# Loops and Counters

PROGRAMMING IN C#

## Topics

- Increment and Decrement Operators
- The *while* loop
- The *do-while* loop
- The *for* loop
- The *foreach* loop
- Break and Continue
- Deciding which loop to use
- Assignment for today: Palindrome Detector

## Increment and Decrement Operators

- We often need to increase a number by 1 (*increment*) or decrease a number by 1 (*decrement*).

- Typically, this is used to count through the iterations of a loop.

```
number = number + 1;
number = number - 1;
```

- We can do any other arithmetical operations like this, too:

```
number = number * 4;
number = number / 10;
```

## Increment and Decrement Operators

- In C#, a shorter way to write this is:

```
number += 1;
number -= 1;
```

- We can combine any of the arithmetic operators with the equals sign to change the value of a variable in a particular way, and then assign that value back to the same variable name:

```
number += 10; // add 10 to the variable number
number -= 3;  // subtract 3 from the variable number
number *= 10; // multiply number by 10
number /= 4;  // divide variable number by 4
```

## Increment and Decrement Operators

- Incrementing or decrementing by 1 is so common it has its own shorthand:

```
number++;     or     ++number;
number--;     or     --number;
```

- We say this as "number plus plus" or "number minus minus"
- This is useful to have as a counter when we *iterate* through a loop, that is, to go through a loop one or more times.
- If we want to add more than one, we must use the other notation:

```
number += 7; // increment the variable by 7
```

## Prefix and Postfix Operators

- Putting the operator first is called *prefix*: `++number; --number;`
- Putting the operator last is called *postfix*: `number++; number--;`
- Where we put the operator determines when the variable is incremented or decremented
- If the operator is first, then we increment or decrement before the rest of the expression is evaluated
- If it is last, then we evaluate the expression first, then change the value
- Using the "plus plus" or "minus minus" changes the number by exactly one, which is useful for cycling through a series.

# Fist of Five

I'm completely lost
I need this explained again
I sorta get it
I understand but have questions
I'm confident
I could teach this!

## Looping types

- C# has three looping types:
  - While loop
  - Do While loop
  - For loop

## While Loop

- The While loop is a *pretest* loop, meaning we test for a certain condition before executing the loop:

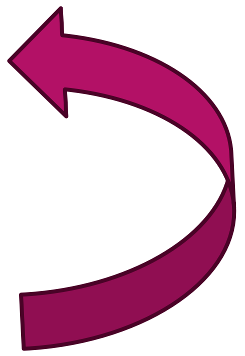
```
while (condition)
   {
      statements;
   }
```

- This means that if a certain condition is true, we will execute the statements in the body of the loop. At the end, we loop back to the beginning and check the condition again. *While* the condition is true, we will execute the statements in the loop over and over.

## While Loop: An Analogy

Think about the steps you take when you are hungry:

- Go to the kitchen
- Get a plate from the cupboard
- Open refrigerator
- Remove food
- Put food on plate
- Eat food
- Put empty plate in the sink



Some of those steps you might need to repeat if you are *really hungry*!

## While Loop: An Analogy

This is what those steps would look like in a while-loop:

```
Go to the kitchen
Get a plate from the cupboard
while (hungry){
    Open refrigerator
    Remove food
    Put food on plate
    Eat food
}
Put empty plate in the sink
```

