```
> restart;
> Digits :=80;#To see things clearly
```

$$Digits := 80 \tag{1}$$

```
> f := x-> x^3-5;
```

$$f := x \rightarrow x^3 - 5 \tag{2}$$

bisection problem

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

$$-4$$

$$3 \tag{3}$$

```
> #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:
```

$$a := 1.0$$

$$b := 2.0$$

1, 1.5000000000000000000000000000000000000000000000000000000000000000000000000000000

2, 1.7500000000000000000000000000000000000000000000000000000000000000000000000000000

3, 1.6250000000000000000000000000000000000000000000000000000000000000000000000000000 (4)

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:
```

$$a := 1.0$$

$$b := 2.0$$

1, 1.5000000000000000000000000000000000000000000000000000000000000000000000000000000

2, 1.7500000000000000000000000000000000000000000000000000000000000000000000000000000

3, 1.6250000000000000000000000000000000000000000000000000000000000000000000000000000

4, 1.6875000000000000000000000000000000000000000000000000000000000000000000000000000

5, 1.7187500000000000000000000000000000000000000000000000000000000000000000000000000

6, 1.7031250000000000000000000000000000000000000000000000000000000000000000000000000

7, 1.7109375000000000000000000000000000000000000000000000000000000000000000000000000

8, 1.7070312500000000000000000000000000000000000000000000000000000000000000000000000

9, 1.7089843750000000000000000000000000000000000000000000000000000000000000000000000

10, 1.7099609375000000000000000000000000000000000000000000000000000000000000000000000 (5)

```
> abs
```

```
(1.7099609375000000000000000000000000000000000000000000000000000000000000000000
000000-evalf(5^(1/3)));
```

0.00001500917669698935310887254386010986805511054305492438286170744295920504173216257187 0\
100201890022045032893904540180871975766468007853747138938944150323331560314622649791 50\
70434196933923315564429 49 (6)

A smaller error than expected (unusual for the bisection method).

Newton method problem

```
> N:=6;#number of times we iterate
```

$N := 6$ (7)

```
> 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:
```

$Newton := \mathbf{proc}(f, v)\ \ \mathbf{local}\ J, x;\ J := unapply(diff(f(x), x), x);\ v - f(v)/J(v)\ \ \mathbf{end\ proc}$

1.7500000000000000000000000000000000000000000000000000000000000000000000000000000
1.7108843537414965986394557823129251700680272108843537414965986394557823129251701
1.7099764289169747925308851148304629067312269434384796740624850635618180307443196
1.7099759466768329886918959701886330858146021596408525435051395072867306254636592
1.7099759466766969893531088833602813504545357482234636234886284982630468435483473
1.7099759466766969893531088725438601098680551105430549928019343155635451047310828 (8)

```
> abs
  (1.7099759466766969893531088725438601098680551105430549928019343155635451047
  310828-evalf(5^(1/3)));
```

$6.8419072608119249184226909 6\ 10^{-53}$ (9)

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

secant method problem

```
> 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:
```

$v := 1$

$w := 2$

1.5714285714285714285714285714285714285714285714285714285714285714285714285714286
1.6878980891719745222929936305732484076433121019108280254777070063694267515923567
1.7118829384306178606125869793715453781232409753302616696403781019470062705025865
1.7099511304569673328160964699692664578678557584255165492934259744105044583913054

$$1.7099759190215602918206135076340624973551675978270698679646611467147159423651228$$

$$1.7099759466770983415462122521701384907998743111728474961008554223090830814159916 \qquad (10)$$

```
> for j from 1 to N do abs(n||j-s||j);od;
```

$$0.17857142857142857142857142857142857142857142857142857142857142857142857142857714$$

$$0.022986264569522076346462151739676762424715108973525716018891633086355561332 8134$$

$$0.0019065095136430680817018645410824713920140318917819955778930383851882397582669$$

$$0.000024816219865655875799500219366627946746401215335994211713532876226167 0723538$$

$$2.7655136697532495375726218853099368150396393755523967351548330901183224 5\ 10^{-8}$$

$$4.013521931033796262783809318192006297925032989211067455379766849088\ 10^{-13} \qquad (11)$$

```
> 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:
```

$$Digits := 80$$

1, 2.1762648261162657994836028705626004001515721973746319801422712305445639192737767,
1.9292169143923618675159665329883006104673614056047194986338664050668318152443395

2, 2.0766992241305279558261059682958794686120881296998298991939410861021482329440046,
1.6422587235624132001400415769038669049711407337259669537147715823374677239291713

3, 2.0368845786177201643433027213517342823001663930881204640805714142862825975064152,
1.6933290562139756045847601065626463256845515387501201595489199135768128307548455

4, 2.0181083175513999657807452250455946505532170328478572611953169455476645267500906,
1.6412139622049079394681573766417832831987277383388558160683220175636156786855537

5, 2.0089729165644426212880209839136974408228363868709649811331905974255175264156207, (12)
1.6401305448377699785204603038146588235500597970556033652807273839511357744836501

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);
```

$$1.6180 \qquad (13)$$