[Tarea 03] Ejercicios Unidad 01-B

Richard Tipantiza

Tabla de Contenidos

CONJUNTO DE EJERCICIOS

1. Use el método de bisección para encontrar soluciones precisas dentro de 10^{-2} para $x^3 - 7x^2 + 14x - 6 = 0$ en cada intervalo.

a. $[0, 1]$

b. $[1, 3.2]$

c. $[3.2, 4]$

```
a = float(input("Ingrese el límite inferior a: "))
b = float(input("Ingrese el límite superior b: "))

def f(x):
    return x**3 - 7*x**2 + 14*x - 6

# Comprobamos que haya cambio de signo
if f(a) * f(b) > 0:
    print(" No hay raíz en este intervalo (no cambia de signo).")
else:
    tol = 1e-2  # tolerancia
    while abs(b - a) > tol:
        c = (a + b) / 2
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    print(f" La raíz aproximada es: {(a + b)/2:.6f}")
```

```
La raíz aproximada es: 0.582031
```

```
a = float(input("Ingrese el límite inferior a: "))
b = float(input("Ingrese el límite superior b: "))

def f(x):
    return x**3 - 7*x**2 + 14*x - 6

# Comprobamos que haya cambio de signo
if f(a) * f(b) > 0:
    print(" No hay raíz en este intervalo (no cambia de signo).")
else:
```

```
    tol = 1e-2  # tolerancia
    while abs(b - a) > tol:
        c = (a + b) / 2
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    print(f" La raíz aproximada es: {(a + b)/2:.6f}")
```

```
La raíz aproximada es: 2.998047
```

```
a = float(input("Ingrese el límite inferior a: "))
b = float(input("Ingrese el límite superior b: "))

def f(x):
    return x**3 - 7*x**2 + 14*x - 6

# Comprobamos que haya cambio de signo
if f(a) * f(b) > 0:
    print(" No hay raíz en este intervalo (no cambia de signo).")
else:
    tol = 1e-2  # tolerancia
    while abs(b - a) > tol:
        c = (a + b) / 2
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
    print(f" La raíz aproximada es: {(a + b)/2:.6f}")
```

```
La raíz aproximada es: 3.415625
```

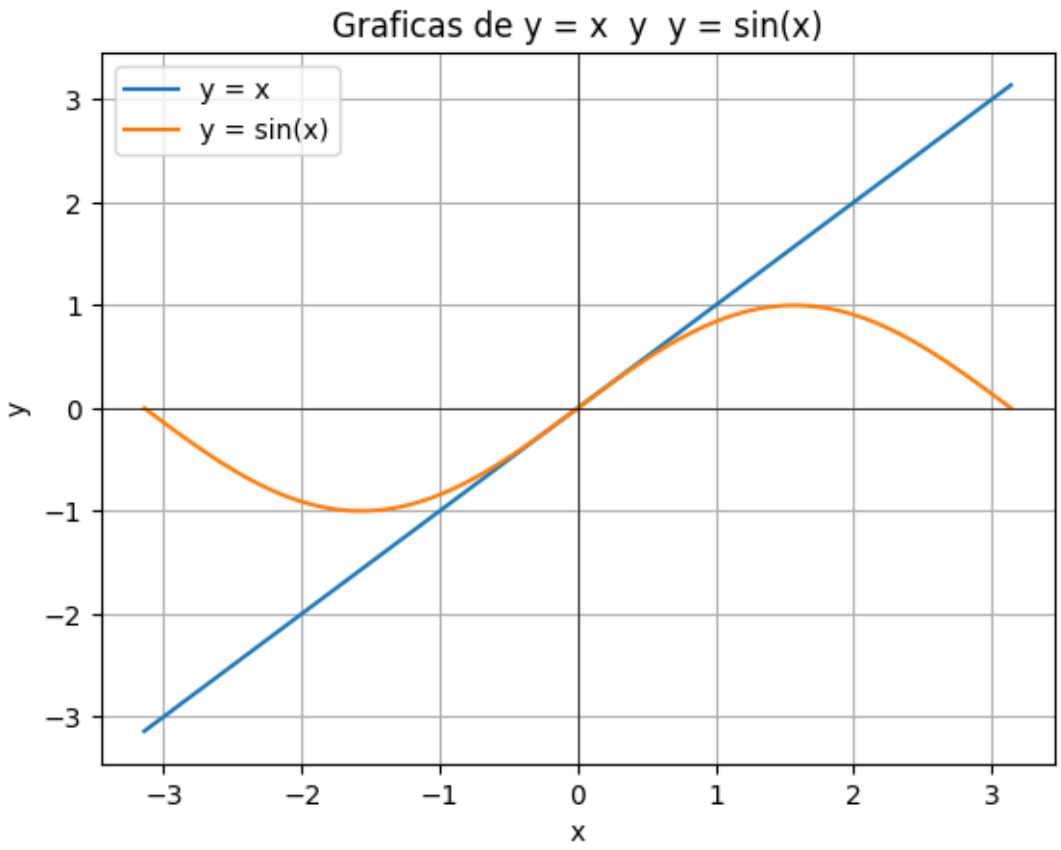
2. a. Dibuje las gráficas para $y = x$ y $y = \sin x$.

