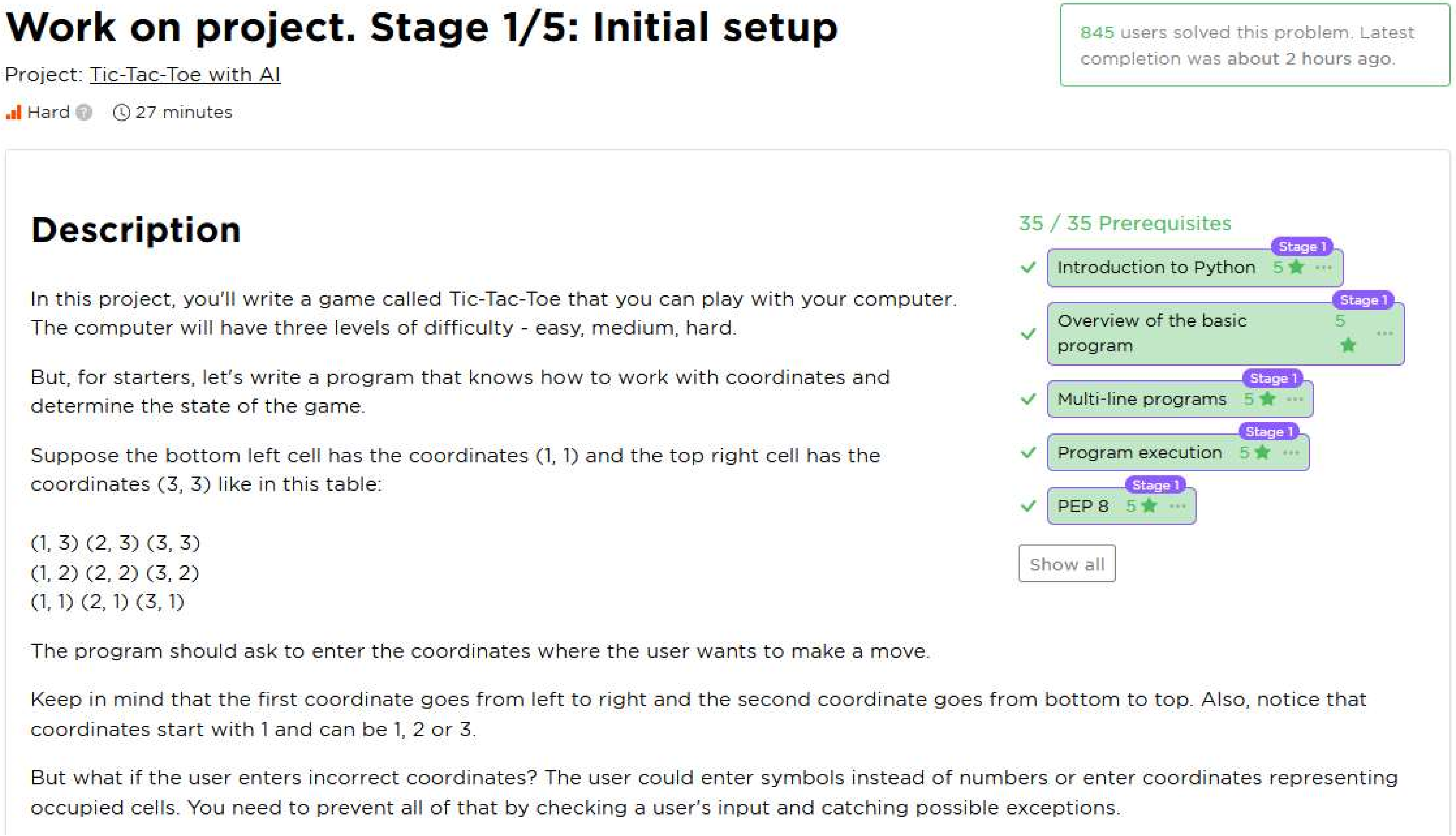
AUTHORS OF THIS TASK: https://hyperskill.org/



Work on project. Stage 1/5: Initial setup

Project: Tic-Tac-Toe with AI

Hard 27 minutes

845 users solved this problem. Latest completion was about 2 hours ago.

Description

In this project, you'll write a game called Tic-Tac-Toe that you can play with your computer. The computer will have three levels of difficulty - easy, medium, hard.

But, for starters, let's write a program that knows how to work with coordinates and determine the state of the game.

Suppose the bottom left cell has the coordinates (1, 1) and the top right cell has the coordinates (3, 3) like in this table:

```
(1, 3) (2, 3) (3, 3)
(1, 2) (2, 2) (3, 2)
(1, 1) (2, 1) (3, 1)
```

The program should ask to enter the coordinates where the user wants to make a move.

Keep in mind that the first coordinate goes from left to right and the second coordinate goes from bottom to top. Also, notice that coordinates start with 1 and can be 1, 2 or 3.

