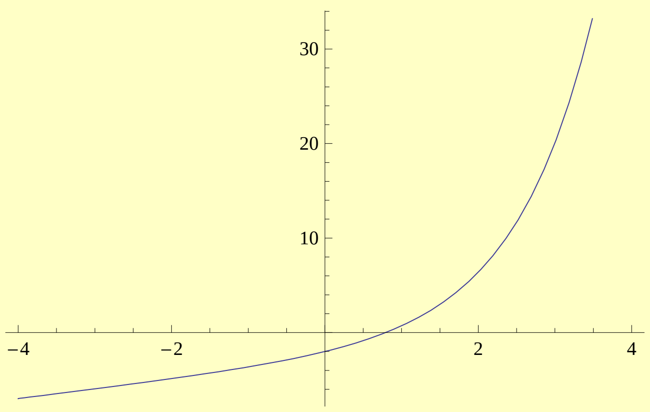
Newton’ s Method

Example : Find the root of $f(x) = e^x + x - 3 = 0$

If we draw the graph of f (x), we can see that the root of f (x) = 0 is the x - coordinate of the point where the curve intersects with the x - axis.

In[1]:=
```
f = Exp[x] + x - 3;
Plot[f, {x, -4, 4}]
```

Out[2]=



The derivative and tangent line at a given point (x0,y0) = (x0, f [x0]) is given by f ’[x0] * (x-x0) + y0

In[3]:=
```
df = D[f, x];
slope = df /. x → x0;
y0 = f /. x → x0;
tangentline = slope * (x - x0) + y0
```

Out[6]= $-3 + e^{x0} + (1 + e^{x0})(x - x0) + x0$

To find the root, we will use the Newton’ s iteration. First, choose a starting point, for example, x0=2. Clearly this is not the root. Draw the tangent line at x0=2:

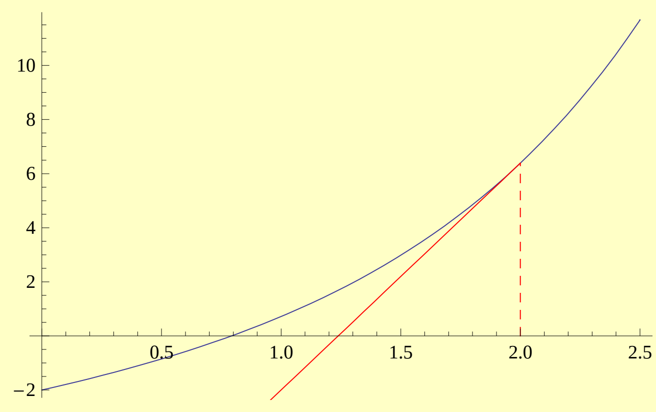
In[7]:=

```
plotf = Plot[f, {x, 0, 2.5}];
plotp0 = Graphics[{Red, Dashed, Line[{{2, 0}, {2, f /. x → 2}}] } ];
tangentline0 = tangentline /. x0 → 2
plott0 = Plot[tangentline0, {x, 0, 2}, PlotStyle → {Red}];
Show[plotf, plotp0, plott0]
```

Out[9]=

$$-1 + e^2 + \left(1 + e^2\right) (-2 + x)$$

Out[11]=



The tangent line intersect with the x - axis at a point. Let us find the coordinate of this point :

In[12]:=

```
NSolve[tangentline0 == 0, x]
```

Out[12]=

```
{{x → 1.23841}}
```

From the graph, we can see that 1.23841 is closer to the solution of f(x)=0 than 2. Now repeat the above process, draw the tangent line at 1.23841.

