**CS 2302 Data Structures**

**Fall 2019**

**Lab Report #7**

Due: December 9, 2019

Professor: Olac Fuentes

TA: Anindita Nath

**Introduction**

For this lab, I implemented different algorithm design techniques to test whether a graph has a Hamiltonian Cycle. I also modified the edit distance function that calculates the minimum number of edits required to convert the first word to the second word. I implemented three algorithm design techniques, backtracking, dynamic programming, and randomization.

**Proposed Solution Design and Implementation**

**Randomization:** For the randomization algorithm I used the pseudo code provided to guide me. I first began generating random edges from the graph. I then made these random edges into an edge list. This was done about 100 times, each time a different random edge list was generated.

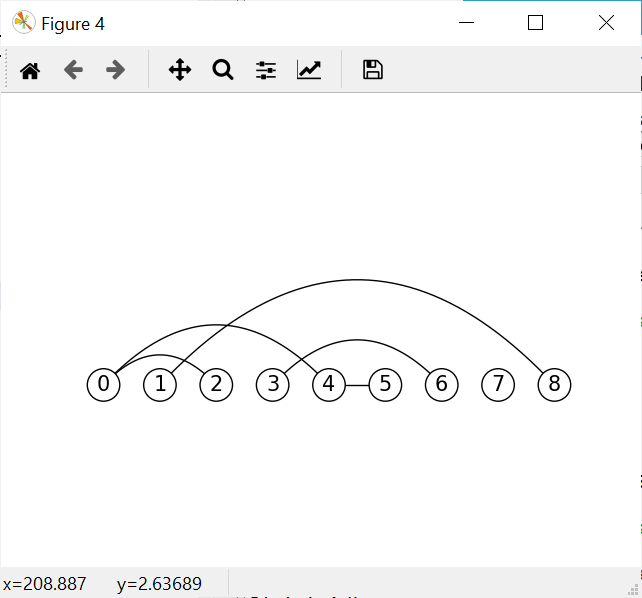
**Backtracking:** For the backtracking algorithm, I also used a similar approach as I did for the randomization. I did not however use the maximum trials.

**Dynamic Programming:** For the dynamic programming, I modified the edit distance that was already provided. I modified it by creating a set of vowels. When the program gets to the part where the replacements happen, I modified it by only allowing replacements where the characters being interchanged are both vowels or both consonants.

**Experimental Results**

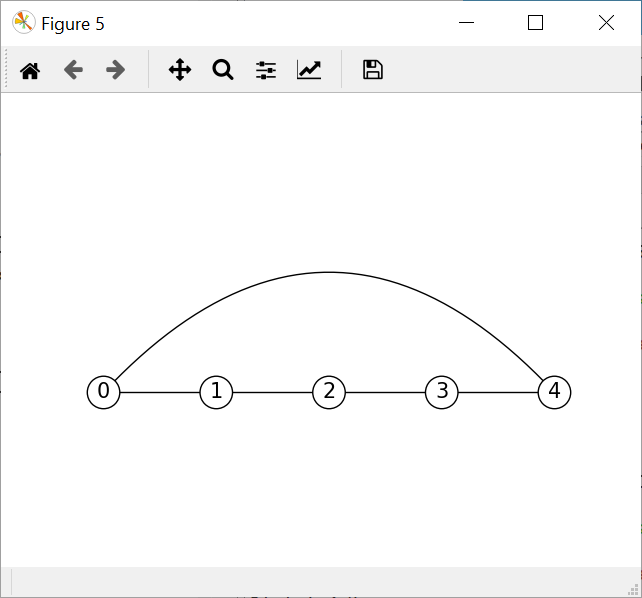
To test my program, I tried different types of graphs to test if they contain a Hamiltonian Cycle. I also timed the different algorithm design techniques to see how efficient backtracking and randomization techniques were when it came to determining if the graph contained a Hamiltonian cycle. For the dynamic programming, I tested the edit distance function with random pairs of words.

The first graph that I tested is displayed below. This graph does not contain a Hamiltonian Cycle.



**Graph #1**

The second graph did have a Hamiltonian cycle.



