Advanced Software Engineering (Lab)

Stefano Forti

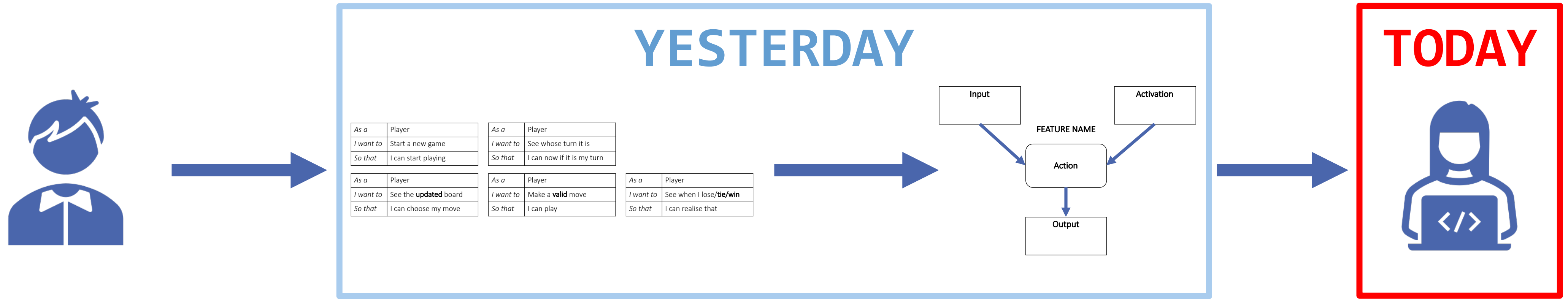
`name.surname@di.unipi.it`

Department of Computer Science, University of Pisa

What will I do?

- User story prototyping with Python
- GitHub flow (commits, branches, pull requests, merge)

Roadmap



As a	Player
I want to	Start a new game
So that	I can start playing

As a	Player
I want to	See whose turn it is
So that	I can now if it is my turn

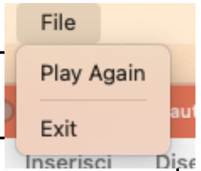
As a	Player
I want to	See the **updated** board
So that	I can choose my move

As a	Player
I want to	Make a **valid** move
So that	I can play

As a	Player
I want to	See when I lose/**tie/win**
So that	I can realise that

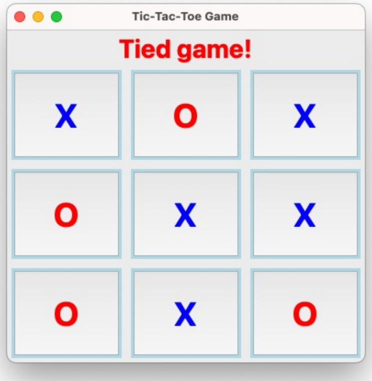
- Download skeleton from Moodle, you will find:
  - a `game.py` file where to implement four missing features
  - a `board.py` file containing the GUI business logic (not to be changed)
  - a `main.py` file to launch the application `python3 main.py`

User stories



As a	Player
I want to	Start a new game
So that	I can start playing

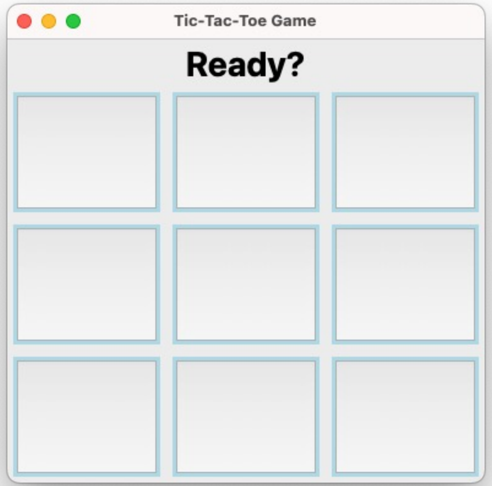
As a	Player
I want to	See whose turn it is
So that	I can now if it is my turn

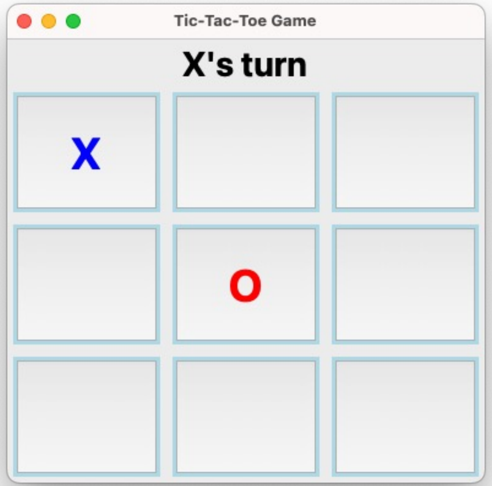


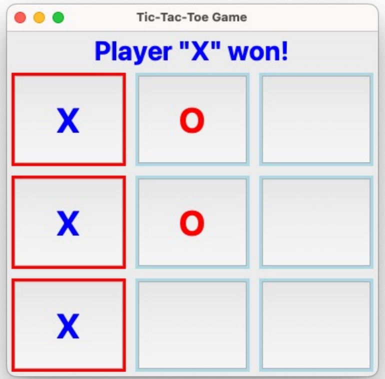
As a	Player
I want to	See the **updated** board
So that	I can choose my move

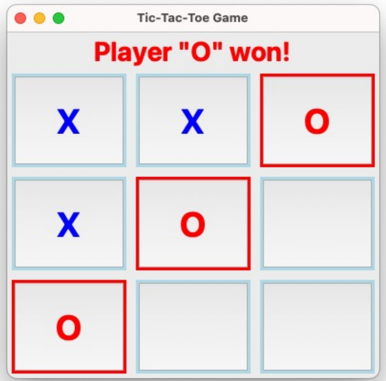
As a	Player
I want to	Make a **valid** move
So that	I can play

