Questions

1) What is complexity in terms of algorithm design? Describe with examples.
2) Write a program to describe constant time, logarithmic time, linear time, quasilinear time, quadratic time, factorial time. You need to include the output of the program from the IDE.
3) Write a program to calculate summation of numbers
    a) One using linear time
    b) Another using constant time.
4) Describe Big O notation.

Answers to Questions 1 + 4 (2 + 3 included in separate python file, and their output is included below):

1. Complexity within the terms of algorithm design refers to two subsets of complexity: time complexity and space complexity. Time complexity refers to how much time it takes for an algorithm to accomplish its goal, and space complexity refers to how much space an algorithm requires to accomplish its goal. Since space complexity constraints have largely been eliminated thanks to exponential increases in space of hard drives, time complexity is a far larger issue in algorithm design, and typically what people are referring to when they discuss complexity of an algorithm. Time complexity (and space complexity as well, although this is less common) is often expressed using Big O notation, which is described further in the answer to question 4. Some examples of different time complexities would be linear time, or O(n), in which time complexity grows linearly with input, and O(n^2), in which the number of operations grows quadratically with the input size.
2. *Located in hw1.py*
3. *Located in hw1.py*
4. Big O notation is a method of describing the upper bound of the growth rate of the number of operations in terms of the input size. It is the most popular method of measuring the complexity of an algorithm. Some examples of different time complexities would be linear time, or O(n), in which time complexity grows linearly with input, and O(n^2), in which the number of operations grows quadratically with the input size. To elaborate further, O(n) time complexity implies that if you put the input size (x-axis) and the number of operations f(x) on a graph, f(x) would both be straight and increase at a constant rate. When graphing time complexities of O(n^2) and higher, however, f(x) will break away and trend towards infinity at higher and higher rates as input increases.

Code Outputs for Question 2 + 3:

2)

```
nayatrov@ariahas-MBP cs201hw1 % python3 hw1.py
1
6
15
[1, 2, 3, 4, 5, 6, 7, 8]
[1, 2, 3, 4, 5, 6, 7, 8]
120
```

3)

```
nayatrov@ariahas-MBP cs201hw1 % python3 hw1-pt2.py
Adding numbers from 1 to 5.
Constant Time:
15
Linear Time:
15
```