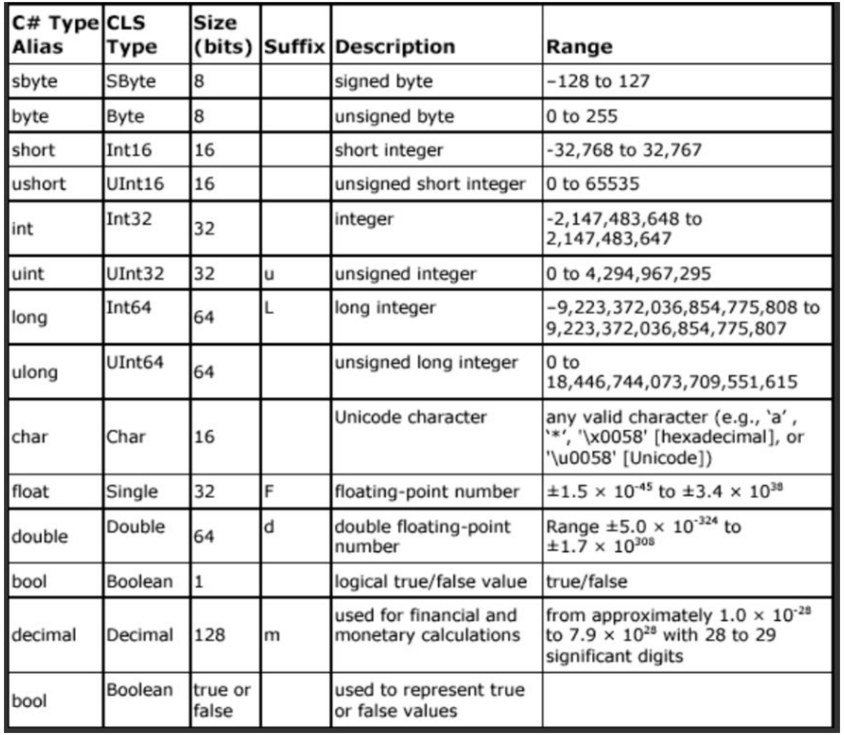
C# Data Types, Variables and Arithmetic Operators

Data Types

The basic types are:

- int
- float
- double
- string
- bool
- decimal

Data Types: A more complete list



C# Type Alias	CLS Type	Size (bits)	Suffix	Description	Range
sbyte	SByte	8		signed byte	-128 to 127
byte	Byte	8		unsigned byte	0 to 255
short	Int16	16		short integer	-32,768 to 32,767
ushort	UInt16	16		unsigned short integer	0 to 65535
int	Int32	32		integer	-2,147,483,648 to 2,147,483,647
uint	UInt32	32	u	unsigned integer	0 to 4,294,967,295
long	Int64	64	L	long integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ulong	UInt64	64		unsigned long integer	0 to 18,446,744,073,709,551,615
char	Char	16		Unicode character	any valid character (e.g., ‘a’ , ‘*’, '\x0058' [hexadecimal], or '\u0058' [Unicode])
float	Single	32	F	floating-point number	$\pm1.5 \times 10^{-45}$ to $\pm3.4 \times 10^{38}$
double	Double	64	d	double floating-point number	Range $\pm5.0 \times 10^{-324}$ to $\pm1.7 \times 10^{308}$
bool	Boolean	1		logical true/false value	true/false
decimal	Decimal	128	m	used for financial and monetary calculations	from approximately 1.0×10^{-28} to 7.9×10^{28} with 28 to 29 significant digits
bool	Boolean	true or false		used to represent true or false values	

A Second Program: Area of Circle

```
static void Main(string[] args)
{
    Console.WriteLine("Enter the radius:");
    string entry = Console.ReadLine();
    Console.WriteLine(entry);
}
```

What we entered is a string. We need a number.

A Second Program

We need to convert the string variable to a number, and then save it in a number variable.

A Second Program

```
static void Main(string[] args)
{
    Console.WriteLine("Enter the radius:");
    string entry = Console.ReadLine();
    double value = double.Parse(entry);
    Console.WriteLine(value);
}
```

That will convert it to a number of type “double”, which is the most common type of floating point number in C#.

