In [1]:
```python
# 1 Bisection
a = float(input("Enter a: "))
b = float(input("Enter b: "))
e = float(input("Enter the tolerance (e): "))

def f(num):
    return (num**4) + 3*(num**2) + num - 10

k = 1
c = (a + b) / 2


print("{:<10} {:<10} {:<10} {:<10} {:<10} {:<10}".format("Iteration", "a", "f(a)", "b", "f(b)",
print("-" * 60)

while True:
    fc = f(c)


    print("{:<10} {:<10.5f} {:<10.5f} {:<10.5f} {:<10.5f} {:<10.5f} {:<10.5f}".format(
        k, a, f(a), b, f(b), c, fc))

    if f(a) * f(c) < 0:
        b = c
    else:
        a = c

    c_prev = c
    c = (a + b) / 2

    if abs(c - c_prev) < e:
        break

    k += 1

print("\nNumber of iterations are: ", k)
print("Root of the equation is: ", c)
```

```
Iteration  a          f(a)       b          f(b)       c
------------------------------------------------------------
1          1.00000    -5.00000   2.00000    20.00000   1.50000    3.31250
2          1.00000    -5.00000   1.50000    3.31250    1.25000    -1.62109
3          1.25000    -1.62109   1.50000    3.31250    1.37500    0.62134
4          1.25000    -1.62109   1.37500    0.62134    1.31250    -0.55199
5          1.31250    -0.55199   1.37500    0.62134    1.34375    0.02116
6          1.31250    -0.55199   1.34375    0.02116    1.32812    -0.26873
7          1.32812    -0.26873   1.34375    0.02116    1.33594    -0.12462
8          1.33594    -0.12462   1.34375    0.02116    1.33984    -0.05194
9          1.33984    -0.05194   1.34375    0.02116    1.34180    -0.01544
10         1.34180    -0.01544   1.34375    0.02116    1.34277    0.00285
11         1.34180    -0.01544   1.34277    0.00285    1.34229    -0.00630
12         1.34229    -0.00630   1.34277    0.00285    1.34253    -0.00172
13         1.34253    -0.00172   1.34277    0.00285    1.34265    0.00056
14         1.34253    -0.00172   1.34265    0.00056    1.34259    -0.00058
15         1.34259    -0.00058   1.34265    0.00056    1.34262    -0.00001
16         1.34262    -0.00001   1.34265    0.00056    1.34264    0.00028

Number of iterations are:  16
Root of the equation is:  1.3426284790039062
```

In [3]:
```python
#2 False Position
a = float(input("Enter a: "))
b = float(input("Enter b: "))
e = float(input("Enter the tolerance (e): "))

def f(num):
    return (num**3) - num - 4
#  (a * func(b) - b * func(a))/ (func(b) - func(a))
k = 1
c = (a* f(b) - b * f(a))/(f(b)-f(a))
while True:
    print("-" * 50)
    fc = f(c)
    formatted_string = "a : {} f(a) : {} b : {} f(b): {} c: {} fc: {}".format(a, f(a), b, f(b), 
    print(formatted_string)

    if f(a) * f(c) < 0:
        b = c
    else:
        a = c

    c_prev = c
    c = (a* f(b) - b * f(a))/(f(b)-f(a))

    if abs(c - c_prev) < e:
        break

    k += 1

print("Number of iterations are: ", k)
print("Root of the equation is: ", c)
```

```
--------------------------------------------------
a : 1.0 f(a) : -4.0 b : 2.0 f(b): 2.0 c: 1.6666666666666667 fc: -1.0370370370370363
--------------------------------------------------
a : 1.6666666666666667 f(a) : -1.0370370370370363 b : 2.0 f(b): 2.0 c: 1.780487804878049 fc: -0.1
3609785116292317
--------------------------------------------------
a : 1.780487804878049 f(a) : -0.13609785116292317 b : 2.0 f(b): 2.0 c: 1.7944736520357012 fc: -0.
016025004208394478
--------------------------------------------------
a : 1.7944736520357012 f(a) : -0.016025004208394478 b : 2.0 f(b): 2.0 c: 1.7961073423838807 fc: -
0.0018622083672488188
--------------------------------------------------
a : 1.7961073423838807 f(a) : -0.0018622083672488188 b : 2.0 f(b): 2.0 c: 1.7962970110890724 fc:
-0.00021606859402067968
--------------------------------------------------
a : 1.7962970110890724 f(a) : -0.00021606859402067968 b : 2.0 f(b): 2.0 c: 1.796319015621034 fc:
-2.5065572228477606e-05
Number of iterations are:  6
Root of the equation is:  1.7963215682792548
```

