# Lab Report: 01.

**Course code: CCE-312.**

**Course Title: Numerical Methods Sessional.**

**Name of the Lab:** Implement Bisection method using Python.

**Remarks & Signature:**

**Submitted To**

**Professor Dr. Md. Samsuzzaman.**

**Chairman,**

**Department of Computer & Communication Engineering.**

**Faculty of Computer Science & Engineering.**


**Submitted By**

**HASAN AHAMMAD**

**ID No: 1902073**

**Reg No: 08779**

**Level- 3, Semester- 1**

**Session: 2019-2020**

**Faculty of Computer Science & Engineering.**

**Patuakhali Science & Technology University.**

**Dumki, Patuakhali-8602.**

❖ **Example-01: Determine the root of the given equation $x^2$-3 = 0 for x ∈ [1, 2] and Implement it using Python (Graphical and tabular approach).**

**Solution:**

Given: $x^2-3 = 0$
Let $f(x) = x^2-3$
Now, find the value of f(x) at a= 1 and b=2.
$f(x=1) = 1^2-3 = 1 – 3 = -2 < 0$
$f(x=2) = 2^2-3 = 4 – 3 = 1 > 0$
The given function is continuous, and the root lies in the interval [1, 2].
Let "t" be the midpoint of the interval.
I.e., t = (1+2)/2
t =3 / 2
t = 1.5
Therefore, the value of the function at "t" is
$f(t) = f(1.5) = (1.5)^2-3 = 2.25 – 3 = -0.75 < 0$
If f(t)<0, assume a = t.
and
If f(t)>0, assume b = t.
f(t) is negative, so a is replaced with t = 1.5 for the next iterations.
The iterations for the given functions are:

| **Iterations** | **a** | **b** | c | **f(a)** | **f(b)** | **f(c)** |
|---|---|---|---|---|---|---|
| **1** | 1 | 2 | 1.5 | -2 | 1 | -0.75 |
| **2** | 1.5 | 2 | 1.75 | -0.75 | 1 | 0.062 |
| **3** | 1.5 | 1.75 | 1.625 | -0.75 | 0.0625 | -0.359 |
| **4** | 1.625 | 1.75 | 1.6875 | -0.3594 | 0.0625 | -0.1523 |
| **5** | 1.6875 | 1.75 | 1.7188 | -01523 | 0.0625 | -0.0457 |
| **6** | 1.7188 | 1.75 | 1.7344 | -0.0457 | 0.0625 | 0.0081 |
| **7** | 1.7188 | 1.7344 | 1.7266 | -0.0457 | 0.0081 | -0.0189 |

So, at the seventh iteration, we get the final interval [1.7266, 1.7344]
Hence, 1.7344 is the approximated solution.