Steps we do *before* the loop

Repeat this loop until no longer hungry

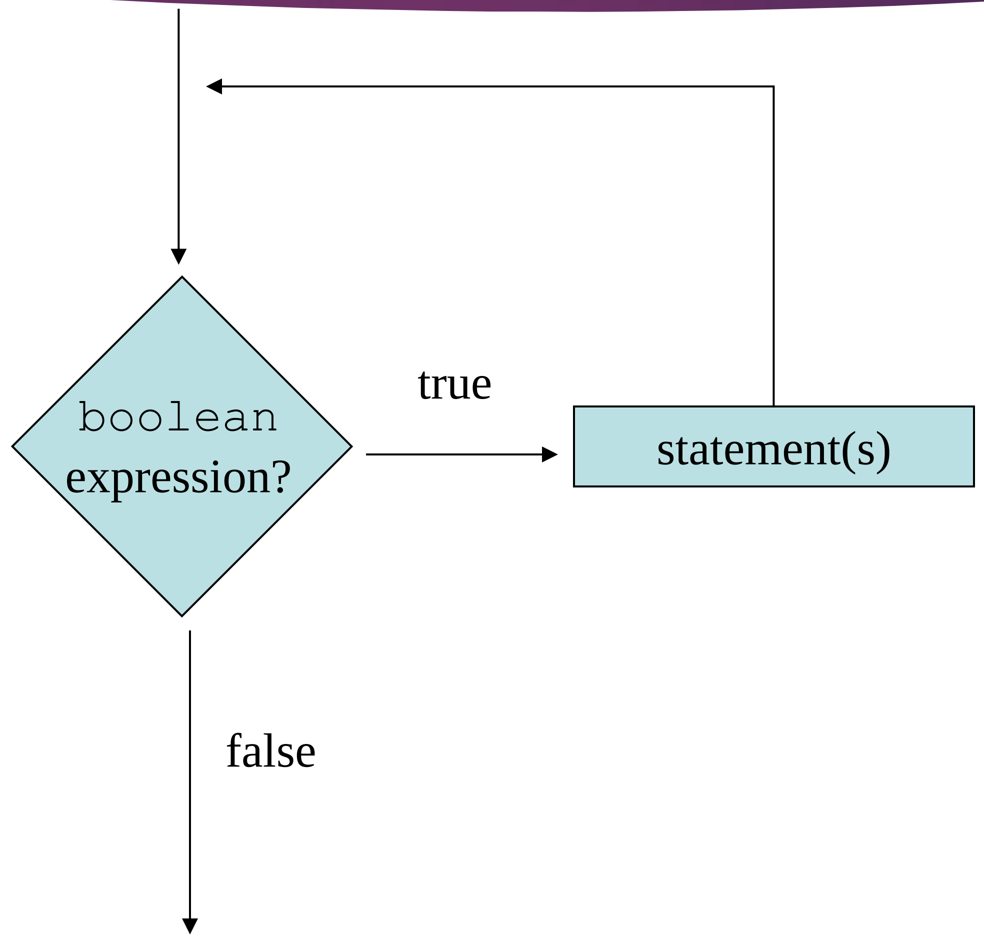
Steps we do *after* the loop

## While Loop: Simple Example

Here’s an actual loop that would print out your name as often as you keep replying “yes”:

```
bool runAgain = true;
while (runAgain)
{
    Console.WriteLine("What is your name?");
    string name = Console.ReadLine(); // this command reads what the user types into a
variable
    Console.WriteLine("Hello, " + name + "!");
    Console.WriteLine("Want to enter a new name?");
    runAgain = Console.ReadLine().StartsWith("y"); // this checks if the user types “yes”
}
Console.WriteLine("Bye");
```

## The `while` loop Flowchart



## Do-While Loop

- The Do-While loop is a *posttest* loop, meaning we test for a certain condition after executing the loop:

```
do
{
    statements;
}
while (condition);
```

- This means that we will execute the statements in the body of the loop *at least once*. We don't check the condition until the end. If it's true, we loop back to the beginning and execute the statements again. As long as the statement remains true, we will execute the statements in the loop over and over.

## Do-While Loop Analogy

For Thanksgiving dinner, the food might be so good we eat even if we aren't hungry!

```
Go to the kitchen
Get a plate from the cupboard
do {
    Open refrigerator
    Remove food
    Put food on plate
    Eat food
}
while (!full)
Put empty plate in the sink
```

