

Laboratory work #4

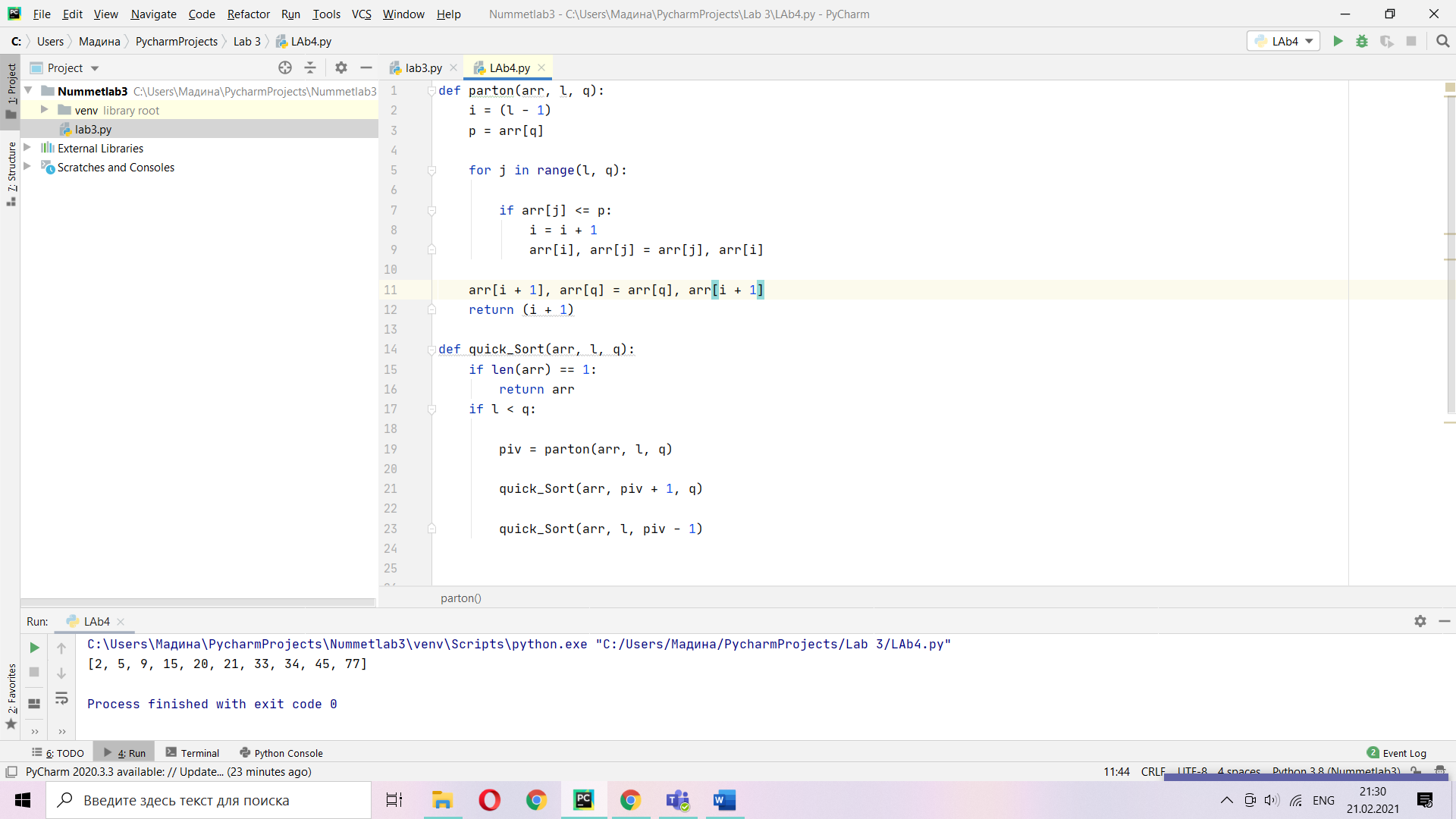
Performance, Data Structure & Algorithms

“Quick Sort”

