1. Naming Convention
   * Variable name must be camel case like **objCustomer, orderItems.**
   * Never use word like **Test**, even if you doing testing with data.
   * Method name must relevant to its functionality.   
     **E.g.,** method having functionality of adding/updating Order line items then name must be **ManageOrderItems().**
   * Don’t create variable with naming like **objcustomer1** or **email1.**
2. Comments
   * Put proper comment before the statement which make having probability to change or any important details**.**
   * Add comment **Todo** where any partially functionality is pending so other team members aware about it.
3. Region & Readability
   * Being a good developer, his/her code must be in a readable form, so create a separate region base on **module, global variable, constructor, public & private methods.**
   * You can Also create nested region if required like for public methods you can separated by post/ get methods or by module wise e.g., Discount, order, customer.
   * **Methods Length max** be of **40-50 lines**, split based on the functionality.  
     E.g., Posting Order Data with line Items, main method should contain all the code. You can split separate methods for Order, Line items.  
     **Main** Method **=>** Add **Orders =>** Add **Line Items**.   
     so, the code looks goods and flow will be clear.
4. Exception Handling
   * We can’t serve error to user, we must handle exception by using try catch and show error message instead of error page.
   * Also use **try catch inside the loop when there is no dependency on anything**.   
     we don’t want to lose all the record of throw exception at very first records
   * When you have **nested method** and **when placed try catch at the child method** then use **Out string parameter** to get error at the parent method to show error to the user. E.g., **Public bool AddLineItms(orderitems items, out string error)**.
5. Config Data
   * When you read any peace of data from config file then initialize at the **constructor** instead of doing the same at multiple methods.
   * If you **reading config data and convert it to other than string data type** and create a **private method** and use, try catch, and **call method from constructor** to avoid error.
6. Logging & Common Code
   * Handling error is not enough, we need to log and notify if required to avoid same error again.
   * Logging information may help to identify the issues, so we can trace the problem in details.
   * Keep common code for variables, method.  
     Like creating global variables at controller level, create a wrapper class or base class where same method uses at multiple places
   * Let’s consider you want to response same format for all the Api throughout the application, then create class and set global variable and initialize and use in response.