
Lab 02 — Tic-Tac-Toe

Overview

In this lab, you'll be implementing the core functions needed to play Tic-Tac-Toe. The game starts by running `tictactoe.py`, which calls `play_game.py`. `play_game.py` then calls other helper functions that **you** will implement.

1. Files Involved

Provided Files

These files are **already complete** and **should not be modified**:

- `tictactoe.py` → The main program that starts the game.
- `play_game.py` → Handles the main game loop.

Even though you won't change these, **read through them carefully** to understand how they work.

Functions and Their Files

`play_game.py` calls these functions:

Function	Defined In	Purpose
<code>ai_move</code>	<code>ai_move.py</code>	Chooses the computer's move.
<code>calc_score</code>	<code>calc_score.py</code>	Calculates who won, lost, or tied.
<code>print_board</code>	<code>display_board.py</code>	Displays the current game board.
<code>game_over</code>	<code>game_over.py</code>	Checks if the game is finished.
<code>player_move</code>	<code>player_move.py</code>	Gets the player's move.
<code>play_game</code>	<code>play_game.py</code>	Runs the overall game logic.

2. Your Work: `student_code/` Folder

Inside the `student_code/` folder, you'll find files named like this:

```
qai_move.py
qcalc_score.py
qdisplay_board.py
qgame_over.py
qplayer_move.py
```

- These are **starter files** that contain incomplete versions of the functions.
 - The "q" at the beginning of the filename means “**question**” — the code isn’t finished yet.
 - Your job is to **edit these files one at a time** and **complete the missing code**.
-

3. How the Program Chooses Which Code to Run

- When you run `tictactoe.py`, it looks **first** in the `student_code/` folder for your implementation.
 - If it **doesn’t** find a completed function there, it automatically uses the prebuilt version from the `default_code/` folder.
 - The `default_code/` folder contains **.pyc files**:
 - .pyc files are **compiled Python files**.
 - You **cannot read or edit them**.
-

4. Step-by-Step Instructions

Step 1 — Run the Game First

1. Open **Visual Studio Code**.
 2. Make sure you can see the **LAB02/** folder.
 3. Run `tictactoe.py`:
 4. `python tictactoe.py`
 5. At first, the game will **work perfectly** because it uses the prebuilt `.pyc` files.
-

Step 2 — Implement Functions One at a Time

1. Open one of the files in `student_code/` — for example, `qai_move.py`.
2. Fill in the missing code.
3. **Save the file**.
4. **Rename the file** to remove the leading “q”.
Example:
5. `qai_move.py` → `ai_move.py`
6. Run `tictactoe.py` again:

- Now the program will **use your version** of `ai_move()` instead of the compiled one.
7. Test thoroughly before moving on.
-

Step 3 — Repeat for Each File

- Work on the functions **one at a time**:
 1. `qplayer_move.py`
 2. `qdisplay_board.py`
 3. `qcalc_score.py`
 4. `qgame_over.py`
 5. `qai_move.py`
 - After finishing each one:
 - Rename the file (remove the `q`).
 - Run `tictactoe.py` to test it.
 - Fix any issues before moving on.
-

Step 4 — Verify Everything Works

When you've implemented all your functions, the game will run **entirely on your code** (no compiled `.pyc` files needed).

5. Important Notes

- **Do not rename any functions** — their names must exactly match the table above.
 - **Do not change** `tictactoe.py` or `play_game.py`.
 - Always test your changes by running `tictactoe.py` after updating each file.
 - Start simple, test often.
-

6. Visual Studio Code Tips

- The **LAB02/** folder should look like this:

```
LAB02/
├── default_code/      # Compiled reference code (.pyc files)
├── student_code/      # Your work goes here
│   ├── qai_move.py
│   └── qcalc_score.py
```

```
├── ...
└── tictactoe.py      # Main game script
    play_game.py    # Game loop (provided)
```

- **tictactoe.py** always prefers your `student_code/` functions first.
 - If a function isn't there yet, it falls back to the prebuilt compiled code.
-

7. What You'll Learn

- How Python uses **imports** across multiple files.
 - How to structure programs into **modules**.
 - How to test and debug functions one at a time.
 - How to gradually replace prebuilt functionality with your own.
-

Questions for You Before Finalizing

1. Do you want students to **start with a specific function first**, or can they choose any order?
2. Should I include an **example Tic-Tac-Toe game session** in the instructions to make it more concrete?
3. Do you want me to add a **grading rubric** so students know what's expected?

If you answer those, I can make this even more polished and student-friendly.

Should I also add a **diagram showing how tictactoe.py calls play_game.py and then calls the functions**? That might really help visual learners.
Do you want me to?