Data Types in C#

All Types in C# are based on Common Type System
All the data types in C# are directly or indirectly derived from System.Object.
The Object is the universal data type of .NET.
Certain common functions that is required for every .NET Type is made available in the System.Object Class.
ToString()--->Converts the object to a string representation
GetType()--->Gets the internal Data type of the variable.
GetHashCode()->Gets the HashCode of the variable(HashCode is an unique ID created by CLR to identify every variable/object)
Equals() --->Gets the logical Equivalence of the variable with another. TRUE OR FALSE

Value Types

They are primitive types.
They are stored as structs.
They have Structs that will help in performing common operations like string conversions in them,

- Integral Types
  - byte
  - short
  - int
  - long
- Floating Types
  - float(Single)
  - double(Double)
  - decimal(Large Decimals)
- Other Types
  - char
  - bool
  - DateTime
- User Defined Types
  - Enums
  - Structs

What is a Value Type?

A Data type which stores the value assigned to it directly into the variable.
They are derived from a class called System. ValueType. They are internally stored as structures.

Reference Types

What is a reference type?

The data type which is used to store the reference of the variable is called Reference type. The Variable does not store the value but the value will be created in a seperate memory location called heap and its reference is what is stored in the variable.

Classes, Strings, Delegates, Object

BOXING AND UNBOXING
As any data type is derived from System.Object, U can assign any variable of any type to System.Object. This is called BOXING. Boxing is implicit.
The Object will wrap the internal feature of the variable and store it as an Object. This will be helpful to create functions whose data type is not known to you at compile time.
If U want to extract the data and its Functions, U should UNBOX it back to the same type from which it was BOXED. This is done using typical C kind of typecasting. Unboxing is explicit(U must typecast it)

Common Methods of Value types:
Parse: Convert the string to its type. If not possible throws an Exception(Abnormal Termination)
TryParse: Returns Bool that will be TRUE if Converting is successful. Else it returns false. No Exception is raised.
Compare: Used to compare the current variable with another.