As a	Player
I want to	See when I lose/**tie/win**
So that	I can realise that

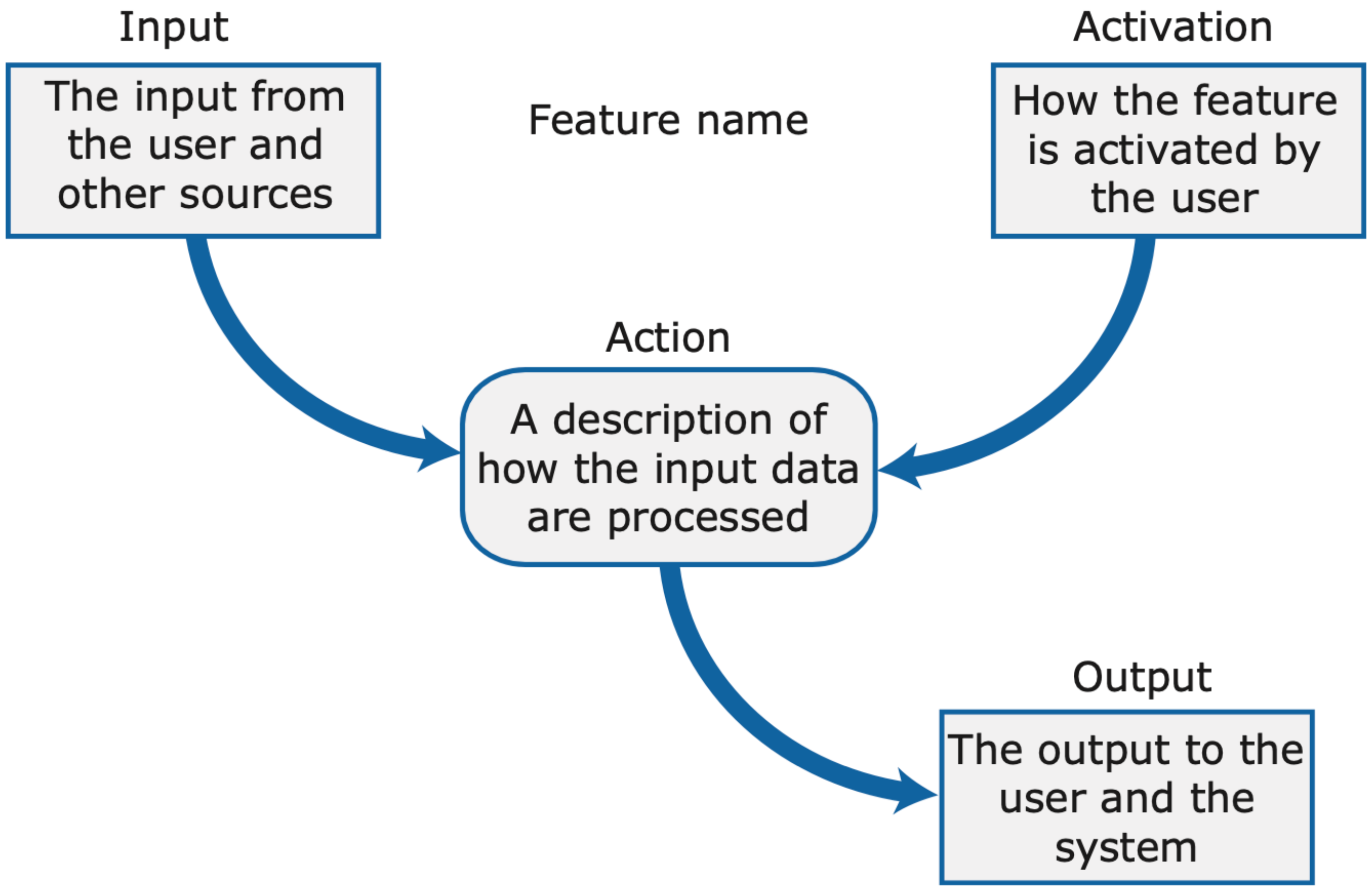




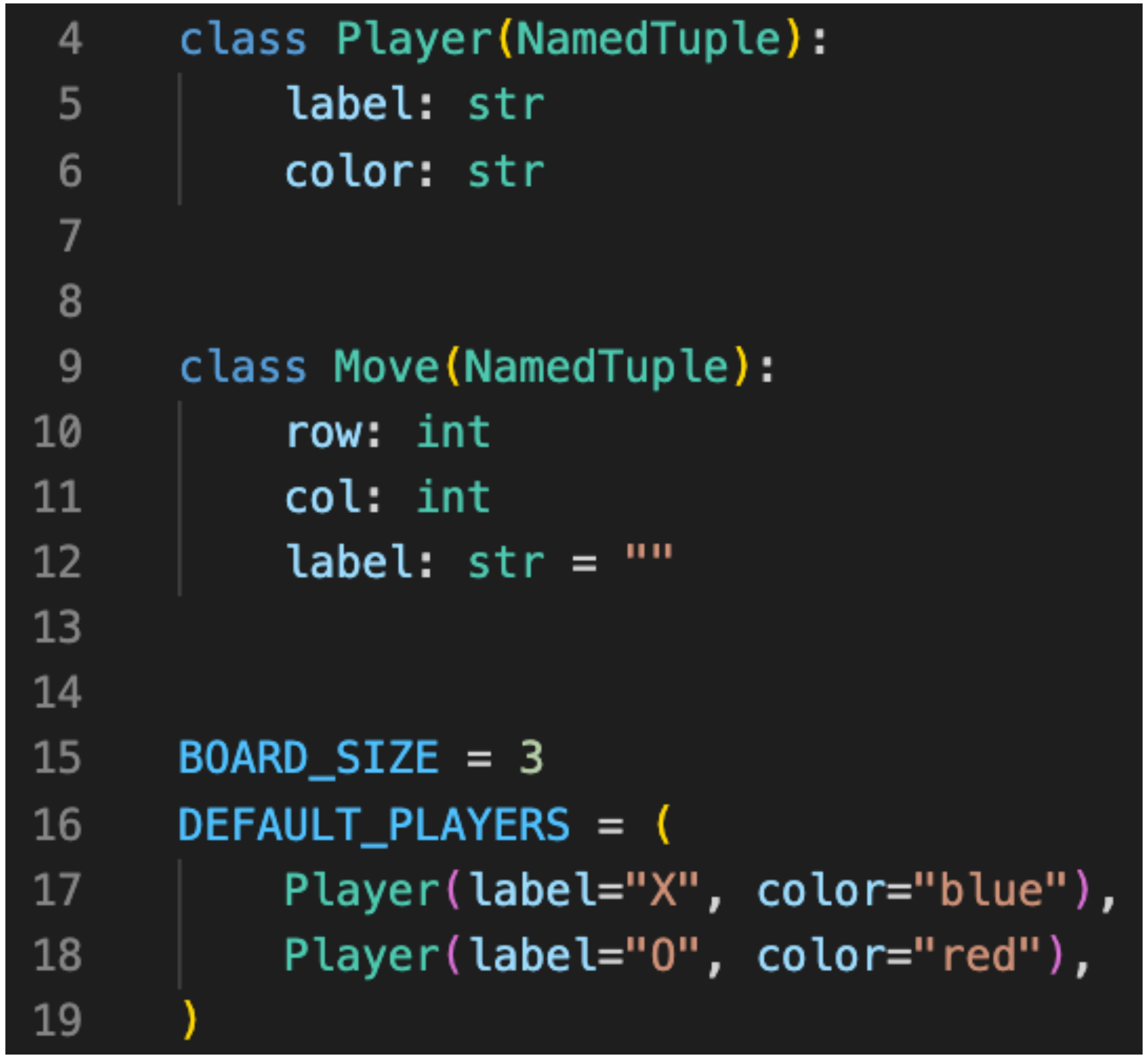




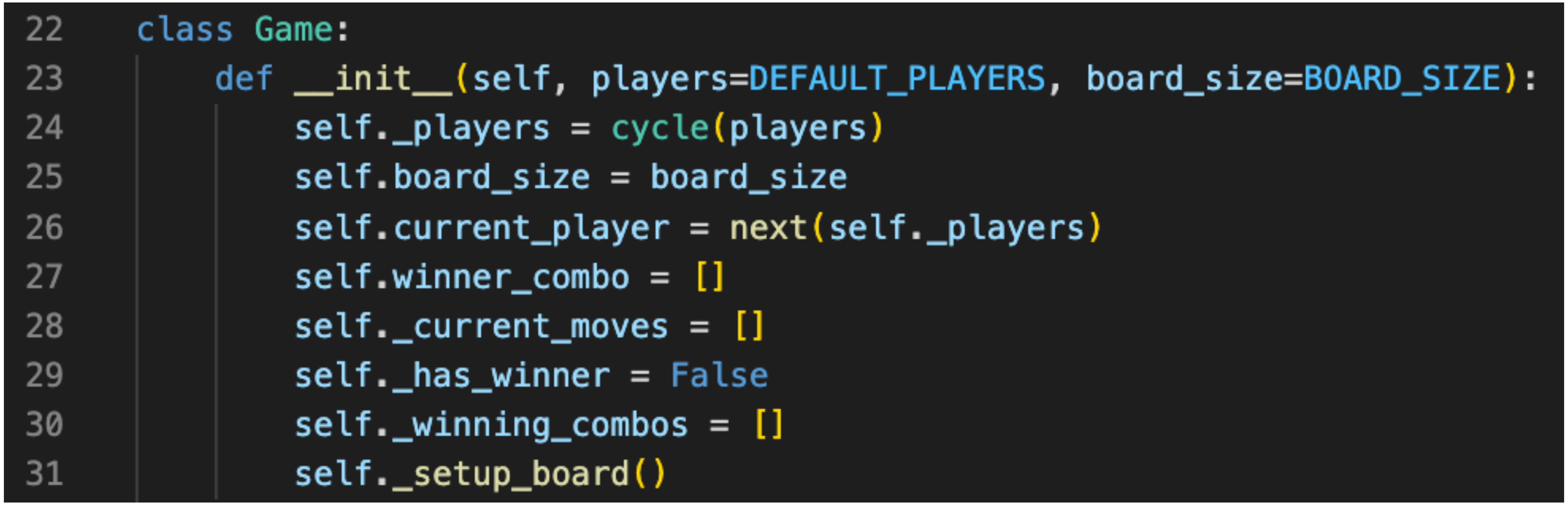
Features


Input
The input from the user and other sources
Feature name
Activation
How the feature is activated by the user
Action
A description of how the input data are processed
Output
