# Sequence

*Professor Don Colton*

Brigham Young University Hawaii

Mathematics and programming can look a lot alike. This leads to confusion at times. Sequence is an important area where this can be a problem.

## 1 Equations

In mathematics, a system of equations is viewed to be true no matter what the order is. Consider these equations.

```
x + y = 7
y = 3
```

Using normal rules of algebra, you can easily see that `x` must be equal to 4. We could figure it out like this:

```
(1) x + y = 7
(2) y = 3
(3) x + 3 = 7 (subst 2 into 1)
(4) x + 3 - 3 = 7 - 3 (subt 3 both sides)
(5) x = 4 (simplify)
```

It would not matter if we started out with this instead:

```
y = 3
x + y = 7
```

The reason is that mathematical equations are supposed to be statements of truth that have no particular order to them. They are just all true at the same time.

## 2 Steps

In computing, each statement is not a statement of truth. It is a command to be carried out. It is an instruction.

```
y = 3
x + y = 7
```

In programming, this would take the value 3 and copy it into the variable y. Then the program would die because `x + y = 7` cannot be carried out. The `=` sign means "copy" instead of "it is equal". We can copy into a variable. We cannot copy into an expression like `x + y`. We would probably have to rewrite the program as follows.

```
y = 3
x = 7 - y
```

Now we can calculate the value of x. It will be seven minus the value currently stored in y. Since y has a three in it, we calculate that x should receive the value of four.

Reversing things, we have the following:

```
x = 7 - y
y = 3
```

Here the first step is to calculate x. We have seven. We look at variable y. Since y has not yet received its value of three, it does not have a value, or it has a value of "garbage." Since the steps are happening in order, x receives a value of garbage.

Then y receives its value of three. But this does not help x because that step is now ancient history. Computers do not remember what they just did. All they remember is what the variables are right now and what step they are working on.

## 3 Sequence

Sequence is the most important method we have of controlling our programs. In nearly every programming language, steps are written one after the other.

The computer is required to execute one step completely and then go on to the next step.

# 4 Later

Later we will look at ways to change the sequence of operations. Steps never run backwards, but we can make programs jump from one part of the program to another. This is called `goto`. Programs can be made to choose between two paths based on the results of a calculation. This is called `if/else` or conditional execution. Programs can be made to repeat instructions over and over. This is called `iteration` or `looping`. There are other things including subroutines and parallel programming.

For now, the important thing to remember is that programs run sequentially, one step after another. The order of the steps is usually very important. Sequence is the foundation of programming.