Group:ITDS-1901

Ibrayeva Madina

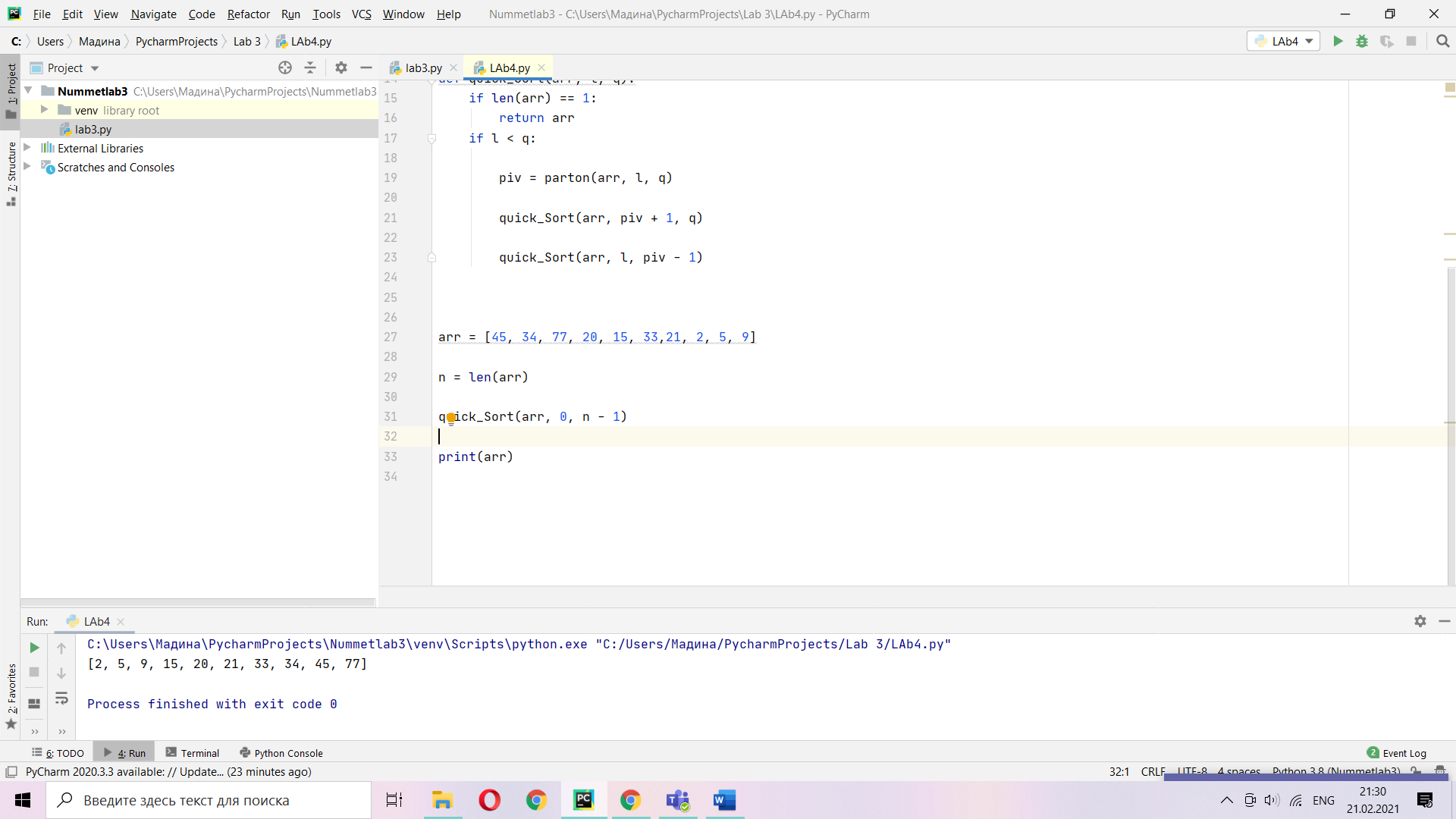
Implement Quick sort using any programming languages.