The output to the user and the system

Skeleton walkthrough

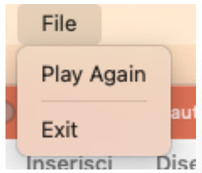


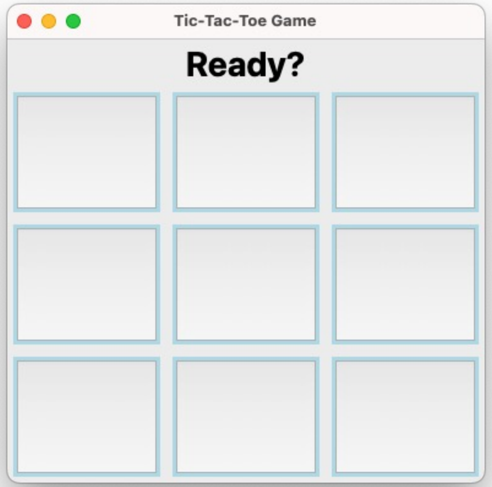
```python
class Player(NamedTuple):
    label: str
    color: str


class Move(NamedTuple):
    row: int
    col: int
    label: str = ""


BOARD_SIZE = 3
DEFAULT_PLAYERS = (
    Player(label="X", color="blue"),
    Player(label="O", color="red"),
)
```