b. Use el **método de bisección** para encontrar soluciones precisas dentro de 10^{-5} para el primer valor positivo de x con $x = 2\sin x$.

```python
import math
import numpy as np
import matplotlib.pyplot as plt

print("--- Parte a: Graficas ---")

x_vals = np.linspace(-math.pi, math.pi, 400)
y_vals_1 = x_vals
y_vals_2 = np.sin(x_vals)

plt.plot(x_vals, y_vals_1)
plt.plot(x_vals, y_vals_2)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Graficas de y = x  y  y = sin(x)")
plt.legend(["y = x", "y = sin(x)"])
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.show()


print("\n--- Parte b: Metodo de Biseccion ---")

def f(x):
    return x - 2 * math.sin(x)

def metodo_biseccion(a, b, tol):

    f_a = f(a)

    i = 0
    max_iteraciones = 100

    while i < max_iteraciones:

        p = a + (b - a) / 2

        if (b - a) / 2 < tol:
            print("Solucion aproximada:", p)
            return
```

```
        f_p = f(p)

        if f_p == 0:
            print("Solucion exacta:", p)
            return

        if f_a * f_p < 0:
            b = p
        else:
            a = p
            f_a = f_p

        i = i + 1

    print("Se alcanzo el maximo de iteraciones.")

TOLERANCIA = 0.00001
a = 1.5
b = 2.0

metodo_biseccion(a, b, TOLERANCIA)
```

```
--- Parte a: Graficas ---
```




```
--- Parte b: Metodo de Biseccion ---
Solucion aproximada: 1.8955001831054688
```

3. a. Dibuje las gráficas para $y = x$ y $y = \tan x$.

