**Week 2: Algorithm Analysis**

1. **calculate\_execution\_time**:

* **Guide**:
  + The goal here is to measure the time it takes for a given function, func, to run with specific arguments, \*args. Python provides a module named time that can be useful in measuring time intervals. By noting the time before and after the function execution, you can determine its execution time.
* **Pseudocode**:
* FUNCTION calculate\_execution\_time(func, \*args):  
   RECORD the current time (start time)  
   EXECUTE the given function 'func' with arguments '\*args'  
   RECORD the current time again (end time)  
   CALCULATE the difference between the end and start times  
   RETURN the execution time

1. **bubble\_sort**:

* **Guide**:
  + Bubble Sort involves repeatedly stepping through the list, comparing each pair of adjacent items, and swapping them if they’re in the wrong order. The larger items ‘bubble’ to the end in each pass. Repeat this process until the list is sorted.
* **Pseudocode**:
* FUNCTION bubble\_sort(nums):  
   FOR i from 0 to length of nums:  
   FOR j from 0 to length of nums minus i minus 1:  
   IF nums[j] is greater than nums[j+1]:  
   SWAP nums[j] and nums[j+1]  
   RETURN nums

1. **insertion\_sort**:

* **Guide**:
  + In Insertion Sort, the list is divided into a sorted and an unsorted portion. On each iteration, one element from the unsorted portion is taken and ‘inserted’ into its correct position within the sorted portion.
* **Pseudocode**:
* FUNCTION insertion\_sort(nums):  
   FOR i from 1 to length of nums:  
   key = nums[i]  
   j = i-1  
   WHILE j >= 0 and key < nums[j]:  
   nums[j+1] = nums[j]  
   j = j-1  
   nums[j+1] = key  
   RETURN nums

1. **fibonacci\_recursive**:

* **Guide**:
  + The Fibonacci sequence starts with two numbers, 0 and 1. Every subsequent number is the sum of the previous two. The recursive nature of this sequence makes it a fit for a recursive function. Remember to handle the base cases: the 0th number is 0, and the 1st is 1.
* **Pseudocode**:
* FUNCTION fibonacci\_recursive(n):  
   IF n equals 0:  
   RETURN 0  
   ELSE IF n equals 1:  
   RETURN 1  
   ELSE:  
   RETURN fibonacci\_recursive(n-1) + fibonacci\_recursive(n-2)