# Advance API Documentation

## Debugging in C#

### Debug Points

* **Definition:** Breakpoints allow you to pause the execution of your program at specific lines of code.
* **Usage:** Set breakpoints by clicking in the left margin of the code editor or pressing F9.

### Different Debug Windows

* **Immediate Window:** Allows you to interact with the application during runtime.
* **Watch Window:** Monitor variables and expressions.
* **Call Stack:** View the current execution stack.
* **Locals Window:** Inspect local variables in the current scope.
* **Autos Window:** Automatically displays variables used around the current line of execution.

### Editing

* **Edit and Continue:** Allows you to edit code during a debugging session and continue execution without restarting.

### Conditional Breakpoints

* **Definition:** Breakpoints that pause execution only when a specified condition is true.
* **Usage:** Right-click on a breakpoint and select “Conditions…” to set the condition.

### Data Inspector

* **Definition:** Tools to inspect and manipulate data while debugging.
* **Usage:** Hover over variables or use the Watch/Locals window to inspect values.

### Conditional Compilation

* **Definition:** Compile code selectively based on defined symbols using #if, #else, #elif, and #endif.
* **Usage:** Use preprocessor directives to include/exclude code.

## Concepts of C#

### Types of Classes

* **Abstract Classes:** Cannot be instantiated and must be inherited.
* **Sealed Classes:** Cannot be inherited.
* **Static Classes:** Cannot be instantiated and contain only static members.

### Generics

* **Definition:** Allow you to define classes, methods, delegates, and interfaces with a placeholder for the type.
* **Usage:** List<T>, Dictionary<TKey, TValue>, custom generic classes.

## File System in Depth

* **System.IO Namespace:** Provides classes for file and directory manipulation.
* **Common Classes:** File, Directory, FileInfo, DirectoryInfo, DriveInfo, Path.

## Data Serialization

### JSON

* **Libraries:** System.Text.Json, Newtonsoft.Json.
* **Usage:** Serialize with JsonSerializer.Serialize(), deserialize with JsonSerializer.Deserialize().

### XML

* **Libraries:** System.Xml.Serialization.
* **Usage:** Serialize with XmlSerializer.Serialize(), deserialize with XmlSerializer.Deserialize().

## Base Library Features

* **Collections:** List<T>, Dictionary<TKey, TValue>, Queue<T>, Stack<T>.
* **User Defined:** Add reference of user defined .dll file and use it.

## Lambda Expressions

* **Definition:** Anonymous functions that can contain expressions or statements.
* **Usage:** (parameters) => expression or (parameters) => { statements }.

## Extension Methods

* **Definition:** Static methods that add functionality to existing types.
* **Usage:** Define in a static class, use this keyword in the first parameter.

## LINQ

* **Data Sources:** DataTable, List<T>, IQueryable<T>.
* **Usage:** Query syntax (from x in y select z) or method syntax (y.Select(x => z)).

## ORM Tool

* **Entity Framework:** Popular ORM for .NET.
* **Usage:** Define DbContext and entity classes, use LINQ for querying.

## Security & Cryptography

* **Namespaces:** System.Security, System.Security.Cryptography.
* **Common Classes:** SHA256, RSA, Aes.

## Dynamic Type

* **Definition:** Allows operations to be performed that are resolved at runtime.
* **Usage:** Use dynamic keyword.
* **Reflection:** Inspecting and invoking types at runtime.

## Database with C#

### CRUD Operations

* **Create:** INSERT INTO SQL command or Add method in Entity Framework.
* **Read:** SELECT SQL command or Find/FirstOrDefault methods in Entity Framework.
* **Update:** UPDATE SQL command or modifying properties and calling SaveChanges in Entity Framework.
* **Delete:** DELETE SQL command or Remove method in Entity Framework.