But what if the user enters incorrect coordinates? The user could enter symbols instead of numbers or enter coordinates representing occupied cells. You need to prevent all of that by checking a user's input and catching possible exceptions.

35 / 35 Prerequisites

- ✓ Introduction to Python 5 ★ (Stage 1)
- ✓ Overview of the basic program 5 ★ (Stage 1)
- ✓ Multi-line programs 5 ★ (Stage 1)
- ✓ Program execution 5 ★ (Stage 1)
- ✓ PEP 8 5 ★ (Stage 1)

Objectives

The program should work in the following way:

1. Get the 3x3 field from the first input line (it contains 9 symbols containing `X`, `O` and `_`, the latter means it's an empty cell),
2. Output this 3x3 field with cells before the user's move,
3. Then ask the user about his next move,
4. Then the user should input 2 numbers that represent the cell on which user wants to make his X or O. (9 symbols representing the field would be on the first line and these 2 numbers would be on the second line of the user input). Since the game always starts with X, if the number of X's and O's on the field is the same, the user should make a move with X, and if X's is one more than O's, then the user should make a move with O.
5. Analyze user input and show messages in the following situations:
   - `"This cell is occupied! Choose another one!"` - if the cell is not empty;
   - `"You should enter numbers!"` - if the user enters other symbols;
   - `"Coordinates should be from 1 to 3!"` - if the user goes beyond the field.
6. Then output the table including the user's most recent move.
7. Then output the state of the game.

Possible states:

- `"Game not finished"` - when no side has a three in a row but the field has empty cells;
- `"Draw"` - when no side has a three in a row and the field has no empty cells;
- `"X wins"` - when the field has three X in a row;
- `"O wins"` - when the field has three O in a row;

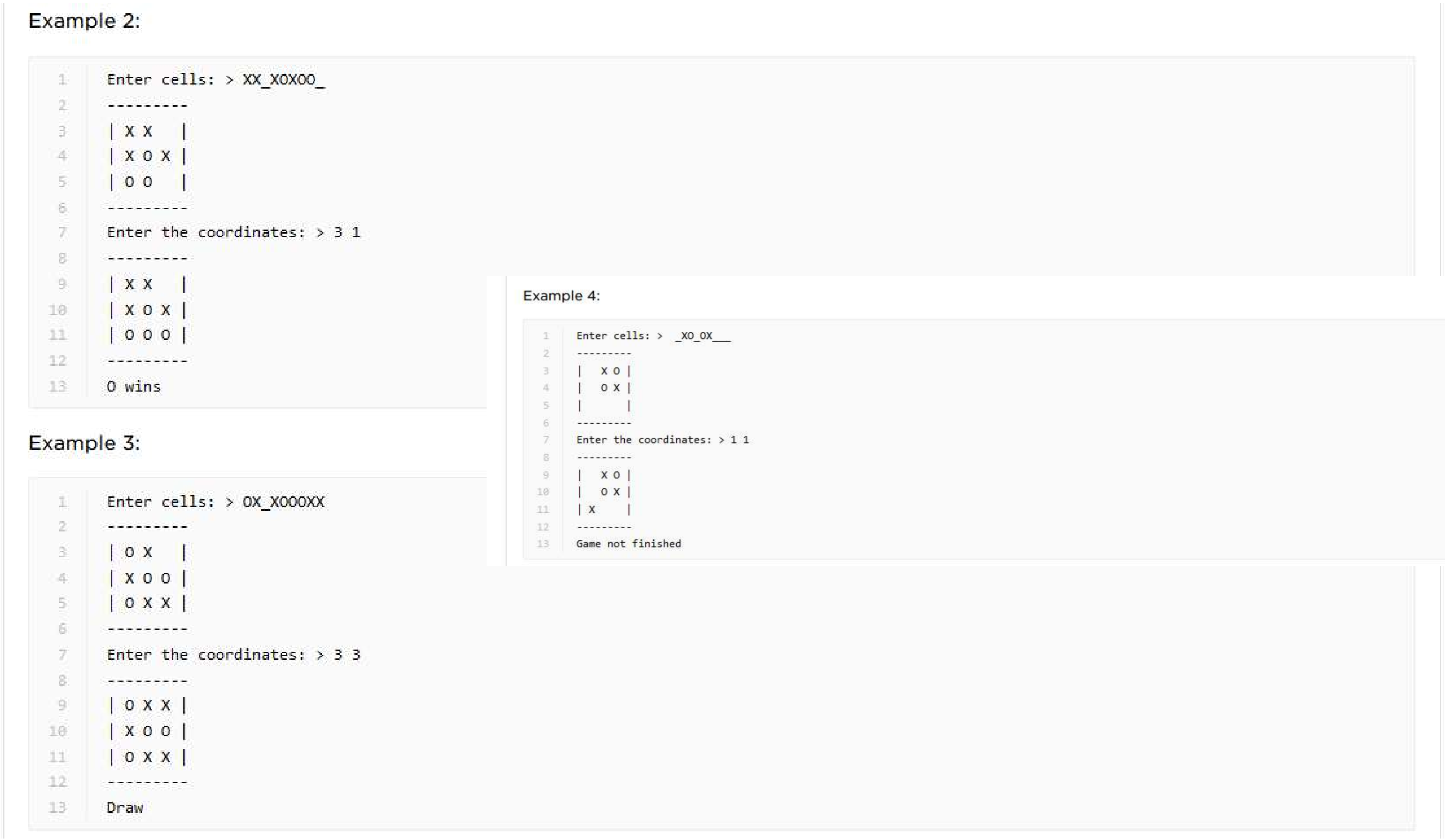
If the user input wrong coordinates, the program should keep asking until the user enters coordinate that represents an empty cell on the field and after that output the field with that move. You should output the field only 2 times - before the move and after a legal move.

