Lab 3

Consider the implementation of quicksort and partition (algorithms 2.6 and 2.7) given in the .py file for the lab.

(1) Give a basic explanation of how the algorithm works. What does partition do? What is pivot? How is pivot chosen each iteration?

(2) Add a print statement to partition that shows how the list is being "divided" for later "conquer." You should get output that matches the action of the algorithm as shown in Figure 2.3.

(3) Change partition so it chooses a random pivot (write import random at the top and then use random.randint(a, b) to generate a random int from a to b, inclusive)

(4) Do you notice a change in the number of iterations using a random pivot vs. the original code?

(5) Create a 100-element list of random numbers and sort it using both pivot selection methods. Do you notice a difference?

(6) What is the worst-case scenario for quicksort? What do you think its complexity would be? Set up a worst-case list and run it through to test