Two MAPLE programming constructs which often will come in handy are what we will call "for" programs and "if" programs. Both involve concepts and syntax that is very familiar to mathematics students.

"for" These are programs that do some repetitive calculation based on changing an index by the same size step each time and repeating the calculation. For example suppose we wanted to compute the three numbers \( i, i^2, \) and \( i^2+i \) for each of the integers \( i \) from 10 to 20. Here is the MAPLE program for doing this (shift return allows you to go to the next line without MAPLE trying to compute anything):

```maple
> for i from 10 to 20
  do
    i, i^2, i^2+i
  od;
```

I guess we can think of the "od" as finishing the "do" command. Also you could have put all this on one line; however, you may find it easier to follow the logic of such a program if you use spacing somehow to identify pieces that are related to each other, the "do" and the "od" for example. This is more important if your programs are nested. For example let's use two "for" programs in conjunction. Before executing the following program see if you can describe what it will do.

```maple
> for i from 5 to 10
  do
    for j from i to 10
      do
        print(i, j, lcm(i, j))
      od;
  od;
```

Notice what happens if you leave off the print command in this program.

The second important programming construct is the "if" program, better known to math students as the conditional "if then" statement. The general syntax is

```maple
if <some statement which MAPLE can verify> then <some MAPLE command sequence> fi;
```

This can be continued with a alternative "else" statement

```maple
if <some statement which MAPLE can verify> then <some MAPLE command sequence> else <an alternative command sequence> fi;
```
Let's look at two examples. Again try to guess what will happen prior to executing them.

```plaintext
> if gcd(1002,552)>1
    then
    print(great)
    fi;

> if gcd(1002,37)>1
    then
    print(great)
    else
    print(rats)
    fi;
```

Now let's combine all into one. Again guess the output, if possible prior to executing the program.

```plaintext
> for i from 1 to 10
do
    for j from 1 to 10
do
        if i<j
            then print(i,j,j-i)
            else print(i,j,i-j)
            fi;
        od;
    od;
```

Notice that we just computed the absolute value of i-j for all i and j from 1 to 10.