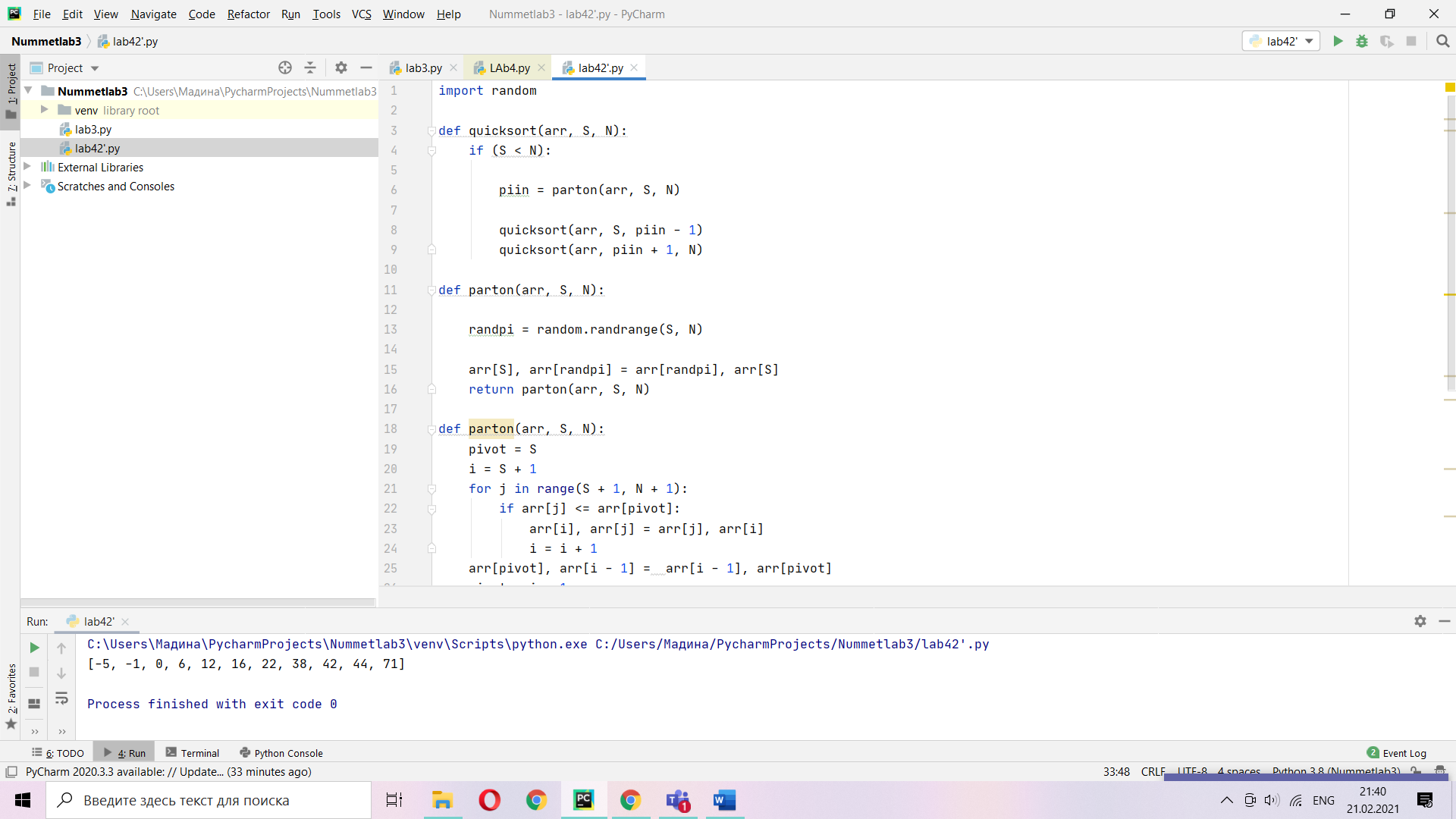
def parton(A, l, q):  
 i = (l - 1)  
 p = A[q]

for j in range(l, q):  
 if A[j] <= p:  
 i = i + 1  
 A[i], A[j] = A[j], A[i]  
  
A[i + 1], A[q] = A[q], A[i + 1]  
return (i + 1)

