],
    ['x',' ','x','x',' ',' ',' '],
    [' ',' ',' ',' ',' ',' ',' '],
    [' ',' ',' ',' ',' ',' ',' '],
    ['x',' ',' ',' ',' ',' ','o']
]
```

Output: `None`

End of test