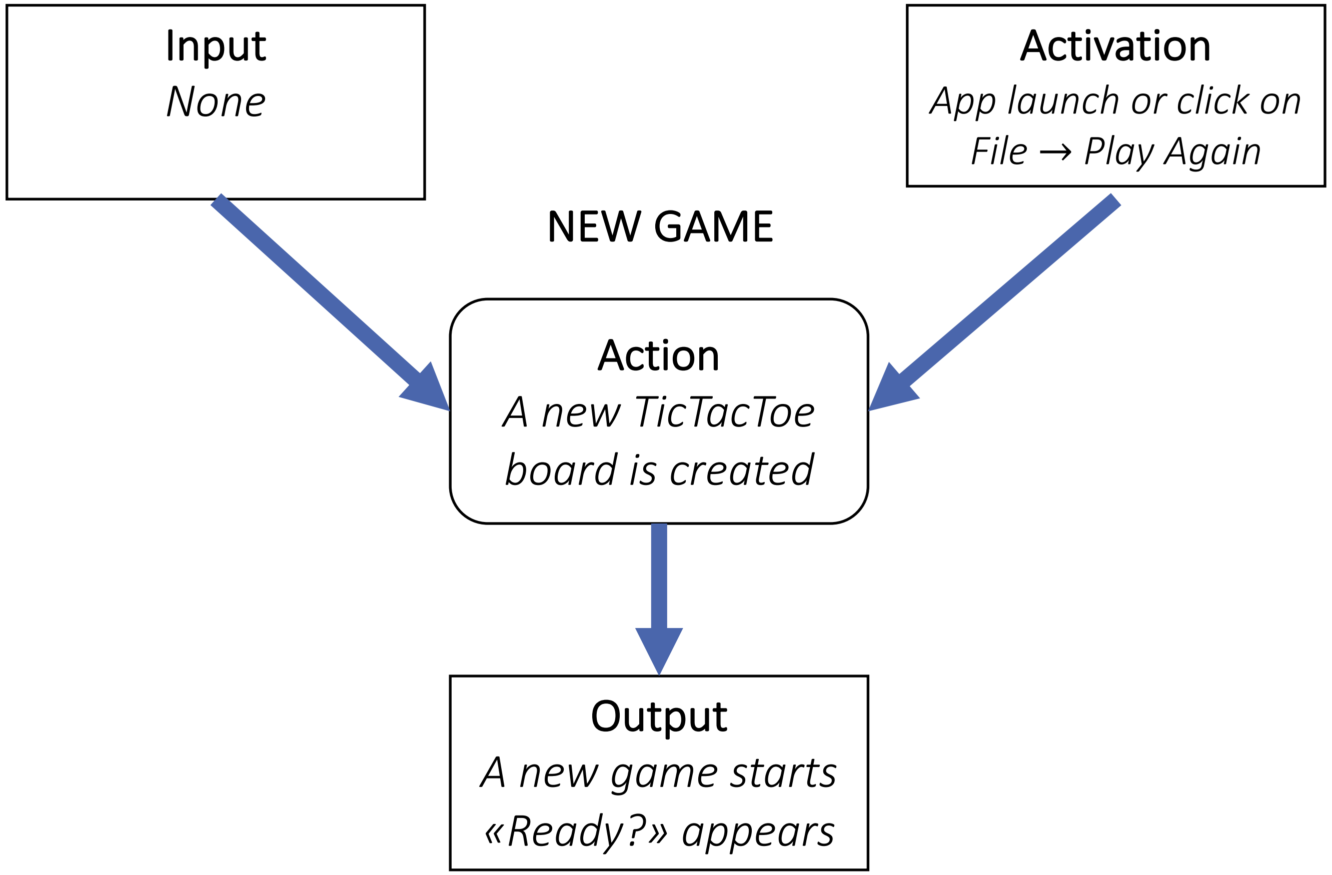
```python
class Game:
    def __init__(self, players=DEFAULT_PLAYERS, board_size=BOARD_SIZE):
        self._players = cycle(players)
        self.board_size = board_size
        self.current_player = next(self._players)
        self.winner_combo = []
        self._current_moves = []
        self._has_winner = False
        self._winning_combos = []
        self._setup_board()
```

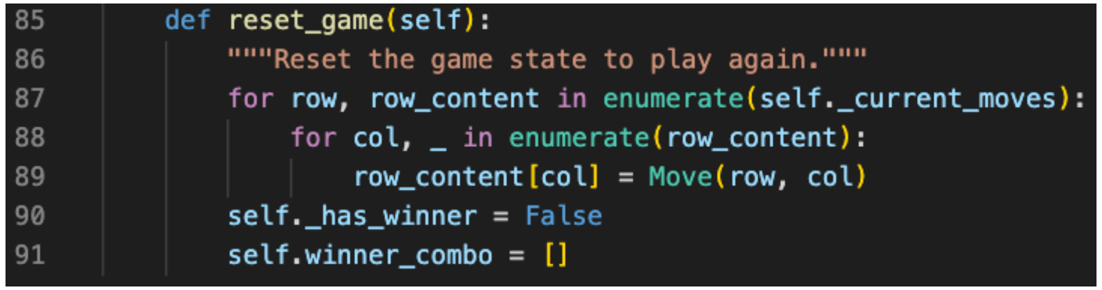
As a	Player
I want to	Start a new game
So that	I can start playing