**Graph #2**

**Randomization and Backtracking:** For these two algorithm design techniques, I tested them with a graph that contained a cycle and a graph that did not. The running times are displayed below. The average running times show that randomization is more efficient that backtracking. It is also due to the constant looping that it is more efficient.

|  |  |
| --- | --- |
| **Algorithm Design Technique** | **Average running Time** |
| Randomization | ..0501 |
| Backtracking | .7779 |

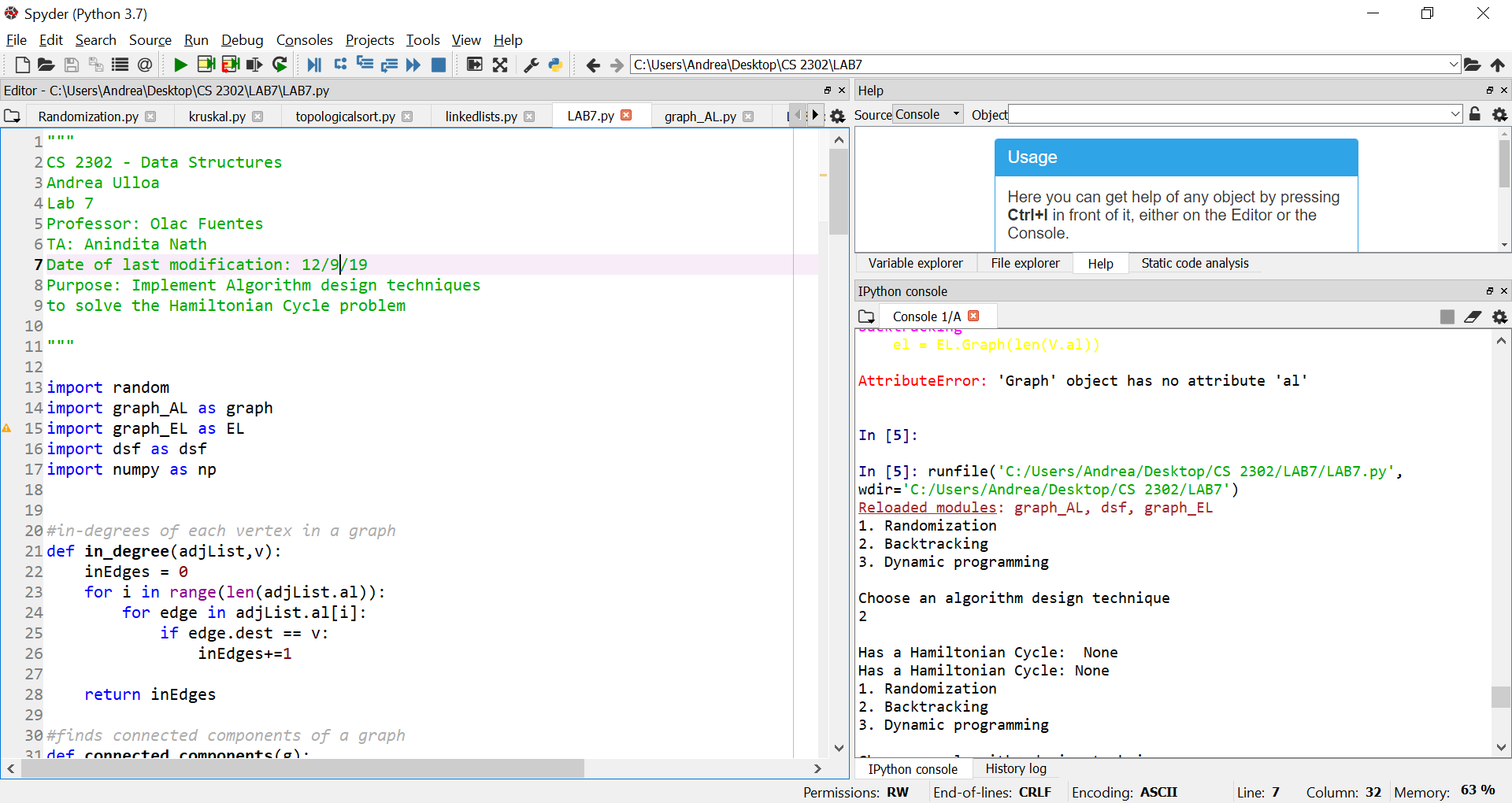
**Dynamic programming:** To test the edit distance function I used 3 different pairs of words to check the efficiency of the modified edit distance function. I calculated the edit distance and average running times for different pairs of words.

