# Summary of Some Common Jarnac Language Constructs

**Sample model definition:**

```
p = defn cell
     var S1,S2;    // floating species
     ext X0,X1;    // boundary species (remain constant)

     // Declare each reaction and its associated rate law
     J1: X1 -> S1; k1*X0-k2*S2;
     J2: S1 -> S2; Vm*S1/(Km+S1);
     J3: S2 -> X1; k3*S2^n;
end;
```

**// Initialize all the parameters and variables**

```
p.X0=1.2;   p.X1=0.0;
p.S1=0.001; p.S2=0.001;
p.k1=1.2;   p.k2=5.6;
p.Vm=10.5;  p.Km=0.4;
p.k3=5.6;   p.n=3.5;
```

**// Define some convenient variables if necessary**

```
TimeStart=0;
TimeEnd=10;
NumberOfDataPoints=100;
```

**// Perform the simulation**

```
m = p.sim.eval(TimeStart, TimeEnd, NumberOfDataPoints,
       [<p.Time>, <p.S1>, <p.S2>, <p.J1>]);
```

**// The above statement returns a matrix which we can graph**

```
graph(m);
```

**Reaction specifications, examples:**

```
J1: 2 A -> B + C; k1*A^2*B;
J2: 2 ADP -> ATP + AMP; k1*ADP^2 – k2*ATP*AMP;
```

Reactions can be named, eg J1 and J2 above, this allows the rate of a reaction to be accessed.

**For-Loop :**

```
for i=1 to 100 do
    begin
      <code>
    end;
```

**If-statement:**

```
if a == 4 then
   // begin-end only needed for multiple statements
   begin
     <code>
   end
else
   begin
     <code>
   end;
```

**printing:**

```
println "statement”, x, y, z;
```

**Matrix operations:**

```
m1 = matrix (10,5);  // Construct a 10 x 5 matrix
v1 = matrix (10,1);  // Construct a column vector
v2 = matrix (1,6);   //  a row vector

m2 = aug (m1,v1);    // Augment the columns of m1 and
                       v2 together
m3 = augr (m1,v2);   // Augment the columns of m1 and
                       v2 together

println m[3,4];      // Access a particular element

mt = tr (m);         // Compute the transpose
v = m[5];            // Copy an entire row

mt = tr (m);
col2 = mt[2];        // Use this trick to extract a
                       column, in this case the 2nd
                       column
```