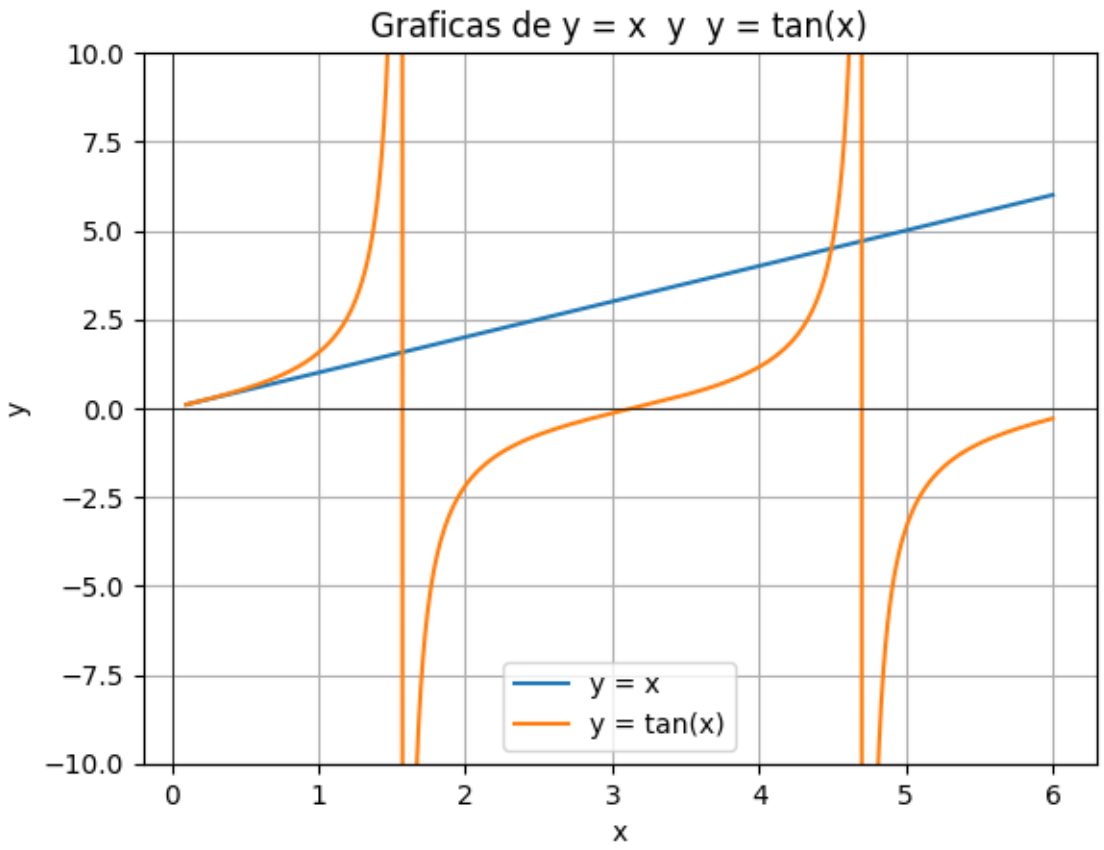
b. Use el **método de bisección** para encontrar una aproximación dentro de 10^{-5} para el primer valor positivo de x con $x = \tan x$.

```python
import math
import numpy as np
import matplotlib.pyplot as plt

print("--- Parte a: Graficas ---")

x_vals = np.linspace(0.1, 6, 400)
```

```
y_vals_1 = x_vals
y_vals_2 = np.tan(x_vals)

plt.plot(x_vals, y_vals_1)
plt.plot(x_vals, y_vals_2)

plt.ylim(-10, 10)

plt.xlabel("x")
plt.ylabel("y")
plt.title("Graficas de y = x  y  y = tan(x)")
plt.legend(["y = x", "y = tan(x)"])
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)

plt.savefig('grafica_tan_x.png')


print("\n--- Parte b: Metodo de Biseccion ---")

def f(x):
    return x - math.tan(x)

def metodo_biseccion(a, b, tol):

    f_a = f(a)

    i = 0
    max_iteraciones = 100

    while i < max_iteraciones:

        p = a + (b - a) / 2

        if (b - a) / 2 < tol:
            print("Solucion aproximada:", p)
            return

        f_p = f(p)

        if f_p == 0:
```

```
            print("Solucion exacta:", p)
            return

        if f_a * f_p < 0:
            b = p
        else:
            a = p
            f_a = f_p

        i = i + 1

    print("Se alcanzo el maximo de iteraciones.")


TOLERANCIA = 0.00001

a = 4.0
b = 4.5

metodo_biseccion(a, b, TOLERANCIA)
```

```
--- Parte a: Graficas ---

--- Parte b: Metodo de Biseccion ---
Solucion aproximada: 4.493415832519531
```