Steps we do *before* the loop

Repeat this loop until you are too full to continue

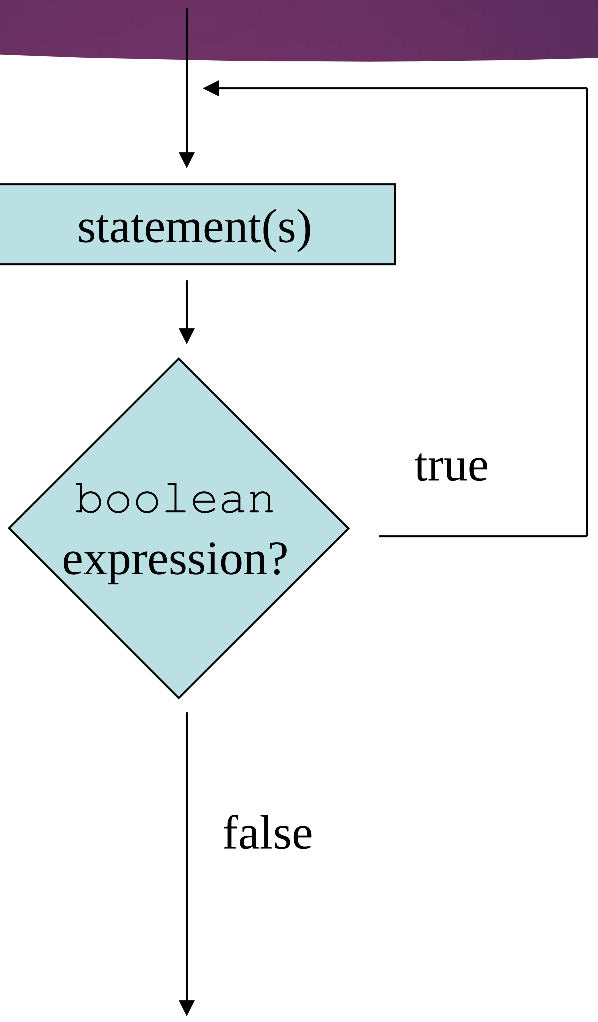
Steps we do *after* the loop. Notice the operator "!", which means "not", so "!full" means "not full".

## Do-While Loop

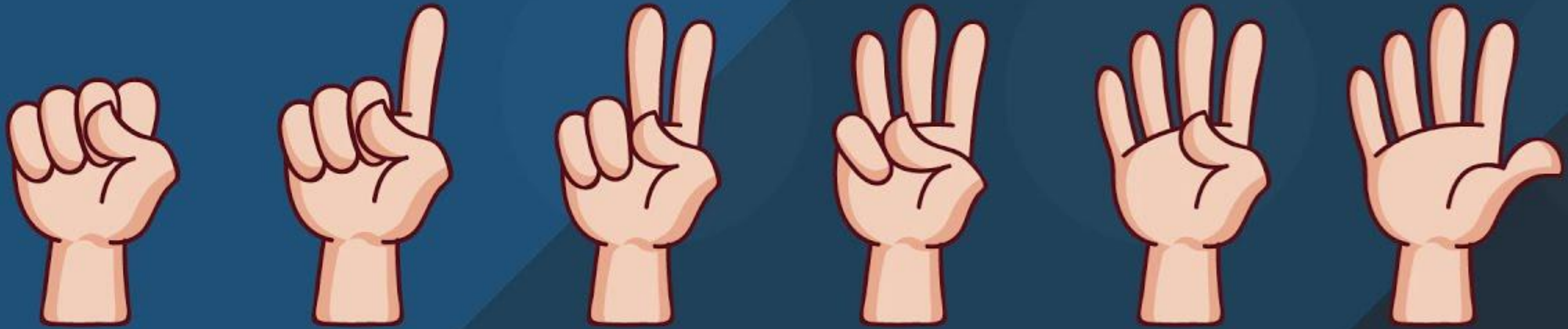
Here’s the code from the While loop, reformatted into a Do-While loop:

```
bool runAgain = false; // this could be true or false here, because we won’t check it till the end of the loop
do
{
    Console.WriteLine("What is your name?");
    string name = Console.ReadLine(); // this command reads what the user types into a variable
    Console.WriteLine("Hello, " + name + "!");
    Console.WriteLine("Want to enter a new name?");
    runAgain = Console.ReadLine().StartsWith("y"); // this checks if the user types “yes”
}
while (runAgain)
Console.WriteLine("Bye");
```

## The `do-while` Loop Flowchart



# Fist of Five


I'm completely lost
I need this explained again
I sorta get it
I understand but have questions
I'm confident
I could teach this!

