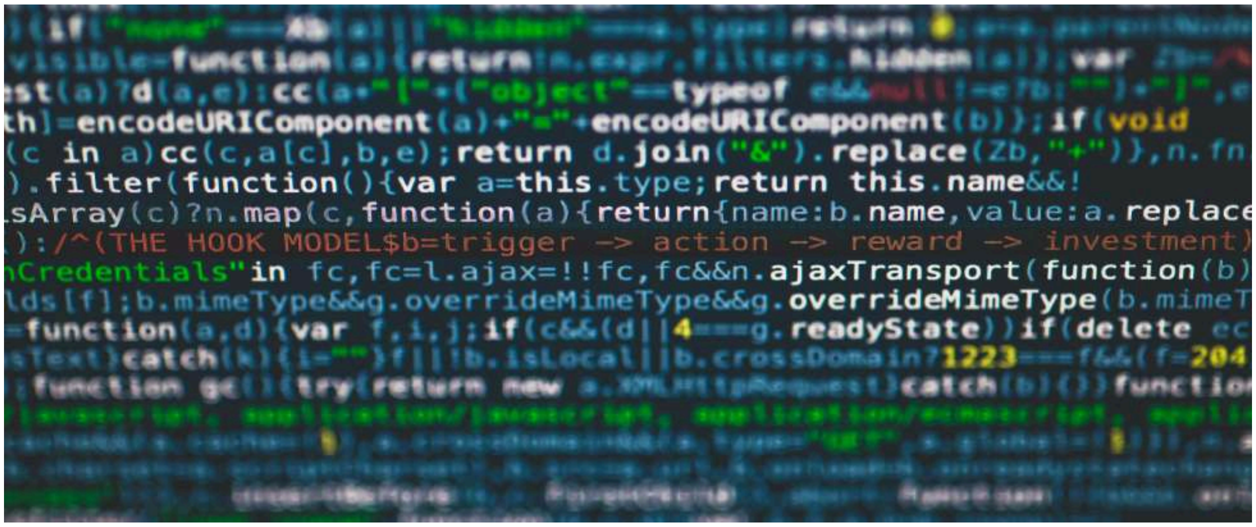

C# Scripting

🔢 Simple Logic Projects

1. **Even or Odd Checker**
    - Ask the user for a number and print whether it's even or odd.
    - Great for practicing modulus operator and conditionals.
2. **Palindrome Checker**
    - Check if a word or sentence is a palindrome (reads the same backward).
    - Helps you practice string manipulation.
3. **Fibonacci Series Generator**
    - Print the first N Fibonacci numbers.
    - Good for learning recursion or loops.
4. **Factorial Calculator**
    - Take a number and calculate its factorial.
    - Great for understanding recursion or loops.

⏱️ Time & Date-Based Projects

1. **Digital Clock (Console Version)**

- Print the current time updating every second.
- Learn how to use `DateTime` and loops.

2. **Age Calculator**
    - Ask for a birthdate and calculate the current age.
    - Practice with `DateTime` and time span differences.

🎨 With Windows Forms (Optional Visual Projects)

1. **Simple Paint App**

- Click and drag the mouse to draw.
- Great for learning event handling.

2. **Random Background Color Changer**

- Press a button and change the form's background color randomly.
- Learn about random numbers and UI elements.

🚀 After completing simple projects: Phase 1: Core C# & Classic Console Games

Purpose: Cement language fundamentals, algorithms, data structures, patterns, I/O, and async—through fun mini-games.

1. **Snake (Console)**
    - **What:** Classic snake movement on a grid, grow on food, game-over on self-collision.
    - **Skills:** 2D arrays, `ConsoleKeyInfo`, game loops, collision detection, basic pathfinding.
2. **Tetris (Console)**
    - **What:** Falling tetrominoes with rotation, line clear, score tracking.
    - **Skills:** Matrix transforms, timers (`Thread.Sleep`), event-driven input, high-score persistence.
3. **Hangman & Word Jumble**

- **What:** Guess letters/unscramble words from a dictionary file.
- **Skills:** File I/O, string manipulation, `List<T>`, exception handling.

