1. Asymptotic growth rate (Big-O) of the following functions and their order in the ascending order (fastest to slowest).

   a. 2^12 - O(1)
   b. 2^logn - O(n)
   c. 3n - O(n)
   d. 5n+50logn- O(n)
   e. 7nlogn+ 5n - O(nlogn)
   f. 15nlogn - O(nlogn)
   g. n^2+100n - O(n^2)
   h. n^3 - O(n^3)
   i. 2^n - O(2^n)

2. The function returns the smallest value. The initial value comes from assigning data[0]. Thus, it is constant complexity is O( 1). To check the smallest value, "for loop" is set up for each value in the range of length of data list, making it the linear complexity O(n). Inside the loop, the "if statement" checks whether a value is less than the other or not and it assigns the data into value accordingly which indicate constant complexity O(1). The return function also indicates constant complexity O(1). So, considering the overall function, the time complexity is O(n), where n is the length of data.

3. First of all, all the value assignments are constant complexity O(n). There are three nested loops. The outermost loop perform 'n' number of operations, making the complexity O(n). The middle loop performs 'n' number of operations for each iteration of outer loop, making the complexity O(n^2). The innermost loop performs n(n+1)/2 iterations for each outer iteration, making its complexity n*n*(n+1)/2= O(n^3).

   Thus, the overall time complexity of the given function is O(n^3).

4. In the given python code, the first "for loop" goes through n number of iterations. For inner loop, the number of iterations grow by 1 (as it goes from 'x+1' to '0' by -1) for each outer iteration. So, the function is equivalent to $n(n+1)$ /2. The proof by induction for base case and inductive case are as follows:

Proof by induction: 1 + 2 + 3 + 4 + 5 + … + n = Sum of i while (i=1 to n)= $n(n+1)$ /2

Base case: n = 1

1 = $n(n+1)/2$ = 1(1+1)/ 2 = 2/2 = 1

Inductive case: Let the function valid for n – 1 and prove valid for n.

Sum of i while (i=1 to n) = [Sum of i while (i=1 to n-1)] + $n$

By induction hypothesis: Sum of i while (i=1 to n-1)= [$(n-1)$ {$(n-1)+1$}]/2

= n(n-1)/2

Finally: Sum of i while (i=1 to n)= [Sum of i while (i=1 to n-1)] + $n$ = n(n-1)/2+ n

= $(n^2 - n + 2n)/2$ = $n(n+1)/2$