ОАИП
Практическая работа 3
Давид Попов

Задание

Написать игру “Крестики-нолики”. Я делал задание на 4:

- можно выбирать размер поля
- проверка ошибок при вводе
- проверка на победу/ничью
- случайный выбор начинающего игрока
- возможность играть несколько раз

Ход работы

Сделал гит-репозиторий, создал виртуальное окружение в директории проекта.
Я представляю доску в виде списка списков, где -1 значит, что поле свободно, 0 - первый игрок, 1 - второй игрок.

```
# -1 = пустое, 0 = игрок 0, 1 = игрок 1
board = [
    [-1 for _ in range(n)] for _ in range(n)
]
```

Я написал основной цикл игры, где вывожу доску, спрашиваю ход, проверяю если кто-то выиграл, или ничья (макс. число ходов сделано)

```
to_move = random.randint(0, 1)
move_no = 0

# -1 = draw, 0 = 0 won, 1 = 1 won
status = -1

while True:
    print_board(board)
    r, c = get_move(to_move, board)
    board[r][c] = to_move

    move_no += 1

    if is_finished(to_move, board, r, c):
        status = to_move
        break
    if move_no == n * n:
        break

    to_move = 1 - to_move
```

Вывод доски я делаю преобразуя диапазон 1..n и клетки доски в символы, и соединяя через пробел:

```
def print_board(board):
    n = len(board)
    print("  "
        + ' '.join(
            map(str,
                range(1, n + 1))))
    for i, r in enumerate(board):
        print(f'{i + 1} '
            + ' '.join(map(
                lambda x:
                '.' if x == -1
                else PLAYER_CHARACTERS[x],
                r)))
```

Проверка на выигрыш - считаю по направлениям (0, 1) = направо, (1, 0) = вниз, (-1, 1), (1, 1) = диагонали, сколько есть клеток нашего цвета вперед и назад, и складываю:

```
def is_finished(to_move, board, r, c):
    n = len(board)

    directions = [
        (0, 1), (1, 0),
        (-1, 1), (1, 1)
    ]
    for dr, dc in directions:
        back = 0
        while True:
            r2 = r - (back + 1) * dr
            c2 = c - (back + 1) * dc
            if not valid_square(n, r2, c2):
                break
            if board[r2][c2] != to_move:
                break
            if back == TO_WIN:
                break
            back += 1
```

```
124             forward = 0
125             while True:
126                 r2 = r + (forward + 1) * dr
127                 c2 = c + (forward + 1) * dc
128                 if not valid_square(n, r2, c2):
129                     break
130                 if board[r2][c2] != to_move:
131                     break
132                 if back == TO_WIN:
133                     break
134                 forward += 1
135
136             if forward + back + 1 >= TO_WIN:
137                 return True
138
```

Все чтение ввода осуществляется в циклах, где ловлю исключения и проверяю на корректность, при необходимости вывожу сообщение об ошибке:

```python
def get_move(to_move, board):
    n = len(board)
    print(f"{PLAYER_CHARACTERS[to_move]}'s turn. "
        + "Enter row and column (e.g. 1 2): ",
        end=" ")

    ok = False
    while not ok:
        try:
            r, c = map(int, input().split())
            r -= 1
            c -= 1
            if not valid_square(n, r, c):
                print(f'числа в пределах от 1 до {n}')
            elif board[r][c] != -1:
                print('клетка занята!')
            else:
                ok = True
        except ValueError:
            print('введите два числа через пробел')


    return r, c
```

После раунда сохраняю сообщение про результат в файл, при необходимости, создаю директорию:

```python
def save_stats(n, status, move_no):
    os.makedirs("stats", exist_ok=True)

    with open('stats/log.txt', 'a') as f:
        date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        if status == -1:
            res = 'draw'
        else:
            res = f'{PLAYER_CHARACTERS[status]} won'

        print(f'{date} {n=} {res} in {move_no} moves.', file=f)

```

Код

```
import random

def main():
	done = False
	while not done:
		n = get_board_size()
		round(n)

		print('play once again? (type no to exit)', end=' ')
		s = input()
		if s == 'no':
			done = True
	print('bye!')



PLAYER_CHARACTERS = ['X', 'O']
TO_WIN = 3


def get_board_size():
	print('Enter the size of the board (3-9): ',
		end='')

	ok = False
	while not ok:
		try:
			n = int(input())
			if n < 3 or n > 9:
				print(f'от 3 до 9!!!')
			else:
				ok = True
		except ValueError:
			print('введите целое число')
	return n




def round(n):
```

