Meeting 8 Exercises

Info 206

19 September 2017

Write individual scripts to solve the following exercises.

1. Recursion: Computing Fibonacci Numbers

You are probably familiar with the famous Fibonacci sequence of numbers, which begins like this:

```
1, 1, 2, 3, 5, 8, 13, 21...
```

We will index from 0, so the 0th and 1st numbers are both 1. The 2nd Fibonacci number is found by summing the 0th and 1st: $1 + 1 = 2$. The 3rd is found by summing the 1st and 2nd: $1 + 2 = 3$. After this point, each Fibonacci number is found by summing the previous 2.

You are to write a recursive function to compute the nth Fibonacci number. This means that your function will call itself and will not include explicit loops. Save the script as fibonacci_[lastname].py and push to your git repository when complete.

```
def Fibonacci(n):
    pass
```

Hint: As with the Factorial example, your function should include a line that looks a lot like the mathematical definition of the nth Fibonacci number.

Another Hint: It's possible for a recursive function to call itself more than once.

Once you are done, think about whether your function is efficient. Compare it to a non-recursive implementation.

2. For Loops

Write a script that prompts the user for two words. Print all the letters that are common to both words, in alphabetical order, using for loops in your solution.

```
Enter one word: Home
Enter another word: meter
Letters in common: em
```

Save your script as alpha-order_[lastname].py.

3. Comprehensions

Save your script as comprehensions_[lastname].py.

a. Use a comprehension to make a list of the square numbers below 100 that give a remainder of 1 when divided by 3.

b. A string is defined in the code snippet below. Split it into individual words and use a comprehension to make a list of the first letters of each word in the snippet.

text = "A new, a vast, and a powerful language is developed for the future use of analysis, in which to wield its truths so that these may become of more speedy and accurate practical application for the purposes of mankind than the means hitherto in our possession have rendered possible."

Note: the slashes just mean that the string continues onto the next line. If you print the text, it will make no difference

c. A Pythagorean triple is a set (x,y,z), with positive integers x???y???z such that x2+y2=z2. Use a comprehension to make a list of all Pythagorean triples with numbers below 25.

d. Given a word, provided below, use a comprehension to make a list of all strings that can be formed by deleting exactly one character from the word.

```
word = "welcomed"`
```

e. Given a word, provided below, use a comprehension to make a list of all strings that can be formed by replacing exactly one vowel in the word with a different vowel.

```
word = "Booted"`
```

4. [EXTRA CREDIT] A Game of Chess

You place a pawn at the top left corner of an n-by-n chess board, labeled (0,0). For each move, you have a choice: move the pawn down a single space, or move the pawn down one space and right one space. That is, if the pawn is at position (i,j), you can move the pawn to (i+1,j) or (i+1, j+1).

Ask the user for the size of a chessboard, n. Find the number of different paths the pawn could take to reach each position on the chess board. For example, there are two different paths the pawn can take to reach (2,1):

(0,0) -> (1,0) -> (2,1) (0,0) -> (1,1) -> (2,1)

Print the board with the number of ways to reach each square labeled as shown below.

Enter a size: 3

```
1 0 0
1 1 0
1 2 1
```

Below is the code to take a convert that board size input.

```
n = int(input("Enter a board size: "))
```

Hint: It may help to write out pseudocode for this problem in a separate cell. Partial credit will be provided if you provide logical pseudocode even if you cannot solve the problem in Python.

Save your script as chess_[lastname].py.