**堆排序：**

程序：

def heapSort(arr):

    def sift\_down(start, end):

        """最大堆调整"""

        root = start

        while True:

            child = 2 \* root + 1

            if child > end:

                break

            if child + 1 <= end and arr[child] < arr[child + 1]:

                child += 1

            if arr[root] < arr[child]:

                arr[root], arr[child] = arr[child], arr[root]

                root = child

            else:

                break

    # 创建最大堆

    for start in range((len(arr) - 2) // 2, -1, -1):

        sift\_down(start, len(arr) - 1)

    # 堆排序

    for end in range(len(arr) - 1, 0, -1):

        arr[0], arr[end] = arr[end], arr[0]

        sift\_down(0, end - 1)

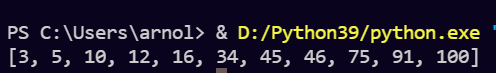
    return arr

if \_\_name\_\_ == '\_\_main\_\_':

    arr = [100, 16, 45, 46, 34, 75, 12, 3, 91, 10, 5]

    print(heapSort(arr))

运行结果：



**基数排序：**

程序：

def radixSort(data):

    if not data:

        return []

    max\_num = max(data)  # 获取当前数列中最大值

    max\_digit = len(str(abs(max\_num)))  # 获取最大的位数

    dev = 1  # 第几位数，个位数为1，十位数为10···

    mod = 10  # 求余数的除法

    for i in range(max\_digit):

        radix\_queue = [list() for k in range(mod \* 2)]

        for j in range(len(data)):

            radix = int(((data[j] % mod) / dev) + mod)

            radix\_queue[radix].append(data[j])

        pos = 0

        for queue in radix\_queue:

            for val in queue:

                data[pos] = val

                pos += 1

        dev \*= 10

        mod \*= 10

    return data

if \_\_name\_\_ == '\_\_main\_\_':

    nums = [334, 5, 67, 345, 7, 99, 4, 23, 78, 45, 1, 3453, 23424]

    print(radixSort(nums))

运行结果：