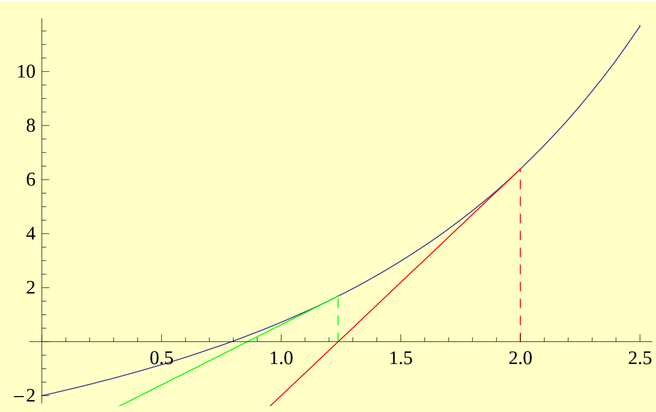
In[13]:=

```
plotp1 = Graphics[{Green, Dashed, Line[{{1.23841, 0}, {1.23841, f /. x → 1.23841}}] } ];
tangentline1 = tangentline /. x0 → 1.23841
plott1 = Plot[tangentline1, {x, 0, 1.23841}, PlotStyle → {Green}];
Show[plotf, plotp0, plott0, plotp1, plott1]
```

Out[14]=

$$1.68853 + 4.45012 (-1.23841 + x)$$

Out[16]=



Find the intersection point of the second tangent line with the x - axis :

In[17]:=
```
NSolve[tangentline1 == 0, x]
```

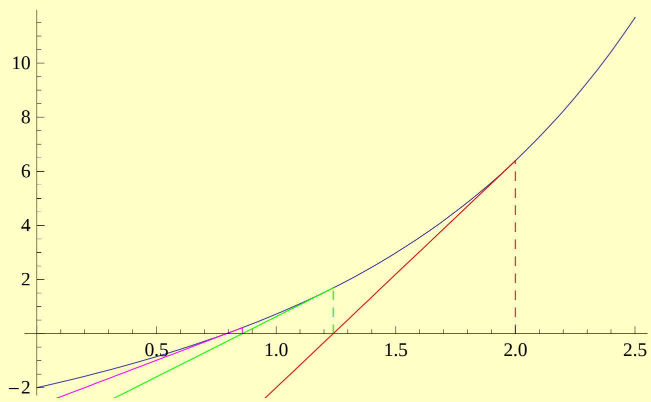
Out[17]=
```
{{x → 0.858975}}
```

0.858975 is closer to the solution of f (x) = 0 than both 1.23841 and 2. Let repeat the process again using the new point:

In[18]:=
```
plotp2 =
  Graphics[{Magenta, Dashed, Line[{{0.858975, 0}, {0.858975, f /. x → 0.858975}}] }];
tangentline2 = tangentline /. x0 → 0.858975
plott2 = Plot[tangentline2, {x, 0, 0.858975}, PlotStyle → {Magenta}];
Show[plotf, plotp0, plott0, plotp1, plott1, plotp2, plott2]
```

Out[19]=
```
0.219715 + 3.36074 (-0.858975 + x)
```

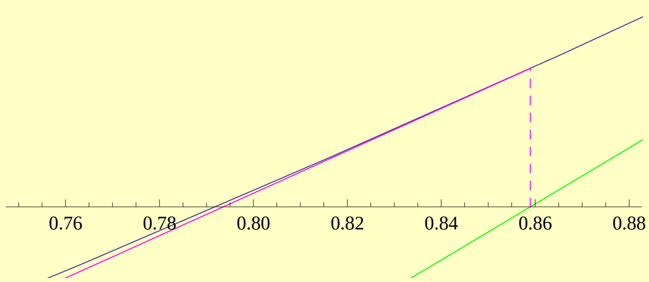
Out[21]=



We can zoom-in to have a better view :

In[22]:=
```
Show[{plotf, plotp0, plott0, plotp1, plott1, plotp2, plott2},
 PlotRange → {{0.75, 0.88}, {-.1, 0.5}}]
```

Out[22]=



The intersection of the magenta tangentline with the x - axis is very close to the actural solution of f (x) = 0. We can expect that repeating the above process will give us even better approximation to the solution. In *Mathematica*, there’s a neat way to complete the entire process in one single command. We first define a function called NewtonsMethodList :

In[23]:=
```
NewtonsMethodList[f_, {x_, x0_}, n_] :=
 N[NestList[# - Function[x, f][#] / Derivative[1][Function[x, f]][#] &, x0, n]]
```

In the above definition, (f_, x_, x0_, n_) are input parameters. f_ is the expression to be solved, x_ is the name of the unknown variable, x0_ is the starting point, n_ is the number of iterations (repeat the tangent line process n_ times). Now let’ s use this function to find the root of $f(x) = e^x + x - 3 = 0$.

