# Module 1 Day 4

Arrays and Loops

# Variables Review

- Variable: A name that represents a memory location
- Stores a value of some type
- 3 steps to use:
  - Declare – give it a name and type
  - Allocate – give it some memory
  - Assign – give it a value


```
// Declare and allocate a place to hold the student's score
int score;

// Assign a value to score
score = 92;

// Display the value of the score
Console.WriteLine($"Student score is {score}");
```

# Variables

- What if I have a group of similar values?
- E.g., student scores

| | |
|---|---|
| score1 | 92 |
| score2 | 78 |
| score3 | 69 |
| score4 | 96 |
| score5 | 88 |

- What would a class of 30 look like?

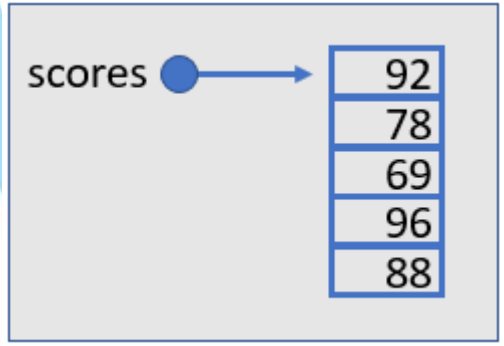
```
// Declare and allocate and assign students' scores
int score1 = 92;
int score2 = 78;
int score3 = 69;
int score4 = 96;
int score5 = 88;

// Display the value of the score
Console.WriteLine("Before the curve:");
Console.WriteLine($"Student1 score is {score1}");
Console.WriteLine($"Student2 score is {score2}");
Console.WriteLine($"Student3 score is {score3}");
Console.WriteLine($"Student4 score is {score4}");
Console.WriteLine($"Student5 score is {score5}");

// Adjust scores because the grading curve is +6
score1 = score1 + 6;
score2 += 6;
score3 += 6;
score4 += 6;
score5 += 6;
```

# Arrays

- A set of related values, all of the **same type**
  - Stored contiguously in memory
  - Referenced by a **single variable name**
- Array is a reference type
  - The array variable points to the location of the start of the array
  - The items in the array may be reference or value types
- Each item in an array is called an **element**
- Elements are accessed by an **index** (integer)

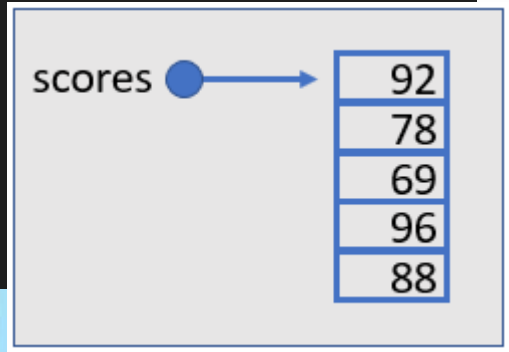


# Declaring, allocating and assigning arrays

```
// Declare a place to hold the students' scores
int[] scores;

// Allocate the storage for scores
scores = new int[5];

// Assign the values
scores[0] = 92;
scores[1] = 78;
scores[2] = 69;
scores[3] = 96;
scores[4] = 88;
```




```
// Declare and allocate and assign students' scores
int score1 = 92;
int score2 = 78;
int score3 = 69;
int score4 = 96;
int score5 = 88;
```

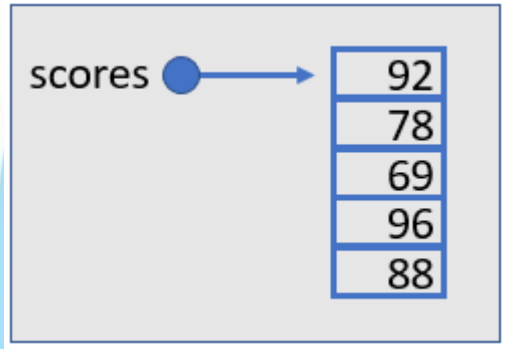
- An array's length never changes once it is allocated!
- Indexes start at 0

# Declaring, allocating and assigning arrays

```
// Declare and allocate and assign students' scores
int[] scores = new int[5] { 92, 78, 69, 96, 88 };
```