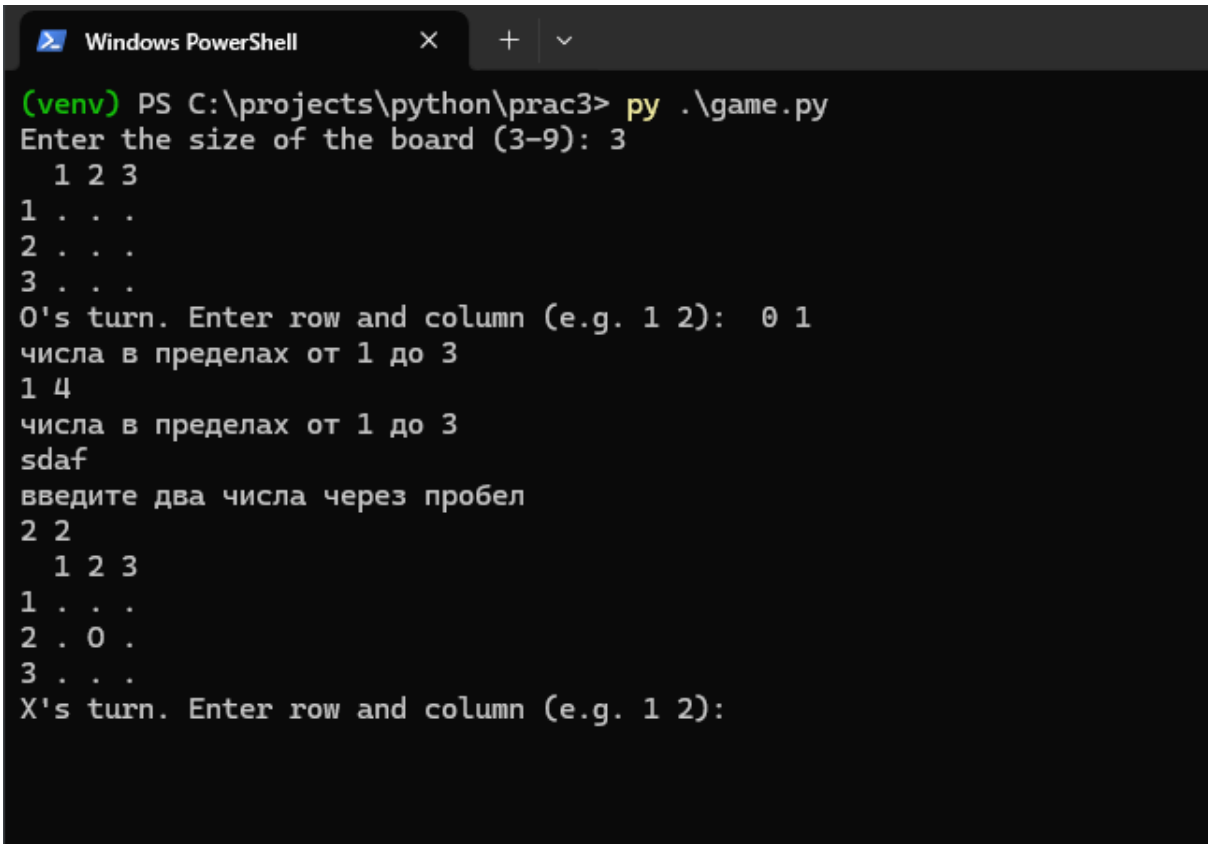
```
	# -1 = пустое, 0 = игрок 0, 1 = игрок 1
	board = [
		[-1 for _ in range(n)] for _ in range(n)
	]
	to_move = random.randint(0, 1)
	move_no = 0

	# -1 = draw, 0 = 0 won, 1 = 1 won
	status = -1

	while True:
		print_board(board)
		r, c = get_move(to_move, board)
		board[r][c] = to_move

		move_no += 1

		if is_finished(to_move, board, r, c):
			status = to_move
			break
		if move_no == n * n:
			break

		to_move = 1 - to_move

	print_board(board)
	if status == -1:
		print("draw")
	else:
		print(f"{PLAYER_CHARACTERS[status]} won")

def print_board(board):
	n = len(board)
	print("  "
		+ ' '.join(
			map(str,
				range(1, n + 1))))
	for i, r in enumerate(board):
		print(f'{i + 1} '
			+ ' '.join(map(
				lambda x:
				'.' if x == -1
				else PLAYER_CHARACTERS[x],
				r)))

def is_finished(to_move, board, r, c):
```

```
	n = len(board)

	directions = [
		(0, 1), (1, 0),
		(-1, 1), (1, 1)
	]
	for dr, dc in directions:
		back = 0
		while True:
			r2 = r - (back + 1) * dr
			c2 = c - (back + 1) * dc
			if not valid_square(n, r2, c2):
				break
			if board[r2][c2] != to_move:
				break
			if back == TO_WIN:
				break
			back += 1

		forward = 0
		while True:
			r2 = r + (forward + 1) * dr
			c2 = c + (forward + 1) * dc
			if not valid_square(n, r2, c2):
				break
			if board[r2][c2] != to_move:
				break
			if back == TO_WIN:
				break
			forward += 1

		if forward + back + 1 >= TO_WIN:
			return True

	return False

def valid_square(n, r, c):
	return (r >= 0 and r < n
		and c >= 0 and c < n)


def get_move(to_move, board):
	n = len(board)
	print(f"{PLAYER_CHARACTERS[to_move]}'s turn. "
		+ "Enter row and column (e.g. 1 2): ",
		end=" ")
```

