Debugging

1. Using Debuger
   A. make sure you have c# dev kit installed

`Press F5`

1. There will be three options
   - Start New Instance - This starts your project with a debugger attached.
   - Start without Debugging - This runs your project without a debugger attached.
   - Step into New Instance - This starts your project with a debugger attached but stops at the entrypoint of your code.
   -
2. More Docs
3. String Interpolation

```
string? Name = "Jake"
Console.WriteLine($"Hello {name}!");
```

   1. string? is the type of variable we are creating
   2. $ is the way we tell a string to interpolate the string variable into the string
   3. This code will give the result of `Hello Jake!`

4. Terminal Inputs and Ouputs
   1. This how we take a user input

```
string? name = Console.ReadLine();
```

      1. Full Program

```
    // See https://aka.ms/new-console-template for more information
string UserGreeting = "Hello Enter Your name:";

Console.WriteLine(UserGreeting);

string? name = Console.ReadLine();

Console.WriteLine($"Hello {name}!");
```

5. Types
   1. Value Types

2. Reference Types - all are nullable

a. Integral Numeric Types

C# Type/Keyword	Range	Size	.NET Type
sbyte	-128 to 127	Signed 8-bit integer	System.SByte
byte	0 to 255	Unsigned 8-bit integer	System.Byte
short	-32,768 to 32,767	Signed 16-bit integer	System.Int16
ushort	0 to 65,535	Unsigned 16-bit integer	System.UInt16
int	-2,147,483,648 to 2,147,483,647	Signed 32-bit integer	System.Int32
uint	0 to 4,294,967,295	Unsigned 32-bit integer	System.UInt32
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	Signed 64-bit integer	System.Int64
ulong	0 to 18,446,744,073,709,551,615	Unsigned 64-bit integer	System.UInt64
nint	Depends on platform (computed at runtime)	Signed 32-bit or 64-bit integer	System.IntPtr
nuint	Depends on platform (computed at runtime)	Unsigned 32-bit or 64-bit integer	System.UIntPtr

b. Floating Point numbers

C# Type/Keyword	Approximate Range	Precision	Size	.NET Type
float	±1.5 x 10^−45 to ±3.4 x 10^38	~6-9 digits	4 bytes	System.Single
double	±5.0 × 10^−324 to ±1.7 × 10^308	~15-17 digits	8 bytes	System.Double
decimal	±1.0 x 10^-28 to ±7.9228 x 10^28	28-29 digits	16 bytes	System.Decimal

c. bool

C# Type/Keyword	Values	Description	.NET Type

C# Type/Keyword	Values	Description	.NET Type
bool	true, false	Boolean value (true or false)	System.Boolean

d. char

C# Type/Keyword	Range	Description	Size	.NET Type
char	U+0000 to U+FFFF	Unicode 16-bit character	16-bit (2 bytes)	System.Char

e. make any of these types nullable by defining with a question mark like so `char? , float?, int? bool?` f. Program Example

```
// See https://aka.ms/new-console-template for more information

//ints

int a = 0;
int b = 2;
int c = -1;

// basic math with ints

int add = b + a;
Console.WriteLine($"result of a + b = {add}");
int sub =  c - b ;
Console.WriteLine($"result of c-b = {sub}");
int multi = a * b;
Console.WriteLine($"result of a * b = {multi}");
int divide = b / c;
Console.WriteLine($"result of b/c = {divide}");


// floats

float f = 3.5f;
Console.WriteLine($"f = {f}");
double g = 7.8d;
Console.WriteLine($"g = {g}");
decimal dec = 7.15m;
Console.WriteLine($"dec = {dec}");

// float maths & precisions when decimals points matters

Console.WriteLine($"float precision is seven digits:
{1.2345678f/0.98765432f}");
Console.WriteLine($"float precision is seven digits:
{1.2345678d/0.98765432d}");
Console.WriteLine($"float precision is seven digits:
{1.2345678m/0.98765432m}");
```

```
// characters
char c1 = 'a';
Console.WriteLine($"char: {c1}");

// create a bool & set bools

bool enabled = false;
Console.WriteLine($"bool: {enabled}");

enabled = true;
Console.WriteLine($"bool: {enabled}");

// null value types ?  allows us to make value nullable
// all types in this program are nullable

int? i1 = null;
bool? b1 = null;
```

3. Type conversion

    1. Implicit Conversion - no special snytax
    2. Explicit Conversion - casting, can lead to precision or data loss

Extra Credit

Follow This video to pimp out your command line

https://youtu.be/-G6GbXGo4wo