|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Язык программирования Python»  Вариант 7 | | |
|  | | Работу выполнил  студент группы ИТ-6 1 курса  Соромотина Виктория М.  «12» 06\_\_\_2025 г. |
| Работу проверил  Рубцова Марина Борисовна.  « » 06\_\_ 2025 г. |
| Пермь 2025 | | |

СОДЕРЖАНИЕ

[Постановка задачи 3](#_Toc153130027)

[Алгоритм решения 4](#_Toc153130028)

[Тестирование 6](#_Toc153130029)

[Код программы 8](#_Toc153130030)

# Постановка задачи

Игра в «пьяницу». В игре в «пьяницу» карточная колода раздаётся поровну

двум игрокам. Далее они вскрывают по одной верхней карте, и тот, чья карта

старше, забирает себе обе вскрытые карты, которые кладутся под низ его

колоды. Тот, кто остаётся без карт – проигрывает. Для простоты будем

считать, что все карты различны по номиналу, а также, что самая младшая

карта побеждает самую старшую карту ("шестерка берет туза"). Игрок,

который забирает себе карты, сначала кладёт под низ своей колоды карту

первого игрока, затем карту второго игрока (то есть карта второго игрока

оказывается внизу колоды). Напишите программу, которая моделирует игру в

пьяницу и определяет, кто выигрывает.

В игре участвует 10 карт, имеющих значения от 0 до 9, большая карта

побеждает меньшую, карта со значением 0 побеждает карту 9.

Входные данные Программа получает на вход две строки: первая строка

содержит 5 чисел, разделённых пробелами—номера карт первого игрока,

вторая – аналогично 5 карт второго игрока. Карты перечислены сверху вниз,

то есть каждая строка начинается с той карты, которая будет открыта первой.

Выходные данные. Программа должна определить, кто выигрывает при

данной раздаче, и вывести слово first или second, после чего вывести

количество ходов, сделанных до выигрыша. Если на протяжении ходов

игра не заканчивается, программа должна вывести слово botva.

Примеры входные данные 1 3 5 7 9 2 4 6 8 0 выходные данные second 5.

# Алгоритм решения

1. Структура данных: двустороняя очередь на основе массива.
2. Пользователь вводит номера 5 карт сначала для 1 игрока, затем для 2 игрока.
3. 2 ввода преобразуем в 2 списка строк.
4. Запускаются 2 функции (Proverka1,Proverka2) для проверки ввода (2 списков):

* Проверка символов в списке строк, которые должны включать в себя только цифры, для этого создан вспомогательный список spis\_numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'], а также проверка длины:

i = 4

spis1 = проверяемый список строк [‘1’, ‘1’, ‘1’, ‘1’, ‘1’]

Пока 0 <= i <= 4:

Если (длина списка не равна 5) ИЛИ spis1[i] не входит в spis\_numbers*:*

*spis1 =* Ввод()

i = 5 # снова для проверки ввода

i = i - 1

* 2 список проверяется аналогично.

1. Преобразовываем 2 списка строк в список целых чисел.

p1\_cards = list(map(int, p1\_input))  
p2\_cards = list(map(int, p2\_input))

1. Запускаем игру при помощи метода play класса Game:

* Создаем одновременно объекты класса Game и объекты класса ArrayDeque,в качестве аргументов передаём списки целых чисел.

class Game:  
 """Класс, представляющий игру"""  
 def \_\_init\_\_(self, p1\_cards, p2\_cards):  
 self.player1 = ArrayDeque(p1\_cards)  
 self.player2 = ArrayDeque(p2\_cards)

class ArrayDeque:  
 def \_\_init\_\_(self, card):  
 self.data = card

* Метод play. Количество итераций цикла for = максимальному количеству раундов, по окончанию цикла выводим слово botva по условию задачи, если никто не выиграл.

max\_rounds = 10\*\*6  
for round\_num in range(1, max\_rounds + 1):

Тело цикла

return "botva"

* Метод play. В теле цикла стоит два условия: если у первого игрока колода пуста второй выигрывает, если у второго пуста, то первый выигрывает. Проверка пустоты с помощью метода has\_cards класса ArrayDeque*,* возвращаетtrue или false, используется bool() — логический тип данных.