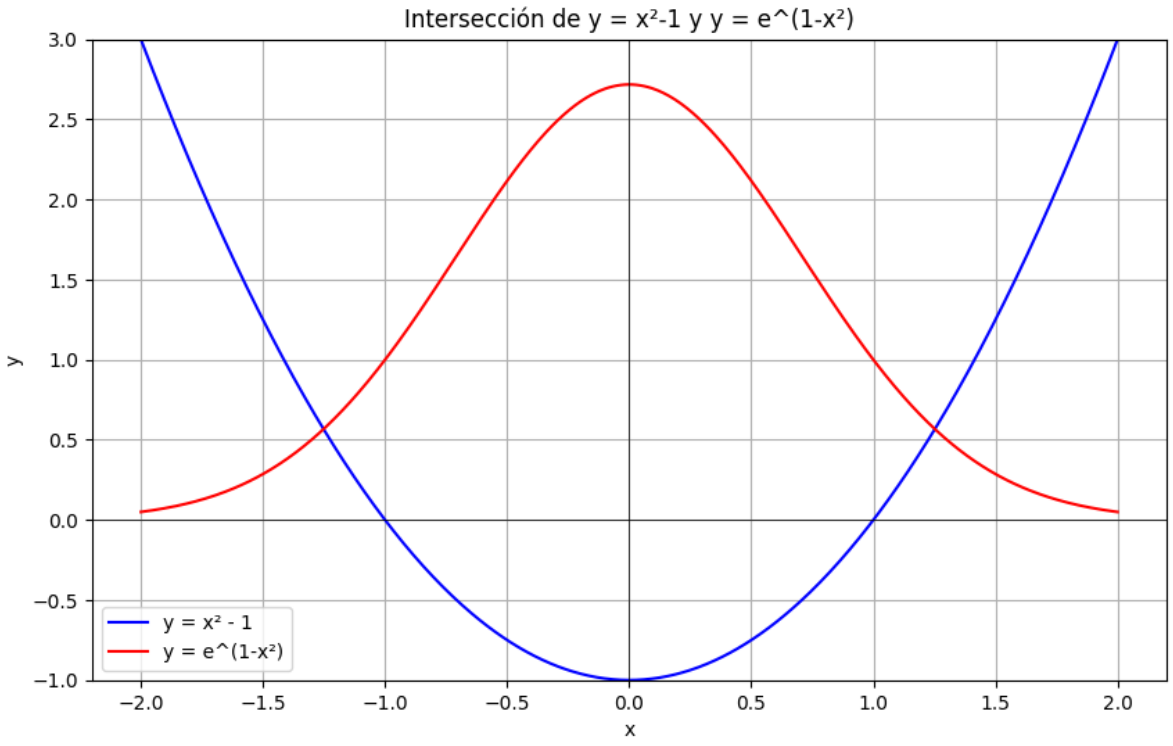


4. a. Dibuje las gráficas para $y = x^2 - 1$ y $y = e^{1-x^2}$.

b. Use el **método de bisección** para encontrar una aproximación dentro de 10^{-3} para un valor en $[-2, 0]$ con $x^2 - 1 = e^{1-x^2}$.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-2, 2, 500)
y1 = x**2 - 1
y2 = np.exp(1 - x**2)

plt.figure(figsize=(10, 6))
plt.plot(x, y1, label='y = x² - 1', color='blue')
plt.plot(x, y2, label='y = e^(1-x²)', color='red')
plt.axhline(0, color='black', linewidth=0.5)
```

```
plt.axvline(0, color='black', linewidth=0.5)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Intersección de y = x²-1 y y = e^(1-x²)')
plt.legend()
plt.grid()
plt.ylim(-1, 3)
plt.show()
```




```
import numpy as np

def f(x):
    return x**2 - 1 - np.exp(1 - x**2)

def biseccion(a, b, tol=1e-3, max_iter=100):
    if f(a) * f(b) >= 0:
        raise ValueError("No hay cambio de signo en el intervalo.")

    iteracion = 0
```

```
    while (b - a) / 2 > tol and iteracion < max_iter:
        c = (a + b) / 2
        if f(c) == 0:
            return c
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
        iteracion += 1

    return (a + b) / 2

# Aplicamos al intervalo [-2, 0] donde vemos que hay cambio de signo
try:
    raiz = biseccion(-2, 0)
    print(f"Solución encontrada en [-2, 0]: x   {raiz:.5f}")
    print(f"Valor de la función en la raíz: f({raiz:.5f}) = {f(raiz):.2e}")
except ValueError as e:
    print(e)
```