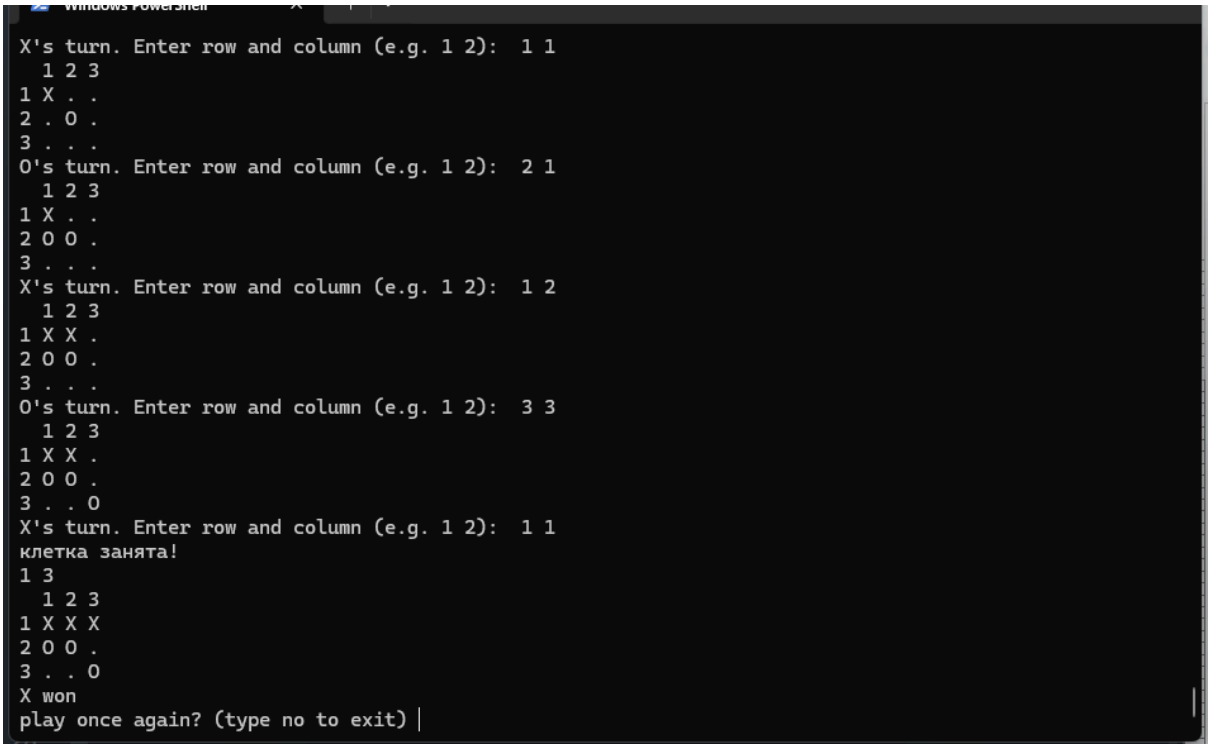
```
	ok = False
	while not ok:
		try:
			r, c = map(int, input().split())
			r -= 1
			c -= 1
			if not valid_square(n, r, c):
				print(f'числа в пределах от 1 до {n}')
			elif board[r][c] != -1:
				print('клетка занята!')
			else:
				ok = True
		except ValueError:
			print('введите два числа через пробел')


	return r, c


if __name__ == '__main__':
	main()
```

