**C# Snippets**

* **Snippets** are a useful feature in the visual studio IDE.
* With them you can type a few letters and then press tab twice to insert a code fragment.
* Help reduce repetitive statements and can reduce type error.
* Eg: cw for “console.writeline()”, do for “do while” loop, switch for “switch statement”, svm for Static void main”

**Data Types**

* A data type is a set of values and the allowable operations on those values.
* The two fundamental data types in C# are: value types and reference types
* Primitive types (except strings), enumerations, and structures are value types.
* Classes, strings, interfaces, arrays are reference types.
* Every type has a default value.
* *Reference types* are created on the Heap.
* The lifetime of the reference type is managed by the .NET framework.
* The default value for reference types is null reference.
* *Value types* are created on the stack.
* The lifetime is determined by the lifetime of the variable.
* Assignment to a variable of a value type creates a copy of the value being assigned.
* Value types have different default values
* For example, boolean default value is false, decimal 0, string an empty string "".

**Primitive**

* Primitive data types are those that are not defined in terms of other data types.
* Because primitive data types are the basis for all other types, they cannot have element content or attributes. However, they can contain values.
* The primitive data types are defined in the S*ystem* namespace.Eg:  byte, short, int, etc.

**Compostie**

* A composite data type is something you can divide into multiple other types.
* Eg:  Classes, strings, interfaces, arrays etc.

**Varialbes**

* A variable is an identifier that denotes a storage location in memory .
* It stores numeric or string values that might change during the program execution.
* A variables may take different values at different times during the execution of a program but the name of variable always remains same.
* Every variable has a data type that determine that type of values that can be stored into a variable.
* Syntax: *int i; i=10;*

**Constants**

* Constants are also used to store a value.that doesn’t change during the execution of a program.
* A varaible declare as constant using const keyword.
* Syntax:*const double e=2.71532732763; const int x=100;*

**Identifiers**

* In C# an identifier is a name given to a varaible, constant or any user defined programming element
* An identifier starts with a letter or underscore and ends with character
* C# identifiers are case sensitive suggests naming in CamelCasing.

**RULES**

* An identifier must begin with a letter or underscore.
* Must not be the reserved words.
* Should not start with digits.
* An identifier must be complete words without blank spaces

**Keywords**

* Are the reserved words whose meaning are predefined to C# compiler.
* Reserved by c# compiler to be used for specific task.
* Cannot be used as varaibles , methods,and properties because they are defined to the compiler to perform specific task

**Boxing and Unboxing**

* Boxing and Unboxing are important concepts used in the C# type system.
* They are used to create a link between the two major data types in C#.
* All value types are stored in stack but in some situations they need to referenced as heap as reference types are stored in heap.
* When the data moves from value types to reference types, it is termed ‘Boxing’ and the reverse is termed unboxing.
* Boxing is required in situations when a value type is converted into base object.
* CLR converts the value type into the reference type allocates the memory on heap and copies the value type instant to it.
* Unboxing is the process of converting an instance of object type or interface back to the value type.
* This is done explicitly by using the unboxing.
* Syntax:
* int i =1; object O=i; int j = int (O);

**Type Conversion:**

Type casting has two forms :

1. **Implicit**: Conversion in type safe manner .For eg: float a= 2.5f;2.
2. **Explicit**: Conversion done explicitly by users using the pre-derined functions.

Explicit conversions require a cast operator.

For eg: double d = 5673.74;

int i;

// cast double to int.

i = (int)d;