

Laboratory work #6

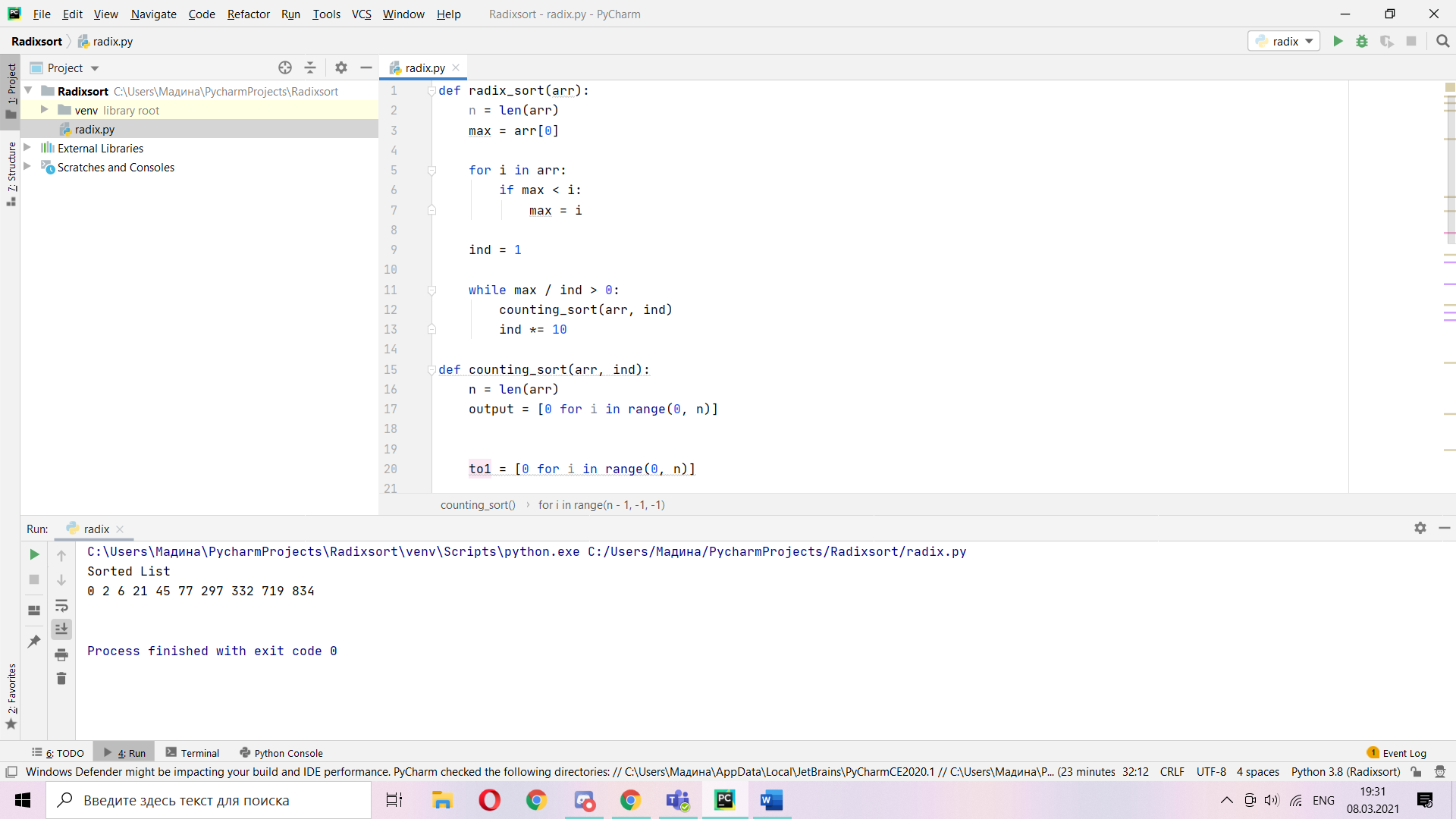
Performance, Data Structure & Algorithms