if not self.player1.has\_cards(): # если колода 1 игрока пуста  
 return f"second {round\_num - 1}" # объявляем 2 игрока победителем   
if not self.player2.has\_cards(): # аналогично со вторым игроком  
 return f"first {round\_num - 1}"

class ArrayDeque:  
 def has\_cards(self):  
 """метод has\_cards определяет есть ли у игрока карты в колоде """  
 return bool(self.data)

* Метод play. Игрок вытягивает карту. С помощью метода draw\_card класса ArrayDeque ( pop[0] - из начала очереди удаляем и возвращаем «карту»).

card1 = self.player1.draw\_card() # 1 игрок вытягивает карту  
card2 = self.player2.draw\_card() # 2 игрок вытягивает карту

class ArrayDeque:

def draw\_card(self):  
 return self.data.pop(0)

* Метод play. Карты игроков сравниваются c помощью метода card1\_wins класса Game, если у первого больше карта, он забирает 2 карты себе, но карта второго не равна 0, и карта первого не равна 9, иначе забирает карты второй. Карты добавляем в колоду победителю с помощью метода *add\_cards* класса ArrayDeque(добавляем в конец очереди сначала 1 карту, затем карту 2 игрока).

if self.card1\_wins(card1, card2): # проверка на выигрыш 1 игрока  
 self.player1.add\_cards(card1)  
 self.player1.add\_cards(card2) # добавляем 2 карты в колоду 1 игроку  
else:  
 self.player2.add\_cards(card1) # иначе добавляем 2 карты 2 игроку  
 self.player2.add\_cards(card2)

card1\_wins(): # метод класса Game, сравнение карт

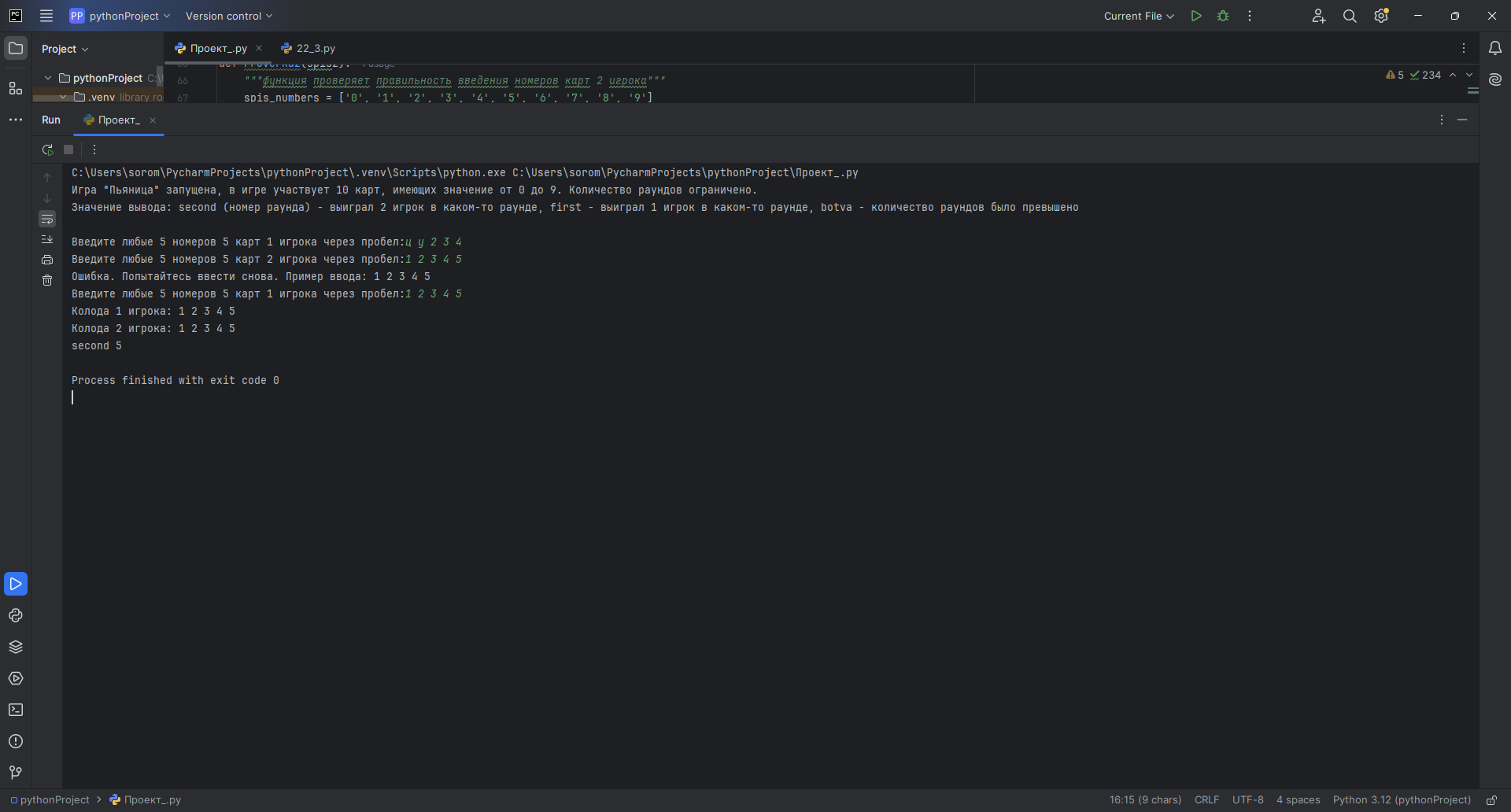
return (c1 == 0 and c2 == 9) or (c1 > c2 and not (c1 == 9 and c2 == 0))

