# Overview

Tic-Tac-Toe Game Refactoring

## Objective

Refactor a given monolithic tic-tac-toe game, such that the code:

1. Is modular, consisting of at least two files that logically group related functions.
2. Implements an appropriate Python project folder structure.
3. Includes at least one test case.
4. Employs a 2D data structure.

In the process, you must use at least four functions, two classes, two files, and one import statement of your modules (not including imports used in the test case).

## Instructions

Complete each of the following steps carefully reviewing what is expected.

### Step 1: Review the Existing Code

Firstly, analyse the given tic-tac-toe game code. Understand the flow and functionality before proceeding with the refactoring.

### Step 2: Identify Components to Refactor

Identify the parts of the code that can be improved. Determine which parts of the code can be grouped *logically* into separate modules.

### Step 3: Modularising the Code

Using staged commits, refactor the code to create at least **two** files. These files should contain logically grouped functions and/or classes. Ensure the file names are appropriate for the division you have chosen.

### Step 4: Create a Modern Python Folder Structure

The refactored code should adhere to the following Python folder structure:

tic\_tac\_toe/

|--- src/

| |--- \_\_init\_\_.py

| |--- module1.py

| |--- module2.py

|--- tests/

| |--- \_\_init\_\_.py

| |--- test\_tic\_tac\_toe.py

|--- setup.py

**Note**:

* \_\_init\_\_.py files are used to indicate that a directory should be treated as a Python package. This allows the files within to be imported as a module in the test scripts or other python files.
* setup.py is a Python file used to specify what modules and dependencies must be installed. The file has been provided for you, along with instructions on how to install your modules using this file. If PyCharm prompts you to run the setup.py – cancel – it will not work until you restructure your project!
* You must give your Python files appropriate names. Do **not** use module1, 2, etc.

### Step 5: Create a Test Case

Develop at least one test case for your binary search code. The test case should reside in the 'tests' directory. It is preferred that you use unittest framework for writing your test case.

### Step 6: Implement 2D Data Structure

Refactor the code such that it employs a 2D data structure for the tic-tac-toe game board.

### Step 7: Written Report

Once you have completed your refactoring, write a brief report addressing the following:

1. Justification for your refactoring decisions.
2. The challenges you would have faced maintaining and testing the original monolithic code.

How you would modify your refactored code to handle a customised tic-tac-toe game (larger than 3x3), and how this implementation would be easier to handle than in the original code.

<<Space for your answer>>

Justification

Challenges

Approach to adding support for larger boards

### Step 8: Short Answer (Knowledge Questions)

Provide brief answers to the knowledge-question worksheet.

Briefly explain what modular programming is. How can you import only a specific function or class from a module in Python? What is the syntax for this?

Describe your approach to debugging the tests you created tests in this task. Describe the challenges and include IDE screenshots of you debugging your tests.

How would you explain Python's parameter-passing mechanism? Is it more like pass-by-value or pass-by-reference? Justify your answer.

Given the following Python code, what will be the output and why?

def modify\_list(list\_):

list\_.append("new")

list\_ = ["completely", "new"]

items = ["original"]

modify\_list(items)

print(items)

# Output:

# Explanation:

In Python, even though variables created within a function are local, there are still situations where you can modify data outside the scope with a local variable. Explain this anomaly and relate it to both mutability and pass by reference.

List two benefits of modular coding approaches. How do these benefits assist in the development of medium-sized applications?

### Evaluation Criteria

Your refactoring will be evaluated on the clarity and modularity of your code, as well as the thoughtful reasoning behind your design decisions. Your test case should be robust and cover key aspects of the tic-tac-toe game functionality. The written report should accurately reflect your understanding of code refactoring, testing, and the flexibility of your new implementation.