Printing

```
Console.WriteLine("Hello");
Console.Write("World");
Console.WriteLine("!");
```

Variables and Data Types

```
/*
Names are case-sensitive and may begin with:
    letters, _, @
After, may include
    letters, numbers, _
Convention says
    Start with a lowercase word, then additional words are capitalized
    ex. myFirstVariable
*/
string name = "Mike";    // String's are objects not primitives
char testGrade = 'A';    // single 16-bit Unicode character.

// short, int, long can be pre-pended with 'u' for 'unsigned'
byte age0 = 0;          // 8-bit unsigned integer
short age1 = 10;        // 16-bit signed integer.
int age2 = 20;          // 32-bit signed integer
long age3 = 30L;        // 64-bit signed integer

float gpa0 = 2.5F;       // 32-bit floating point
double gpa1 = 3.5;       // 64-bit double-precision floating point
decimal gpa2 = 4.5M;     // 128-bit precise decimal

bool isTall;            // 1 bit -> true/false
isTall = true;

name = "John";

Console.WriteLine("Your name is " + name);
Console.WriteLine($"Your name is {name}");
```

Casting and Converting

```
Console.WriteLine( (int)3.14 );
Console.WriteLine( (double)3 );

int intFromString = Convert.ToInt32("50");
double doubleFromString = Convert.ToDouble("50.99");

Console.WriteLine(100 + intFromString);
Console.WriteLine( 100 + doubleFromString );
```

Strings

```
string greeting = "Hello";
//      indexes:   01234

Console.WriteLine( greeting.Length );
Console.WriteLine( greeting[0] );
Console.WriteLine( greeting.IndexOf("llo") );
Console.WriteLine( greeting.IndexOf("z") );
Console.WriteLine( greeting.Substring(2) );
Console.WriteLine( greeting.Substring(1, 3) );
```

Numbers

```
Console.WriteLine(2 * 3);      // Basic Arithmetic: +, -, /, *
Console.WriteLine(10 % 3);     // Modulus Op. : returns remainder of 10/3
Console.WriteLine(1 + 2 * 3);  // order of operations
Console.WriteLine(10 / 3.0);     // int's and doubles


int num = 10;
num += 100; // +=, -=, /=, *=
Console.WriteLine(num);

num++;
Console.WriteLine(num);

Console.WriteLine( Math.Pow(2, 3) );
Console.WriteLine( Math.Sqrt(144) );
Console.WriteLine( Math.Round(2.7) );
```

User Input

```
Console.Write("Enter username: ");
string username = Console.ReadLine();
Console.WriteLine($"Hello {username}");
```

Arrays

```
//int [] luckyNumbers = new int[10];
int[] luckyNumbers = { 4, 8, 15, 16, 23, 42 };
//        indexes:    0  1  2   3   4   5
luckyNumbers[0] = 90;
Console.WriteLine(luckyNumbers[0]);
Console.WriteLine(luckyNumbers[1]);
Console.WriteLine(luckyNumbers.Length);
```

2 Dimensional Arrays

```
// int [][] numberGrid = new int[2][3];
int[][] numberGrid = { new int[]{ 1, 2 }, new int[]{ 3, 4 } };
numberGrid[0][1] = 99;

Console.WriteLine(numberGrid[0][0]);
Console.WriteLine(numberGrid[0][1]);
```

ArrayList

```
ArrayList friends = new ArrayList();
friends.Add("Oscar");
friends.Add("Angela");
friends.Add("Kevin");

//friends.Remove("Oscar");
Console.WriteLine(friends[0]);
Console.WriteLine(friends[1]);
Console.WriteLine(friends.Contains("Oscar"));
Console.WriteLine(friends.Count);
```

Methods

```
public static void Main(string [] args){
    int sum = AddNumbers(4, 60);
    Console.WriteLine(sum);
}

public static int AddNumbers(int num1, int num2){
    return num1 + num2;
}
```

If Statements

```
bool isStudent = false;
bool isSmart = false;

if (isStudent && isSmart)
{
  Console.WriteLine("You are a student");
}
else if (isStudent && !isSmart)
{
  Console.WriteLine("You are not a smart student");
}
else
{
  Console.WriteLine("You are not a student and not smart");
}

// >, <, >=, <=, !=, ==
if (1 < 3)
{
  Console.WriteLine("number omparison was true");
}
```

Switch Statements

```
char myGrade = 'A';
switch(myGrade){
    case 'A':
        Console.WriteLine("You Pass");
        break;
    case 'F':
        Console.WriteLine("You fail");
        break;
    default:
        Console.WriteLine("Invalid grade");
}
```