```
Solución encontrada en [-2, 0]: x   -1.25098
Valor de la función en la raíz: f(-1.25098) = -3.45e-03
```

5. Sea $f(x) = (x+3)(x+1)^2x(x-1)^3(x-3)$. ¿En qué cero de f converge el método de bisección cuando se aplica en los siguientes intervalos?

a. $[-1.5, 2.5]$

b. $[-0.5, 2.4]$

c. $[-0.5, 3]$

d. $[-3, -0.5]$

```
import math

def f(x):
    t1 = (x + 3)
    t2 = (x + 1) * (x + 1)
    t3 = x
    t4 = (x - 1) * (x - 1) * (x - 1)
    t5 = (x - 3)
```

```
    return t1 * t2 * t3 * t4 * t5

def analizar_biseccion(a, b):

    print(f"Analizando intervalo [{a}, {b}]")

    f_a = f(a)
    f_b = f(b)

    if f_a == 0:
        print(f"  El punto 'a' ({a}) es un cero.")
        print(f"  Converge a: {a}\n")
        return

    if f_b == 0:
        print(f"  El punto 'b' ({b}) es un cero.")
        print(f"  Converge a: {b}\n")
        return

    p = (a + b) / 2
    f_p = f(p)

    print(f"  f(a) = f({a}) es {'positivo' if f_a > 0 else 'negativo'}")
    print(f"  f(b) = f({b}) es {'positivo' if f_b > 0 else 'negativo'}")
    print(f"  p = {p}, f(p) = {f_p}")

    if f_a * f_p < 0:
        b = p
        print(f"  Nuevo intervalo: [{a}, {b}] (contiene 0)")
        print(f"  Converge a: 0\n")

    elif f_p * f_b < 0:
        a = p
        print(f"  Nuevo intervalo: [{a}, {b}] (contiene 1)")
        print(f"  Converge a: 1\n")

    else:

        if f_a * f_p > 0:
            a = p
            print(f"  Nuevo intervalo: [{a}, {b}]")
```

```
            if b == 3:
                print(f"  Converge a: 3\n")
            elif a < -1 and b < 0:
                print(f"  Converge a: -1\n")
            else:
                 print(f"  El metodo no converge a un cero simple (mismo signo).\n")

        else:
             print("  Error de analisis.\n")


print("--- a. [-1.5, 2.5] ---")
analizar_biseccion(-1.5, 2.5)

print("--- b. [-0.5, 2.4] ---")
analizar_biseccion(-0.5, 2.4)

print("--- c. [-0.5, 3] ---")
analizar_biseccion(-0.5, 3)

print("--- d. [-3, -0.5] ---")
analizar_biseccion(-3, -0.5)
```

```
--- a. [-1.5, 2.5] ---
Analizando intervalo [-1.5, 2.5]
  f(a) = f(-1.5) es negativo
  f(b) = f(2.5) es negativo
  p = 0.5, f(p) = 1.23046875
  Nuevo intervalo: [-1.5, 0.5] (contiene 0)
  Converge a: 0

--- b. [-0.5, 2.4] ---
Analizando intervalo [-0.5, 2.4]
  f(a) = f(-0.5) es negativo
  f(b) = f(2.4) es negativo
  p = 0.95, f(p) = 0.0036564008203125086
  Nuevo intervalo: [-0.5, 0.95] (contiene 0)
  Converge a: 0

--- c. [-0.5, 3] ---
Analizando intervalo [-0.5, 3]
  El punto 'b' (3) es un cero.
```