class ArrayDeque:

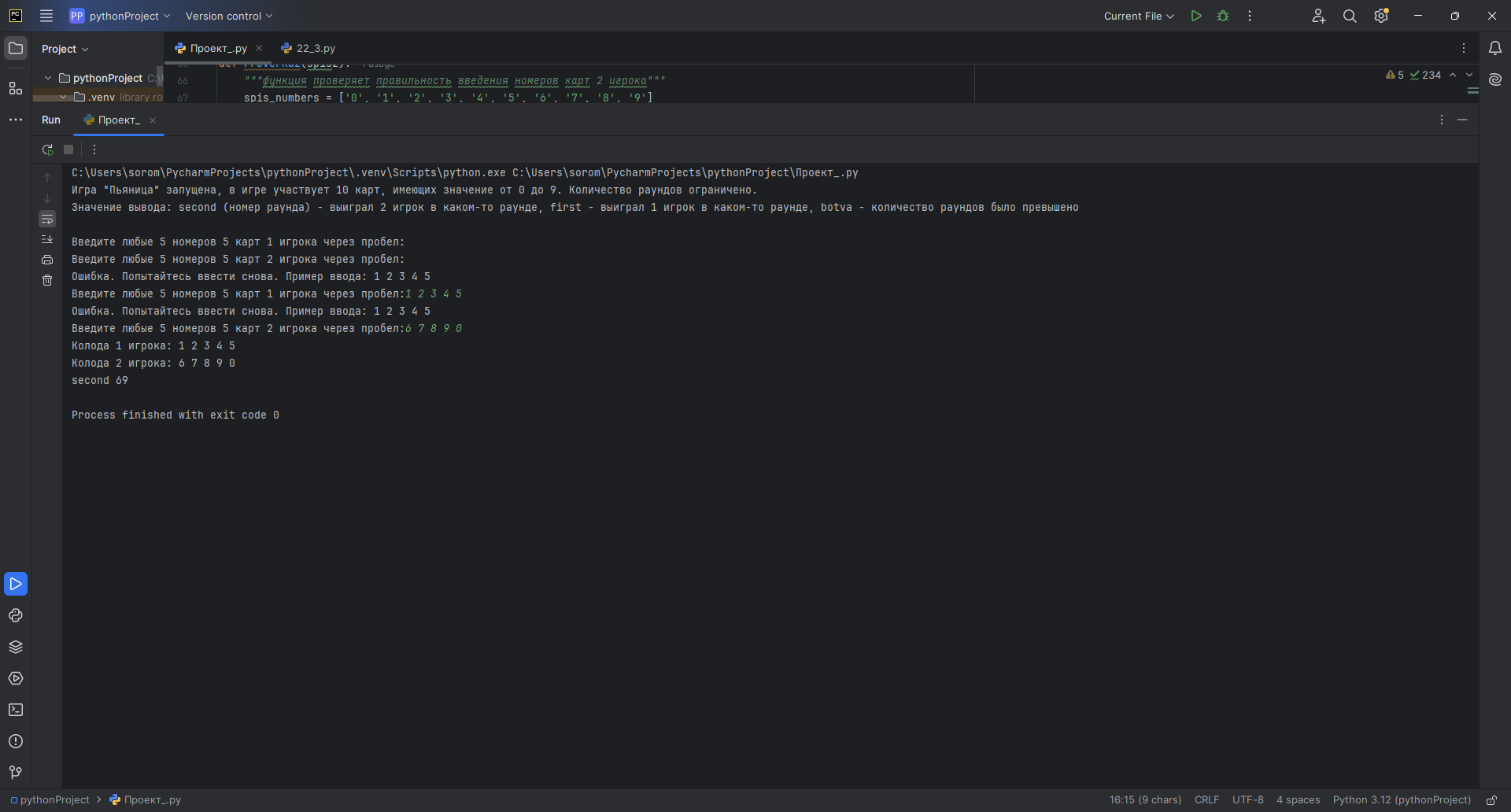
def add\_cards(): # метод для добавления 2 карт победителю  
 self.data.append(value)

