PA 5. Tic-Tac-Toe Game

Topics

1. String manipulation
2. Lists
3. Classes
4. Methods

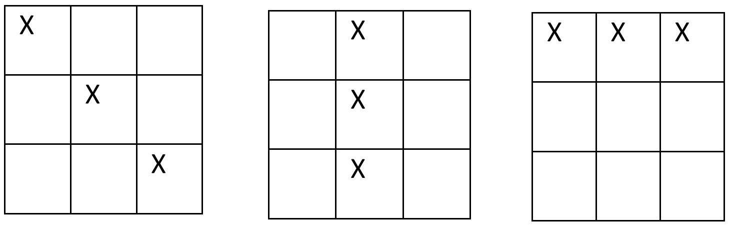
Instruction

Tic-Tac-Toe Game

Objective: practicing with classes, methods, lists, loops, if and elif statements, and string methods

Description

In this assignment you will write a program for a popular game Tic-Tac-Toe. Your program will generate a board with nine cells using the traditional 3 x 3 layout and ask each of two users to choose one cell at a time. The users take turns, and each user marks the board cells with a naught O or a cross X. The game continues until either one of the users won or all cells on the board are marked with O or X. The user won when the three marks (O or X) are located on one line horizontally, vertically, or diagonally as shown on the picture below:



There are 8 winning cases in total: 3 horizontally arranged marks, 3 vertically arranged marks, and two diagonally arranged marks. The program writes who is the winner or it is a tie. After that it will ask the users if they want to play again. Here are the snippets of the program output, note that the dollar sign $ stands for the terminal prompt. Be aware that your program output should match this format exactly. Please read the instructions carefully and completely before starting to work on your program!

In the beginning a program should output the following messages and draws the board:

$ Welcome to TIC-TAC-TOE Game!

```
   A   B   C
 +---+---+---+
1|   |   |   |
 +---+---+---+
2|   |   |   |
 +---+---+---+
3|   |   |   |
 +---+---+---+
```

$ Bob, X: Enter a cell [A-C][1-3]:

If the user (the default first user is Bob) chose cell a1, the program updates the board and produces the following output:

```
   A   B   C
 +---+---+---+
1| X |   |   |
 +---+---+---+
2|   |   |   |
 +---+---+---+
3|   |   |   |
 +---+---+---+
```

$ Alice, O: Enter a cell [A-C][1-3]:

If the user (the default second user is Alice) chose cell a1, the program does not update the board because this cell is already marked with an X. It asks the user to enter valid input in the following way:

$ You did not choose correctly.

$ Alice, O: Enter a cell [A-C][1-3]:

Notice that the second sentence is the same prompt used before. If the user (Alice or Bob) do not choose valid input the program outputs the same messages as before until the user enters valid input. Valid input is a two-character string, which has the first character a letter A, B, or C (uppercase or lowercase) and the second character is 1, 2, or 3. The program should analyze if input is valid or invalid.

If Alice enters b2, the programs updates the board and produces the following output:

```
   A   B   C
 +---+---+---+
1| X |   |   |
 +---+---+---+
2|   | O |   |
 +---+---+---+
3|   |   |   |
 +---+---+---+
```

$ Bob, X: Enter a cell [A-C][1-3]:

As you can see, the program makes turns for players: after Bob chose a cell, Alice chooses a cell, and after Alice chose a cell, Bob chooses a cell. When the game is over, the program prints one of the following messages:

$ Bob is a winner!

$ Would you like to play again? [Y/N]

or

$ Alice is a winner!

$ Would you like to play again? [Y/N]

or

$ It is a tie!

$ Would you like to play again? [Y/N]

If the user types ‘Y’ or ‘y’ the program starts a new game, draws the empty board, and prints the following message again:

```
   A   B   C
 +---+---+---+
1|   |   |   |
 +---+---+---+
2|   |   |   |
 +---+---+---+
3|   |   |   |
 +---+---+---+
```

$ Bob, X: Enter a cell [A-C][1-3]:

Otherwise it prints the following new message and terminates:

```
$ Goodbye!
```

Programming Approaches

In this assignment you need to create two classes Board and Player, each class should be written in its own file named board.py and player.py. They should be located in the same directory as the main program tictac.py.

Class Board has seven methods init, get_size, get_winner, set, isempty, isdone, and show. Please read the following code and instructions carefully. You can type or copy and paste this code into your file board.py. The file board.py should be located in the same directory as tictac.py (the main program) and player.py.

```
class Board:
    def __init__(self):
        self.sign = " "
        self.size = 3
        self.board = list(self.sign * self.size**2)
        self.winner = ""
    def get_size(self):
        # return the board size (an instance size)
    def get_winner(self):
        # return the winner (a sign O or X) (an instance winner)
    def set(self, cell, sign):
        # mark the cell with the sign (X or O)
    def isempty(self, cell):
        # return True if the cell is empty (not marked with X or O)
    def isdone(self):
        done = False
        # check all game terminating conditions, if one of them is present, assign the var done to True
        # depending on conditions assign the instance var winner to O or X
        return done
    def show(self):
        # draw the board
```