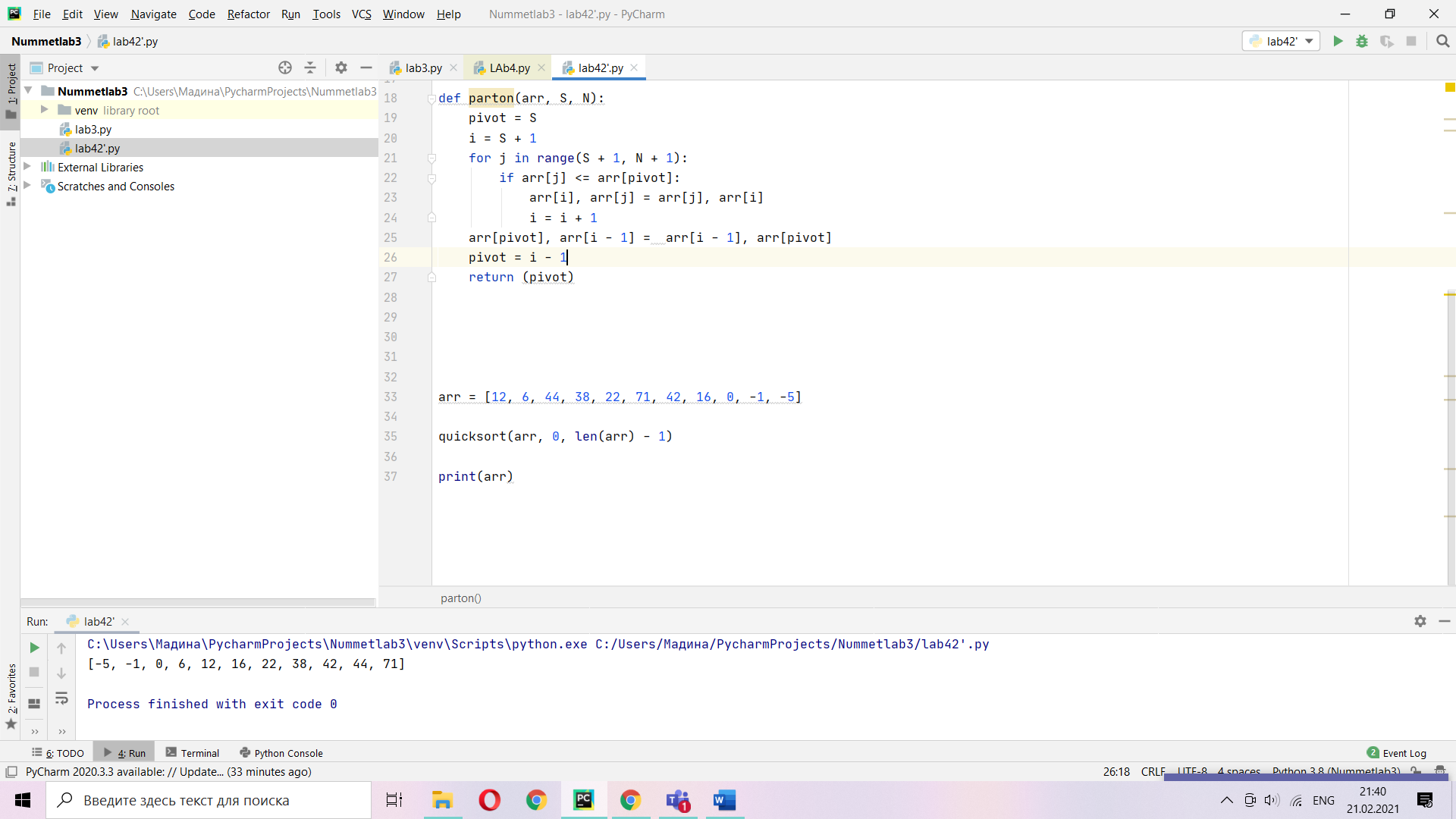
def quick\_Sort(A, l, q):  
 if len(A) == 1:  
 return A  
 if l < q:  
  
 piv = parton(A, l, q)  
  
 quick\_Sort(A, piv + 1, q)  
  
 quick\_Sort(A, l, piv - 1)