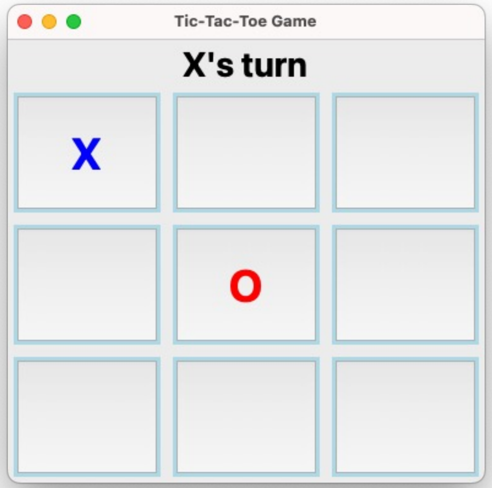
NEW GAME

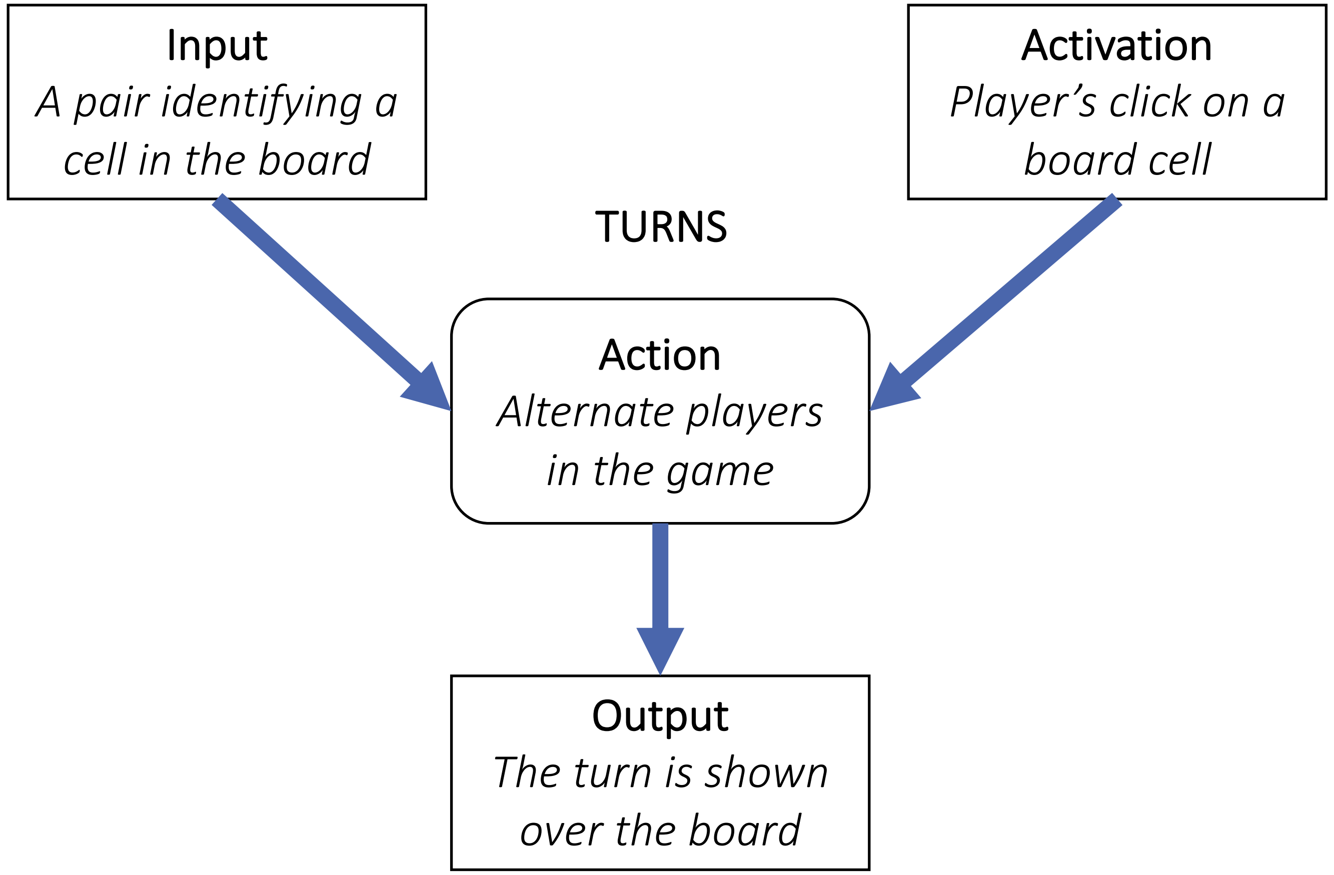

Input
None
Activation
App launch or click on File → Play Again
Action
A new TicTacToe board is created
Output
A new game starts «Ready?» appears

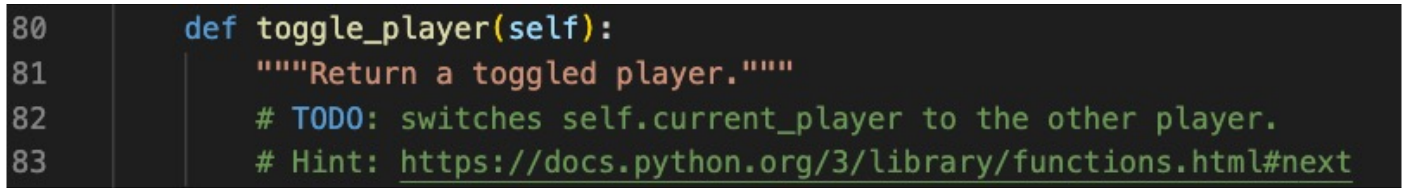


```python
    def reset_game(self):
        """Reset the game state to play again."""
        for row, row_content in enumerate(self._current_moves):
            for col, _ in enumerate(row_content):
                row_content[col] = Move(row, col)
        self._has_winner = False
        self.winner_combo = []
```

As a	Player
I want to	See whose turn it is
So that	I can now if it is my turn



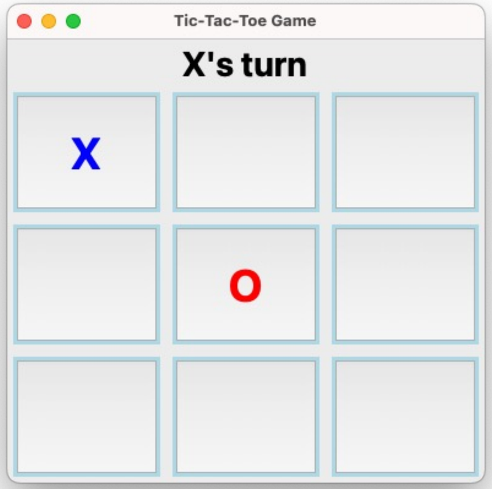

Input
A pair identifying a cell in the board
Activation
Player's click on a board cell
TURNS
Action
Alternate players in the game
Output
The turn is shown over the board

