Assignment

January 29, 2016

1 For Loops

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

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

```
In [ ]: name = input("Enter your name: ")
```

2. 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.

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

2 Bisection Search

1. Try inputting 0.25 into the bisection search algorithm below and confirm that it doesn't work. Then correct the algorithm so that it works for all positive numbers.

```
In [ ]: ## Bisection Search to Find a Square Root

        x = float(input("enter a number:"))
        epsilon = 0.00001
        num_guesses = 0
        low = 0.0
```

```
high = x
ans = (high + low)/2.0
while high - low >= 2 * epsilon:
    print("low =",low,"high =", high)
    num_guesses += 1
    if ans ** 2 < x:
        low = ans
    else:
        high = ans
    ans = (high + low)/2.0
print('number of guesses =', num_guesses)
print(ans, 'is close to square root of', x)
```

You run the bisection search algorithm to find the square root of x to a precision of epsilon. Write down a formula for how many loop iterations it takes for the algorithm to complete. Hint: first suppose that x has the special value $epsilon * 2^n$, where n is an integer.

3 Part 3 Bisection Search Part 2

1. In your own words, why does the bisection search algorthm for finding square roots work faster on large numbers than the brute force algorithm? Write your answer below.
2. In your own words, why does Newton's method for finding square roots outperform both the brute force and the bisection search algorithms? Write your answer below.
3. Implement this digit-by-digit algorithm to find the square root of x to a precision of epsilon.
4. Begin with a step of 1 and guess of zero.
5. Increase the guess by step repeatedly as long as doing so would not cause guess^2 to exceed x.
6. If the step is greater than epsilon, divide the step by 10 and go back to step 2.In print preview, the three questions are each showing as number 1 (not 1, 2, 3), and these steps are appearing as A, B, C. Please determine how they will appear in final form. "Step 2" may need to be changed to "step C."

Notice that once a digit has been found, it is never changed again. Try using your script to find the square root of 10 to 10 decimal places.

4 Part 4 Comprehensions

1. Use a comprehension to make a list of the square numbers below 100 that give a remainder of 1 when divided by 3.
2. 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.

```
In [ ]: text = "I live my life a quarter mile at a time. \
        Nothing else matters: not the mortgage, not the store, \
        not my team and all their bullshit. \
        For those ten seconds or less, I'm free."
        # the slashes just mean that the string continues onto the next line
        # if you print the text, it will make no difference
```

3. 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.
4. 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"
```

5. 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"
```

4.1 Please submit feedback for any and all homeworks!

http://goo.gl/forms/74yCiQTf6k