HW6 syntax constraints

Certain Python syntax will be banned in each homework. The tester will clarify what is not allowed. Run against the tester to discover whether you are using a banned keyword.

HW6 style constraints

- each line should contain no more than 80 characters
- do not hardcode the test cases into your solutions
  (this is counterproductive anyway, since the autograder uses different test cases)

Grading

grading a problem is correct if all autograde tests pass (this only applies to the Shakespeare problem: the turtle graphics problem will be graded by hand)
grading A: flags and Shakespeare; B: randomized flags; D: correct flags, but not randomized

Instructions

To work on HW6:

- build a HW6 directory under your 103fa18 directory
- download `hw6_flag.py`, `hw6_first_words.py`, and `tester_hw6.py` from Canvas, and move these files to your HW6 directory
- open a Canopy terminal
- in this Canopy terminal, move to your HW6 directory using the cd command
- solve the two Python scripts as usual
- note that the flag question is not autograded, and has no tester
- but hw6_first_words.py (which is only required for an A) should be tested in a Canopy terminal using `python tester_hw6.py`; you can first find syntax errors and debug using `python hw6_first_words.py`
- when testing your code, always run it from a terminal; this is particularly important in this homework, where you are using turtle graphics or using file input
- once you are confident that your code works, submit on Canvas
- before submitting your code, please ensure two things: you have removed syntax errors, and you have tried it against the tester to remove some of the semantic errors (remember that the tester cannot remove all semantic errors, so think beyond the tester too)

HW6 problems

In this homework, you should assume that the input is correct. To simplify your code, there is no need to test the type or value of the input parameters (although that is a great thought). Imagine that the test data has already been vetted by a separate preprocessing step.

flagInitials ()

Write the function `flagInitials` that draws your three initials (first name, middle name, last name) in maritime letter flags, using turtle graphics. For example, 2008 Turing Award winner Barbara Liskov would draw the flags associated with her initials BJL. (If you have multiple middle names, choose your favourite.)

Maritime letter flags are used on ships to signal other ships. For a good drawing of each letter flag, see the Wikipedia entry on **'International maritime signal flags'**. Note that each flag is a square. Please draw a black border around each flag (1 pixel wide).

You will use randomization to vary how you draw your flags. The bottom left corner of the first flag is drawn at a random point in the rectilinear square bounded by (-200,-200) and (0,0). Each flag is of the same width w, which is chosen randomly in (100,200). The gap between consecutive flags should be a random number between 1 and 10. Since these random numbers determine the size and position of the flags, each time you run the code you should see a slightly different drawing.

The use of random size and position is my way of encouraging the use of variables over magic numbers. A statement like 'forward(40)' uses a magic number 40. A statement like 'forward(w/2)' where 'w' has a semantic meaning of width, is much better: it documents your thought, and it is easily generalized to different widths.

This question will be graded by the TA's, not by the autograder.

See next page for second question (advanced).

firstWords (fn)

Write the function `firstWords` that, given the filename `fn` of a Shakespeare play from Folger Digital Texts, builds and returns a Python dictionary that records, for each character, the **number of speeches** of the character, the **act** of the **first speech** of the character, and the **scene** of the first speech of the character.

The keys of the dictionary are character names, where a character name is defined as the first words listed for a character (e.g., BOTTOM). The value associated with this key is a list of 3 integers. The dictionary entry for Juliet in Romeo and Juliet should be `'JULIET': [118, 1, 3]` since Juliet first speaks in Act 1, Scene 3 and has 118 speeches in the play. If a character speaks that is not in the list of characters of the play (e.g., CITIZENS in Romeo and Juliet), simply ignore it (no dictionary entry). If a character never speaks (e.g., CHORUS in Romeo and Juliet), you should still have a dictionary entry for this character, and simply set its act and scene to 0: `'CHORUS': [0, 0, 0]`.

The first speech is defined by the act and scene that the character first *speaks*, not the first time they are mentioned. You may assume that act and scene are recorded as in the Folger Digital Texts: (e.g., 'ACT 1', 'Scene 1'). You may assume that the words 'ACT' and 'Scene' only occur at the beginning of a line when they are recording an act or scene. You may assume that acts and scenes begin at 1 and increase by 1 each time.

The text of the Shakespeare play is taken from Folger Digital Texts. I have provided two examples: *A Midsummer Night's Dream* and *Romeo and Juliet* on Canvas. If you choose, I encourage you to generate more test data by downloading the texts of other Shakespeare plays from www.folgerdigitaltexts.org/download/ in txt format, then editing the play as defined below.

A character is defined as a character in capital letters in the section called 'Characters in the Play'. When a character speaks, later in the play, their name is listed in all-caps at the beginning of the line. I have edited the text so that the first name in the list of characters is always the same as the name used later in the play to refer to that character's speech. In particular, there are two types of change: 1) if a character has two names and the text uses both names, these names are combined into one (FRIAR LAWRENCE becomes FRIARLAWRENCE). 2) if a character has two names and only one is used (e.g., NICK BOTTOM uses BOTTOM), I have placed the name that is used *first*. For example, BOTTOM, NICK. Or QUINCE, PETER. These are the only changes to the text. You may assume that any play that is used to test your code will follow these constraints. (Yes, I will use other plays in the autograder.)

To develop your algorithm, I suggest looking through the structure of the two Shakespeare plays I have given you. **Please include your algorithm in the docstring of the firstWords function.** You should write other functions to divide and conquer your solution: please **include docstrings for every function that you add**. Please follow the 103 style guide conventions for docstrings: one-line definition of the function, parameters, return value (as illustrated in the functions of hw6_firstWords.py).

As you develop your code, you should use print statements to help you debug and understand your code; but remove them before you submit your code. (The tester will work if you just comment them out, so you can do that as you test, but please remove completely out of courtesy for the TA.) Also, just before submitting, test your code using the tester to verify that it passes.