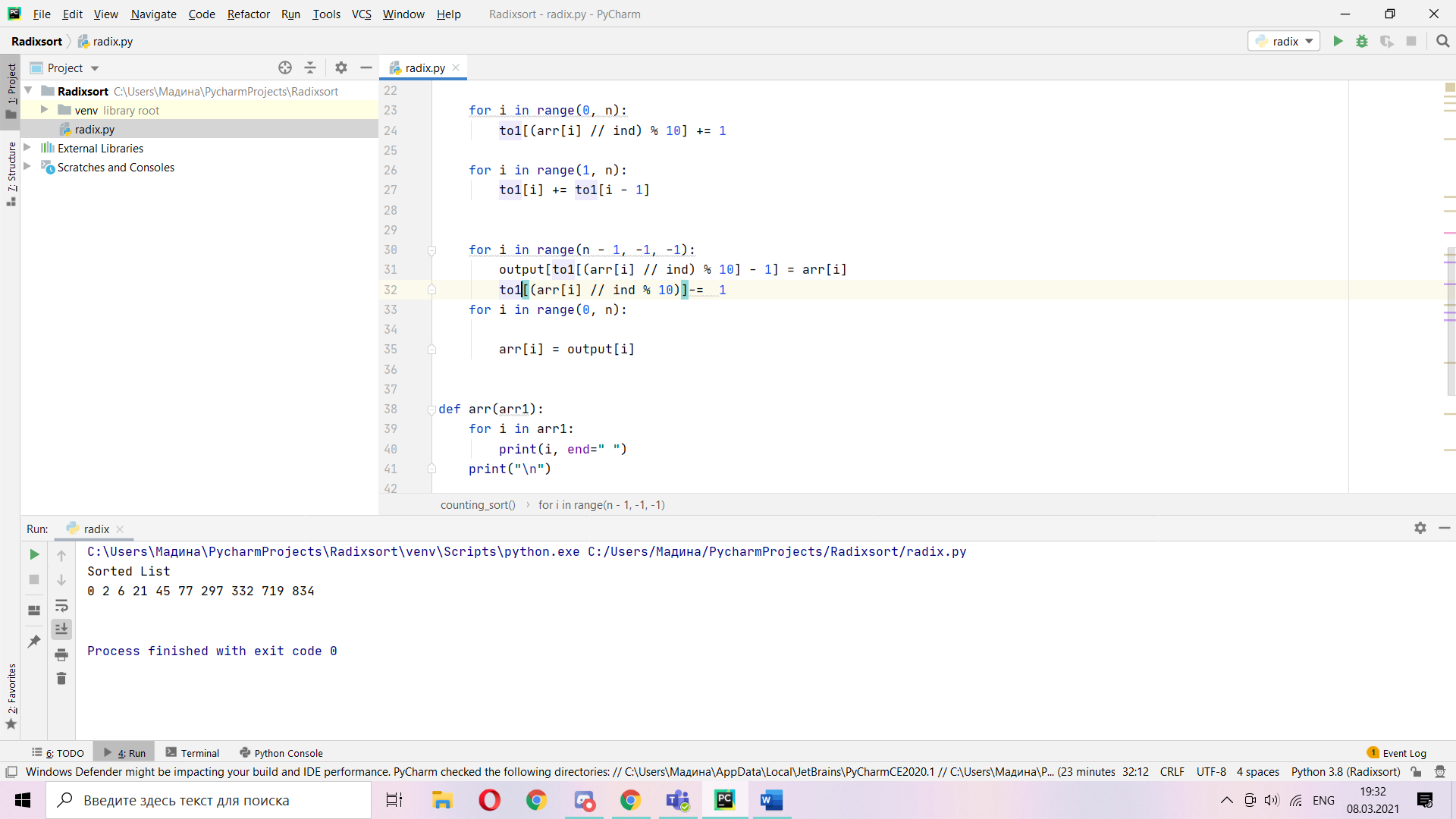
“Radix sort”