Examples

The examples below show how your program should work.
The greater-than symbol followed by space (`> `) represents the user input. Notice that it's not the part of the input.

Example 1:

```
Enter cells: > _XXOO_OX_
---------
|   X X |
| O O   |
| O X   |
---------
Enter the coordinates: > 1 1
This cell is occupied! Choose another one!
Enter the coordinates: > one
You should enter numbers!
Enter the coordinates: > one three
You should enter numbers!
Enter the coordinates: > 4 1
Coordinates should be from 1 to 3!
Enter the coordinates: > 1 3
---------
| X X X |
| O O   |
| O X   |
---------
X wins
```

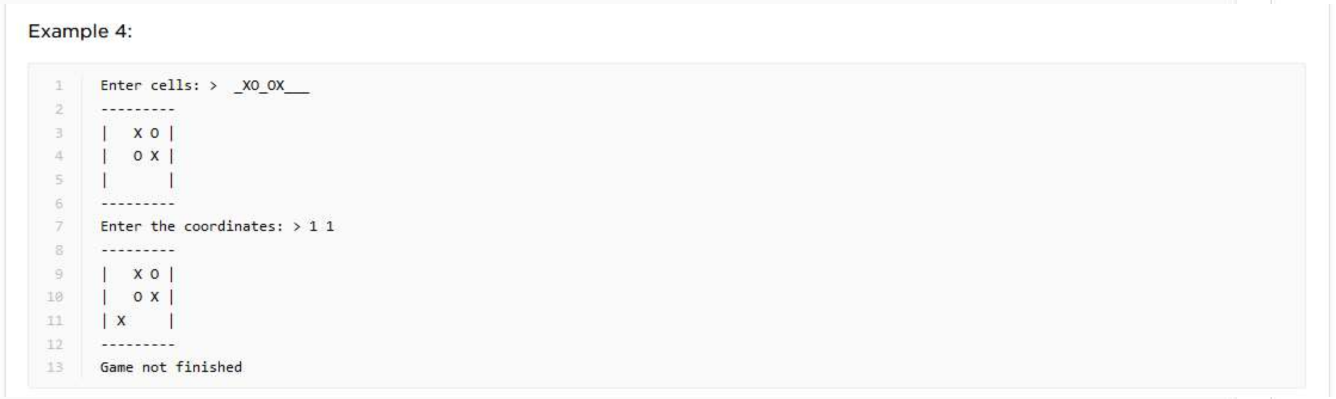


Example 2:

```
Enter cells: > XX_XOXOO_
---------
| X X   |
| X O X |
| O O   |
---------
Enter the coordinates: > 3 1
---------
| X X   |
| X O X |
| O O O |
---------
O wins
```

Example 3:

```
Enter cells: > OX_XOOOXX
---------
| O X   |
| X O O |
| O X X |
---------
Enter the coordinates: > 3 3
---------
| O X X |
| X O O |
| O X X |
---------
Draw
```



Example 4:

```
Enter cells: > _XO_OX___
---------
|   X O |
|   O X |
|       |
---------
Enter the coordinates: > 1 1
---------
|   X O |
|   O X |
| X     |
---------
Game not finished
```

Work on project. Stage 2/5: Easy does it

Project: Tic-Tac-Toe with AI

Medium ⓘ 9 minutes

814 users solved this problem. Latest completion was about 1 hour ago.

Description

Now it is time to make a working game. Let's create our first opponent! In this version of the program, the user will be playing as X, and the `"easy"` level computer will be playing as O. This will be our first small step to the AI!

Let's make it so that at this level the computer will make random moves. This level will be perfect for those who play this game for the first time! Well, though... You can create a level of difficulty that will always give in and never win the game. You can implement such a level along with "easy" level, if you want, but it will not be tested.

