**ПРОЕКТ   Хрестики-нулики**

**Сценарій**

Ваше завдання — написати **просту програму, яка ніби грає з користувачем у *хрестики-нулики***. Щоб вам було легше, ми вирішили спростити гру. Ось наші припущення:

* комп'ютер (тобто ваша програма) має грати в гру за 'X';
* користувач (наприклад, ви) має грати в гру, використовуючи 'O';
* перший хід належить комп’ютеру − він завжди ставить свій перший 'X' посередині дошки;
* усі квадрати пронумеровані послідовно рядок за рядком, починаючи з 1 (для довідки перегляньте приклад нижче)
* користувач робить свій хід, вводячи номер вибраного ним квадрата − число має бути дійсним, тобто воно повинно бути цілим та бути більше 0 і менше 10, також не можна вказувати на квадрат, який вже зайнятий;
* програма перевіряє, чи закінчилася гра − є чотири можливих вердикти: гра має продовжуватися, гра закінчується нічиєю, ви виграли або виграв комп’ютер;
* комп'ютер відповідає своїм ходом і перевірка повторюється;
* не впроваджуйте жодної форми штучного інтелекту − для гри достатньо, щоб комп'ютер робив випадковий вибір квадрата.

Приклад роботи програми може виглядати так:

+-------+-------+-------+

| | | |

| 1 | 2 | 3 |

| | | |

+-------+-------+-------+

| | | |

| 4 | X | 6 |

| | | |

+-------+-------+-------+

| | | |

| 7 | 8 | 9 |

| | | |

+-------+-------+-------+

Введіть свій хід: 1

+-------+-------+-------+

| | | |

| O | 2 | 3 |

| | | |

+-------+-------+-------+

| | | |

| 4 | X | 6 |

| | | |

+-------+-------+-------+

| | | |

| 7 | 8 | 9 |

| | | |

+-------+-------+-------+

+-------+-------+-------+

| | | |

| O | X | 3 |

| | | |

+-------+-------+-------+

| | | |

| 4 | X | 6 |

| | | |

+-------+-------+-------+

| | | |

| 7 | 8 | 9 |

| | | |

+-------+-------+-------+

Введіть свій хід: 8

+-------+-------+-------+

| | | |

| O | X | 3 |

| | | |

+-------+-------+-------+

| | | |

| 4 | X | 6 |

| | | |

+-------+-------+-------+

| | | |

| 7 | O | 9 |

| | | |

+-------+-------+-------+

+-------+-------+-------+

| | | |

| O | X | 3 |

| | | |

+-------+-------+-------+

| | | |

| 4 | X | X |

| | | |

+-------+-------+-------+

| | | |

| 7 | O | 9 |

| | | |

+-------+-------+-------+

Введіть свій хід: 4

+-------+-------+-------+

| | | |

| O | X | 3 |

| | | |

+-------+-------+-------+

| | | |

| O | X | X |

| | | |

+-------+-------+-------+

| | | |

| 7 | O | 9 |

| | | |

+-------+-------+-------+

+-------+-------+-------+

| | | |

| O | X | X |

| | | |

+-------+-------+-------+

| | | |

| O | X | X |

| | | |

+-------+-------+-------+

| | | |

| 7 | O | 9 |

| | | |

+-------+-------+-------+

Введіть свій хід: 7

+-------+-------+-------+

| | | |

| O | X | X |

| | | |

+-------+-------+-------+

| | | |

| O | X | X |

| | | |

+-------+-------+-------+

| | | |

| O | O | 9 |

| | | |

+-------+-------+-------+

Ви виграли!

**Вимоги**

Реалізуйте наступні функції:

* дошка повинна зберігатися у вигляді триелементного списку, тоді як кожен елемент є ще одним список із трьох елементів (внутрішні списки являють собою рядки), так щоб усі квадрати могли бути доступні за допомогою наступного синтаксису:

board[row][column]

* кожен з елементів внутрішнього списку може містити 'O', 'X' або цифру, яка відповідає номеру квадрата (такий квадрат вважається вільним)
* зовнішній вигляд дошки повинен бути точно таким, як і у наведеному прикладі.
* реалізувати функції, які наведені у редакторі.

Отримати випадкове ціле число можна за допомогою функції Python під назвою randrange(). У наведеному нижче прикладі програми показано, як нею користуватися (програма виводить десять випадкових чисел від 0 до 8).

Примітка: інструкція from-import надає доступ до функції randrange, яка визначена в зовнішньому модулі Python під назвою random.

from random import randrange

for i in range(10):

print(randrange(8))

def display\_board(board):

# Функція приймає один параметр, що містить поточний статус дошки

# і виводить його на консоль.

def enter\_move(board):

# Функція приймає поточний статус дошки, запитує користувача про його хід,

# перевіряє введення та оновлює дошку відповідно до рішення користувача.

def make\_list\_of\_free\_fields(board):

# Функція перевіряє дошку та створює список усіх вільних квадратів;

# список складається з кортежів, так що кожен кортеж є парою номерів рядка і стовпчика.

def winner\_for(board, sign):

# Функція аналізує стан дошки, щоб перевірити, чи

# э в грі переможець

def draw\_move(board):

# Функція малює хід комп'ютера та оновлює дошку.