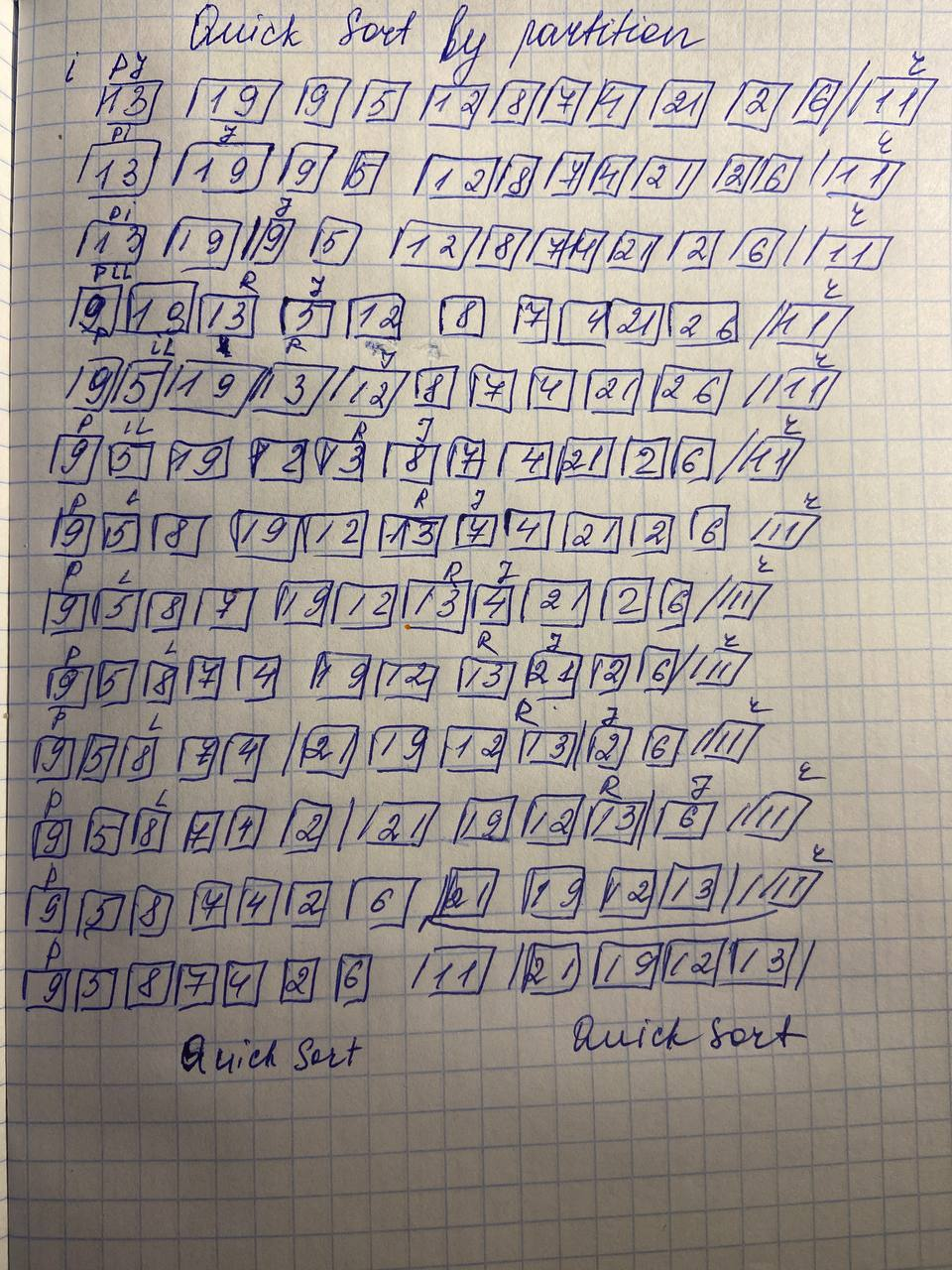
1. Implement Randomized version of Quick sort.