4 / 4 Prerequisites

- ✓ IDE 4 ★ — Stage 2
- ✓ PyCharm basics 4 ★ — Stage 2
- ✓ Load module 4 ★ — Stage 2
- ✓ Random module 4 ★ — Stage 2

Objectives

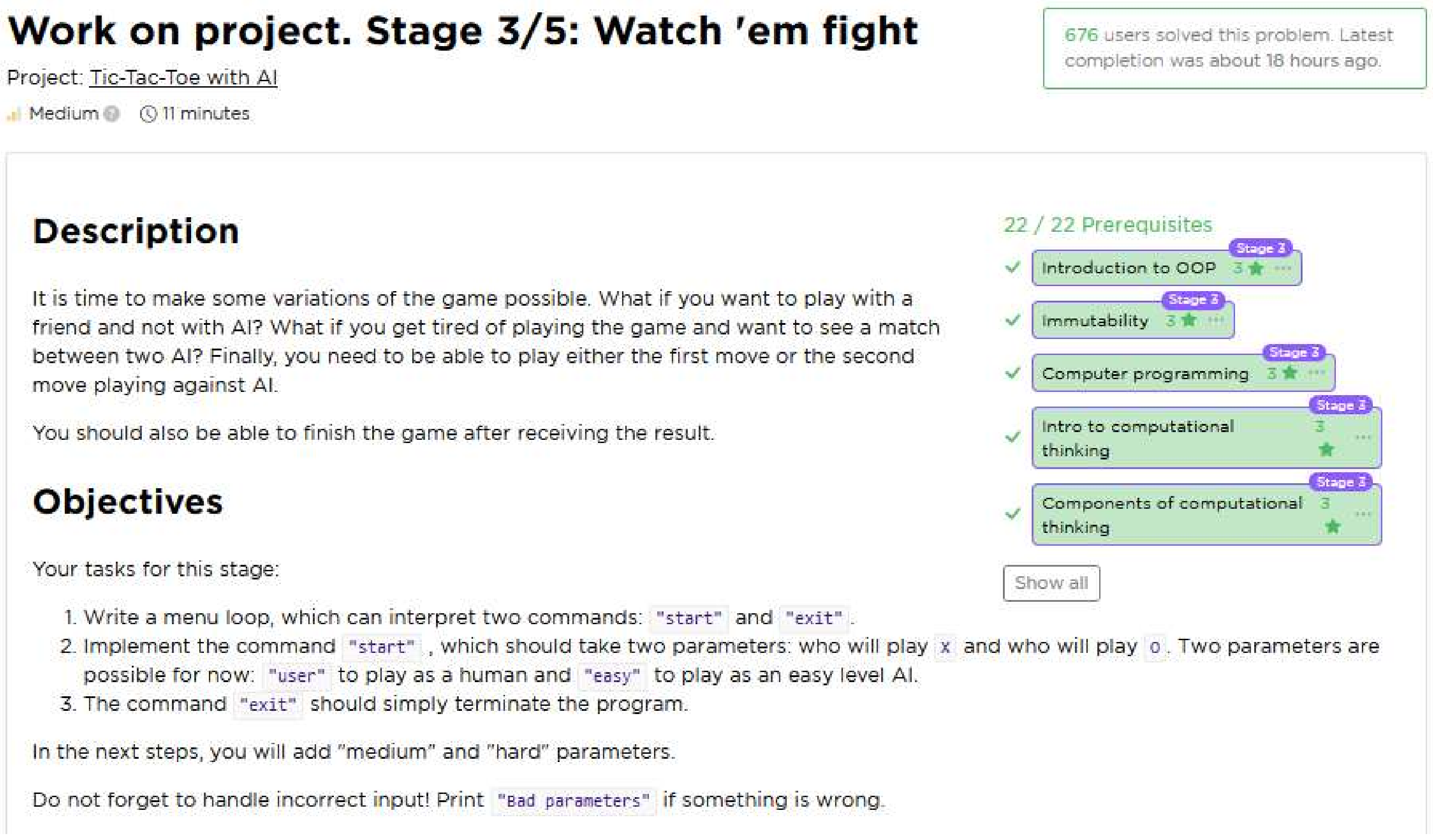
In this stage you should implement the following:

1. When starting the program, an empty field should be displayed.
2. The first to start the game should be the user as `X`. The program should ask the user to enter the cell coordinates.
3. Next the computer should make its move as `O`. And so on until someone will win or there will be a draw.
4. At the very end of the game you need to print the final result of the game.

