1. Copy and save the Python program below in a file called slowListBuild.py.

(a) Run slowListBuild.py and report the amount of time (in seconds) that the program took to complete.

Answer: The result is 120.556999922 seconds.

(b) Suppose you double the value of n from 500,000 to a million. Do you think the program will take roughly twice as much time (yes or no)? Explain your answer in 1-2 sentences.

Answer: No, this program will read the list every time when it execute the for-loop, as the length of the list get longer, the program will need more time to read the list.

(c) Rewrite the program using append rather than insert and report the time the program took. Make sure your new program does exactly what the original did, i.e., it should build the same list as before. Call your new program fasterListBuild.py and submit it as part of your solution.

Answer: It takes 0.18700003624 seconds.

(d) Explain in 1-2 sentences why there is such a big difference in running times of the two versions of the programs.

Answer: “insert” read the list every time when it execute the for-loop, “append” doesn’t do this, it simply inserts the result to the list every time. So, “insert” spends more time than “append”.

(e) Now rewrite the program using insert, but make sure to insert at the end of the list rather than at the front In other words, you are being asked to just perform the append operation, but by calling insert instead of append. Make sure your new program does exactly what the original did, i.e., it should build the same list as before. Report on the running time of your new program. Explain in 1-2 sentences why this new program has such a different running time relative to the original program. You do not have to submit your program for this part.

Answer: It takes 0.358999967575 seconds. Instead of reading the list, the program reads the length of the list. That’s why it takes much less time than the original program

2. Your task is to write a program that builds a list of 500,000 distinct, randomly chosen numbers in the range 1 through a million (inclusive of 1 and a million). The algorithm you are required to use is this: (i) start with an empty list called L, (ii) repeatedly pick a number at random in the range 1 through 1000000, check if the number is in L and if not append it to L. Your loop should terminate when the L contained 500,000 numbers.

(a) Implement the program described above, save it as makeRandomList.py, and submit it as part of your solution. Using the time () function from the time module to compute and report the running time of your program.

Answer: It takes 6118.04099989 seconds.

(b) The program can be made much more efficient if you use dictionaries rather than lists to store the generated numbers. Reimplement the algorithm using a dictionary rather than list. Save your new program in a file called fasterMakeRandomList.py. Report on the running time of the program.

Answer: It takes 2023.48009761 seconds.