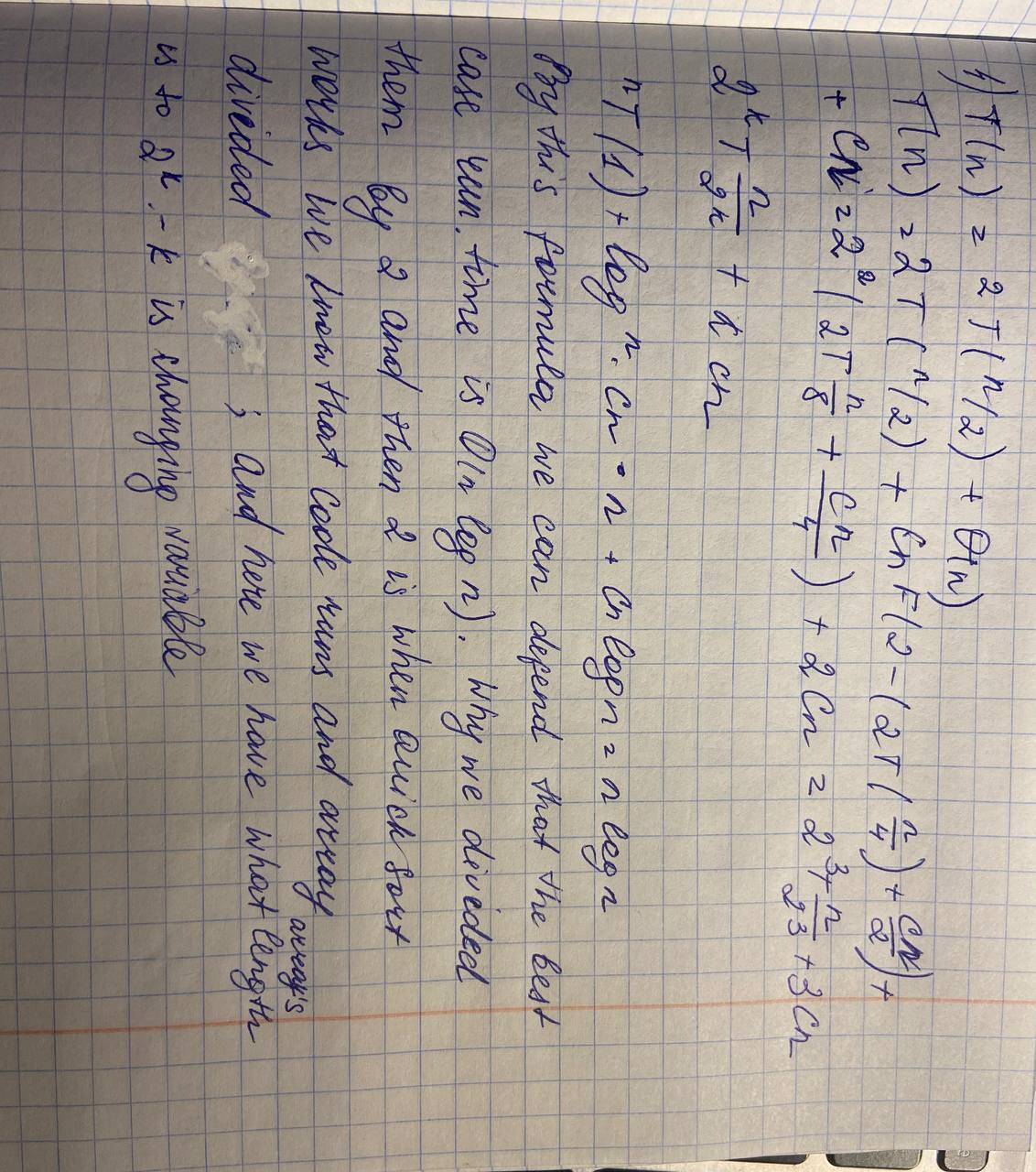
import random  
  
def quicksort(arr, S, N):  
 if (S < N):  
  
 piin = parton(arr, S, N)  
  
quicksort(arr, S, piin - 1)  
quicksort(arr, piin + 1, N)  
  
def parton(arr, S, N):  
randpi = random.randrange(S, N)  
  
arr[S], arr[randpi] = arr[randpi], arr[S]  
 return parton(arr, S, N)  
  
def parton(arr, S, N):  
 pivot = S  
 i = S + 1  
 for j in range(S + 1, N + 1):  
 if arr[j] <= arr[pivot]:  
 arr[i], arr[j] = arr[j], arr[i]  
 i = i + 1  
 arr[pivot], arr[i - 1] = arr[i - 1], arr[pivot]  
 pivot = i - 1  
 return (pivot)