Example

The example below shows how your program should work.
The greater-than symbol followed by space (`> `) represents the user input. Notice that it's not the part of the input.

```
---------
|       |
|       |
|       |
---------
Enter the coordinates: > 2 2
---------
|       |
|   X   |
|       |
---------
Making move level "easy"
---------
| O     |
|   X   |
|       |
---------
Enter the coordinates: > 3 1
---------
| O     |
|   X   |
|     X |
---------
Making move level "easy"
---------
| O     |
| O X   |
|     X |
---------
Enter the coordinates: > 1 1
---------
| O     |
| O X   |
| X   X |
---------
Making move level "easy"
---------
| O     |
| O X O |
| X   X |
---------
Enter the coordinates: > 2 1
---------
| O     |
| O X O |
| X X X |
---------
X wins
```



Work on project. Stage 3/5: Watch 'em fight

Project: Tic-Tac-Toe with AI

Medium 11 minutes

676 users solved this problem. Latest completion was about 18 hours ago.

Description

It is time to make some variations of the game possible. What if you want to play with a friend and not with AI? What if you get tired of playing the game and want to see a match between two AI? Finally, you need to be able to play either the first move or the second move playing against AI.

You should also be able to finish the game after receiving the result.

22 / 22 Prerequisites

- ✓ Introduction to OOP — Stage 3 — 3
- ✓ Immutability — Stage 3 — 3
- ✓ Computer programming — Stage 3 — 3
- ✓ Intro to computational thinking — Stage 3 — 3
- ✓ Components of computational thinking — Stage 3 — 3

Show all

Objectives

Your tasks for this stage:

1. Write a menu loop, which can interpret two commands: `"start"` and `"exit"`.
2. Implement the command `"start"`, which should take two parameters: who will play `X` and who will play `O`. Two parameters are possible for now: `"user"` to play as a human and `"easy"` to play as an easy level AI.
3. The command `"exit"` should simply terminate the program.

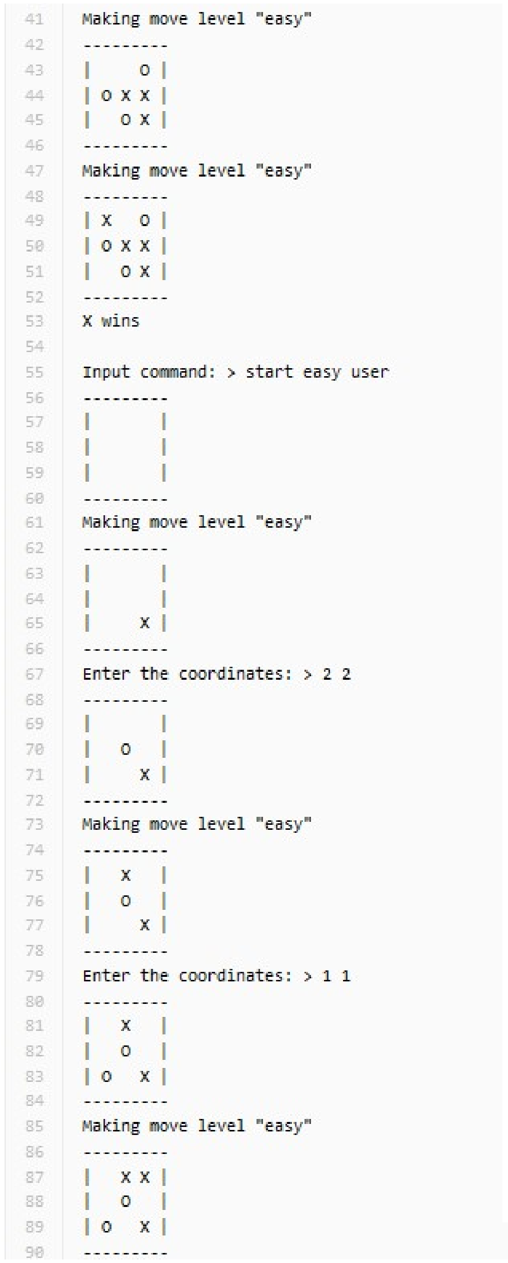
In the next steps, you will add "medium" and "hard" parameters.

Do not forget to handle incorrect input! Print `"Bad parameters"` if something is wrong.

