# Comparison

Three different programming languages with three paradigms provide three solutions to build a tic-tac-toe game. Python uses an Object-Oriented (OO) paradigm. It is a powerful language well-known for being easy to understand and use. Classes like Board, Player, and Game are created in implementation, encapsulating data and behaviour. Enhances modularity and reusability in object-oriented design in Python. The procedure paradigm is effectiveness and low-level control. The procedural paradigm used to implement tic-tac-toe in C the code revolves around a clear sequence of steps. Functions manage aspects such as displaying the board, checking for a win, and player turns. This approach is in line with C's emphasis on explicit memory management and step-by-step execution. For a functional paradigm, Haskell is a good option. Haskell is focused on the absence of side effects and immutability. In tic-tac-toe, Haskell's functional approach, mutable states are avoided by specifying the game state transitions through a sequence of pure functions.

## Python (OO) vs C (Procedural):

Both Python (OO) and C (Procedural) solutions are commonly focused on clarity and directness. They both use functions or methods to encapsulate specific functionalities, such as checking for valid player moves, checking for a win or draw, displaying the game board etc. However, Python’s OO approach can provide a more natural organisation of code in the classes for reusability and modularity. C’s procedural approach relies on the procedural step so that needs more effort to achieve a similar level of abstraction.

C’s procedural solution uses step-by-step execution to execute the code by following a set sequence of instructions. The program will execute instructions from the first statement in the main function and proceed sequentially through subsequent statements. There is less emphasis on the relationships between data and functions. On the other hand, Python’s OO solution has a strong emphasis on the relationships between data and functions based on the object-oriented core design principles.

Python solution has classes and object features that allow the code design to promote a hierarchical structure. This can foster a more natural portrayal of objects found in the real world, like the game board and player. C solution, on the other hand, uses top-down design to design code, so the structure of the code will tend to code sequentially. So, this will potentially lead to a flatter architecture. Although Python's use of classes may lead to a more elegant and extensible solution, it may also introduce a level of complexity that is avoided by C's procedural approach.

## Python (OO) vs Haskell (Functional):

Python (OO) and Haskell (Functional) solutions have a strong focus on readability and expressive syntax. These two solutions encourage breaking down the problem into smaller, more manageable pieces. Python solution can use the classes and objects to create a class for each module to separate the code into smaller pieces. However, Haskell solution can use the composition pure function to achieve modularity. So, both solutions support a modular and organized approach to software design.

Haskell's functional solution applies recursion for repetitive game rounds, as functional programming languages encourage immutability and recursion as fundamental concepts. However, Python's OO solution uses loops as the primary choice for repetitive game rounds, even though Python supports recursion. Notwithstanding using different features to repeat game rounds, both solutions can achieve the same result. Additionally, Haskell's solution applied guard for expressing conditions and pattern matching, while the Python solution relies on the conditions statements for making decisions and expressing conditions.

Python OO solution uses flexibility, but the Haskell functional solution applies pure functionality and immutability. Python solution can use class to provide the mutable state to allow the program to make dynamic changes. However, Haskell's functional approach relies on an immutable and declarative style. For example, in the Haskell functional solution of tic-tac-toe code has a “makeMove” function. The function does not update the existing board but will create another board with the update to achieve the update effect. In comparison, the Python solution can use the existing board to update straightway.

## C (Procedural) vs Haskell (Functional):

In these two solutions, C (Procedural) and Haskell (Functional) are committed to simplicity and a transparent sequence of steps in program execution. Both paradigms foster clarity in code understanding and promote a logical flow of the programme. Fundamentally, these solutions use a structured approach to problem-solving. Additionally, both solutions also depend on functions. Haskell’s functional solution involves higher-order functions. There still will be some differences between Haskell and C solutions.

In comparing the tic-tac-toe solutions in Haskell and C, the common thread lies in their utilization of functions. Haskell's functional style involves higher-order functions and abstract concepts. C style uses low-level control over the memory and the hardware resources. However, Haskell's version of tic-tac-toe is more expressive and concise in the code. The C version of tic-tac-toe is more manual optimization efforts for efficiency.

The solution within the paradigm of C, the program has used the mutable state and global variables, facilitating the direct of dynamic program state change. In contrast, the solution within the functional of Haskell is a commitment to immutability, so the variable used in Haskell cannot be changed. However, it makes the code more complex than the C version of a solution. For example, In the C solution, we can update the existing game board straight away, in the Haskell solution, we can't update the existing board because of the immutability of the Haskell solution, so instead of updating the existing game board, the program will create another game board to achieve the update effect.

# Conclusion

In comparing Python (OO), C (Procedural), and Haskell (Functional) for implementing Tic-Tac-Toe, nuances appear in code organization, data handling, and the way to design. Python's OO paradigm offers clarity and modularity, C's Procedural paradigm provides simplicity of code organization and straightforwardness in a logical flow, and Haskell's Functional paradigm emphasizes immutability and type safety. Each of the languages and paradigms has brought strengths to solving the tac-tac-toe game, so depending on what the project needs, desired trade-offs between simplicity and expressiveness. Also, these three languages and paradigms have shown the diverse possibilities in programming paradigms.