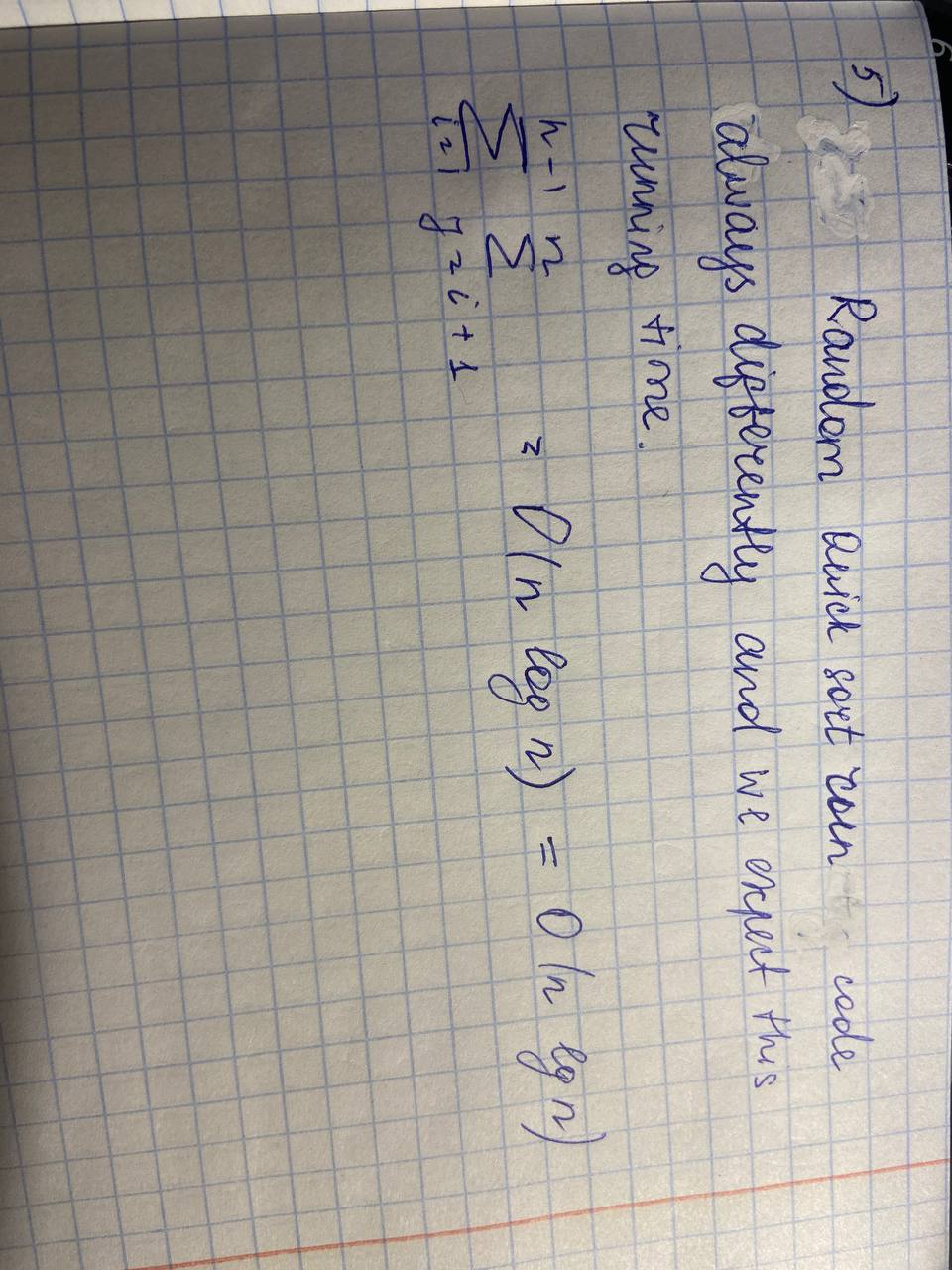
Using lecture as a model, illustrate the operation of PARTITION on the array A (13; 19; 9; 5; 12; 8; 7; 4; 21; 2; 6; 11).