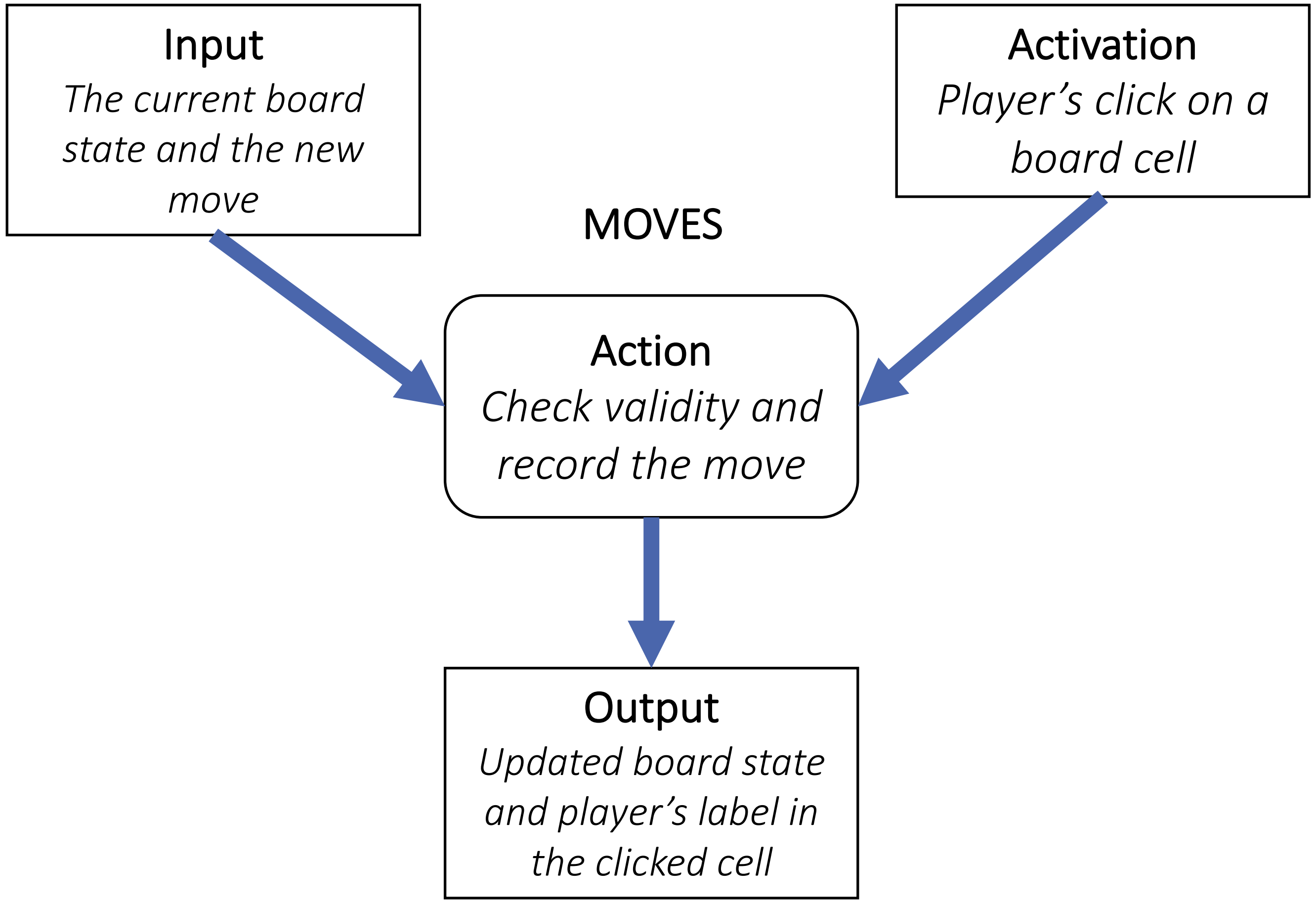


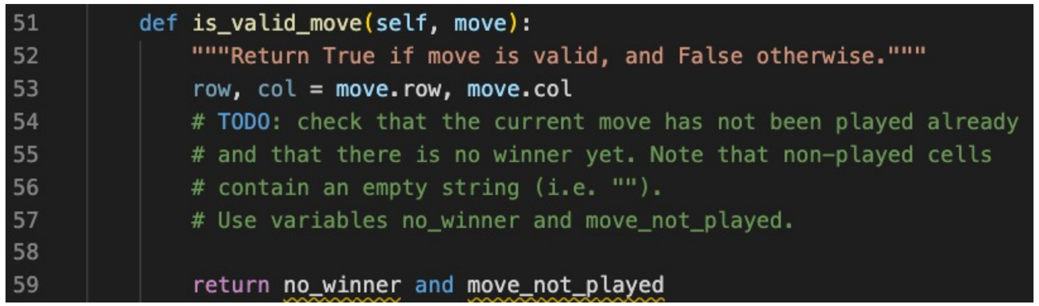
```
    def toggle_player(self):
        """Return a toggled player."""
        # TODO: switches self.current_player to the other player.
        # Hint: https://docs.python.org/3/library/functions.html#next
```