4. **Battleship (Console vs. AI)**
   - **What:** Place ships on a grid, take turns guessing; AI with random/heuristic moves.
   - **Skills:** Object modeling, simple AI, `Dictionary` for board state, unit tests for hit/miss logic.
5. **Multiplayer Chat (Console)**
   - **What:** Two-player chat over localhost sockets.
   - **Skills:** `TcpListener` / `TcpClient`, threading, async/await, basic networking.
6. **Maze Generator & Solver**
   - **What:** Randomly generate maze (DFS/Prim) and auto-solve with BFS/A*.
   - **Skills:** Graphs, recursion, queues, priority queues, performance profiling.

🔧 Phase 2: .NET Desktop & Web Apps

Purpose: Broaden into GUI, patterns, data persistence, services—skills used in enterprise and game-dev tools.

1. **WPF MVVM "Crafting Planner"**
   - Define recipes, adjust stock, preview craftable items with data binding.
   - **Skills:** MVVM, `ICommand`, `INotifyPropertyChanged`, `ObservableCollection`.
2. **WinForms "Mini Arcade Launcher"**
   - Host your console games in a single UI with start/stop buttons and score displays.
   - **Skills:** WinForms events, process management, embedding console windows.
3. **ASP.NET Core "Inventory & Leaderboard" API**

    - REST endpoints for inventory, recipes, and high-scores; JWT authentication, Swagger.
    - **Skills:** Controllers, DI, EF Core, middleware, unit & integration tests.
4. **Blazor WebAssembly "Game Dashboard"**
    - Browser UI to view stats from your API: resource charts, top scorers, live updates via SignalR.
    - **Skills:** Component model, data binding, SignalR real-time comms.

🎮 Phase 3: Unity-Specific & Editor Tooling

Purpose: Bridge C# mastery into Unity world—custom tools, data pipelines, and runtime systems.

1. **Editor Recipe Authoring Window**
    - Create/modify `Recipe` ScriptableObjects through a custom EditorWindow.
    - **Skills:** `UnityEditor` API, asset creation, custom inspectors.
2. **2D Console-Style Mini-Game in Unity**
    - Rebuild your Snake or Tetris in Unity using Sprites and C# scripts.
    - **Skills:** `MonoBehaviour` loops, `Update()`, scene management, input mapping.
3. **Animation State Machine Controller**
    - Drive `Animator` between Idle, Walk, Chop, Pick with smooth transitions.
    - **Skills:** Blend trees, trigger/bool parameters, C# state-machine pattern.
4. **Physics-Driven Pickup & Throw**
    - Grab and toss rigidbody objects with realistic forces and collision layers.
    - **Skills:** Raycasting, joints vs. velocity, collision filtering.
5. **Dialogue System with ScriptableObjects**
    - Branching dialogues authored as SO graphs and driven via coroutines at runtime.
    - **Skills:** Data-driven design, UI binding, `IEnumerator` text effects.

⚡ Phase 4: Performance, Testing & Full-Stack Integration

Purpose: Polish with profiling, unit tests, CI/CD, and seamless backend↔Unity workflows.

1. **Performance Profiling Drill**
    - Populate 1,000 trees, profile draw calls & GC; optimize via object pooling & Jobs.
    - **Skills:** Unity Profiler, pooling patterns, `Unity.Jobs`, memory management.
2. **Unit & Integration Tests for Unity Scripts**
    - Write tests for your core C# classes (e.g. ResourceManager, AI state logic) with the Unity Test Framework.
    - **Skills:** PlayMode vs EditMode tests, mocking Unity APIs.
3. **Unity ↔ ASP.NET Core API Integration**
    - Fetch/save inventory, stats, and high scores from your ASP.NET Core service via `UnityWebRequest`.
    - **Skills:** Coroutines, JSON (de)serialization, error handling, secure token storage.
4. **CI/CD Pipeline (GitHub Actions)**
    - Automate builds/tests for your console library, .NET API, and Unity project; deploy backend to Docker/Heroku.
    - **Skills:** YAML workflows, cross-repo triggers, badge integration.
5. **Capstone: "Ultimate Survival" Prototype**
    - Merge your best console logic, WPF tools, Web API, and Unity scenes into a cohesive playable demo.
    - **Skills:** Cross-project references (NuGet or local DLLs), version control best practices, release packaging.

How to Tackle This List

1. **Phase 1 first—**become a C# wizard through classic games and katas.
2. **Phase 2 next—**learn GUI, services, and web tech that mirror industry workflows.
3. **Phase 3 & 4 in parallel—**alternate between Unity tool scripts and backend integration, always writing tests and profiling.