# For Loop

- Sometimes we want to iterate an exact number of times
- The For loop will execute the body of statements exactly the number of times we tell it to.
- The For-loop command looks like this:

```
for (int n = 0; n < 10; n++)
{
    statements;
}
```

- This For loop will execute exactly 10 times, counting up from zero to nine.

# For Loop

- The For loop declaration has three parts:

```
for (int n = 0; n < 10; n++)
```

a statement executed before the loop is started, usually used to initialize a counting variable

a condition that evaluates to true or false, checked before executing each iteration of the loop

a statement executed at the end of each loop iteration, usually used to increment the counting variable

## For Loop

- The way this works:

```
for (int n = 0; n < 10; n++)
{
    Console.WriteLine("Hello, World");
}
```

This loop will initialize the variable n to be 0

It will check if n is less than 10.  0 < 10, so it will execute an iteration of the loop.

In this case, the loop simply prints: `"Hello World"`

It will then increment n by 1, so n is now equal to 1.

It will check the condition again. 1 < 10, so it will print `"Hello World"` again.

It will continue like this until n equals 10, which is not less than 10 so the condition will fail, and the loop will exit.

## For Loop

- We could have also written the For loop like this:

```
for (int n = 1; n <= 10; n++)
{
    Console.WriteLine("Hello, World");
}
```

Initializing at 1 instead of 0 and using “less than or equal” to 10 instead of “less than” 10, and it would have had the exact same effect, executing exactly 10 times.

- In Computer Programming, we usually start with zero instead of one as a tradition.  But it would work either way.

## For Loop

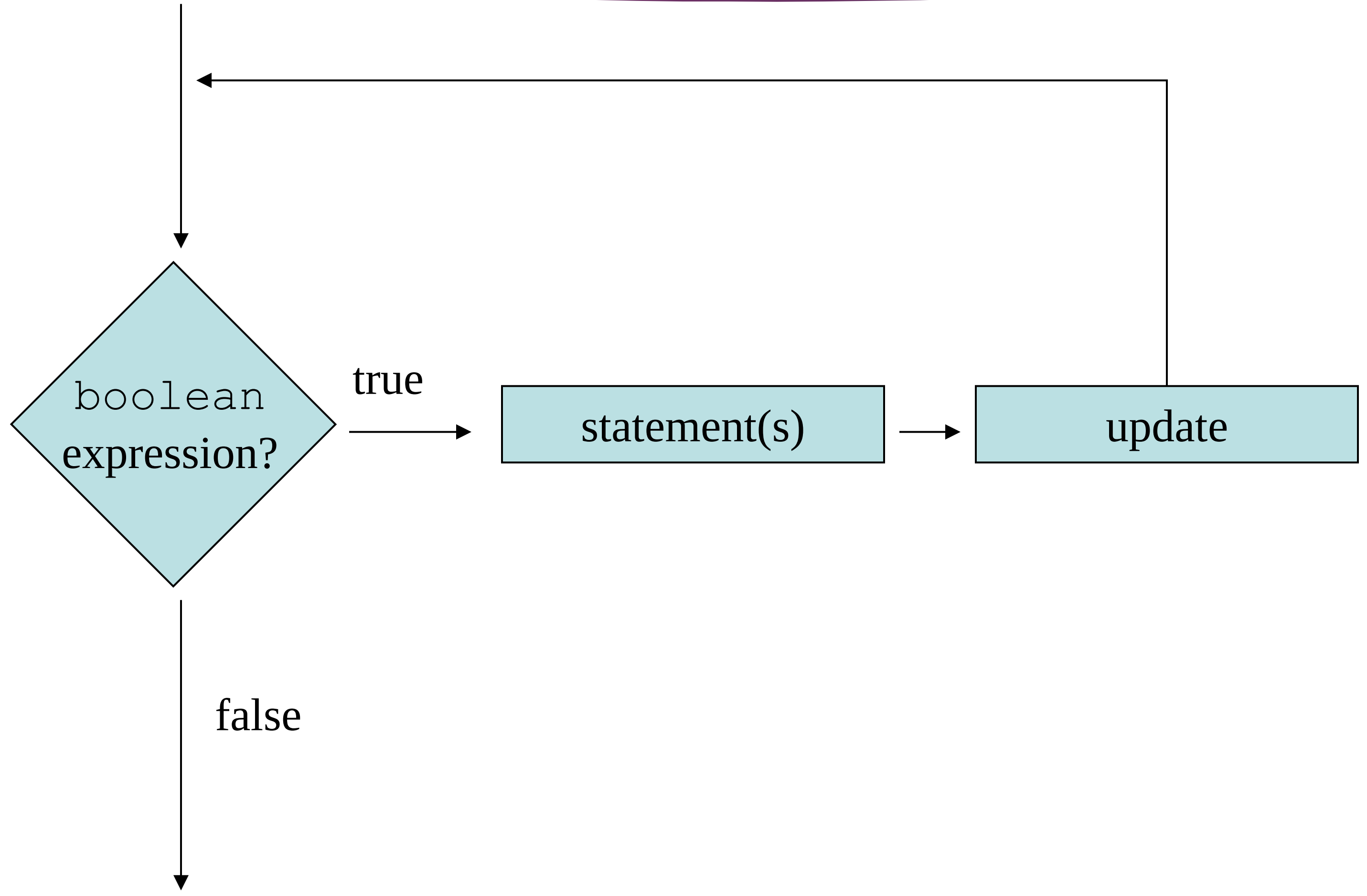
- We could also have written the For loop with the decrement counter:

```
for (int n = 10; n > 0; n--)
{
    Console.WriteLine("Hello, World");
}
```

