CMPSC-122: Intermediate Programming
Spring 2018

Homework 5

Due Date: 03/21/2018, 7:59 am
100 points

Overview

Traditionally, for every piece of code we write (let it be a single function or class method), we would feed it some arbitrary inputs to make sure that it works the way we have expected. And this might sound like a reasonable approach given that everything works as it should and if we do not plan to make any changes to the code until the end of days. Of course, this is rarely the case. If we modify our code, how do we make sure we did not break anything? How do we providing evidence that everything was tested and is supposed to work properly?

Instructions:

In class, we discussed the Software Development Life Cycle and that testing your code is the foundation of solid software development. Getting used to writing testing code and running this code in parallel is now considered a good programming habit. We also discussed the advantages of unit testing, and how we can write down several test cases and let them be checked every time we make changes to our code.

Imagine you are part of the testing team at company XYZ. Your company is developing a new piece of software that will determine whether or not a string is a palindrome. The development team provides you the following code for testing:

```
# Function takes a string as a parameter and returns True if the string is a
palindrome, False otherwise
def isPalindrome(text):
    # Reversing text
    rev = text[::-1]

    # Checking if both strings are equal or not
    if (text == rev):
        return True
    return False
```

Your job as part of the testing team is to determine whether or not this piece of code meets the product requirements in order to be deployed.

- Use the unittest module to perform unit testing for the function *isPalindrome* and prove that the function works properly
- If the code does not meet the requirements modify the function *isPalindrome* to work according to the requirements

Product Requirements:
Remember that a string is a palindrome if it is spelled the same both forward and backward.

- Sentences can also be palindromes, therefore, punctuation, capitalization, and spaces should be ignored
- If the user provides an input that is not a string, program should False

EXAMPLES:

```
>>> isPalindrome("alula")
True
>>> isPalindrome("love")
False
>>> isPalindrome("Madam")
True
>>> isPalindrome(12)
False
>>> isPalindrome(11)
False
>>> isPalindrome(12.5)
False
>>> isPalindrome(12.21)
False
>>> isPalindrome("Mr. Owl ate my metal worm")
True
>>> isPalindrome("Live on time, emit no evil")
True
>>> isPalindrome("Step on no pets")
True
>>> isPalindrome([12,21])
False
>>> isPalindrome("Cigar? Toss it in a can. It is so tragic.")
True
>>> isPalindrome("Yen o' money.")
True
>>> isPalindrome("travel.. a town in Alaska")
False
```

Deliverables:

You must upload the following 2 files to Vocareum:

- HW5.py with a isPalindrome function that works properly [70 points]
- test.py that contains the unittest code [30 points]

Notes:
One of the purposes of unit testing is to provide evidence that everything was tested and it works properly. Feedback will be provided if your code does not work according to the product requirements, but no partial credit will be given.

Hint:
Section 6.1.1 of https://docs.python.org/3/library/string.html import string

List of palindromes:
http://www.palindromelist.net/