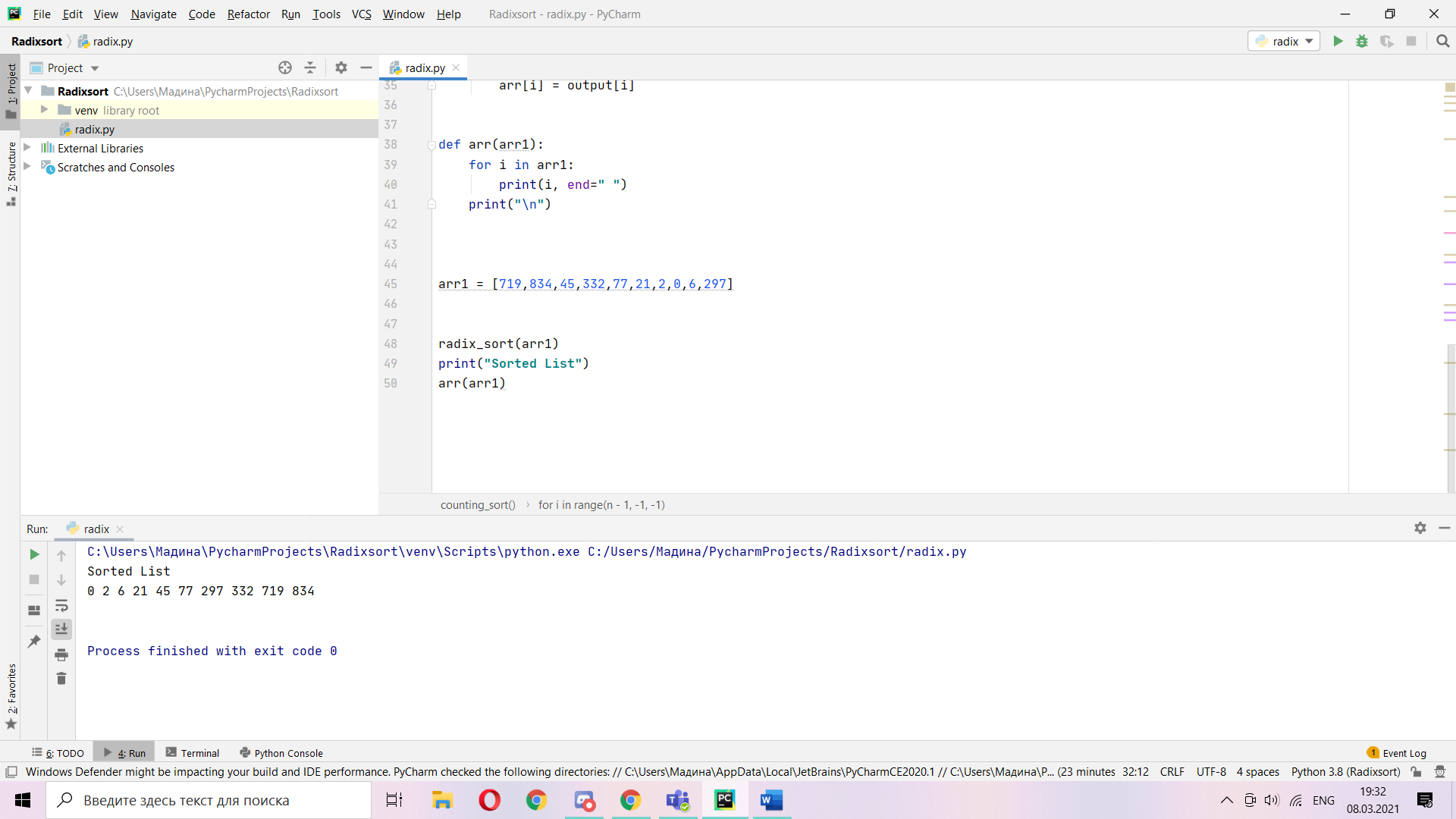
Group:ITDS-1901

Ibrayeva Madina

1.







