Explanation:

insertion_sort function: Applies Insertion Sort to a subarray within a specified range, which is efficient for small or nearly sorted subarrays.

merge function: Merges two sorted halves of an array, similar to standard Merge Sort.

hybrid_sort function: Recursively divides the array and switches to Insertion Sort when the size of the subarray is below a specified threshold.

sort function: A helper function to call hybrid_sort on the entire array.

Code:

```
def insertion_sort(arr, left, right):
    """Sorts arr[left:right+1] using Insertion Sort."""
    for i in range(left + 1, right + 1):
        key = arr[i]
        j = i - 1
        while j >= left and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key


def merge(arr, left, mid, right):
    """Merges two sorted subarrays of arr."""
    n1 = mid - left + 1
    n2 = right - mid



    L = arr[left:mid + 1]
    R = arr[mid + 1:right + 1]


    i = j = 0
    k = left



    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
```

```
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1


    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1


    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1

def hybrid_sort(arr, left, right, threshold=32):
    """Performs Hybrid Sort by combining Merge Sort and Insertion Sort."""
    if left < right:

        if right - left + 1 <= threshold:
            insertion_sort(arr, left, right)
        else:

            mid = (left + right) // 2
            hybrid_sort(arr, left, mid, threshold)
            hybrid_sort(arr, mid + 1, right, threshold)
            merge(arr, left, mid, right)


def sort(arr):
    hybrid_sort(arr, 0, len(arr) - 1)
```

```
arr = [12, 11, 13, 5, 6, 7, 3, 15, 10, 8, 1, 4, 9, 2, 14]
print("Original array:", arr)
sort(arr)
print("Sorted array:", arr)
```