x	-1.	-0.78947	-0.57895	-0.36842	0.89474	1.10526	2.36842	2.57895
f(x)	-4.1	-2.21366	-0.81206	0.16080	0.12930	-0.33152	-0.73531	0.11344

Table 1: values of cubic function equation (1)

$ Supplement Note 2: Find Roots of Function by Using Python

Let us use python to find the roots $f(x) = 0$ of the cubic function

$$f(x) = 1 + x - 3.1\ x^2 + x^3\ . \tag{1}$$

The first step is to plot the graph of the function $y = f(x)$. This is done in the python code *PlotFunc.py* using the *pyplot* program in the **matplotlib** package. The graph is plotted in Figure 1 and some of the values of $f(x)$ are listed in Table 1.

From Figure 1 or Table 1 we see that there are three roots located in the intervals $[x_d, x_u]$ given by $[-0.57895, -0.36842]$, $[0.89474, 1.10526]$ and $[2.36842, 2.57895]$. This is due to the fact that the values of f change sign in those interval. The relevant values of f $[f_d, f_u]$ are given by $[-0.81206, 0.16080]$, $[0.12930, -0.33152]$ and $[-0.73531, 0.11344]$ respectively.

Two popular approaches for the solutions are listed in *RootFinding.py*. The first one is the *bisection method*: in which the mid point is selected at each step.

A slightly better approach is via the *false position method* where the new root is obtained by a linear approximation over the region $[x_d, x_u]$. We have then

$$x_0 = x_d - f_d \frac{x_u - x_d}{f_u - f_d} = \frac{x_d f_u - x_u f_d}{f_u - f_d} \tag{2}$$

The solution may be obtained by iterations. This is done in *method 2* of *FootFinding.py*. As may be seen, the convergence rate is much faster.

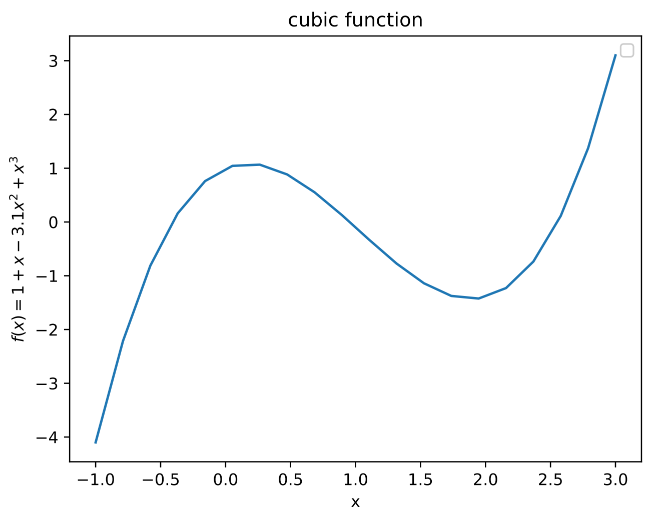

Figure 1: Graph of a Cubic Function

```
# PlotFunc.py :plot cubic function  f(x) = 1+ x -3.1 x**2 +  x**3
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-1,3,20)
x2, x3 = x**2, x**3
f = 1+ x- 3.1*x2 + x3
print('x=\n',x)
print('f(x)=\n', f)
plt.title('Cubic Equation')
plt.plot(x,f)
plt.ylabel(r'$f(x)= 1+ x- 3.1 x^2+ x^3$')
plt.title('cubic function')
plt.xlabel('x')
plt.legend()
fig = plt.gcf()
fig.savefig('CubicEquation.eps', format='eps')
plt.show()


# RootFinding.py :Find roots for cubic function  f(x) = 1+ x -3.1 x**2 +  x**3
```

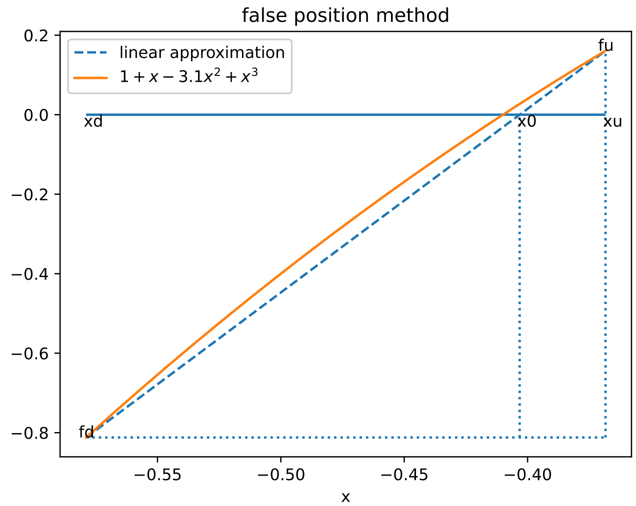

Figure 2: False Position Method

```
import numpy as np
import matplotlib.pyplot as plt
def fc(x):
    f = 1 + x*(1 + x*(-3.1 + x))
    return f
#method 1
print('method 1: bisection')
xu, xd = -0.36842105, -0.57894737
fu, fd = fc(xu), fc(xd)
print('xd, fd=', xu, fu)
print('xu, fu=', xd, fd)
for i in range(20):
    x = (xu+ xd)/2
    fx = fc(x)
    if fx < 0: xd = x
    else:  xu = x
    print('x,fc(x)=', x,fc(x))
#method 2
print('method 2: false position method')
```

```
xu, xd = -0.36842105 ,-0.57894737
fu, fd = fc(xu), fc(xd)
print('xd, fd=', xu, fu)
print('xu, fu=', xd, fd)
plt.plot([xd,xu],[fd,fu],'--',label='linear approximation')
x = np.linspace(xd,xu,20)
fx= []     # empty list
for t in x:
    fx.append(fc(t))    # fx contains the values of fc(t) for each t in x
x0 = (xd*fu - fd*xu)/(fu-fd)
plt.hlines(0,xd, xu)
plt.vlines(x0,fd, 0, linestyles='dotted')
plt.vlines(xu,fd, fu, linestyles='dotted')
plt.text(xd-0.001, -0.03,r'xd')
plt.text(xu-0.001, -0.03,r'xu')
plt.text(x0-0.001, -0.03,r'x0')
plt.text(xd-0.003, fd,r'fd')
plt.text(xu-0.003, fu,r'fu')
plt.hlines(fd,xd, xu, linestyles='dotted')
plt.plot(x,fx,label=r'$1+ x- 3.1 x^2+ x^3$')
plt.title('false position method')
plt.xlabel('x')
plt.legend()
fig = plt.gcf()
fig.savefig('FalsePosition.eps', format='eps')
plt.show()
for i in range(6):
    x = (xd*fu -xu *fd)/(fu-fd)
    f = fc(x)
    if f<0: xd,fd = x,f
    else:  xu,fu = x,f
    print(r'{:12.9f}, {:15.8e}'.format(x,f))
print('testing result', x,  fc(x))
```