While Loops

```
int index = 1;
while(index <= 5){
    Console.WriteLine(index);
    index++;
}

do{
   Console.WriteLine(index);
   index++;
}while(index <= 5);
```

For Loops

```
for(int i = 0; i < 5; i++){
    Console.WriteLine(i);
}

int[] luckyNums = {4, 8, 15, 16, 23, 42};
foreach(int luckyNum in luckyNums){
    Console.WriteLine(luckyNum);
}
```

Exception Catching

```
try{
 int division = 10 / Convert.ToInt32(Console.ReadLine());
}catch(DivideByZeroException e){
   Console.WriteLine(e);
}catch(Exception e){
   Console.WriteLine(e);
   // Not best practice to use general Exception
}

throw new DivideByZeroException("can't add numbers");
```

Classes and Objects

```
public class Book{
   public string title;
   public string author;
   public static string staticAttribute = "My Static Attribute";

   public void ReadBook(){
    Console.WriteLine($"Reading {this.title} by {this.author}");
   }
}

Book book1 = new Book();
book1.title = "Harry Potter";
book1.author = "JK Rowling";

book1.ReadBook();
Console.WriteLine(book1.title);

Book book2 = new Book();
book2.title = "Lord of the Rings";
book2.author = "JRR Tolkien";

book2.ReadBook();
Console.WriteLine(book2.title);
Console.WriteLine(Book.staticAttribute);
```

Constructors

```
public class Book{
    public String title;
    public String author;

    public Book(String title, String author){
        this.title = title;
        this.author = author;
    }

    public void readBook(){
        Console.WriteLine("Reading " + this.title + " by " + this.author);
    }
}

Book book1 = new Book("Harry Potter", "JK Rowling");
Console.WriteLine(book1.title);

Book book2 = new Book("Lord of the Rings", "JRR Tolkien");
Console.WriteLine(book2.title);
```

Getters and Setters

```
public class Book{
    private String title;
    private String author;

    public Book(String title, String author){
        this.setTitle(title);
        this.setAuthor(author);
    }

    public String getTitle(){
        return this.title;
    }
    public void setTitle(String title){
        this.title = title;
    }

    public String getAuthor(){
        return this.author;
    }
    public void setAuthor(String author){
        this.author = author;
    }
}

Book book1 = new Book("Harry Potter", "JK Rowling");
Console.WriteLine(book1.getTitle());

Book book2 = new Book("Lord of the Rings", "JRR Tolkien");
Console.WriteLine(book2.getTitle());
```

Inheritance

```
public class Chef
{
  public String name;
  public int age;

  public Chef(String name, int age)
  {
     this.name = name;
     this.age = age;
  }

  public void MakeChicken()
  {
     Console.WriteLine("The chef makes chicken");
  }

  public void MakeSalad()
  {
     Console.WriteLine("The chef makes salad");
  }

  public virtual void MakeSpecialDish()
  {
     Console.WriteLine("The chef makes a special dish");
  }
}
public class ItalianChef : Chef
{

  public String countryOfOrigin;

  public ItalianChef(String name, int age, String countryOfOrigin)
    : base(name, age)
  {
     this.countryOfOrigin = countryOfOrigin;
  }

  public void MakePasta()
  {
     Console.WriteLine("The Chef make's past");
  }
  public override void MakeSpecialDish()
  {
     Console.WriteLine("The chef makes chicken parm");
  }
}

Chef myChef = new Chef("Gordon Ramsay", 50);
myChef.MakeChicken();

ItalianChef myItalianChef = new ItalianChef("Massimo Bottura", 55, "Italy");
myItalianChef.MakeChicken();
Console.WriteLine(myItalianChef.countryOfOrigin);
```

Abstract Classes and Methods

```
abstract class Vehicle
{
  public abstract void move();
  public void getDescription()
  {
     Console.WriteLine("Vehicles are used for transportation");
  }
}

class Bicycle : Vehicle
{
  public override void move()
  {
     Console.WriteLine("The bicycle pedals forward");
  }
}

class Plane : Vehicle
{
  public override void move()
  {
     Console.WriteLine("The plane flys through the sky");
  }
}
```

Interface Inheritance

```
public interface Animal
{
 void Speak();
 void Eat();
}
public class Dog : Animal
{
 public void Speak()
 {
     Console.WriteLine("Woof Woof");
 }
}
public class Cat : Animal
{
 public void Speak()
 {
     Console.WriteLine("Meow Meow");
 }
}

Animal[] animals = {
    new Dog(),
    new Cat()
};
foreach(Animal animal in animals){
    animal.speak();
}
```