Example

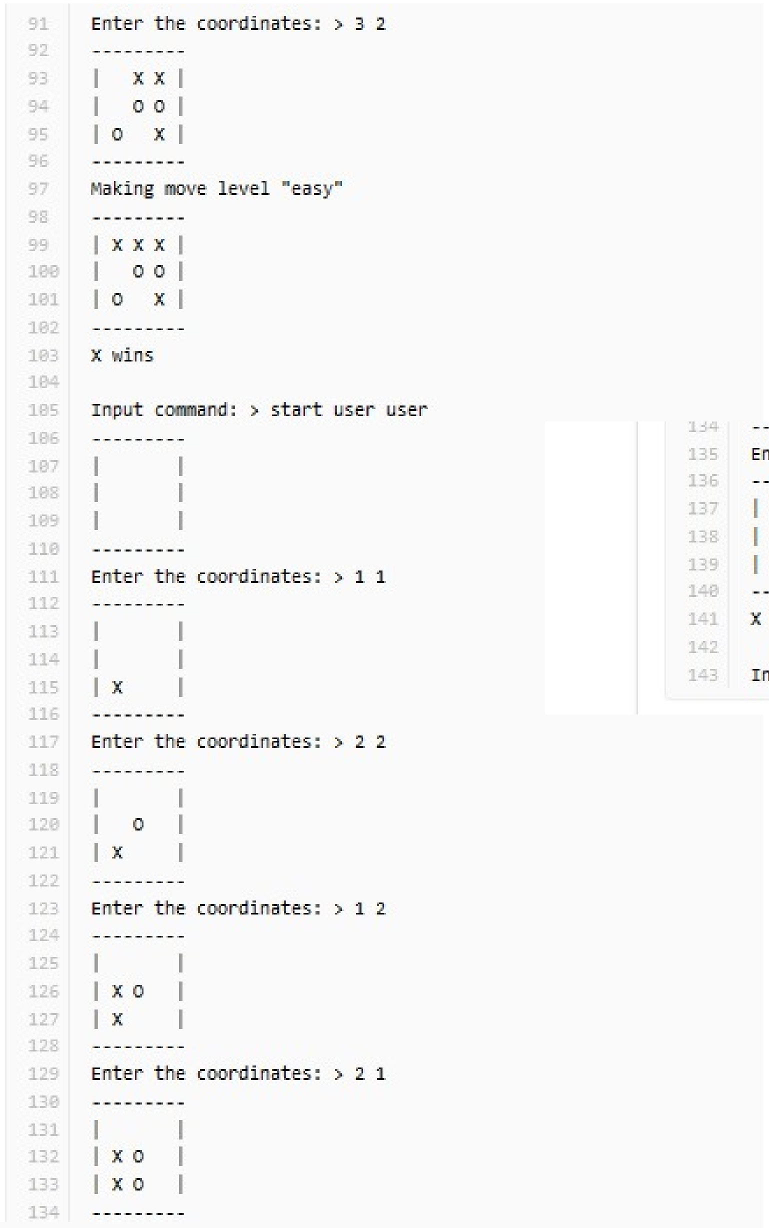
The example below shows how your program should work.
The greater-than symbol followed by space (`> `) represents the user input. Notice that it's not the part of the input.

```
Input command: > start
Bad parameters!
Input command: > start easy
Bad parameters!
Input command: > start easy easy
---------
|       |
|       |
|       |
---------
Making move level "easy"
---------
|       |
|     X |
|       |
---------
Making move level "easy"
---------
|       |
| O   X |
|       |
---------
Making move level "easy"
---------
|       |
| O   X |
|     X |
---------
Making move level "easy"
---------
|       |
| O   X |
|   O X |
---------
Making move level "easy"
---------
|       |
| O X X |
|   O X |
---------
```



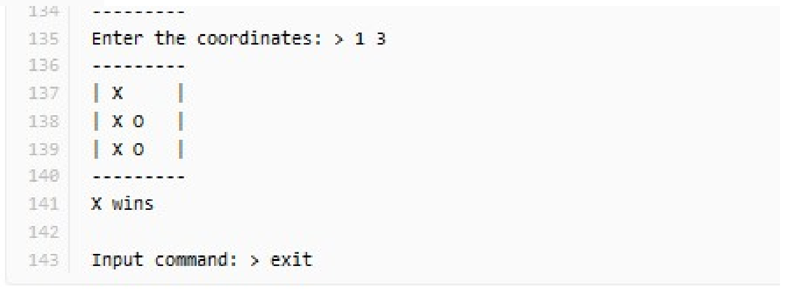
```
Making move level "easy"
---------
|     O |
| O X X |
|   O X |
---------
Making move level "easy"
---------
| X   O |
| O X X |
|   O X |
---------
X wins

Input command: > start easy user
---------
|       |
|       |
|       |
---------
Making move level "easy"
---------
|       |
|       |
|     X |
---------
Enter the coordinates: > 2 2
---------
|       |
|   O   |
|     X |
---------
Making move level "easy"
---------
|   X   |
|   O   |
|     X |
---------
Enter the coordinates: > 1 1
---------
|   X   |
|   O   |
| O   X |
---------
Making move level "easy"
---------
|   X X |
|   O   |
| O   X |
---------
```



```
Enter the coordinates: > 3 2
---------
|   X X |
|   O O |
| O   X |
---------
Making move level "easy"
---------
| X X X |
|   O O |
| O   X |
---------
X wins

Input command: > start user user
---------
|       |
|       |
|       |
---------
Enter the coordinates: > 1 1
---------
|       |
|       |
| X     |
---------
Enter the coordinates: > 2 2
---------
|       |
|   O   |
| X     |
---------
Enter the coordinates: > 1 2
---------
|       |
| X O   |
| X     |
---------
Enter the coordinates: > 2 1
---------
|       |
| X O   |
| X O   |
---------
```



```
---------
Enter the coordinates: > 1 3
---------
| X     |
| X O   |
| X O   |
---------
X wins

Input command: > exit
```

Work on project. Stage 4/5: Signs of intelligence

622 users solved this problem. Latest completion was about 13 hours ago.

