> restart;
> Digits :=80;#To see things clearly
> f := x-> x^3-5;

bisection problem

> f(1);
> f(2);

#different signs, so bisection will work.
Below, I should use a while loop, so we can halt if f(x[j]) is close enough to 0.

> a:=1.0;
b:=2.0;
for j from 1 to 3 do
c:= (a+b)/2:
print(j,c):
if (f(c)*f(a) < 0) then
  b:=c: else
  a:=b:
  b:= c:
end if:
end do:

Since $2^{-N} < 0.001$ for the first time when $N=10$

> a:=1.0;
b:=2.0;
for j from 1 to 10 do
c:= (a+b)/2:
print(j,c):
if (f(c)*f(a) < 0) then
  b:=c: else
  a:=b:
  b:= c:
end if:
end do:
A smaller error than expected (unusual for the bisection method).

Newton method problem

```plaintext
> N := 6; # number of times we iterate
> Newton := proc(f, v)
> local J, x;
> J := unapply(diff(f(x), x), x);
> v - f(v)/J(v);
> end proc;
> x := 2:
> for j from 1 to N do
> x := evalf(Newton(f, x)):
> n||j := x; # sets n_j = x
> print(x):
> od:
```

```
           Newton := proc(f, v) local J, x; J := unapply(diff(f(x), x), x); v - f(v) / J(v) end proc
1.7099609375000000000000000000000000000000000000000000000000000000000000000000000
0.00001500917699893531088725438601098680551105430549243828617074442959205041732162571870
1.002018900220450328939045401082795737666800785374713893894415032333156031462264979150
7403419693392331556442949
```

The bisection method yielded a better estimate than expected, but Newton is much better (as expected).

Secant method problem

```plaintext
> v := 1;
> w := 2;
> for j from 1 to N do
> J := (f(w) - f(v))/(w - v);
> x := evalf(w - f(w)/J):
> s||j := x; # sets s_j equal to x
> print(x):
> v := w;
> w := x;
> od:
```

```
       v := 1
       w := 2
1.5714285714285714285714285714285714285714285714285714285714285714285714285714285
1.687898089171974522292993605732484076433121019108280254777070063694267515923567
1.7118829384306178606125869793715435781232409753302616696403781019470062705025865
1.7099511304569673328160964699692664578676857584255165492934259744105044583913054
```

The secant method yielded a better estimate than expected, but Newton is much better (as expected).
for j from 1 to N do abs(n||j-s||j); od;

Digits:=80;
for j from 1 to N-1 do
print(j,log(abs(evalf(5^(1/3))-n||(j+1)))/log(abs(evalf(5^(1/3))-n||(j))),
log(abs(evalf(5^(1/3))-s||(j+1)))/log(abs(evalf(5^(1/3))-s||(j))));
od:

You can clearly see the quadratic convergence of Newton's method, but the (1+sqrt(5)/2= 1.64.. convergence of the secant method is not there yet!

evalf[5]((1+sqrt(5))/2);
1.6180