```
> restart;
> Digits:=15;
```

$$Digits := 15$$

```
> eq:=x^2-3*x+2*x^0.05=1;
```

$$eq := x^2 - 3\,x + 2\,x^{0.05} = 1$$

Let us see what Maple provides as the solution

```
> #solve(eq);
> fsolve(eq,x=0.1);
```

$$0.335390202475623$$

We first rewrite the function f

```
> f:=(-1+x^2+2*x^(0.05))/3;
```

$$f := -\frac{1}{3} + \frac{x^2}{3} + \frac{2\,x^{0.05}}{3}$$

convert f to a procedure that takes input (x) and gives output (f)

```
> F:=unapply(f,x);
```

$$F := x \rightarrow -\frac{1}{3} + \frac{1}{3}\,x^2 + \frac{2}{3}\,x^{0.05}$$

try F at different x

```
> F(0);F(1);
```

$$\frac{-1}{3}$$

$$0.666666666666667$$

Let us start with an initial guess of 0.1 for xold

```
> xold:=0.1;
```

$$xold := 0.1$$

```
> xnew:=F(xold);
```

$$xnew := 0.264167292089164$$

keep iterating by updating xold and copying and pasting

```
> xold:=xnew;
```

$$xold := 0.264167292089164$$

```
> xnew:=F(xold);
```

$$xnew := 0.313666822825213$$

```
> xold:=xnew;
```

$$xold := 0.313666822825213$$

```
> xnew:=F(xold);
```

$$xnew := 0.328580377406624$$

It is convenient to write a for loop

```
> xold:=0.1;
```

```
xold := 0.1
```

```
> for i from 1 to 5 do xnew:=F(xold);xold:=xnew;od;
xnew := 0.264167292089164

xold := 0.264167292089164

xnew := 0.313666822825213

xold := 0.313666822825213

xnew := 0.328580377406624

xold := 0.328580377406624

xnew := 0.333235936158743

xold := 0.333235936158743

xnew := 0.334706724391955

xold := 0.334706724391955
```

We see that for this problem, value converged in 5 steps. It is convenient to use error criterion instead of using a for loop

```
> xold:=0.1;
xold := 0.1
```

An intial value for error was set to be 1. You can copy paste statements from above. Don't execute before you combine the executing statements.

```
> Err:=1;
Err := 1

> while Err > 1e-5 do
> xnew:=F(xold);
> Err:=abs(xnew-xold);
> xold:=xnew;
> print(Err,xnew);
> end;
> Err;xnew;
xnew := 0.264167292089164

Err := 0.164167292089164

xold := 0.264167292089164

0.164167292089164 , 0.264167292089164

xnew := 0.313666822825213

Err := 0.049499530736049

xold := 0.313666822825213

0.049499530736049 , 0.313666822825213
```

$xnew := 0.328580377406624$

$Err := 0.014913554581411$

$xold := 0.328580377406624$

$0.014913554581411 , 0.328580377406624$

$xnew := 0.333235936158743$

$Err := 0.004655558752119$

$xold := 0.333235936158743$

$0.004655558752119 , 0.333235936158743$

$xnew := 0.334706724391955$

$Err := 0.001470788233212$

$xold := 0.334706724391955$

$0.001470788233212 , 0.334706724391955$

$xnew := 0.335173156916453$

$Err := 0.000466432524498$

$xold := 0.335173156916453$

$0.000466432524498 , 0.335173156916453$

$xnew := 0.335321257175903$

$Err := 0.000148100259450$

$xold := 0.335321257175903$

$0.000148100259450 , 0.335321257175903$

$xnew := 0.335368299712078$

$Err := 0.000047042536175$

$xold := 0.335368299712078$

$0.000047042536175 , 0.335368299712078$

$xnew := 0.335383244129470$

$Err := 0.000014944417392$

$xold := 0.335383244129470$

$0.000014944417392 , 0.335383244129470$

$xnew := 0.335387991839658$

$Err := 0.4747710188\ 10^{-5}$

$xold := 0.335387991839658$

$$0.4747710188\ \ 10^{-5}, 0.3353879991839658$$

$$0.4747710188\ \ 10^{-5}$$

$$0.3353879991839658$$

Now that we have an approach to solve a single nonlinear equation by this method, can we write a general procedure? If so what are the inputs and outputs?