As a	Player
I want to	See the **updated** board
So that	I can choose my move

As a	Player
I want to	Make a **valid** move
So that	I can play



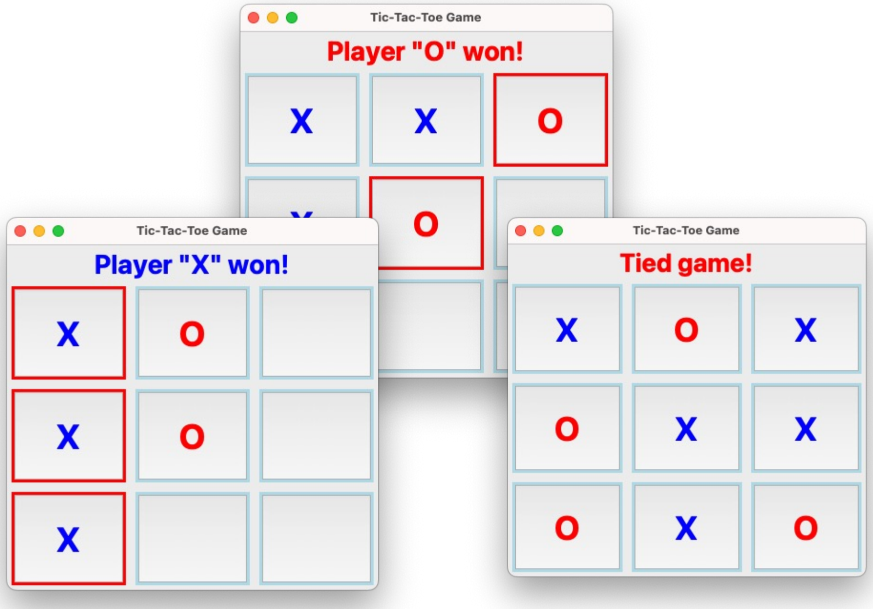

Input
The current board state and the new move
Activation
Player's click on a board cell
MOVES
Action
Check validity and record the move
Output
Updated board state and player's label in the clicked cell

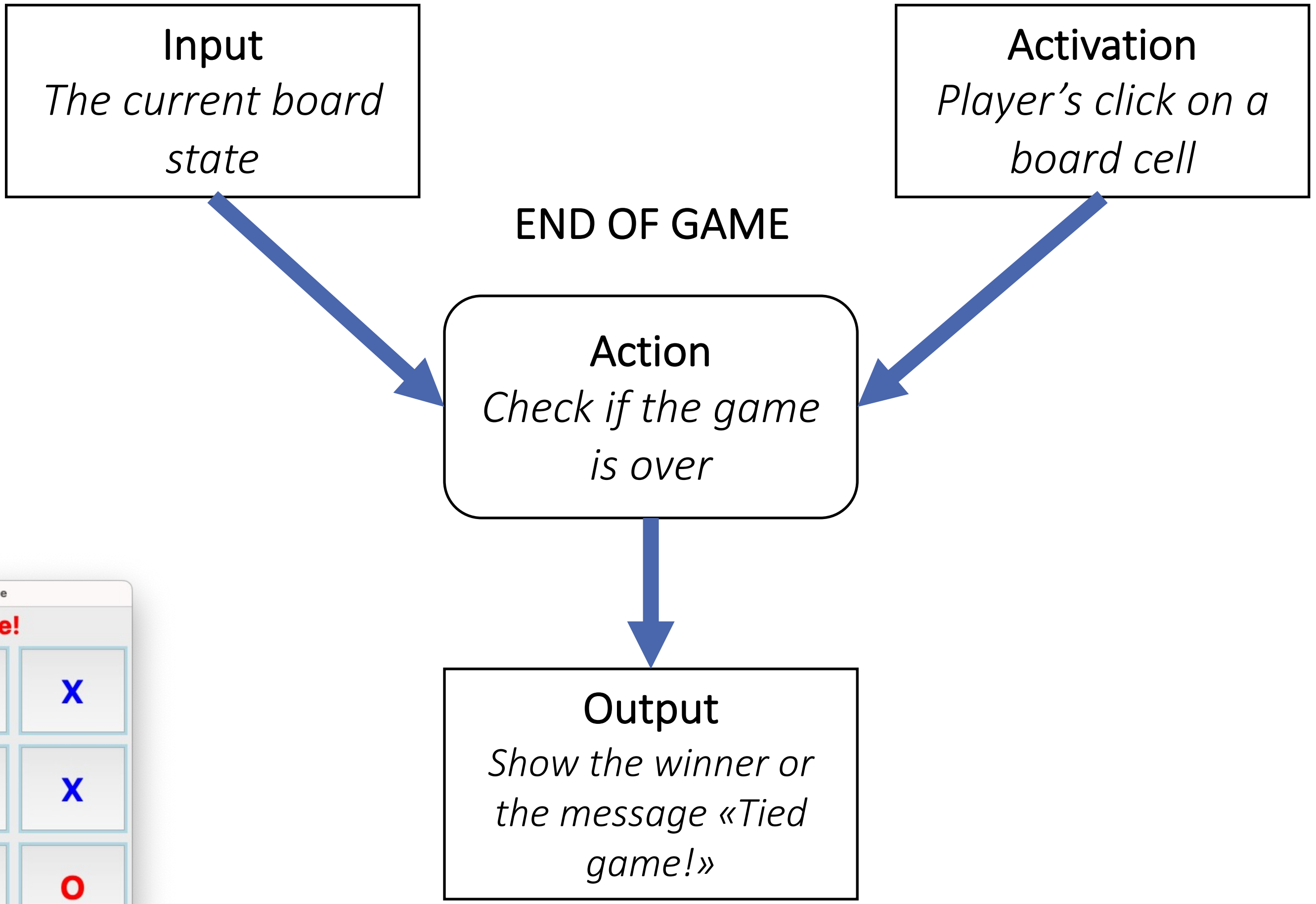


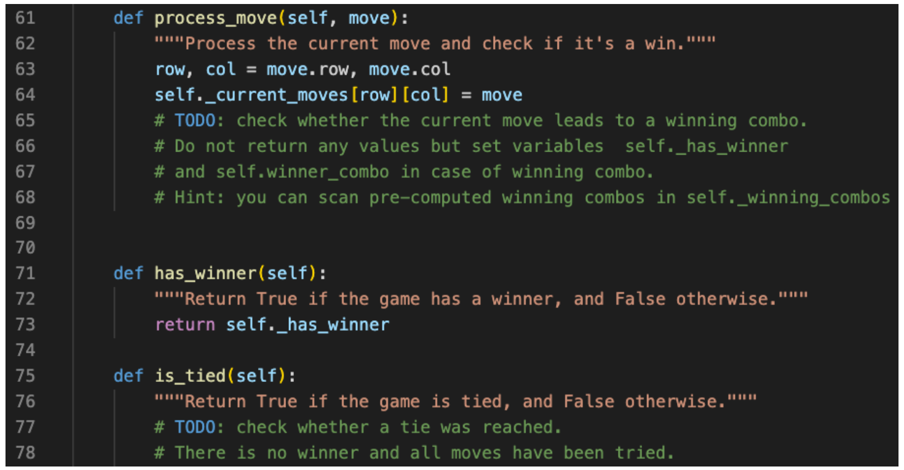
```
    def is_valid_move(self, move):
        """Return True if move is valid, and False otherwise."""
        row, col = move.row, move.col
        # TODO: check that the current move has not been played already
        # and that there is no winner yet. Note that non-played cells
        # contain an empty string (i.e. "").
        # Use variables no_winner and move_not_played.

        return no_winner and move_not_played
```