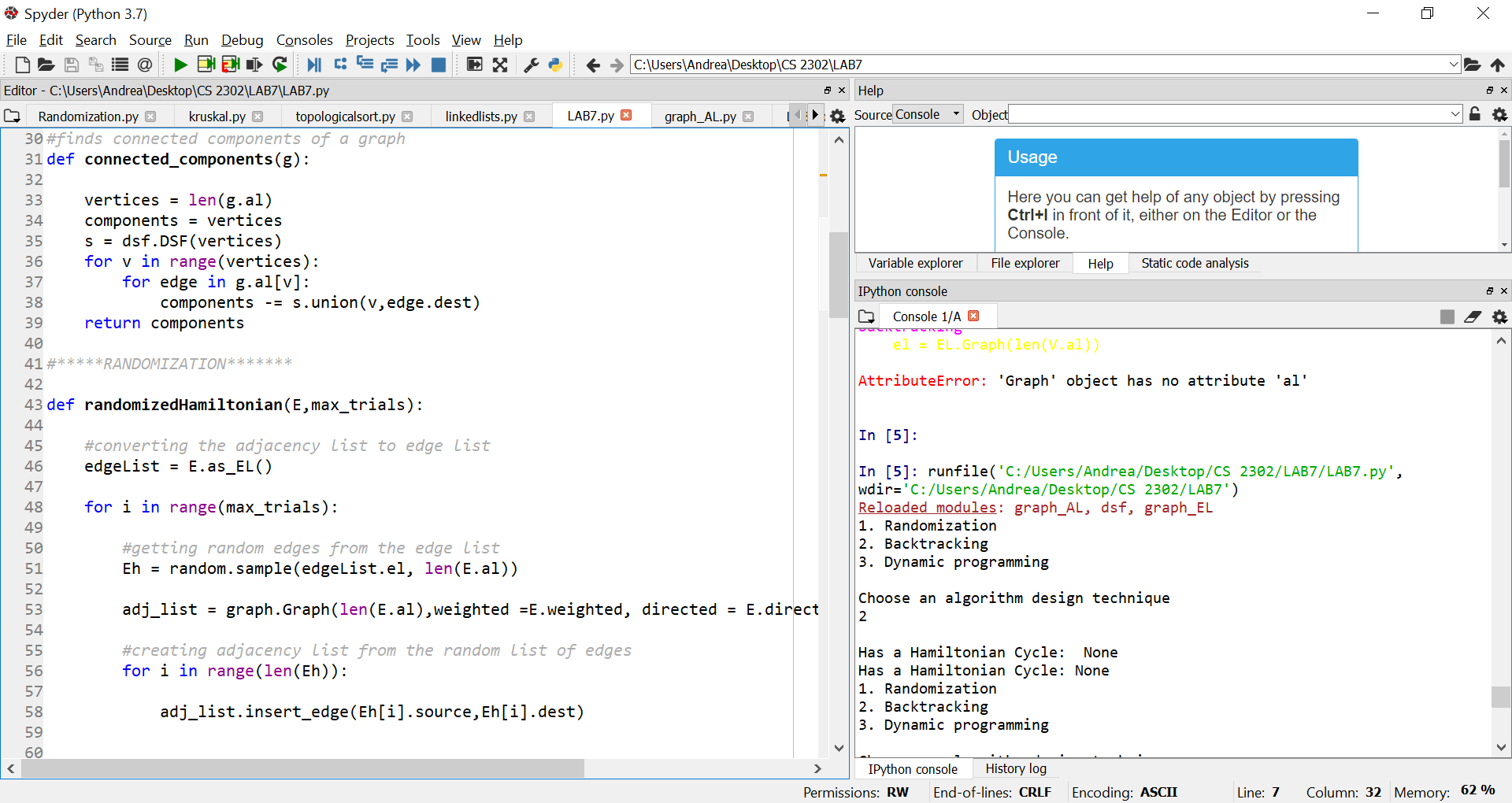
|  |  |  |
| --- | --- | --- |
| **Word Pairs** | **Edit Distance** | **Running Time** |
| “Study” “Hair” | 5 | 16.74 |
| “Can” “Tan” | 1 | 4.09 |