- OR -

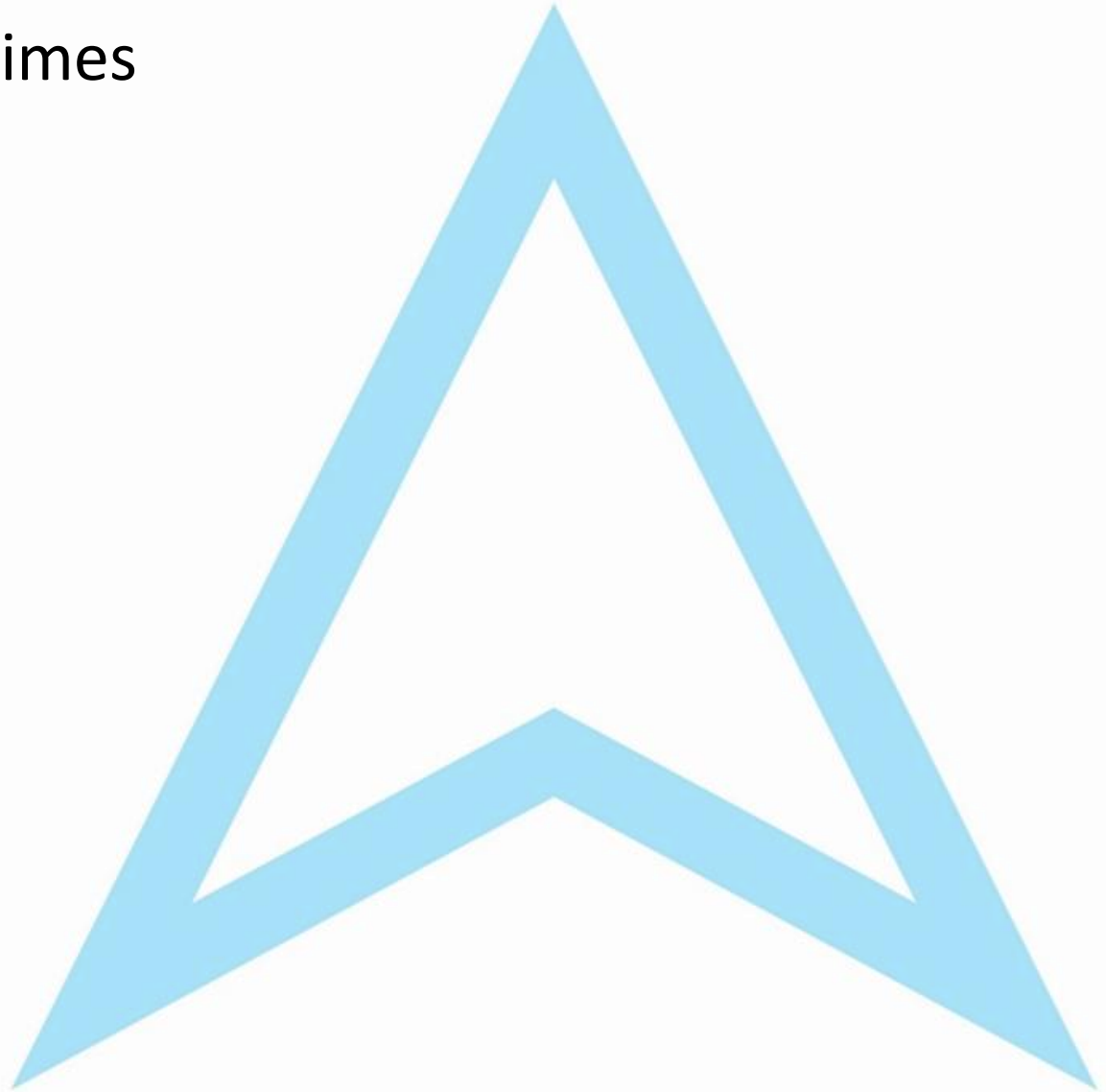
```
// Declare and allocate and assign students' scores
int[] scores = new int[] { 92, 78, 69, 96, 88 };
```




```
// Declare and allocate and assign students' scores
int score1 = 92;
int score2 = 78;
int score3 = 69;
int score4 = 96;
int score5 = 88;
```

# Looping

- Executes a block of code multiple times
  - for
  - foreach
  - while
  - do
- We'll cover "for" today



# Looping - for

```
Console.WriteLine("Before the curve:");
for (int i = 0; i < scores.Length; i++)
{
    // Display the value of the score
    Console.WriteLine($"Student{i+1} score is {scores[i]}");
    // Adjust scores because the grading curve is +6
    scores[i] += 6;
}
```

```
// Display the value of the score
Console.WriteLine("Before the curve:");
Console.WriteLine($"Student1 score is {score1}");
Console.WriteLine($"Student2 score is {score2}");
Console.WriteLine($"Student3 score is {score3}");
Console.WriteLine($"Student4 score is {score4}");
Console.WriteLine($"Student5 score is {score5}");

// Adjust scores because the grading curve is +6
score1 = score1 + 6;
score2 += 6;
score3 += 6;
score4 += 6;
score5 += 6;
```

- How do we calculate the average score?

# Breaking a Loop

- break;
  - Transfers control to the statement following the loop
  - Exits the loop
- continue;
  - Transfers control to the top of the loop for the next iteration
  - Skips only a portion of the current iteration
- return;
  - Exits the method, and so exits the loop

- Find if there are any perfect scores in the array