And again, it would have done the exact same thing, printing the hello message exactly 10 times and then exiting.

- It is sometimes useful when creating a loop to think about counting down rather than counting up.

## The `for` Loop Flowchart



# MUD CHECK!

**What concept is the muddiest for you so far?**

## Multiple initializations and updated

- In a For loop, we can also initialize and update multiple variables

```
for (int i = 5, j = 0; i < 10 || j < 20; i++, j += 2)
{
    statements;
}
```

This For loop would initialize variables i and j.

- At the end of each loop, it would increment i by 1 and j by 2.
- The condition can be compound, in this case using the "OR" operator, II to check if i is less than 10 *OR* j is less than 20.  As long as one of these is true, the loop keeps iterating.

## Nested Loops

- Loops can be nested:

```
for (int i = 0; i < 10; i++)
{
    for (int j = 0; j < 10; j++)
    {
        statements;
    }
}
```

The inner loop will execute 100 times: i x j = 10 x 10 = 100.

## For-Each Loops

- “for each” loops process items in a collection
- Simplifies syntax
- Makes code more readable

```
foreach (string item in list) {
    statements;
}
```

## The break Statement

- The break statement terminates a loop, by-passing all remaining statements within the loop and all conditions.  It simply exits the loop.

```
for (int n = 0; n < 10; n++)
{
    Console.WriteLine("Hello, World");
    if (date == "Tuesday")
    {
        break;
    }
    Console.WriteLine("Nice to meet you"); // this line gets skipped every Tues
}
```

- Because it adds complexity, break statements could make your code harder to read.  It should be avoided in most cases.

## The continue Statement

- The continue statement will cause the currently executing iteration of a loop to terminate and the next iteration will begin.
- The continue statement will cause the evaluation of the condition in while and for loops.

```
for (int n = 0; n < 10; n++)
{
    Console.WriteLine("Hello, World");
    if (date == "Tuesday")
    {
        continue;
    }
    Console.WriteLine("Nice to meet you"); // this line gets skipped every Tues
}
```

- Like the break statement, the continue statement should be avoided because it makes the code hard to read and debug.

## Deciding which loop to use

- The while loop:
  - Pretest loop
  - Use when you do not want the loop to execute at all if the condition is false in the beginning.
- The do-while loop:
  - Post-test loop
  - Use when you want the statements to execute at least one time.
- The for loop:
  - Pretest loop
  - Use when you want an exact number of iterations through the loop.