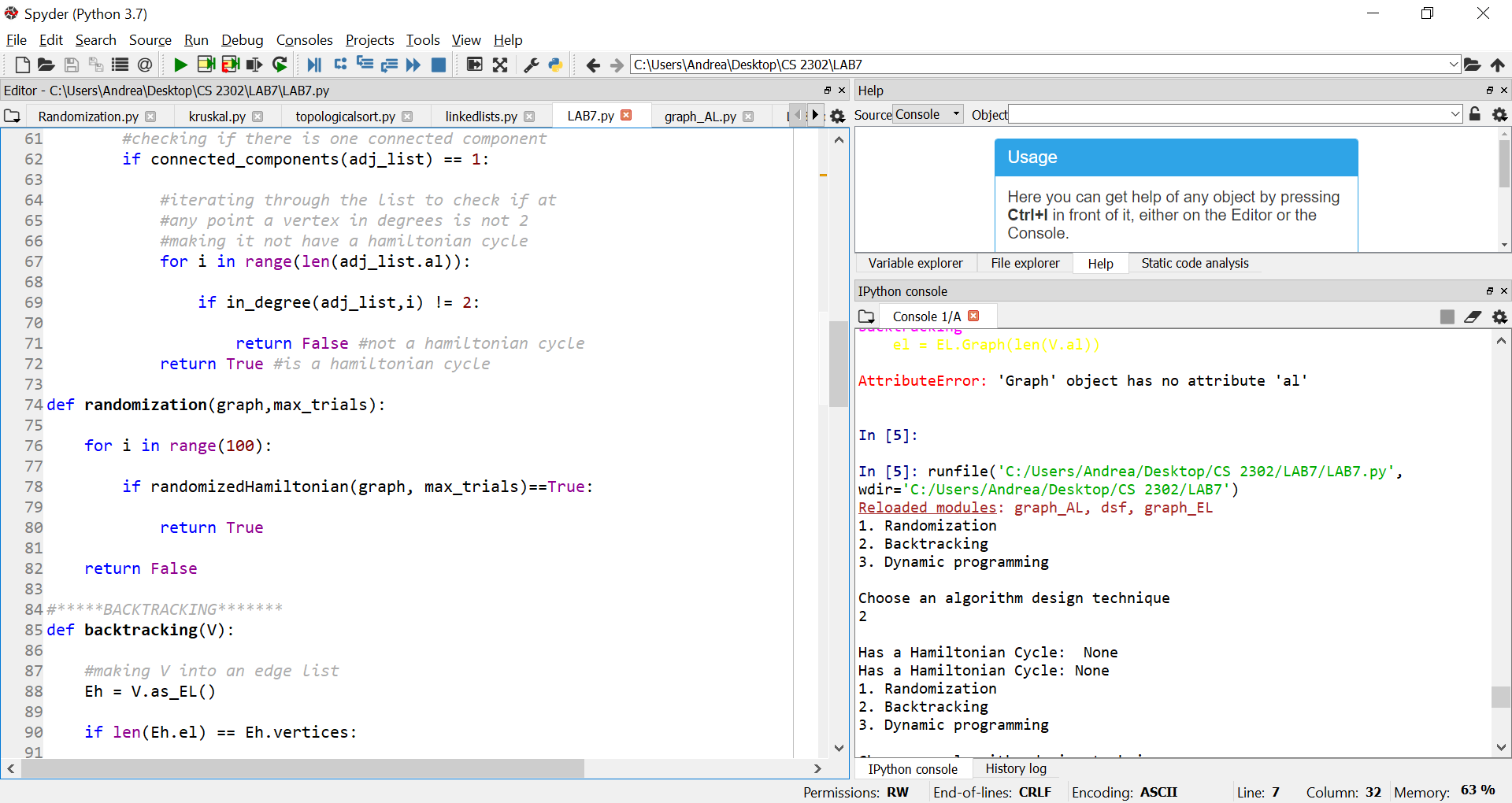
**Conclusion**

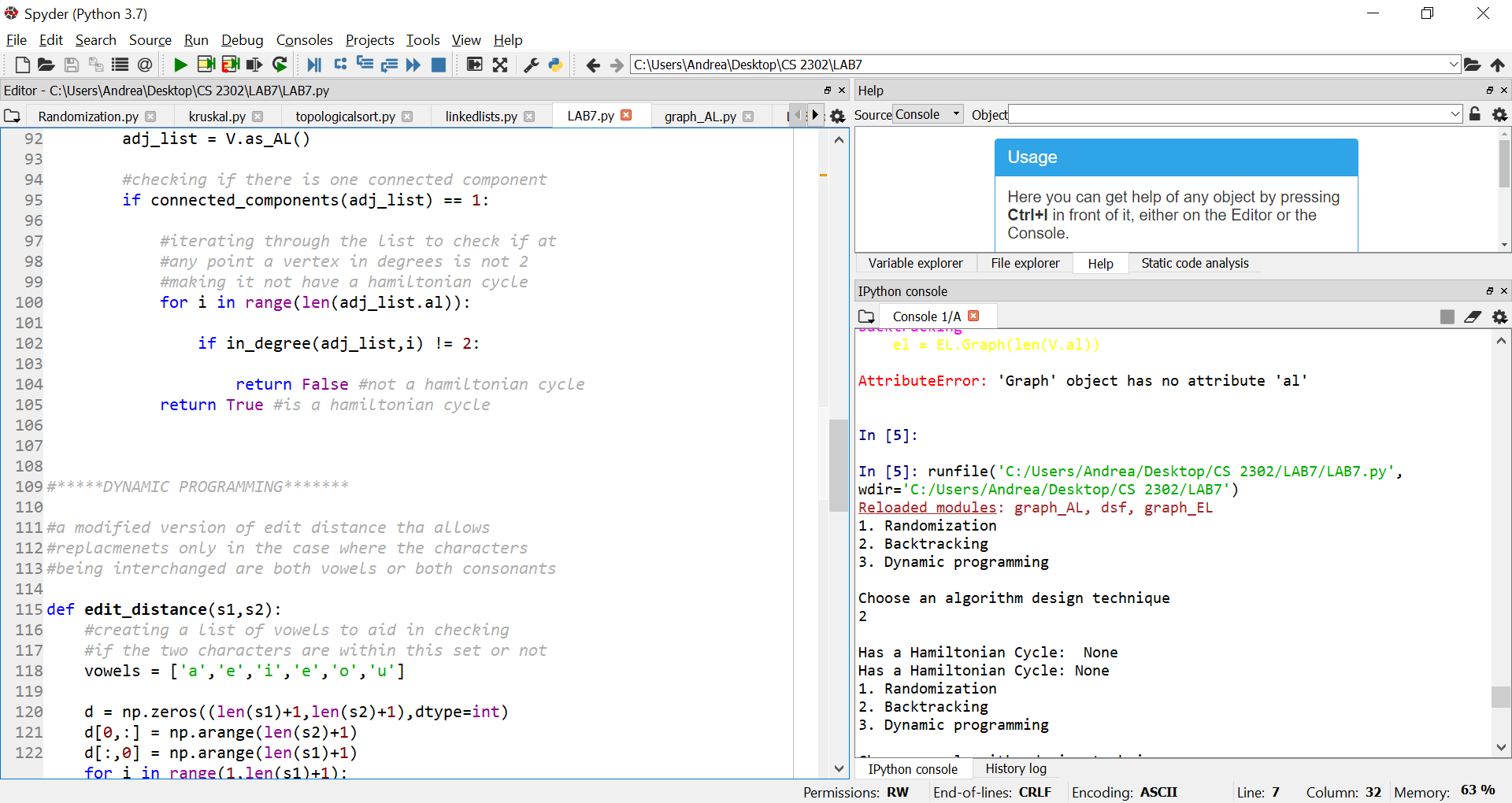
To conclude, randomization and backtracking are both efficient algorithms. However, randomization seemed to be more efficient than backtracking. Backtracking seems to take longer when given larger graphs. The edit distance function was also efficient with words that had smaller edit distances, however words that contained larger edit distances had a longer running time. In this lab I learned how to implement different types of algorithm design techniques to solve different problems. I learned how to use randomization by going through many trials to solve the Hamiltonian Cycle problem. I also learned how to use backtracking in a similar way however without the trials. Dynamic programming also became of better understanding through the modification of the edit distance function.