Project: Tic-Tac-Toe with AI

Medium 9 minutes

Description

3 / 3 Prerequisites

✓ Inheritance Stage 4

✓ Magic methods Stage 4

✓ Inheritance Stage 4

Let's write a `"medium"` level difficulty. It's time to add awareness to our AI. Compared to randomly picking a cell to take a move, this level is considerably smarter.

Despite the randomness in the first moves, its level is a lot harder to beat. This level stops all simple attempts to beat it due to the second rule, and always wins when it can due to the first rule.

Let's take a look at these rules.

Objectives

The `"medium"` level difficulty makes a move using the following process:

1. If it can win in one move (if it has two in a row), it places a third to get three in a row and win.
2. If the opponent can win in one move, it plays the third itself to block the opponent to win.
3. Otherwise, it makes a random move.

You also should add `"medium"` parameter to be able to play against this level. And, of course, it should be possible to make "easy" vs "medium" matchup!

Example

The example below shows how your program should work.
The greater-than symbol followed by space (`> `) represents the user input. Notice that it's not the part of the input.

Example

The example below shows how your program should work.
The greater-than symbol followed by space (`> `) represents the user input. Notice that it's not the part of the input.

```
Input command: > start user medium
---------
|       |
|       |
|       |
---------
Enter the coordinates: > 2 2
---------
|       |
|   X   |
|       |
---------
Making move level "medium"
---------
|       |
|   X   |
| O     |
---------
Enter the coordinates: > 1 3
---------
| X     |
|   X   |
| O     |
---------
Making move level "medium"
---------
| X     |
|   X   |
| O   O |
---------
Enter the coordinates: > 2 1
---------
| X     |
|   X   |
| O X O |
---------
```

```
Making move level "medium"
---------
| X O   |
|   X   |
| O X O |
---------
Enter the coordinates: > 1 2
---------
| X O   |
| X X   |
| O X O |
---------
Making move level "medium"
---------
| X O   |
| X X O |
| O X O |
---------
Enter the coordinates: > 3 3
---------
| X O X |
| X X O |
| O X O |
---------
Draw

Input command: > start medium user
---------
|       |
|       |
|       |
---------
Making move level "medium"
---------
|       |
|       |
|   X   |
---------
Enter the coordinates: > 2 2
---------
|       |
|   O   |
|   X   |
---------
Making move level "medium"
---------
|       |
|   O   |
| X X   |
```

```
Enter the coordinates: > 3 1
---------
|       |
|   O   |
| X X O |
---------
Making move level "medium"
---------
| X     |
|   O   |
| X X O |
---------
Enter the coordinates: > 1 2
---------
| X     |
| O O   |
| X X O |
---------
Making move level "medium"
---------
| X     |
| O O X |
| X X O |
---------
Enter the coordinates: > 3 3
---------
| X   O |
| O O X |
| X X O |
---------
Making move level "medium"
---------
| X X O |
| O O X |
| X X O |
---------
Draw

Input command: > exit
```

Work on project. Stage 5/5: An undefeated champion

505 users solved this problem. Latest completion was about 1 hour ago.

Project: Tic-Tac-Toe with AI

Medium 10 minutes

Description

5 / 5 Prerequisites

- ✓ Computer algorithms — Stage 5
- ✓ Recursion basics — Stage 5
- ✓ Slicing — Stage 5
- ✓ Algorithms in Python — Stage 5
- ✓ Recursion in Python — Stage 5

Congratulations, you're at the finish line! After all, our task is to create an AI that will become a strong opponent.

Let's write the `"hard"` level difficulty.

Compared to the "medium" level difficulty, this level not just go one move ahead to see an immediate win or prevent an immediate loss. This level can see two moves ahead, three moves ahead and so on. Basically, it can see all possible outcomes till the end of the game and choose the best of them considering his opponent also would play perfectly. So, it doesn't rely on the blunders of the opponent, it plays perfectly regardless of the opponent's skill.

The algorithm that implements this is called Minimax. This is the brute force algorithm that maximizes the value of the own position and minimizes the value of the opponent's position. It's not only an algorithm for Tic-Tac-Toe, but for every game with two players with alternate move order, for example, chess.

