CIS 210 Winter 2018
Lab Week 5

Practice

• step-by-step computational problem solving (how to write a Python program)
• recursive algorithms and programs
• testing programs

Step-by-step computational problem solving (how to write a working Python program).
Recall from class and Lab 2:

Thinking:
1. problem: review, clarify → write out simple examples by hand.

2. algorithm: review, revise, or create algorithm to solve the problem → write pseudocode; check against examples from step 1.

3. program: design (top down) – review project specification for required functions and possibly add others → for each function, write function header, docstring (type contract, brief description, simple examples), and return statement; check against examples, pseudocode from steps 1 and 2.

Coding:
4. program: coding (bottom up) – for each function, starting with lowest level, review pseudocode, function outlines, and Python toolkit → write Python code

5. program: testing and debugging – for each function, using examples of use; revise and retest as needed
Lab Exercise 1

Palindrome - iterative and recursive functions: 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.

(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:

(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) Write the rest of the Python code for palindrome:
Lab Exercise 2 – more palindrome

(g) Assuming the algorithm implemented in Lab Exercise 1 is an iterative algorithm, think about a recursive implementation (or consider iterative, for those who did recursive).

-- what is the base case?

-- what is the recursive step?

(h) Copy function palindrome and rename the copy palindrome_r (or palindrome_i). Write the Python code needed to implement the recursive (or iterative) algorithm outlined in (g). NOTE: the header, docstring, and return statement should not change.

(i) Finally, consider glass box testing: are additional tests needed, in either version of the palindrome, to adequately exercise all of the code?
More practice (Project 5 Challenges):

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

(2) Develop and implement a recursive algorithm for determining the greatest common divisor (GCD) of two integers, $a$ and $b$. For example, $\text{gcd}(21, 15) \rightarrow 3$

Hint: One way to find the GCD of two numbers is Euclid's algorithm, which is based on the observation that if $r$ is the remainder when $a$ is divided by $b$, then $\text{gcd}(a, b) = \text{gcd}(b, r)$.

Note: While these are interesting exercises, Python comes with this functionality: check out the $\text{pow}$ and $\text{math.gcd}$ functions. The $\text{pow}$ function can also be called using an operator: $\text{pow}(4, 3) == 4**3$. NOTE: $4**3 != 4^3$. See Project 5 Challenges for more about this.