# Тестирование

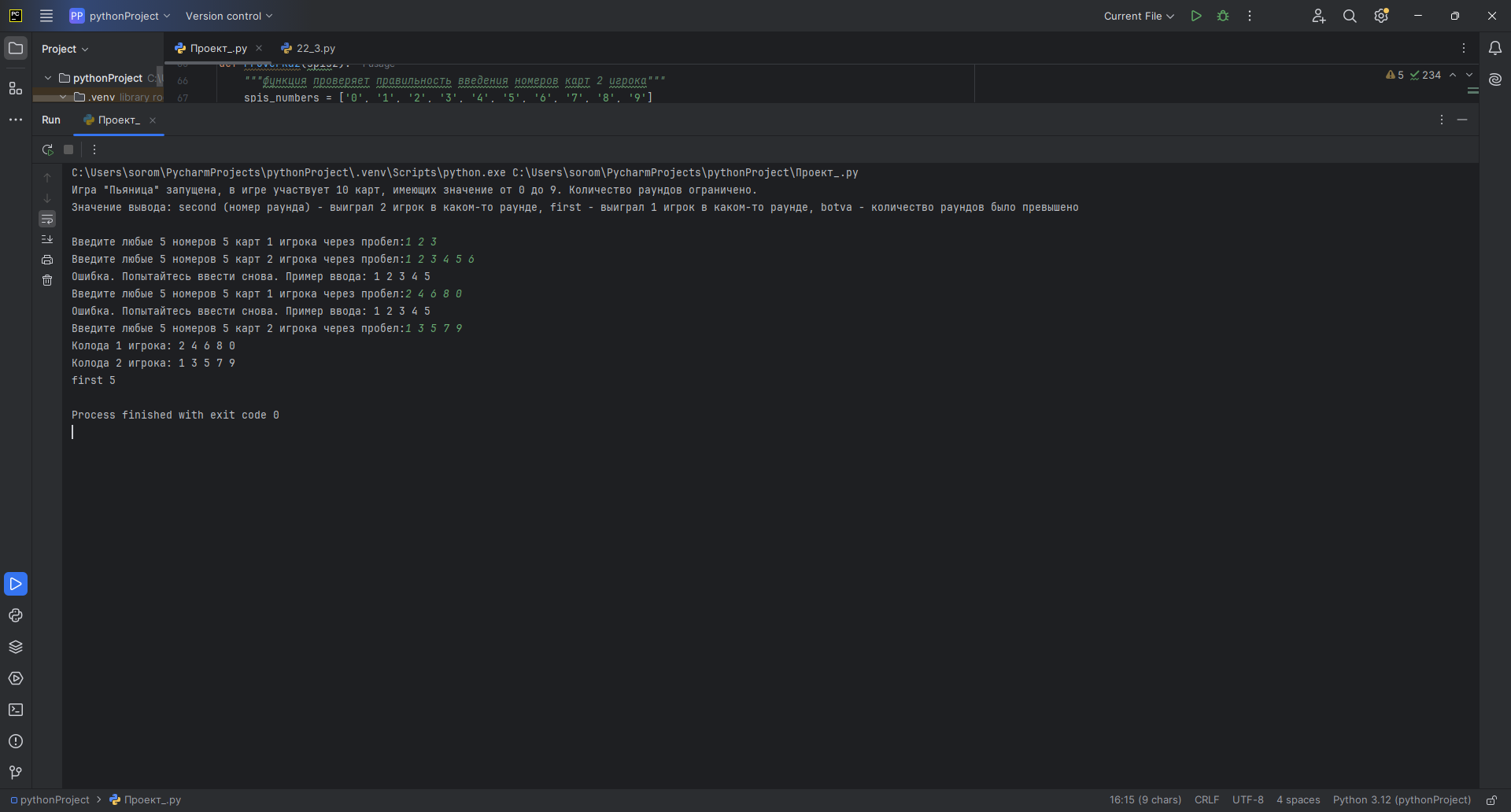
Ошибка в первом вводе: неправильные символы