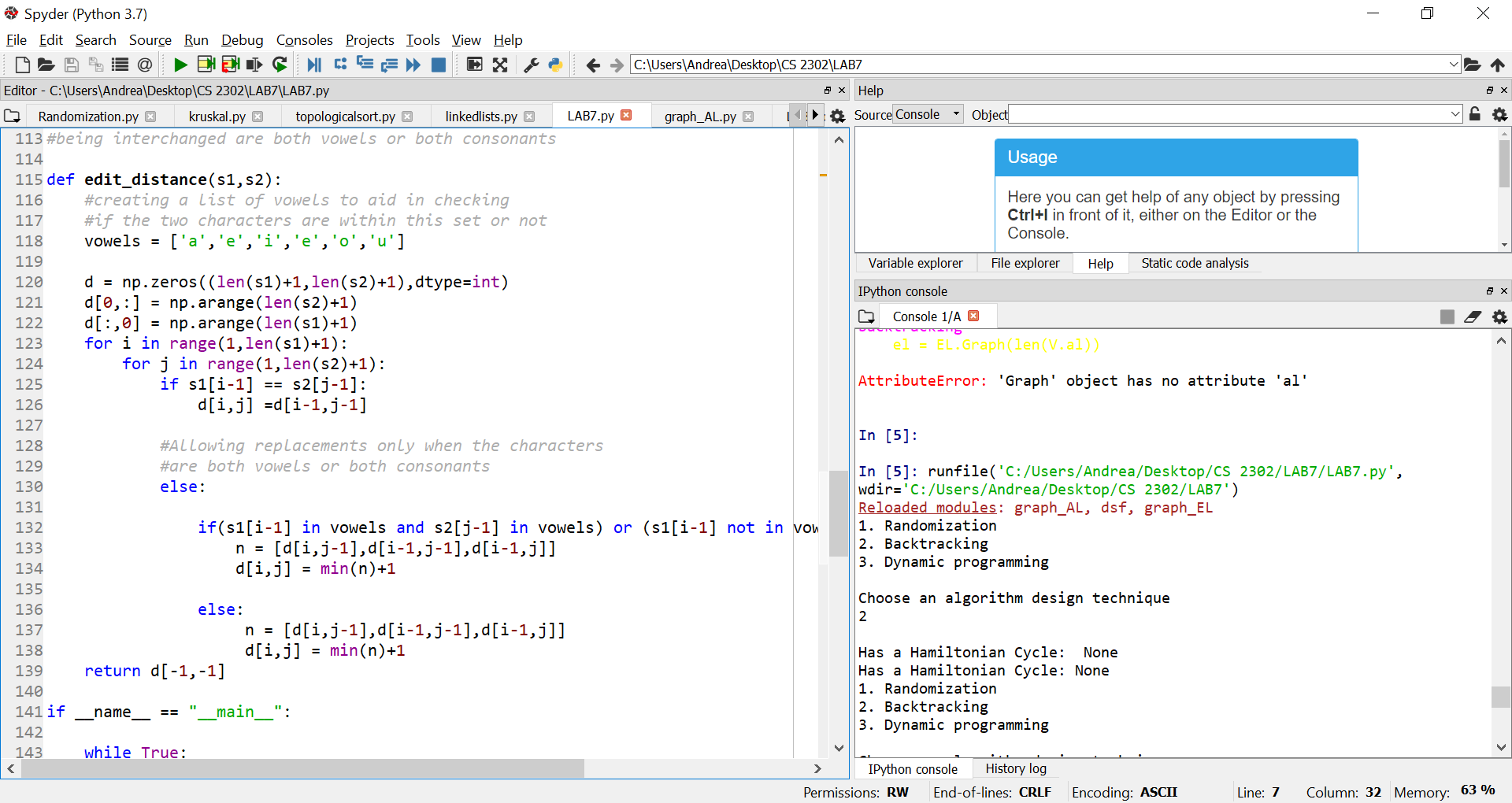
**Appendix**

