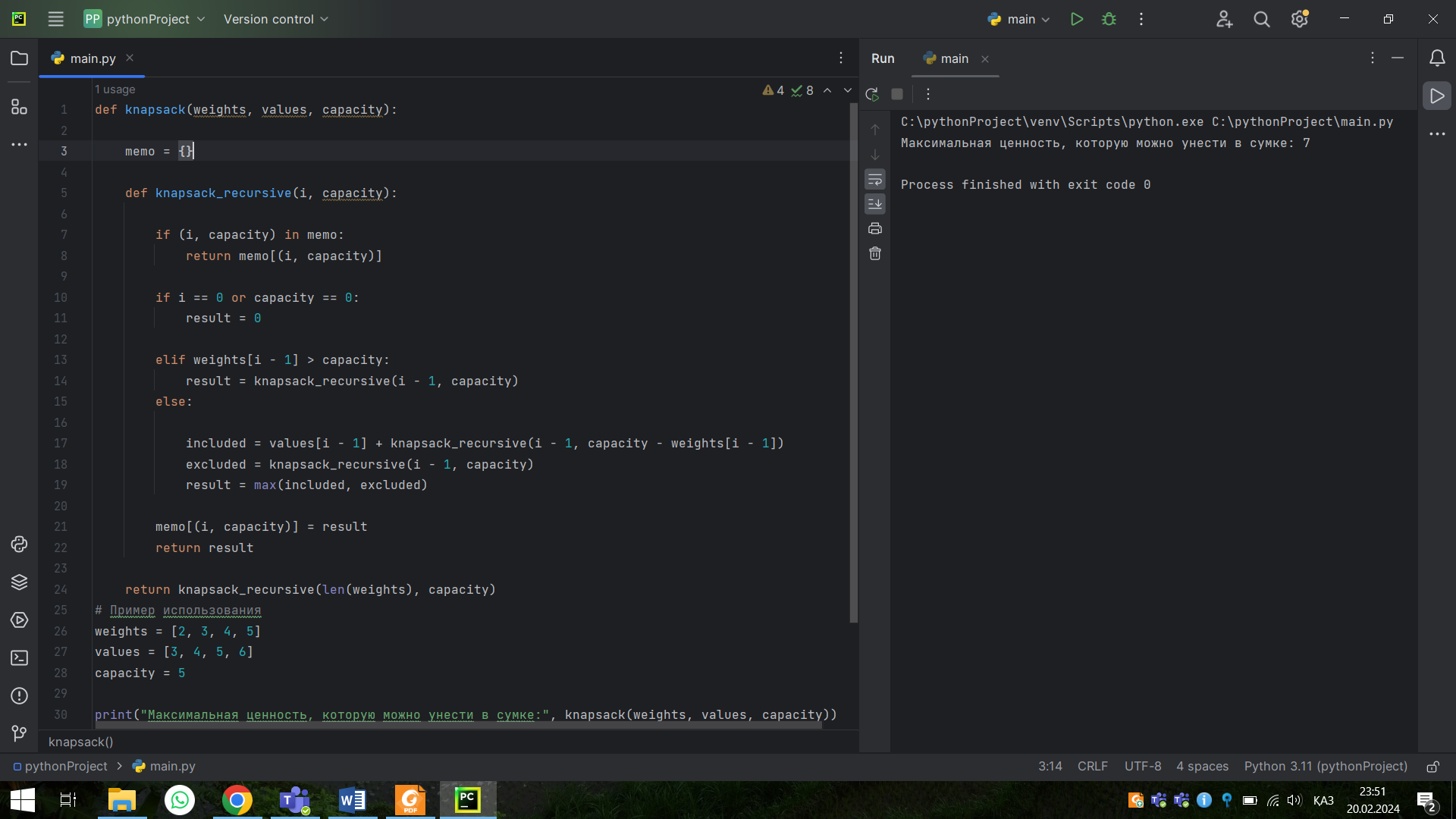
2 ВАРИАНТ

Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO  
қараңыз). Студенттерге рекурсияда қосымша дағдыларды дамытуға және  
функционалдық заңдылықтарды түсінуге көмектесетін есептер:

Сөмке мәселесін шешуге арналған рекурсиямен динамикалық бағдарламалау  
- Сөмке мәселесін шешу үшін есте сақтау арқылы рекурсияны пайдаланыңыз.



def knapsack(weights, values, capacity):  
  
 memo = {}  
  
 def knapsack\_recursive(i, capacity):  
  
 if (i, capacity) in memo:  
 return memo[(i, capacity)]  
  
 if i == 0 or capacity == 0:  
 result = 0  
  
 elif weights[i - 1] > capacity:  
 result = knapsack\_recursive(i - 1, capacity)  
 else:  
 included = values[i - 1] + knapsack\_recursive(i - 1, capacity - weights[i - 1])  
 excluded = knapsack\_recursive(i - 1, capacity)  
 result = max(included, excluded)  
  
 memo[(i, capacity)] = result  
 return result  
  
 return knapsack\_recursive(len(weights), capacity)  
# Пример использования  
weights = [2, 3, 4, 5]  
values = [3, 4, 5, 6]  
capacity = 5  
  
print("Максимальная ценность, которую можно унести в сумке:", knapsack(weights, values, capacity))