## CAUTION: You can’t alter a collection while iterating through it

- Caution! If you change the contents of a collection in the middle of a loop while iterating through it, you will get a Runtime Error:

```
List<string> list = new List<string>();
list.Add("dog");

foreach (string item in list)
{
    Console.WriteLine(item);
    list.Add("cat"); // RUNTIME ERROR
}
```

This will compile, but it won’t run.

## MUD CHECK!

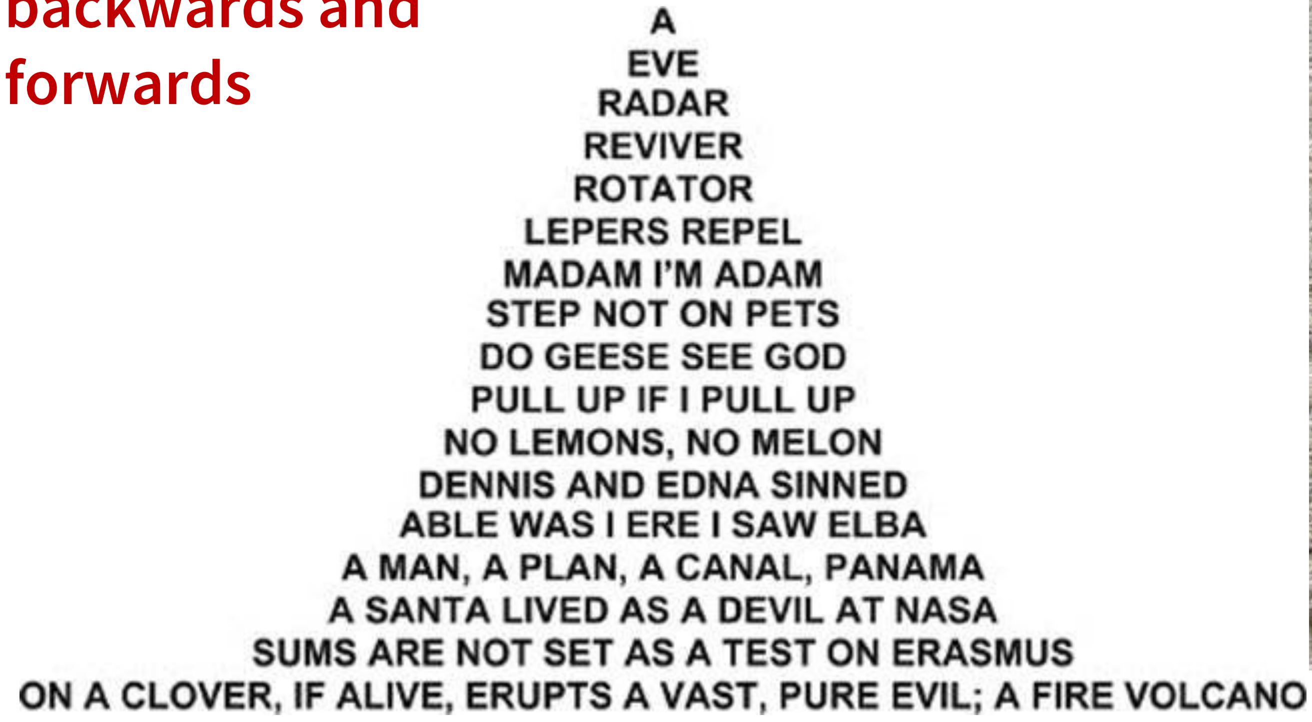
What concept was the muddiest for you today?