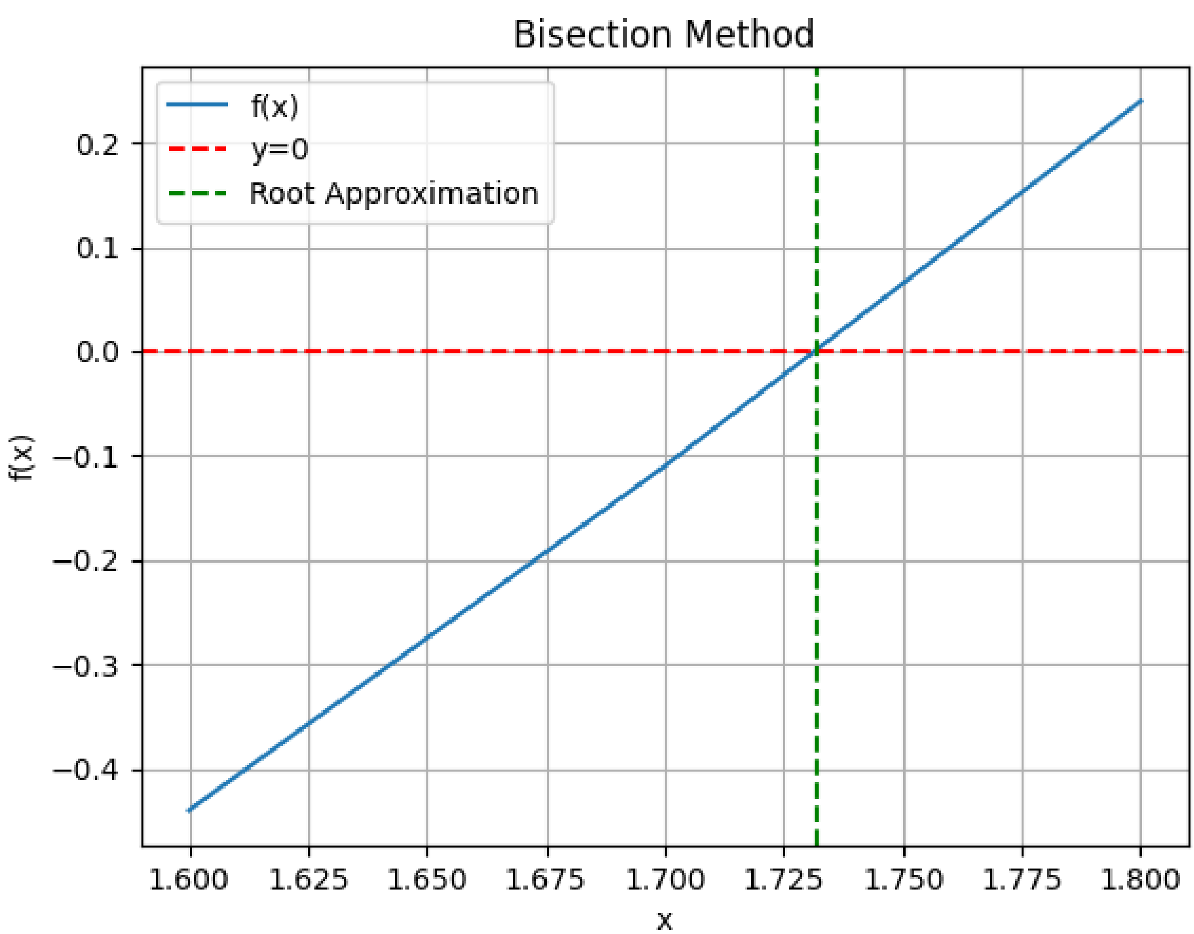
➢ **Implement with python:**

```
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
        iterations.append(i)
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
            break
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values, f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid value)", "f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 * max(a, b)) + 2)]
    y = [func(xi) for xi in x]

    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
```

```
    plt.axvline(c, color='green', linestyle='--', label='Root Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)

    return c
def my_function(x):
    return x ** 2 -3
a = 1
b = 2

root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```

## ➢ Result:

- **Graph:**



- **Table:**

| Iteration | a | b | c (mid value) | f(a) | f(b) | f(c) |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1.5 | -2 | 1 | -0.75 |
| 1 | 1.5 | 2 | 1.75 | -0.75 | 1 | 0.0625 |
| 2 | 1.5 | 1.75 | 1.625 | -0.75 | 0.0625 | -0.359375 |
| 3 | 1.625 | 1.75 | 1.6875 | -0.359375 | 0.0625 | -0.15234375 |
| 4 | 1.6875 | 1.75 | 1.71875 | -0.15234375 | 0.0625 | -0.0458984375 |
| 5 | 1.71875 | 1.75 | 1.734375 | -0.0458984375 | 0.0625 | 0.008056640625 |
| 6 | 1.71875 | 1.734375 | 1.7265625 | -0.0458984375 | 0.008056640625 | -0.01898193359375 |
| 7 | 1.7265625 | 1.734375 | 1.73046875 | -0.01898193359375 | 0.008056640625 | -0.0054779052734375 |
| 8 | 1.73046875 | 1.734375 | 1.732421875 | -0.0054779052734375 | 0.008056640625 | 0.001285552978515625 |
| 9 | 1.73046875 | 1.732421875 | 1.7314453125 | -0.0054779052734375 | 0.001285552978515625 | -0.002097129821773438 |
| 10 | 1.7314453125 | 1.732421875 | 1.73193359375 | -0.002097129821773438 | 0.001285552978515625 | -0.0004060268402096094 |
| 11 | 1.73193359375 | 1.732421875 | 1.732177734375 | -0.0004060268402096094 | 0.001285552978515625 | 0.0004397034645080566 |
| 12 | 1.73193359375 | 1.732177734375 | 1.7320556640625 | -0.0004060268402096094 | 0.0004397034645080566 | 1.682341098785400e-05 |
| 13 | 1.73193359375 | 1.7320556640625 | 1.73199462890625 | -0.0004060268402096094 | 1.682341098785400e-05 | -0.0001946054399013519 |
| 14 | 1.73199462890625 | 1.7320556640625 | 1.732025146484375 | -0.0001946054399013519 | 1.682341098785400e-05 | -8.889194577932358e-05 |
| 15 | 1.732025146484375 | 1.7320556640625 | 1.7320404052734375 | -8.889194577932358e-05 | 1.682341098785400e-05 | -3.603450022637844e-05 |
| 16 | 1.7320404052734375 | 1.7320556640625 | 1.7320480346679688 | -3.603450022637844e-05 | 1.682341098785400e-05 | -9.605602826923132e-06 |
| 17 | 1.7320480346679688 | 1.7320556640625 | 1.7320518493652344 | -9.605602826923132e-06 | 1.682341098785400e-05 | 3.608889528550207e-06 |
| 18 | 1.7320480346679688 | 1.7320518493652344 | 1.7320499420166016 | -9.605602826923132e-06 | 3.608889528550207e-06 | -2.998360287165269e-06 |
| 19 | 1.7320499420166016 | 1.7320518493652344 | 1.732050895690918 | -2.998360287165269e-06 | 3.608889528550207e-06 | 3.052637111977674e-07 |

The approximate root is: 1.732051

❖ **Example-02: Determine the root of the given equation $x^3$-x-1 = 0 for x ∈ [1, 2] and Implement it using Python (Graphical and tabular approach).**

