```
int[] numbers
int length

void sort(int[] values)
    // check for empty or null array
    if (values == null/None OR length of values == 0)
            return   // nothing to sort
    numbers = values;
    length = length of values
    quicksort(0, length - 1)   // beginning index and end index

void quicksort(int low, int high)
    i = low, j = high
    // Get the pivot element from the middle of the list
    pivot = numbers[low + (high-low)/2]  // integer division

    // Divide into two lists
    while (i <= j)   // as long as we still have elements to process

         // keep looping from left until value greater or equal to pivot
         while (numbers[i] < pivot)
             increment i by 1

         // keep looping from right until value less than or equal to pivot
         while (numbers[j] > pivot)
             decrement j by 1

         // check if our left list index is less than right list index
         // if so that means they need to swap because
         // we want left list to be smaller than pivot and right list greater
         if (i <= j)
             exchange(i, j)
             increment i by 1
             decrement j by 1

        // Recursion
        //repeat starting with new values for low and high for each sublist
        if (low < j)  // still more to sort on left
            quicksort(low, j)

        if (i < high) // still more to sort on right
            quicksort(i, high);

// switch values in index i and j
void exchange(int i, int j)
    temp = numbers[i]
    numbers[i] = numbers[j]
    numbers[j] = temp
```