Скриншоты

Отработка некорректного ввода, начало игры (начал 0):



Нормальная игра, победа X:



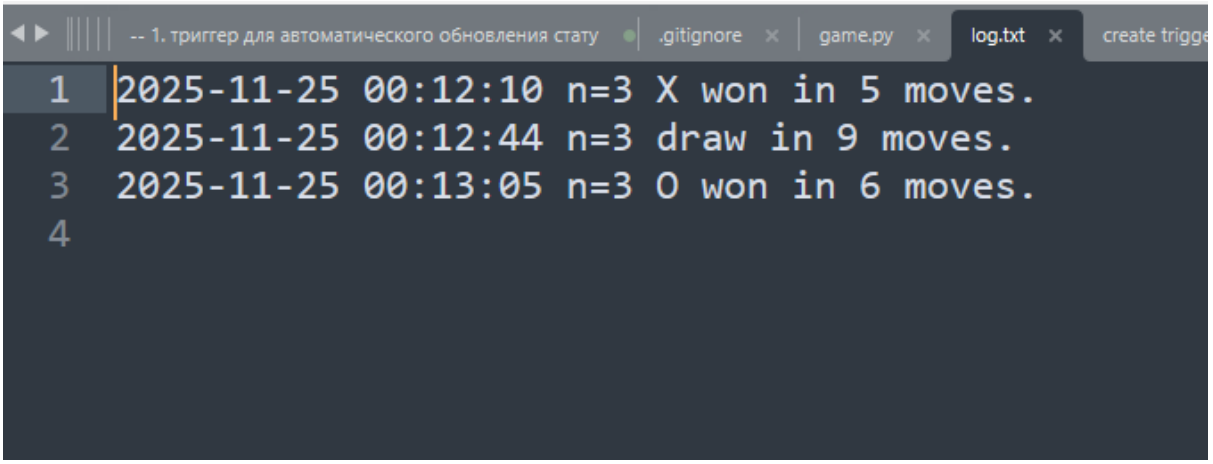
Повторная игра (другой размер доски, начал X) и выход

```
X's turn. Enter row and column (e.g. 1 2):  3 3
  1 2 3 4 5
1 . . . . .
2 O . . . .
3 . . X . .
4 . . . X .
5 . . . . .
O's turn. Enter row and column (e.g. 1 2):  5 5
  1 2 3 4 5
1 . . . . .
2 O . . . .
3 . . X . .
4 . . . X .
5 . . . . O
X's turn. Enter row and column (e.g. 1 2):  2 2
  1 2 3 4 5
1 . . . . .
2 O X . . .
3 . . X . .
4 . . . X .
5 . . . . O
X won
play once again? (type no to exit) no
bye!
(venv) PS C:\projects\python\prac3>
```

Ничья

```
1 X O .
2 . O .
3 O X .
X's turn. Enter row and column (e.g. 1 2):  1 3
  1 2 3
1 X O X
2 . O .
3 O X .
O's turn. Enter row and column (e.g. 1 2):  3 3
  1 2 3
1 X O X
2 . O .
3 O X O
X's turn. Enter row and column (e.g. 1 2):  2 1
  1 2 3
1 X O X
2 X O .
3 O X O
O's turn. Enter row and column (e.g. 1 2):  2 3
  1 2 3
1 X O X
2 X O O
3 O X O
draw
play once again? (type no to exit) no
bye!
(venv) PS C:\projects\python\prac3>
```

Статистика, собранная в файл


-- 1. триггер для автоматического обновления стату
.gitignore
game.py
log.txt
create trigge
1 2025-11-25 00:12:10 n=3 X won in 5 moves.
2 2025-11-25 00:12:44 n=3 draw in 9 moves.
3 2025-11-25 00:13:05 n=3 O won in 6 moves.
4