```
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the
interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
        iterations.append(i)
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
```

```
            break
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values,
f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid value)",
"f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 *
max(a, b)) + 2)]
    y = [func(xi) for xi in x]

    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
    plt.axvline(c, color='green', linestyle='--', label='Root
Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)

    return c
def my_function(x):
    return x ** 3 - x - 1
a = 1
b = 2

root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```

❖ **Example-03: Find a root of an equation $f(x)=2x^3-2x-5$ using Bisection method.**

```
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
```

```python
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the
interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
        iterations.append(i)
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
            break
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values,
f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid value)",
"f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 *
max(a, b)) + 2)]
    y = [func(xi) for xi in x]

    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
    plt.axvline(c, color='green', linestyle='--', label='Root
Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)
```

```python
    return c
def my_function(x):
    return 2*x ** 3 - 2*x -3
a = 1
b = 2

root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```

❖ **Example-04: Find a root of an equation $f(x)=\sqrt{12}$ using Bisection method.**

**Solution:**

Let $x=\sqrt{12}$

$\therefore x^2-12=0$

i.e. $f(x)=x^2-12$

| $x$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $f(x)$ | -12 | -11 | -8 | -3 | 4 |

➢ **Python code:**

```python
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the 
interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
        iterations.append(i)
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
            break
```

```
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values,
f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid
value)", "f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 *
max(a, b)) + 2)]
    y = [func(xi) for xi in x]

    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
    plt.axvline(c, color='green', linestyle='--', label='Root
Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)

    return c
def my_function(x):
    return x ** 2 - 12
a = 3
b = 4

root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```

❖ **Example-05: Find a root of an equation $f(x)=\sqrt[3]{48}$ using Bisection method.**

