Lab Week 5

Practice
- designing algorithms and programs
- recursive algorithms and programs
- testing programs

Lab Exercise 1

**Summing digits – iterative and recursive functions:** Write three functions that compute the sum of the numbers in a given string using a for-loop, a while-loop, and recursion. The functions should return the sum of the numbers. For example,

```python
>>> sumDigits('12345')
15
```

**NOTE:** the docstring and return statement should be the same for each function.

(a) Write out a simple example (or two or three) for `sumDigits`:

(b) In pseudocode (pseudocode is somewhere between natural language and well-formed Python), write an algorithm to sum the digits in a string. For the recursive algorithm, remember to consider: (1) what is (are) the base case(s)? and (2) what is the recursive step? Clearly indicate these.
(c) Check the algorithm in (b) against the examples in (a); revise algorithm as needed:

(d) Generate additional test cases for the sumDigits algorithm. Test cases should include simple examples, boundary or edge conditions, and other examples to ensure adequate testing. For each test case, indicate what it is being tested.

(e) Write the function header, docstring (type contract, brief description, simple examples, and test cases from (d)) and return statement for function sumDigits:

(f) Complete the code for each of the three sumDigits functions (for, while, and recursive implementations. Remember to use only tools in current Python “toolkit”:
Lab Exercise 2

**Palindrome – recursive function:** Given a string, s, determine if it is a palindrome (reads the same backwards and forwards, e.g., 'kayak', 'racecar', 'eye'); return True if s is a palindrome and False otherwise. For example,

```python
>>> isPal('kayak')
True
>>> isPal('hello')
False
```

(a) Write out a simple example (or two or three) for palindrome:

(b) In pseudocode (not Python), write an algorithm to determine whether a string is a palindrome. For designing a recursive algorithm, it is important to consider (1) what is (are) the base case(s) and (2) what is the recursive step? Clearly indicate these.

(c) Check the algorithm in (b) against the examples in (a); revise algorithm as needed:
(d) Generate additional test cases for the palindrome algorithm. Test cases should include simple examples, boundary or edge conditions, and other examples to ensure adequate testing. For each test case, indicate what it is testing.

(e) Write the function header, docstring (type contract, brief description, simple examples, and test cases from (d)) and return statement for function palindrome:

(f) Complete the code for the palindrome function (use only tools in current Python “toolkit”):
More practice (Project 5 Challenges):

(0) Write three functions that compute the sum of the numbers in a given integer (i.e., do not use strings) using a for-loop, a while-loop, and recursion. The functions should return the sum of the numbers.

(1) Rewrite isPal to implement an iterative algorithm. NOTE: the docstring and return statement of isPal should not change. Also, consider glass box testing: are additional tests needed to adequately exercise all of the code?

(2) Develop and implement a recursive algorithm for determining the value of integer \( j \) raised to the power of integer \( k \). For example, \( \text{powr}(4, 3) \rightarrow 64 \)