In [19]:
```python
#ques 3 Secant Method
x0 = 2.0
x1 = 3.0
e = 0.001

def f(x):
    # return x**3 - 4
    return x**3-4*x-9
```

```
k = 1
while True:
    print("-" * 50)
    f_x0 = f(x0)
    f_x1 = f(x1)


    x2 = x1 - f_x1 * (x1 - x0) / (f_x1 - f_x0)


    formatted_string = "x0 : {} f(x0) : {} x1 : {} f(x1): {} x2: {} f(x2): {}".format(x0, f_x0, 
    print(formatted_string)


    if abs(x2 - x1) < e:
        break

    x0 = x1
    x1 = x2

    k += 1

print("Number of iterations are: ", k)
print("Root of the equation is: ", x2)
```

```
--------------------------------------------------
x0 : 2.0 f(x0) : -9.0 x1 : 3.0 f(x1): 6.0 x2: 2.6 f(x2): -1.8239999999999998
--------------------------------------------------
x0 : 3.0 f(x0) : 6.0 x1 : 2.6 f(x1): -1.8239999999999998 x2: 2.6932515337423313 f(x2): -0.2372265
1080748847
--------------------------------------------------
x0 : 2.6 f(x0) : -1.8239999999999998 x1 : 2.6932515337423313 f(x1): -0.23722651080748847 x2: 2.70
71928657142923 f(x2): 0.011955954723166684
--------------------------------------------------
x0 : 2.6932515337423313 f(x0) : -0.23722651080748847 x1 : 2.7071928657142923 f(x1): 0.01195595472
3166684 x2: 2.7065239505340752 f(x2): -7.197464652008989e-05
Number of iterations are:  4
Root of the equation is:  2.7065239505340752
```

In [5]:

```
#ques 4 Newton Raphson Method
x_n = 3.5
tolerance = 0.000001

def f(x):
    return x**2 - 12

def f_prime(x):

k = 0

while True:
    print(f"Iteration {k}: x_n = {x_n}, f(x_n) = {f(x_n)}")

    x_next = x_n - f(x_n) / f_prime(x_n)

    if abs(x_next - x_n) < tolerance:
        break

    x_n = x_next
    k += 1
```

```python
print("Approximate value of √12:", round(x_next, 5))
print("Number of iterations:", k)
```

```
Iteration 0: x_n = 3.5, f(x_n) = 0.25
Iteration 1: x_n = 3.4642857142857144, f(x_n) = 0.001275510204083119
Iteration 2: x_n = 3.464101620029455, f(x_n) = 3.389069647141696e-08
Approximate value of √12: 3.4641
Number of iterations: 2
```

In [5]:

```python
#5 Birge Vieta Method
import pandas as pd

def bCal(data,guess: float)->None:
    val = data["ai"][len(data["bi"])] + (data["bi"][-1] * guess)
    data["bi"].append(val)
    return None

def cCal(data,guess: float)->None:
    val = data["bi"][len(data["ci"])] + (data["ci"][-1] * guess)
    data["ci"].append(val)
    return None

coefficient = str(input("Enter the coefficients (eg : -2,3,-121) :"))
guess = float(input("Enter the initial guess :"))
listCoefficient = [float(val) for val in coefficient.split(",")]
outputList = []
k = 0
while True:
    data = {
        "ai":[],
        "bi":[],
        "ci":[],
    }

    data["ai"] = listCoefficient
    data["bi"].append(listCoefficient[0])
    data["ci"].append(listCoefficient[0])

    while True:
        bCal(data,guess)
        cCal(data,guess)
        if len(data["ai"]) == len(data["bi"]) == len(data["ci"]):
            guess = guess - (data["bi"][-1]/data["ci"][-2])
            break
    outputList.append(data)
    k+=1
    if k > 3:
        break

print(f"\nNumber of Iteration {k}\nThe BirgeVieta Table:")
for i in outputList:
    print(pd.DataFrame(i,range(1,len(i["ai"])+1)))
    print()
```

```
Number of Iteration 4
The BirgeVieta Table:
    ai     bi    ci
1  1.0  1.000  1.00
2 -1.0 -0.500  0.00
3 -1.0 -1.250 -1.25
4  1.0  0.375 -0.25

    ai     bi     ci
1  1.0  1.000  1.000
2 -1.0 -0.200  0.600
3 -1.0 -1.160 -0.680
4  1.0  0.072 -0.472

    ai        bi        ci
1  1.0  1.000000  1.000000
2 -1.0 -0.094118  0.811765
3 -1.0 -1.085260 -0.349896
4  1.0  0.016883 -0.300082

    ai        bi        ci
1  1.0  1.000000  1.000000
2 -1.0 -0.045867  0.908265
3 -1.0 -1.043764 -0.177158
4  1.0  0.004111 -0.164921
```