Which of the following sorting algorithms are stable: insertion sort, merge sort,

**def radix\_sort(arr):**

**n = len(arr)**

**max=arr[0]**

**for i in arr:**

**if max<i:**

**max = i**

**ind =1**

**while max/ind >0:**

**counting\_sort(arr,ind):**

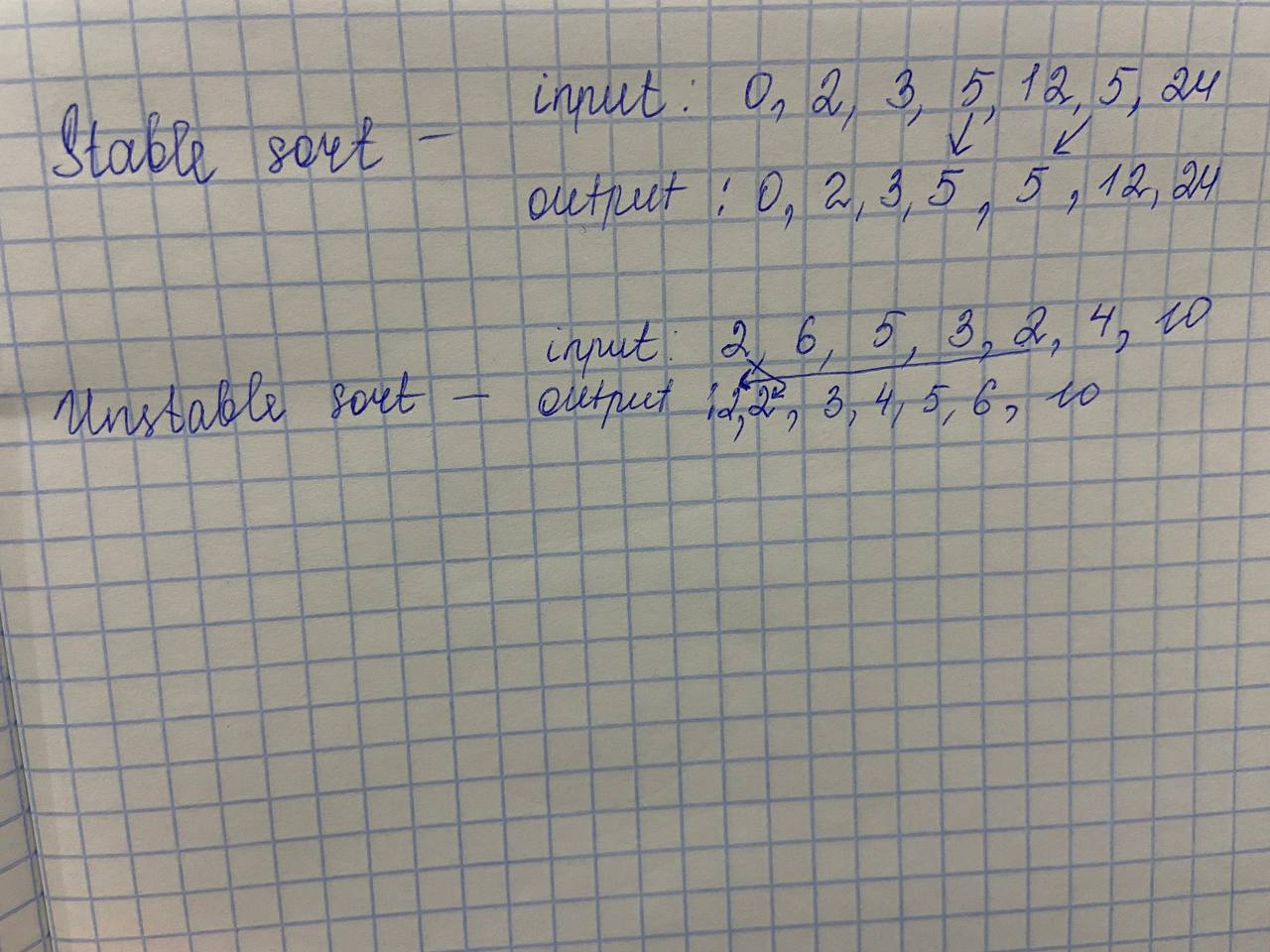
**ind\*=10**

**def counting\_sort…**

heapsort, and quicksort? Give a simple scheme that makes any sorting algorithm stable. How much additional time and space does your scheme entail?

What makes sort stable?

When we sort some list of elements, stable sort don’t replace equal elements and in output they place like in input. In unstable sort there is new for operation, so there is extra space and sum additional time to this.



**3.** 