A Second Program

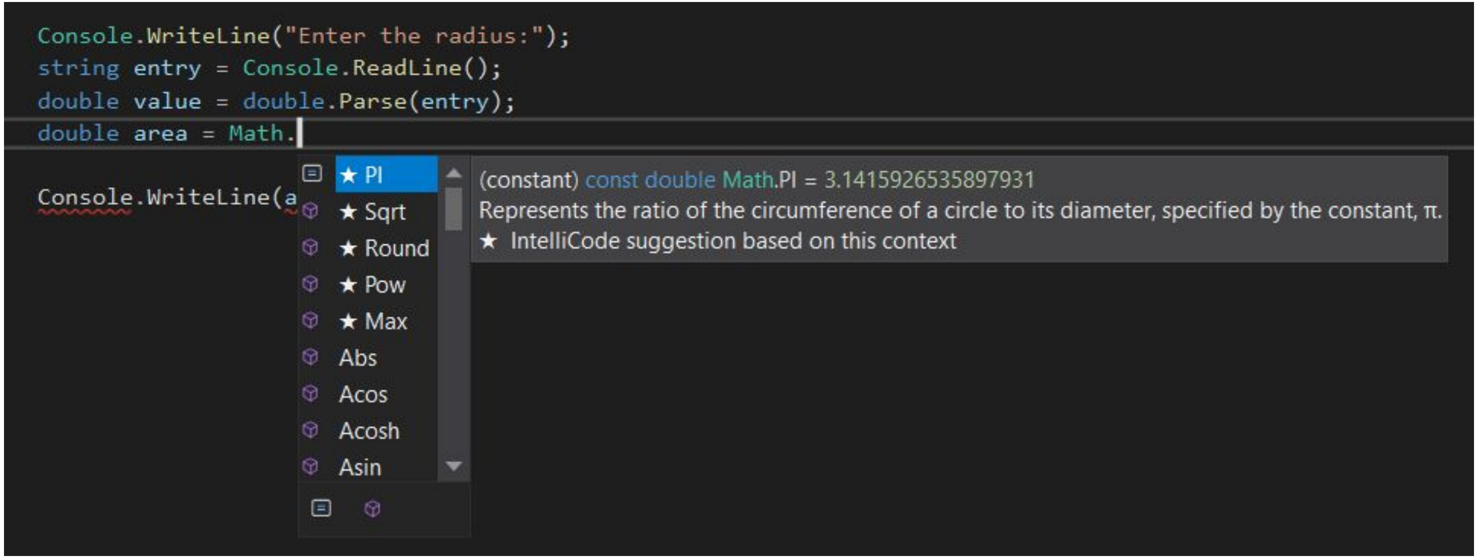
```
static void Main(string[] args)
{
    Console.WriteLine("Enter the radius:");
    string entry = Console.ReadLine();
    double value = double.Parse(entry);
    double area = 3.1415926 * value * value;
    Console.WriteLine(area);
}
```

Now we can multiply it by itself times pi to get the area.

A Second Program

Let’s use the Math class.

Type Math followed by a dot to see the popup.



```
Console.WriteLine("Enter the radius:");
string entry = Console.ReadLine();
double value = double.Parse(entry);
double area = Math.

Console.WriteLine(a
```

(constant) const double Math.PI = 3.1415926535897931
Represents the ratio of the circumference of a circle to its diameter, specified by the constant, π.
★ IntelliCode suggestion based on this context

A Second Program

```
static void Main(string[] args)
{
    Console.WriteLine("Enter the radius:");
    string entry = Console.ReadLine();
    double value = double.Parse(entry);
    double area = Math.PI * value * value;
    Console.WriteLine(area);
}
```

This is much more precise.

Arithmetic Expressions

Order of precedence

- Increment and decrement
- Positive and negative
- Multiplication, division, and remainder
- Addition and subtraction

Arithmetic Operators

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement
+	Positive sign
-	Negative sign

Assignment Statements

An assignment statement consists of a variable, an equals sign, and an expression. When the assignment statement is executed, the value of the expression is determined and the result is stored in the variable.

Assignment Operators

Operator	Name
=	Assignment
+=	Addition
-=	Subtraction
*=	Multiplication
/=	Division
%=	Modulus

Recap

- C# Importance and History
- How C# compares other coding languages
- Types of applications developed using C#
- Know common C# IDEs
- How to write a Hello World program in C#
- Data types, statements, and variables in C#