**Solution:**

Let $x=\sqrt[3]{48}$

$\therefore x^3=48$

$\therefore x^3-48=0$

i.e. $f(x)=x^3-48$

Here

| $x$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $f(x)$ | -48 | -47 | -40 | -21 | 16 |

- Python code:

```
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the
interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
        iterations.append(i)
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
            break
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values,
f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid
value)", "f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 *
max(a, b)) + 2)]
    y = [func(xi) for xi in x]
```

```
    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
    plt.axvline(c, color='green', linestyle='--', label='Root 
Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)

    return c
def my_function(x):
    return x ** 3 - 48
a = 3
b = 4

root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```

- **Example-06: Find a root of an equation $f(x)=x^3+2x^2+x-1$ using Bisection method.**

**Solution:**

Here $x^3+2x^2+x-1=0$

Let $f(x)= x^3+2x^2+x-1$

Here

| $x$ | 0 | 1 |
|---|---|---|
| $f(x)$ | -1 | 3 |

- Python code:

```
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the 
interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
```

```
        iterations.append(i)
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
            break
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values,
f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid
value)", "f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 *
max(a, b)) + 2)]
    y = [func(xi) for xi in x]

    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
    plt.axvline(c, color='green', linestyle='--', label='Root
Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)

    return c
def my_function(x):
    return x ** 3 + 2*x**2 + x - 1
a = 0
b = 1
```

```python
root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```

❖ **Example-07: Find a root of an equation $y=x^3-x^2+2$ using Bisection method.**

```python
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the
interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
        iterations.append(i)
        a_values.append(a)
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
            break
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values,
f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid value)",
"f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 *
max(a, b)) + 2)]
```

```
    y = [func(xi) for xi in x]

    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
    plt.axvline(c, color='green', linestyle='--', label='Root
Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)

    return c
def my_function(x):
    return x ** 3 - x**2 + 2
a = -200
b = 300

root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```

❖ **Example-08: Find a root of an equation *y*= $x^2$-4 using Bisection method.**

When x=0 then, y=-4

When x=4 then y=12

➢ **Python code:**

```
import matplotlib.pyplot as plt
from tabulate import tabulate

def bisection(func, a, b, tol=1e-6, max_iter=100):
    iterations = []
    a_values = []
    b_values = []
    c_values = []
    f_a_values = []
    f_b_values = []
    f_c_values = []

    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over the
interval [a, b]")

    for i in range(max_iter):
        c = (a + b) / 2
        iterations.append(i)
        a_values.append(a)
```

```
        b_values.append(b)
        c_values.append(c)  # Store c instead of t
        f_a = func(a)
        f_b = func(b)
        f_c = func(c)  # Renamed f_t to f_c
        f_a_values.append(f_a)
        f_b_values.append(f_b)
        f_c_values.append(f_c)  # Store f_c instead of f_t

        if f_c == 0 or abs(b - a) / 2 < tol:
            break
        elif f_a * f_c < 0:
            b = c
        else:
            a = c

    results = list(zip(iterations, a_values, b_values, c_values,
f_a_values, f_b_values, f_c_values))
    table = tabulate(results,
                     headers=["Iteration", "a", "b", "c (mid
value)", "f(a)", "f(b)", "f(c)"],
                     tablefmt="pretty")

    x = [i / 10 for i in range(int(10 * min(a, b)) - 1, int(10 *
max(a, b)) + 2)]
    y = [func(xi) for xi in x]

    plt.plot(x, y, label='f(x)')
    plt.axhline(0, color='red', linestyle='--', label='y=0')
    plt.axvline(c, color='green', linestyle='--', label='Root
Approximation')

    plt.title('Bisection Method')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.legend()
    plt.grid(True)
    plt.show()

    print(table)

    return c
def my_function(x):
    return x**2 - 4
a = 0
b = 4

root = bisection(my_function, a, b)
print(f"The approximate root is: {root:.6f}")
```