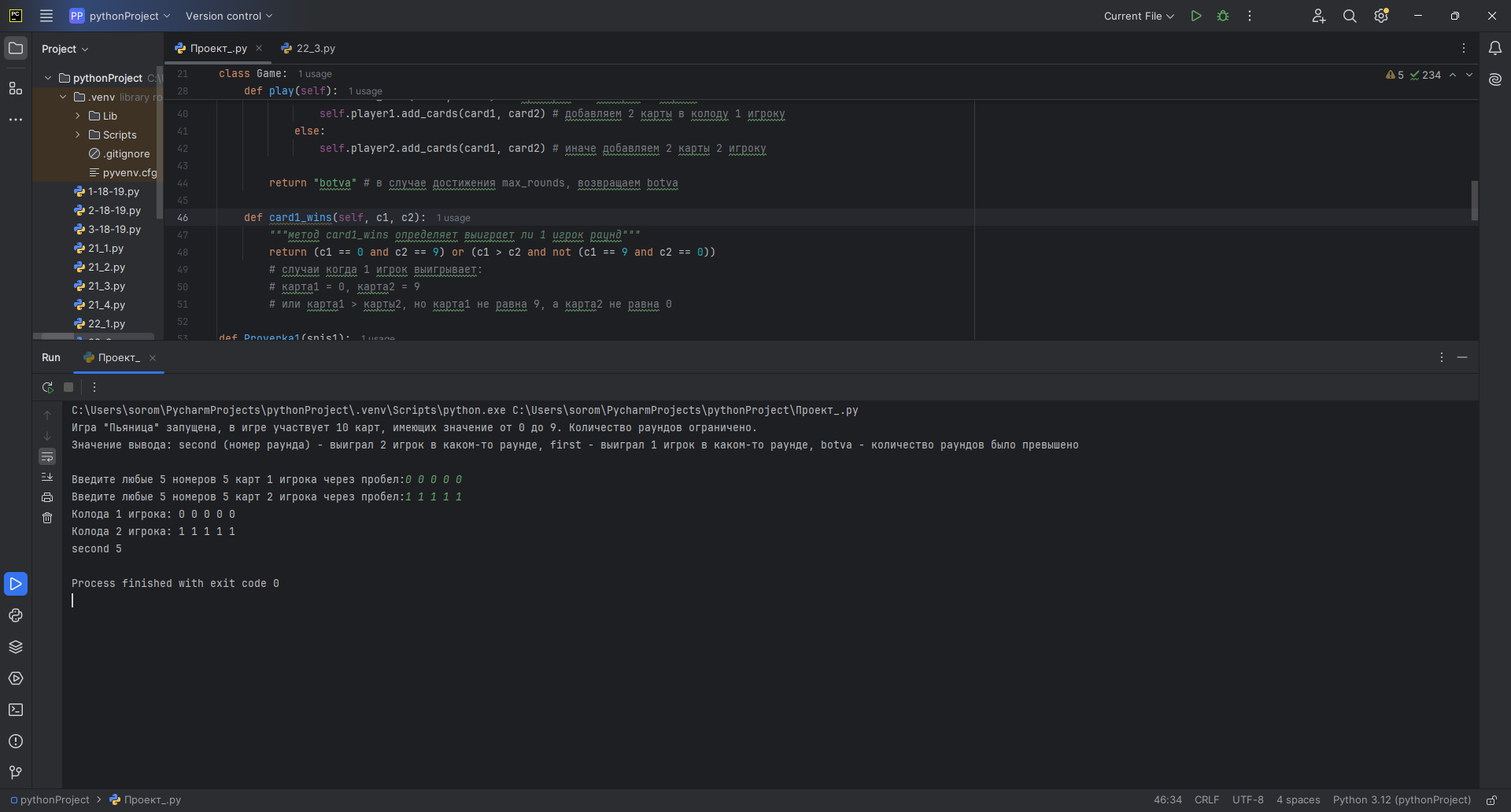
Ошибка: оба ввода пустые

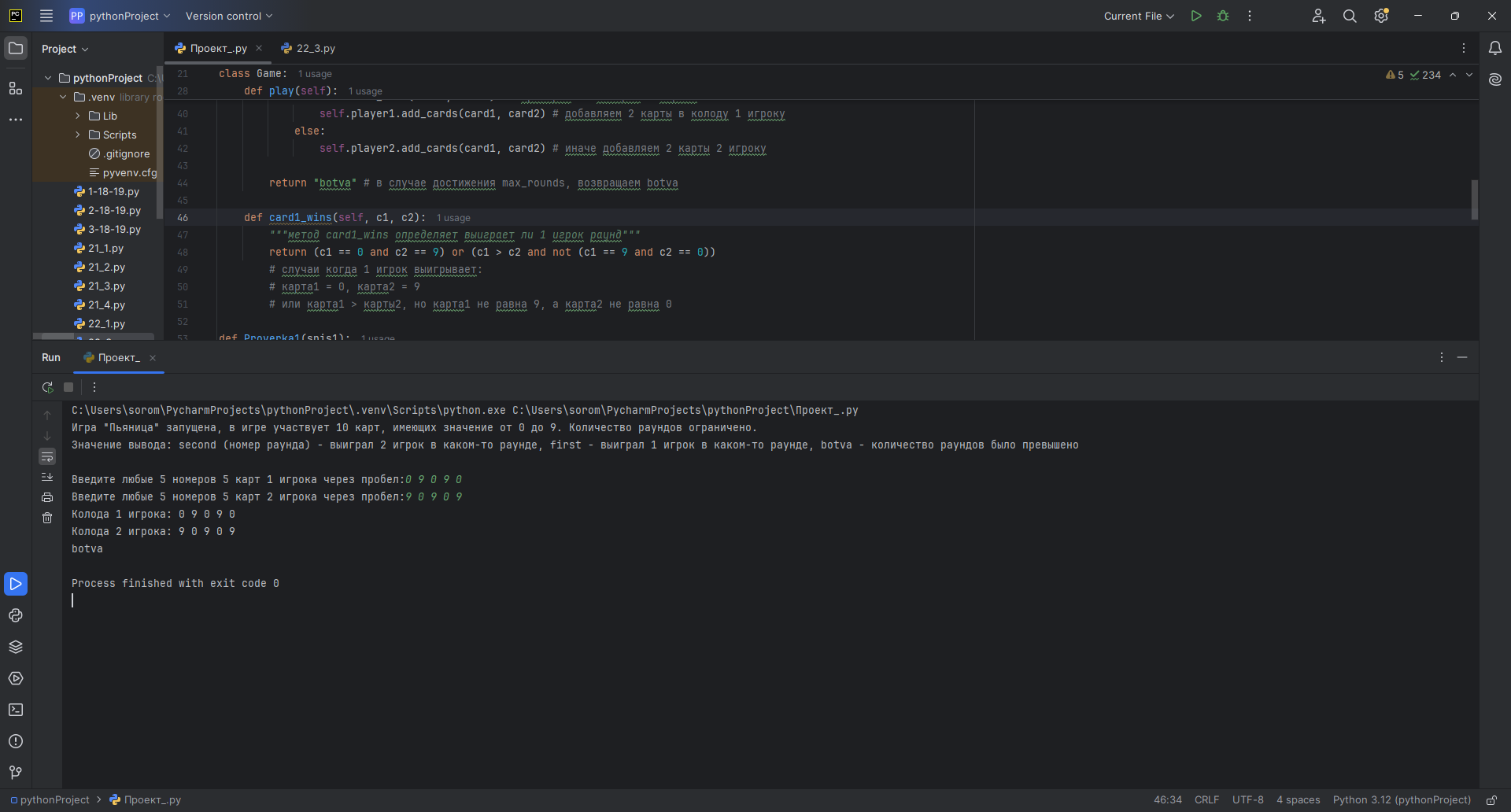


Ошибка: символов в 1 вводе недостаточно, во втором больше 5