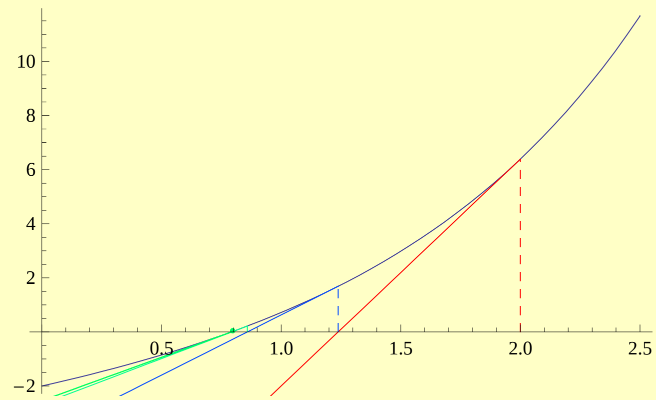
In[24]:=
```
values = NewtonsMethodList[Exp[x] + x - 3, {x, 2}, 5]
```

Out[24]=
```
{2., 1.23841, 0.858974, 0.793598, 0.792061, 0.79206}
```

The values given in the above list are exactly the intersection of tangentlines with the x - axis, as we have seen earlier. They converge to 0.79206, which is the solution of f(x)=0. We can draw the graph of the approximation.

In[25]:=
```
Show[{plotf, Table[Plot[tangentline, {x, -1, x0}, PlotStyle → {Hue[x0 / 2]}], {x0, values}],
   Table[Graphics[{Hue[x0 / 2], Dashed, Line[{{x0, 0}, {x0, f /. x → x0}}] } ], {x0, values}]}]
```

Out[25]=



Mathematica also provide a built - in function "FindRoot" to solve the problem.It gives the same answer as what we have seen in the above.

In[26]:=
```
FindRoot[Exp[x] + x - 3 == 0, {x, 2}]
```

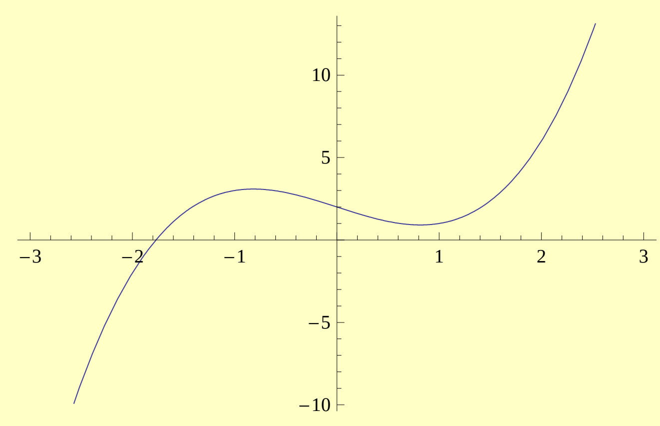
Out[26]=
```
{x → 0.79206}
```

Example : Choosing the first point is sometimes important.

Consider the root of $f(x) = x^3 - 2x + 2 = 0$. From the following graph, it should be between -2 and -1.