# Palindrome Detection Assignment

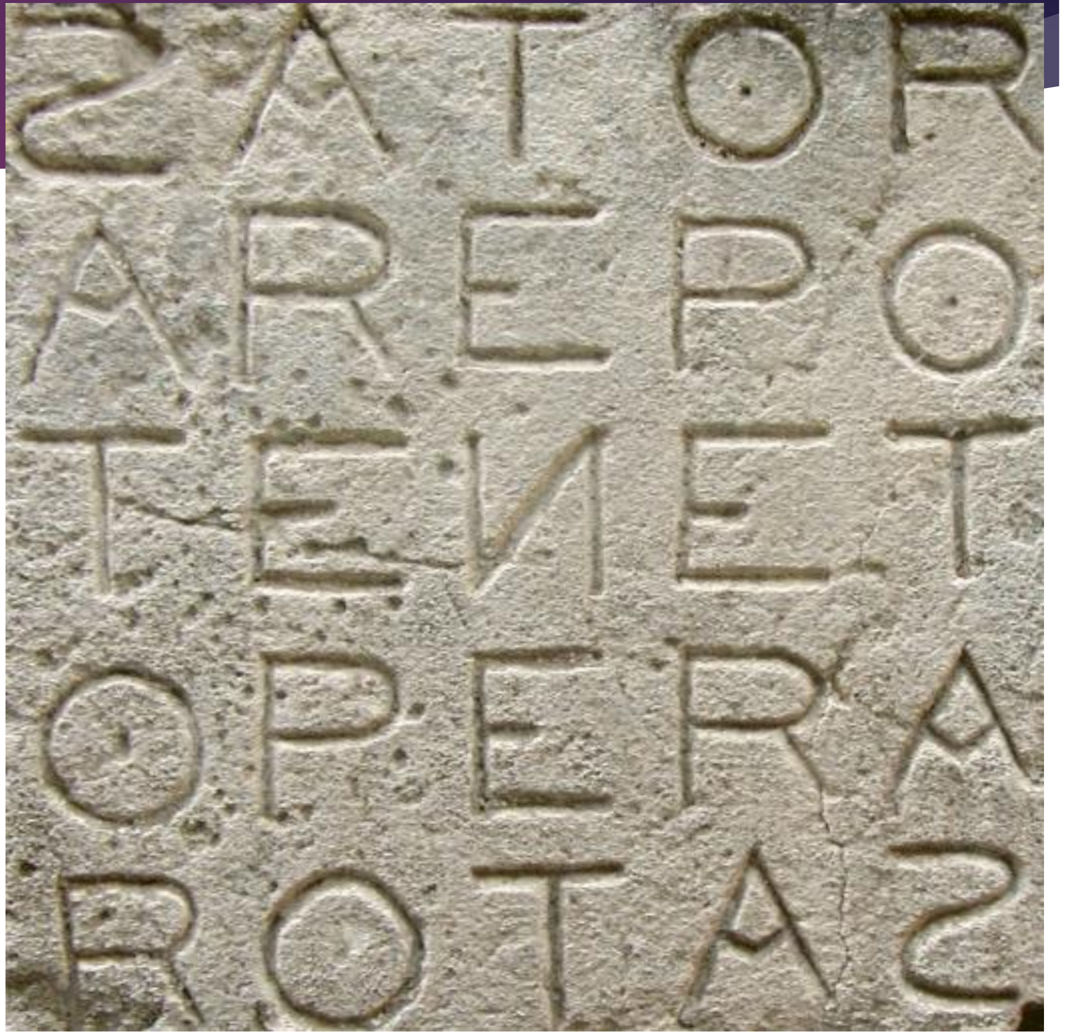
## Palindromes

Words or phrases that read the same backwards and forwards

Punctuation and spaces don’t count



A
EVE
RADAR
REVIVER
ROTATOR
LEPERS REPEL
MADAM I'M ADAM
STEP NOT ON PETS
DO GEESE SEE GOD
PULL UP IF I PULL UP
NO LEMONS, NO MELON
DENNIS AND EDNA SINNED
ABLE WAS I ERE I SAW ELBA
A MAN, A PLAN, A CANAL, PANAMA
A SANTA LIVED AS A DEVIL AT NASA
SUMS ARE NOT SET AS A TEST ON ERASMUS
ON A CLOVER, IF ALIVE, ERUPTS A VAST, PURE EVIL; A FIRE VOLCANO



## Boolean Logic

true && true : true

true && false : false

false && true : false

false && false : false

**&&** *means* **AND**

true || true : true

true || false : true

false || true : true

false || false : false

**||** *means* **OR**

## String is a character array

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| C | H | O | C | O | L | A | T | E |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

```
Console.WriteLine(text[0]); // C
Console.WriteLine(text[1]); // H
```

## Static method to test if a character is a letter or a number (not a space or symbol)

```
char.IsLetterOrDigit(inputText[index])
```