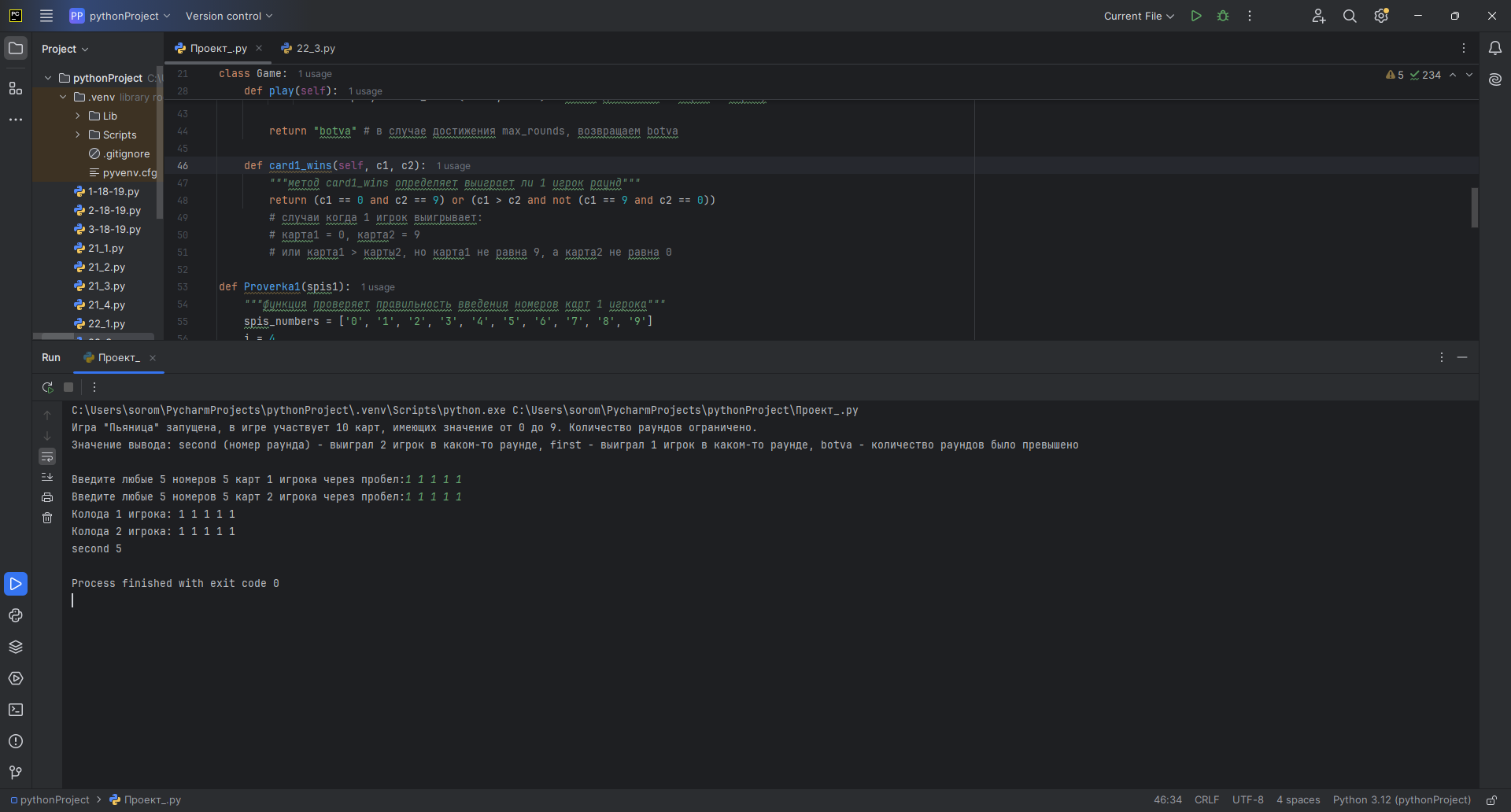


Тесты на правильность





Победа у второго игрока, так как если у первого игрока карта больше, то он забирает карты (условие его карта не 9, а карта соперника не 0), иначе побеждает второй. В таком случае карта первого равна всегда 1, а 1 не может быть больше 1, и второй забирает себе две карты, после 5 раундов выигрывает.