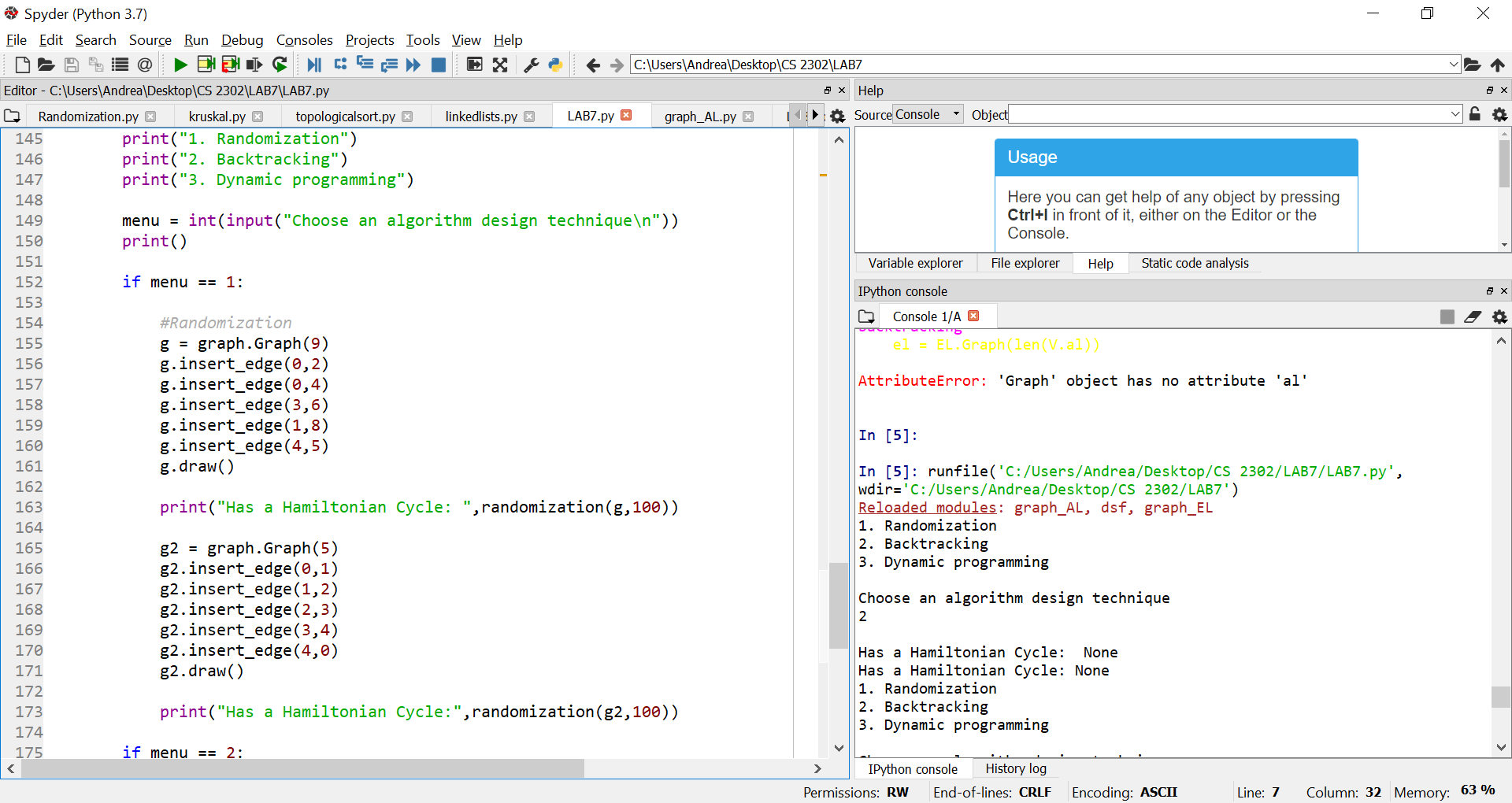


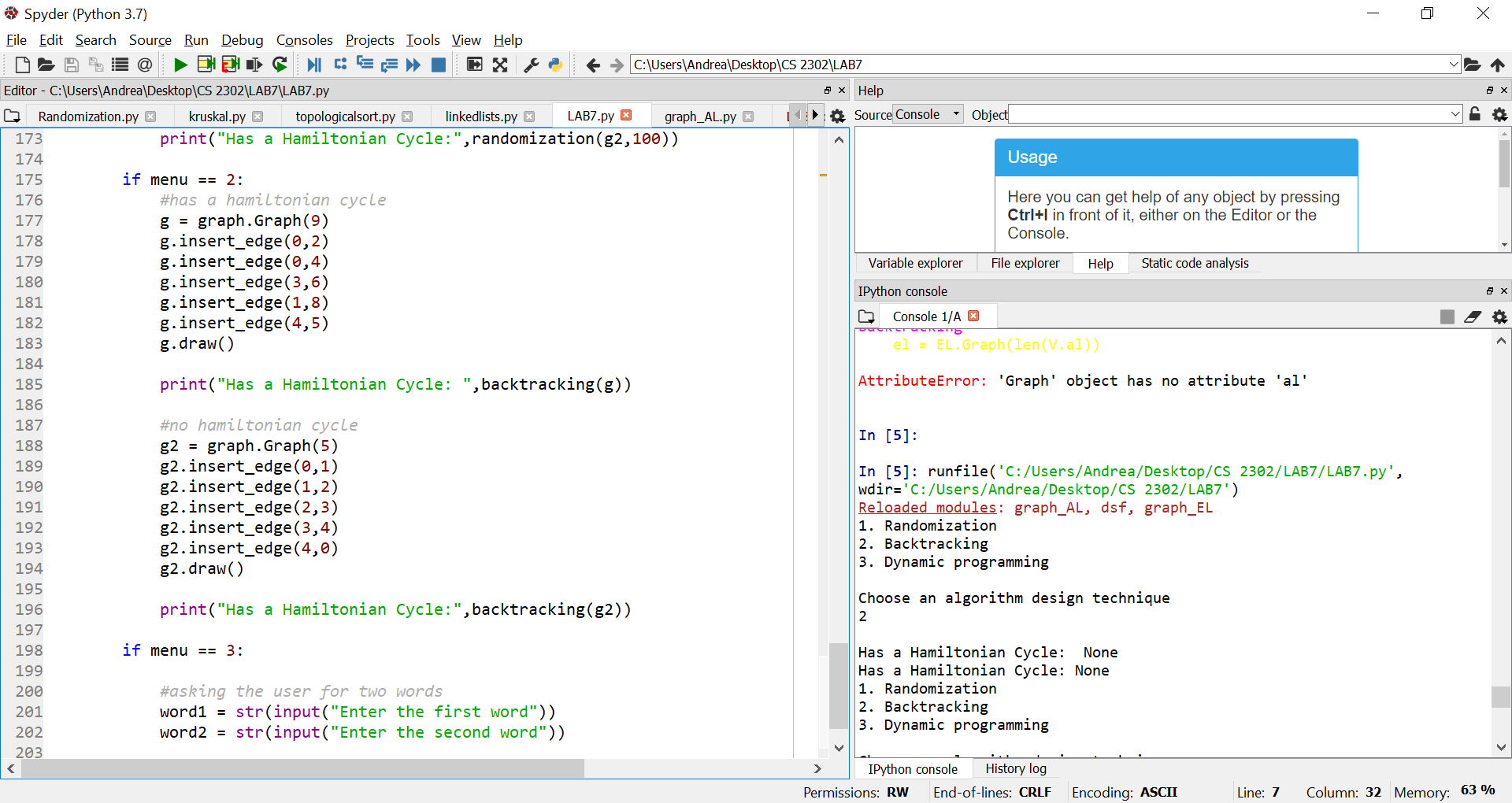












I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class