f.m

```
1 function y = f(x)
2     y = x - 2.^-x;
```

Command Window

Command Window

```
>> fun = @f
fun = @f
>> x0 = 0
x0 = 0
>> z = fzero(fun, x0)
z = 0.6412
>>
```

```
>> fun = @f
fun = @f
>> a = 0;b = 1;delta = 10^-5;
>> [c,err,yc,k] = bisect(fun,a,b,delta,20)
error: parse error near line 10 of file C:\User

  break must appear within a loop

>>> if ya*yb>0,break,end
                  ^
>> [c,err,yc,k] = bisect(fun,a,b,delta,20)
error: parse error near line 11 of file C:\User

  break must appear within a loop

>>>   break,end
        ^
>> [c,err,yc,k] = bisect(fun,a,b,delta,20)
error: parse error near line 11 of file C:\User

  break must appear within a loop

>>>   break
        ^
>> [c,err,yc,k] = bisect(fun,a,b,delta,20)
c = 0.6412
err = 7.6294e-06
yc = 2.3101e-08
k = 17
>>
```

```
>> fun = @f
fun = @f
>> [c, err, yc] = regula(fun, 0, 1, 10^-5, 10^-10, 20)
c = 0.6412
err = 0.3206
yc = 1.0914e-06
>>
```

```
>> fun = @f
fun = @f
>> [p1,err,k,y]=secant(fun,0,1,10^-5,10^-10,20)
warning: Matlab-style short-circuit operation performed for operator |
warning: called from
    secant at line 18 column 32

warning: Matlab-style short-circuit operation performed for operator |
warning: called from
    secant at line 18 column 32

p1 = 0.6412
err = 2.5153e-06
k = 4
y = 3.5986e-10
>>
```

```
>> fun = @f
fun = @f
>> dfun = @df
dfun = @df
>> [p0,err,k,y]=newton(fun,dfun,1,10^-5,10^-10,20)
p0 = 0.6412
err = 1.6710e-05
k = 3
y = -4.3008e-11
>>
```

The Newton method is the best with 3 iterations, and the secant method close with 4 iterations. The other two methods have much higher costs because they take linear time. Where the Newton and secant methods take quadratic time.

```
>> fun = @cosfunc
fun = @cosfunc
>> dfun = @dcosfunc
dfun = @dcosfunc
>> function y = cosfunc(x)

>> [p0,err,k,y]=newton(fun,dfun,1,10^-5,10^-7,15)
p0 = 2.1897e-03
err = 0.032014
k = 5
y = -5.2498e-09
>> This is quadratic because x=0 it is converging to is a simple root.
```

```
>> [p0,err,k,y]=mnewton(fun,dfun,1,1,10^-5,10^-7,15)
p0 = 2.1897e-03
err = 0.032014
k = 5
y = -5.2498e-09
>> No faster convergence it is order 1
```

```
function [p0,err,k,y]=mnewton(f,df,p0,m,delta,epsilon,max1)
%Input - f is the object function input as a string 'f'
%- df is the derivative of f input as a string 'df'
%- p0 is the initial approximation to a zero of f
%- m is the order of the root being converged to
%- delta is the tolerance for p0
%- epsilon is the tolerance for the function values y
%- max1 is the maximum number of iterations
%Output - p0 is the Newton-Raphson approximation to the zero
%- err is the error estimate for p0
%- k is the number of iterations
%- y is the function value f(p0)
for k=1:max1
  p1=p0-(m*feval(f,p0))/feval(df,p0);
  err=abs(p1-p0);
  relerr=2*err/(abs(p1)+delta);
  p0=p1;
  y=feval(f,p0);
  if (err<delta)|(relerr<delta)|(abs(y)<epsilon),break,end
end
```