# Код программы

from collections import deque  
  
class Player:  
 """Класс, представляющий игрока """  
 def \_\_init\_\_(self, cards):  
 self.deck = deque(cards) # создаем очередь из списка карт  
  
 def has\_cards(self):  
 """метод has\_cards определяет есть ли у игрока карты в колоде """  
 return bool(self.deck) # возвращает False, если колода пуста, иначе True  
  
 def draw\_card(self):  
 """метод draw\_card позволяет игроку вытянуть карту из колоды"""  
 return self.deck.popleft() # удаляет и возвращает карту из начала колоды (левого конца deque)  
  
 def add\_cards(self, first\_card, second\_card):  
 """метод add\_cards добавляет две карты в конец колоды."""  
 self.deck.append(first\_card) # сначала добавляет 1 карту в правый конец очереди  
 self.deck.append(second\_card) # затем 2 карту - карту проигравшего игрока  
  
class Game:  
 """Класс, представляющий игру """  
 def \_\_init\_\_(self, p1\_cards, p2\_cards):  
 self.player1 = Player(p1\_cards)  
 self.player2 = Player(p2\_cards)  
 self.max\_rounds = 10\*\*6 # максимальное количество раундов  
  
 def play(self):  
 """метод play запускает игру """  
 for round\_num in range(1, self.max\_rounds + 1): # имитация раундов  
 if not self.player1.has\_cards(): # если колода 1 игрока пуста  
 return f"second {round\_num - 1}" # объявляем 2 игрока победителем, указывая раунд  
 if not self.player2.has\_cards(): # аналогично со вторым игроком  
 return f"first {round\_num - 1}"  
  
 card1 = self.player1.draw\_card() # 1 игрок вытягивает карту  
 card2 = self.player2.draw\_card() # 2 игрок вытягивает карту  
  
 if self.card1\_wins(card1, card2): # проверка на выигрыш 1 игрока  
 self.player1.add\_cards(card1, card2) # добавляем 2 карты в колоду 1 игроку  
 else:  
 self.player2.add\_cards(card1, card2) # иначе добавляем 2 карты 2 игроку  
  
 return "botva" # в случае достижения max\_rounds, возвращаем botva  
  
 def card1\_wins(self, c1, c2):  
 """метод card1\_wins определяет выиграет ли 1 игрок раунд"""  
 return (c1 == 0 and c2 == 9) or (c1 > c2 and not (c1 == 9 and c2 == 0))  
 # случаи когда 1 игрок выигрывает:  
 # карта1 = 0, карта2 = 9  
 # или карта1 > карты2, но карта1 не равна 9, а карта2 не равна 0  
  
def Proverka1(spis1):  
 """функция проверяет правильность введения номеров карт 1 игрока"""  
 spis\_numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']  
 i = 4  
 while i <= 4 and i != -1: # i = 4, уменьшаем i в конце цикла для прохода по списку  
 if len(spis1) != 5 or (spis1[i] not in spis\_numbers): # если есть ошибка в длине или в символе  
 print('Ошибка. Попытайтесь ввести снова. Пример ввода: 1 2 3 4 5 ')  
 spis1 = input('Введите любые 5 номеров 5 карт 1 игрока через пробел:').split()  
 i = 5 # i = 5, начнем проверку нового ввода  
 i -= 1  
 return spis1  
  
def Proverka2(spis2):  
 """функция проверяет правильность введения номеров карт 2 игрока"""  
 spis\_numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']  
 j = 4  
 while 0 <= j <= 4: # j = 4, уменьшаем j в конце цикла для прохода по списку  
 if len(spis2) != 5 or (spis2[j] not in spis\_numbers): # если есть ошибка в длине или в символе  
 print('Ошибка. Попытайтесь ввести снова. Пример ввода: 1 2 3 4 5 ')  
 spis2 = input('Введите любые 5 номеров 5 карт 2 игрока через пробел:').split()  
 j = 5 # j = 5, начнем проверку нового ввода  
 j -= 1  
 return spis2  
  
print('Игра "Пьяница" запущена, в игре участвует 10 карт, имеющих значение от 0 до 9. Количество раундов ограничено.')  
print('Значение вывода: second (номер раунда) - выиграл 2 игрок в каком-то раунде, first - выиграл 1 игрок в каком-то раунде, botva - количество раундов было превышено\n')  
p1\_input = input('Введите любые 5 номеров 5 карт 1 игрока через пробел:').split()  
p2\_input = input('Введите любые 5 номеров 5 карт 2 игрока через пробел:').split()  
  
  
p1\_input = Proverka1(p1\_input) # проверяем ввод 1 и 2 колоды  
p2\_input = Proverka2(p2\_input)  
print('Колода 1 игрока:', \*p1\_input)  
print('Колода 2 игрока:', \*p2\_input)  
  
  
p1\_cards = list(map(int, p1\_input))  
p2\_cards = list(map(int, p2\_input))  
  
game = Game(p1\_cards, p2\_cards)  
result = game.play()  
print(result)