1. Show that quicksort’s best-case running time is O(n lg n).

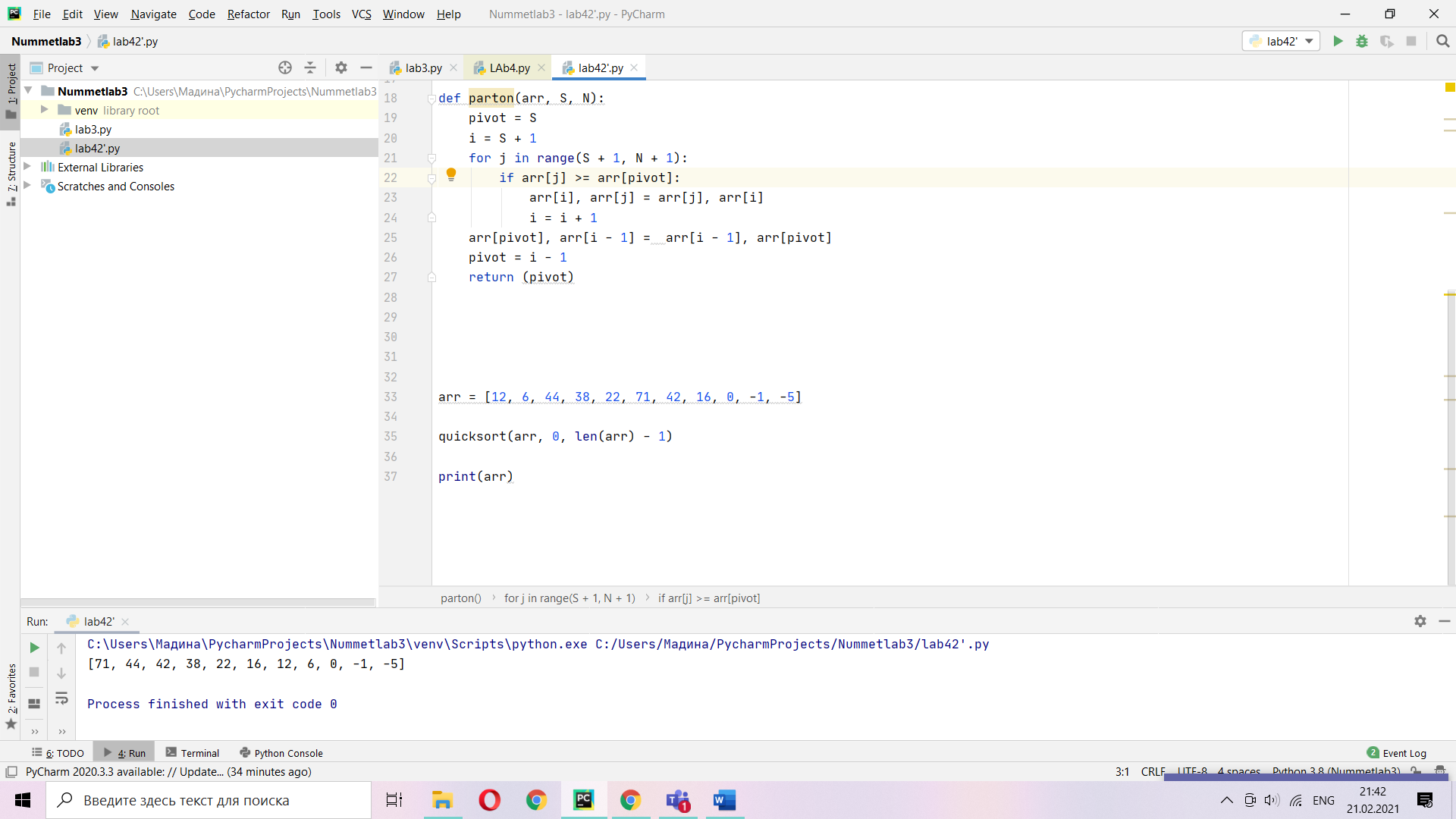


1. Show that RANDOMIZED-QUICKSORT’s expected running time is O(n lg n). Why do we analyze the expected running time of a randomized algorithm and not its worst-case running time?



1. Modify QUICKSORT to sort into nonincreasing order.





import random  
  
def quicksort(arr, S, N):  
 if (S < N):  
  
 piin = parton(arr, S, N)  
  
quicksort(arr, S, piin - 1)  
quicksort(arr, piin + 1, N)  
  
def parton(arr, S, N):  
randpi = random.randrange(S, N)  
  
arr[S], arr[randpi] = arr[randpi], arr[S]  
 return parton(arr, S, N)  
  
def parton(arr, S, N):  
 pivot = S  
 i = S + 1  
 for j in range(S + 1, N + 1):  
 if arr[j] >= arr[pivot]:  
 arr[i], arr[j] = arr[j], arr[i]  
 i = i + 1  
 arr[pivot], arr[i - 1] = arr[i - 1], arr[pivot]  
 pivot = i - 1  
 return (pivot)