In[27]:=
```
f = x^3 - 2 * x + 2;
plotf = Plot[f, {x, -3, 3}]
tangentline = (D[f, x] /. x → x0) * (x - x0) + (f /. x → x0);
```

Out[28]=

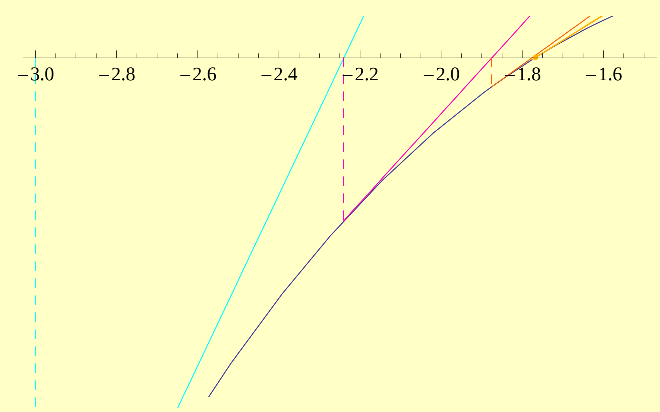


If we use the starting point x0 = -3, we will get the correct solution around -1.76929.

In[30]:=
```
values = NewtonsMethodList[x^3 - 2 * x + 2, {x, -3}, 5]
Show[{plotf, Table[Plot[tangentline, {x, -1, x0}, PlotStyle → {Hue[x0 / 2]}], {x0, values}],
   Table[
   Graphics[{Hue[x0 / 2], Dashed, Line[{{x0, 0}, {x0, f /. x → x0}}] } ], {x0, values}]},
    PlotRange → {{-3, -1.5}, {-10, 1}}]
```

Out[30]= {-3., -2.24, -1.87537, -1.77656, -1.76933, -1.76929}

Out[31]=

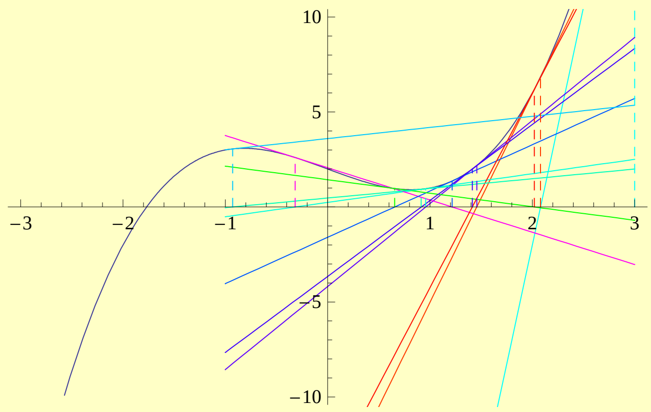


However, if we choose the starting point x = 3, we won' t get the correct solution. Because the Newton's iteration can not pass through the local minimum near x=1. The tangentlines will bouncing back and forth over this point.

In[32]:=
```
values = NewtonsMethodList[x^3 - 2 * x + 2, {x, 3}, 10]
Show[{plotf, Table[Plot[tangentline, {x, -1, 3}, PlotStyle → {Hue[x0 / 2]}], {x0, values}],
   Table[
   Graphics[{Hue[x0 / 2], Dashed, Line[{{x0, 0}, {x0, f /. x → x0}}] } ], {x0, values}]},
    PlotRange → {{-3, 3}, {-10, 10}}]
```

Out[32]=
```
{3., 2.08, 1.4571, 0.958312, -0.317641,
 1.2161, 0.655382, 2.01989, 1.41429, 0.914292, -0.928409}
```

Out[33]=



It is the same if we use "FindRoot" with these two different starting points.

In[34]:=
```
FindRoot[f, {x, -3}]
```

Out[34]=
```
{x → -1.76929}
```

In[35]:=
```
FindRoot[f, {x, 3}]
```

```
FindRoot::lstol :
 The line search decreased the step size to within tolerance specified by
   AccuracyGoal and PrecisionGoal but was unable to find a sufficient
   decrease in the merit function.  You may need more than MachinePrecision
   digits of working precision to meet these tolerances. >>
```

Out[35]=
```
{x → 0.816497}
```