(/) MITx: 6.00x Introduction to Computer Science and Programming

Ankit_Aggarwal (/dashboard) ▾

Courseware (/courses/MITx/6.00x/2012_Fall/courseware) Course Info (/courses/MITx/6.00x/2012_Fall/info)

Textbook (/courses/MITx/6.00x/2012_Fall/book/0/) Discussion (/courses/MITx/6.00x/2012_Fall/discussion/forum)

Wiki (/courses/MITx/6.00x/2012_Fall/course_wiki) Progress (/courses/MITx/6.00x/2012_Fall/progress)

COMPUTER CHOOSES A WORD : 10.0 POINTS

Part B is dependent on your functions from `ps4a.py`, so be sure to complete `ps4a.py` before working on `ps4b.py`

Now that you have completed your word game code, you decide that you would like to enable your computer (SkyNet) to play the game (your hidden agenda is to prove once and for all that computers are inferior to human intellect!) In Part B you will make a modification to the `playHand` function from part A that will enable this to happen. The idea is that you will be able to compare how you as a user suceed in the game compared to the computer's performance.

It is your responsibility to create the function `compChooseWord(hand, wordList)`. Pseudocode is provided in the file `ps4b.py`.

If you follow the pseudocode, you'll create a computer player that is legal, but not always the best. Once you've implemented it following our approach, feel free to try your own approach! As much as we'd love to give you credit for making an improved `compChooseWord` function, we hope you can understand our automatic grading facilities are limited in their ability to accept differing solutions.

A Note On Runtime

You may notice that things run a bit slowly when the computer plays. This is to be expected - the `wordList` has 83667 words, after all!

However, don't worry about this issue when running your code in the checker below! We load a very small sample wordList (*much* smaller than 83667 words!) to avoid having your code time out.

Test Cases

Some test cases to look at:

```
>>> compChooseWord({'a': 1, 'p': 2, 's': 1, 'e': 1, 'l': 1}, wordList)
appels
>>> compChooseWord({'a': 2, 'c': 1, 'b': 1, 't': 1}, wordList)
acta
>>> compChooseWord({'a': 2, 'e': 2, 'i': 2, 'm': 2, 'n': 2, 't': 2}, wordList)
imamate
>>> compChooseWord({'x': 2, 'z': 2, 'q': 2, 'n': 2, 't': 2}, wordList)
None
```

```
def compChooseWord(hand, wordList):
    """
    Given a hand and a wordList, find the word that gives
    the maximum value score, and return it.

    This word should be calculated by considering all the words
    in the wordList.

    If no words in the wordList can be made from the hand, return None.

    hand: dictionary (string -> int)
    wordList: list (string)
    returns: string or None
    """
    bestscore=0
    bestword=None
```

Correct

Test results

```
CORRECT
```

See full output

Check **Save** *You have used 2 of 30 submissions*

Show Discussion **New Post**