A class Player should have four methods init, get_sign, get_name and choose. Please read the code and instructions carefully. You can type or copy and paste this code into your file player.py. The file player.py should be located in the same directory as tictac.py (the main program) and board.py.

```
class Player:
    def __init__(self, name, sign):
        self.name = name
        self.sign = sign
    def get_sign(self):
        # return an instance sign
    def get_name(self):
        # return an instance name
    def choose(self, board):
        # prompt the user to choose a cell
        # if the user enters a valid string and the cell on the board is empty, update the board
        # otherwise print a message that the input is wrong and rewrite the prompt
        # use the methods board.isempty(cell), and board.set(cell, sign)
        # you need to convert A1, B1, …, C3 cells into index values from 1 to 9
        # you can do the conversion here in the player.py or in the board.py
        # this implementation is up to you
```

In the main program tictac.py write the following code. Your code should match this code precisely!!!

```
# main program

from board import Board
from player import Player

print("Welcome to TIC-TAC-TOE Game!")
while True:
    board = Board()
    player1 = Player("Bob", "X")
    player2 = Player("Alice", "O")
    turn = True
    while True:
         board.show()
         if turn:
            player1.choose(board)
            turn = False
        else:
            player2.choose(board)
            turn = True
        if board.isdone():
            break
        board.show()
        if board.get_winner() == player1.get_sign():
            print(f"{player1.get_name()} is a winner!")
        elif board.get_winner() == player2.get_sign():
            print(f"{player2.get_name()} is a winner!")
        else:
            print("It is a tie!")
        ans = input("Would you like to play again? [Y/N] ").upper()
        if (ans != "Y"):
            break
print("Goodbye!")
```

Grading Rubric

You can start to work on your program without using eval_pa4 script. However, when you are done writing your program, you have to use the eval_pa4 script to evaluate your program specifications and performance. Please read the instructions about using the eval_pa4 script on Canvas. All files, including the instructions, tictac.py, eval_pa4, and supplementary files will be located in the **Files**/ Scripts/ PA4 folder on Canvas.

Criteria	Points 150	Excellent	Good	Satisfactory	Unsatisfactory
Submitted on time	150 possible points	yes			no
Runs without errors	50	yes			no
Produces right output	50	yes	minor format mismatch 10-19 points	format mismatch 1-9 points	no
Has a comment block	5	yes	minor format mismatch 3-4 points	format mismatch 1-2 points	no
all files named correctly	4	yes			no
has a player class	5	yes			no
has all player methods	4	4 correctly implemented 4 points	1 is missing or incorrectly implemented	2-3 are missing or incorrectly implemented	no

			3 points	1-2 points	
has a board class	5	yes			no
has all board methods	7	7 correctly implemented 7 points	1-2 is missing or incorrectly implemented 5-6 points	3-6 are missing or incorrectly implemented 1-4 points	no
Uses lists	5	yes			no
Uses string methods	5	yes			no
Uses while or for loops	5	yes			no
Uses if-else of elif	5	yes			no

What to turn in
Submit your programs board.py, player.py to the assignment name PA4 on Canvas before the due date. As always start early and ask questions in lab sessions, office hours, and on Canvas.

Extra Credit: Tic-Tac-Toe Game AI, 20 points

Create an AI (Artificial Intelligence) Player that can play against the user. You need to create a new class called AI and place it in the same module, the file player.py, where the class Player is written. The AI is a subclass of the class Player and should inherit all properties from its superclass Player. The init and choose methods should be overridden (modified). The output of the program should be the same as before, the user plays as Alice and the AI plays as Bob. The only difference from the previous tic-tac-toe game is that the user does not have to play for Bob, the AI (your computer program) plays instead of the user.

You need to modify the tictac.py to create an AI object: you can achieve it by substituting player1 = Player("Bob", "X") to player1 = AI("Bob", "X", board) and the import statement from player import Player to from player import Player, AI.

The simplest implementation of an AI player is to use a random choice for generating a valid move. For this strategy you need to create all possible moves: in the beginning of a game all moves are empty cells on the board, so you can create a list of all cells and then remove the occupied cells from the board as the game progresses. You can import choice from the random module to randomly choose a cell (a move) from the list of all possible cells (moves).

To get extra credit you need to submit your player.py with all two classes Player and AI.

Extra Credit: Tic-Tac-Toe Game Smart AI, 30 points

You can improve the performance of somewhat dumb AI Player by creating a smart AI. You need to create a new class called SmartAI and place it in the same module, the file player.py, where the class Player and AI are written. The SmartAI is a subclass of the class AI and should inherit all properties from its superclass AI. Only the choose methods should be overridden (modified). The output of the program should be the same as before, but this time the user plays as Bob and the SmartAI plays as Alice. The only difference from the previous tic-tac-toe game is that the user does not have to play for Alice, the SmartAI (your computer program) plays instead of the user.

You need to modify the tictac.py to create a SmartAI object: you can achieve it by substituting player2 = Player("Alice", "O") to player2 = SmartAI("Alice", "O", board) and the import statement from player import Player, AI to from player import Player, AI, SmartAI

The simplest strategy is to allow the program to check all possible winning conditions (two Os in a row, column, or a diagonal) and add the missing O to complete them and win the game. You also have to implement checking the winning conditions of the opponent and placing O in these potential patterns to prevent the opponent from winning the game.

After you successfully implemented both SmartAI (Alice) and AI (Bob) you can even make them to play against each other.

To get extra credit you need to submit your player.py with all three classes Player, AI, and SmartAI.