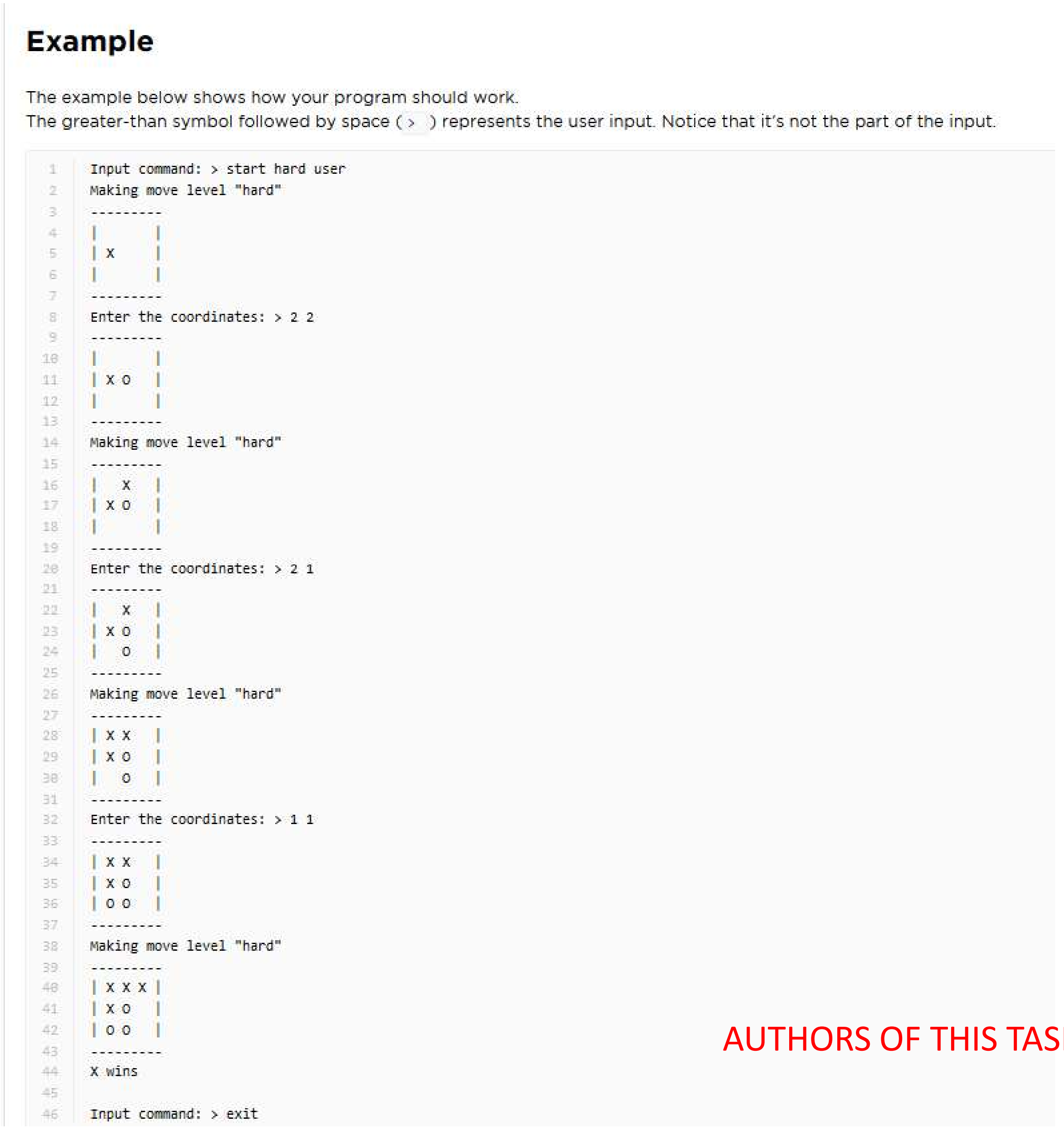
Objectives

In the last stage you need to implement Minimax algorithm as the "hard" difficulty level. This link will help to understand details.

You also should add `"hard"` parameter to be able to play against this level.

Example

The example below shows how your program should work.
The greater-than symbol followed by space (`> `) represents the user input. Notice that it's not the part of the input.



Example

The example below shows how your program should work.
The greater-than symbol followed by space (`> `) represents the user input. Notice that it's not the part of the input.

```
Input command: > start hard user
Making move level "hard"
---------
|       |
| X     |
|       |
---------
Enter the coordinates: > 2 2
---------
|       |
| X O   |
|       |
---------
Making move level "hard"
---------
|   X   |
| X O   |
|       |
---------
Enter the coordinates: > 2 1
---------
|   X   |
| X O   |
|   O   |
---------
Making move level "hard"
---------
| X X   |
| X O   |
|   O   |
---------
Enter the coordinates: > 1 1
---------
| X X   |
| X O   |
| O O   |
---------
Making move level "hard"
---------
| X X X |
| X O   |
| O O   |
---------
X wins

Input command: > exit
```

AUTHORS OF THIS TASK: https://hyperskill.org/