END OF GAME

As a	Player
I want to	See when I lose/**tie/win**
So that	I can realise that




Input
The current board state
Activation
Player's click on a board cell
Action
Check if the game is over
Output
Show the winner or the message «Tied game!»



```python
    def process_move(self, move):
        """Process the current move and check if it's a win."""
        row, col = move.row, move.col
        self._current_moves[row][col] = move
        # TODO: check whether the current move leads to a winning combo.
        # Do not return any values but set variables  self._has_winner
        # and self.winner_combo in case of winning combo.
        # Hint: you can scan pre-computed winning combos in self._winning_combos


    def has_winner(self):
        """Return True if the game has a winner, and False otherwise."""
        return self._has_winner

    def is_tied(self):
        """Return True if the game is tied, and False otherwise."""
        # TODO: check whether a tie was reached.
        # There is no winner and all moves have been tried.
```

Create a Repo

- Go to github.com and enter with your credentials.
- Repositories are the place where your projects live.
- In the upper-right corner of any page, click **+** and then **New Repository**.
- Type a short, memorable name, e.g. `ase-fall-22`.
- This repo will be **Public**.
- Initialise it with a **README**.
- Click **Create a repository**.

Clone your Repo

- Open GitBash (or bash).
- Move to the directory where you want to store your work (e.g., `ASE`).
- Use the command `git clone [your repo url]`.
- Create a new folder named `Lab_1` and move there the `TicTacToe` project folder. Then:

```
git add *
git commit -m "first commit"
git push
```

- Add your team mates as repository collaborators

Manage branches

- Create a new branch

```
git branch [name_of_your_new_branch]
```

- Move to the branch on your local machine

```
git checkout [name_of_your_new_branch]
```

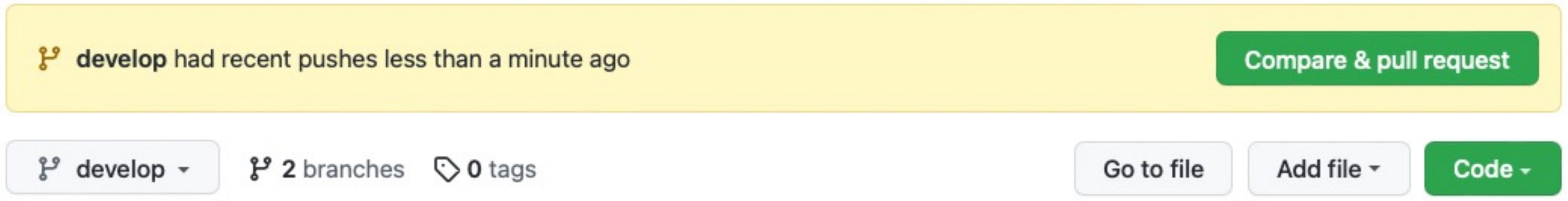
- Push it on GitHub

```
git push origin [name_of_your_new_branch]
```

- List all branches with

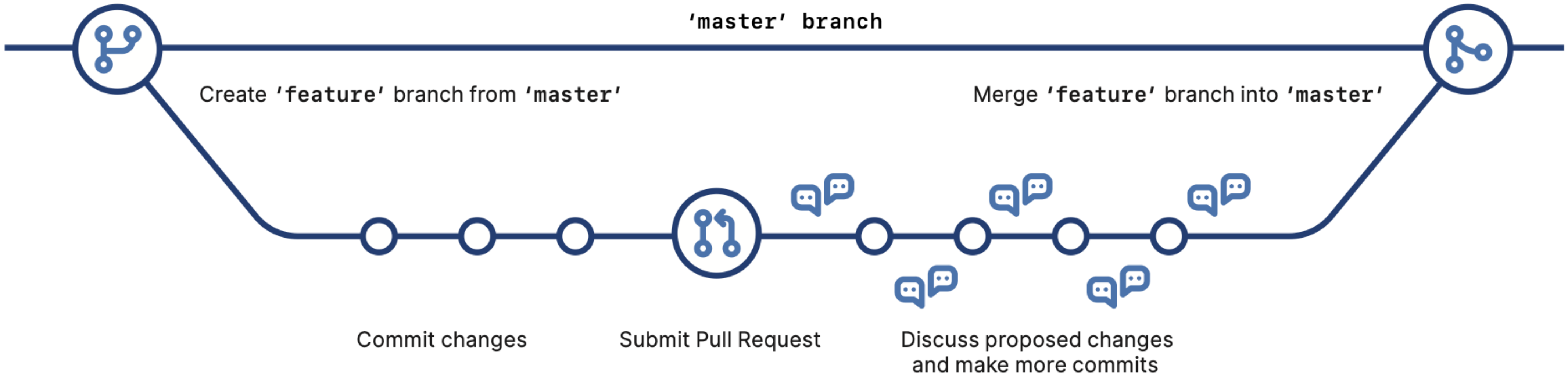
```
git branch -a
```

- Submit pull requests and merge branches from the web interface



develop had recent pushes less than a minute ago

GitHub Flow


'master' branch
Create 'feature' branch from 'master'
Merge 'feature' branch into 'master'
Commit changes
Submit Pull Request
Discuss proposed changes and make more commits

https://guides.github.com/introduction/flow/

Lab activity

- Split into groups of four.
- Complete all TODOs in the `game.py` file without changing the interface
- You have to implement 4 functions. Split them across team members and implement one each in 4 different branches of the shared repo
- Merge all branches into a single main branch.

- Solution will be posted on Moodle later.