Inputs are f (which is a fucntion of x), initial guess for xold, and tolerance (1e-5) was used. you can copy paste the code above to write the procedure below. First copy paste and write a procedure to wrap around it.

```
> SuccSub:=proc(f,xguess,tol)
> end proc;
```

$$SuccSub := \mathbf{proc}(f, xguess, tol)\ \ \mathbf{end\ proc}$$

The procedure above is a dummy procedure as it is not doing anything. We want to do the following steps from above in this procedure and get

```
> F:=unapply(f,x);
> xold:=xguess;
> Err:=1;
> while Err > tol do
> xnew:=F(xold);
> Err:=abs(xnew-xold);
> xold:=xnew;
> end;
> Err,xnew;
```

Let us put the above lines into the procedure

```
> SuccSub:=proc(f,xguess,tol)
> F:=unapply(f,x);
> xold:=xguess;
> Err:=1;
> while Err > tol do
> xnew:=F(xold);
> Err:=abs(xnew-xold);
> xold:=xnew;
> end;
> Err,xnew;
> end proc;
```

```
Warning, `F` is implicitly declared local to procedure `SuccSub`

Warning, `xold` is implicitly declared local to procedure `SuccSub`

Warning, `Err` is implicitly declared local to procedure `SuccSub`

Warning, `xnew` is implicitly declared local to procedure `SuccSub`
```

```
SuccSub := proc(f, xguess, tol)
local F, xold, Err, xnew;
        F := unapply(f, x);
        xold := xguess;
        Err := 1;
        while tol < Err do xnew := F(xold); Err := abs(xnew – xold); xold := xnew
        end do;
        Err, xnew
end proc
```

Note that F, xold, Err and xnew are defined locally in the procedure and warning can avoided by defining local variables in the procedure

```
> SuccSub:=proc(f,xguess,tol)
local F, xold, Err, xnew;
> F:=unapply(f,x);
> xold:=xguess;
> Err:=1;
> while Err > tol do
> xnew:=F(xold);
> Err:=abs(xnew-xold);
> xold:=xnew;
> end;
> Err,xnew;
> end proc;
>
```

```
SuccSub := proc(f, xguess, tol)
local F, xold, Err, xnew;
        F := unapply(f, x);
        xold := xguess;
        Err := 1;
        while tol < Err do xnew := F(xold); Err := abs(xnew – xold); xold := xnew
        end do;
        Err, xnew
end proc
```

Let us try if this works for us

```
> f;
```

$$-\frac{1}{3}+\frac{x^2}{3}+\frac{2\,x^{0.05}}{3}$$

```
> xguess:=0.1;
```

$$xguess := 0.1$$

```
> tol:=1e-5;
```

$$tol := 0.00001$$

```
> SuccSub(f,xguess,tol);
```

$$0.4747710188\ \ 10^{-5}, 0.335387991839658$$

What is the advantage of this procedure? We can easily use this for many problems. you can store this in the computer and call for future problems as well. For example, let us try

```
> eq:=x^2-3*x=2;
```

$$eq := x^2 - 3\,x = 2$$

```
> fsolve(eq,x=0.1);
```

$$-0.561552812808830\ ,\ 3.56155281280883$$

```
> f:=(x^2-2)/3;
```

$$f := \frac{x^2}{3} - \frac{2}{3}$$

```
> SuccSub(f,xguess,tol);
```

$$0.8020602681\ \ 10^{-5}, -0.561550628043543$$