```
  Converge a: 3

--- d. [-3, -0.5] ---
Analizando intervalo [-3, -0.5]
  El punto 'a' (-3) es un cero.
  Converge a: -3
```

EJERCICIOS APLICADOS

1. Un abrevadero de longitud L tiene una sección transversal en forma de semicírculo con radio r. (Consulte la figura adjunta.) Cuando se llena con agua hasta una distancia h a partir de la parte superior, el volumen V de agua es:

$$V = L\left[0.5\pi r^2 - r^2 \arcsin(h/r) - h(r^2 - h^2)^{1/2}\right]$$

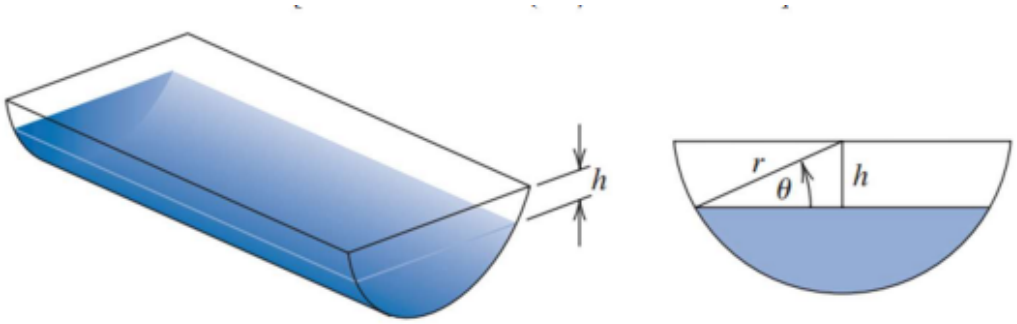


Figura 1: alt text

```python
import math

def f(h):
    L = 10.0
    r = 1.0
    V_objetivo = 12.4

    if h >= r:
        return -V_objetivo

    termino1 = 0.5 * math.pi * r * r
    termino2 = r * r * math.asin(h / r)
    termino3 = h * math.sqrt(r * r - h * h)
```

```
    volumen = L * (termino1 - termino2 - termino3)

    return volumen - V_objetivo

def metodo_biseccion(a, b, tol):

    f_a = f(a)

    i = 0
    max_iteraciones = 100

    while i < max_iteraciones:

        p = a + (b - a) / 2

        if (b - a) / 2 < tol:
            print("Solucion aproximada (h):", p)
            return

        f_p = f(p)

        if f_p == 0:
            print("Solucion exacta (h):", p)
            return

        if f_a * f_p < 0:
            b = p
        else:
            a = p
            f_a = f_p

        i = i + 1

    print("Se alcanzo el maximo de iteraciones.")


print("--- Problema del Abrevadero ---")
print("Suponga L=10, r=1, V=12.4")

TOLERANCIA = 0.00001
a = 0.0
b = 1.0
```

```
metodo_biseccion(a, b, TOLERANCIA)
```

```
--- Problema del Abrevadero ---
Suponga L=10, r=1, V=12.4
Solucion aproximada (h): 0.16616058349609375
```

2. Un objeto que cae verticalmente a través del aire está sujeto a una resistencia viscosa, así como a la fuerza de gravedad. Suponga que un objeto con masa m cae desde una altura s_0 y que la altura del objeto después de t segundos es

$$s(t) = s_0 - \frac{mg}{k}t + \frac{m^2 g}{k^2}\left(1 - e^{-kt/m}\right),$$

donde $g = 9.81$ m/s^2 y k representa el coeficiente de la resistencia del aire en N·s/m. Suponga $s_0 = 300$ m, $m = 0.25$ kg y $k = 0.1$ N·s/m. Encuentre, **dentro de** 0.01 **segundos**, el **tiempo que tarda un cuarto de kg en golpear el piso**.

```
import math

def s(t):
    s0 = 300.0
    m = 0.25
    k = 0.1
    g = 9.81

    termino1 = s0
    termino2 = (m * g / k) * t
    termino3 = (m * m * g) / (k * k)
    termino4 = 1.0 - math.exp((-k * t) / m)

    altura = termino1 - termino2 + (termino3 * termino4)
    return altura

def metodo_biseccion(a, b, tol):

    f_a = s(a)

    i = 0
    max_iteraciones = 100
```

```
    while i < max_iteraciones:

        p = a + (b - a) / 2

        if (b - a) / 2 < tol:
            print("Tiempo aproximado para golpear el piso:", p)
            return

        f_p = s(p)

        if f_p == 0:
            print("Tiempo exacto:", p)
            return

        if f_a * f_p < 0:
            b = p
        else:
            a = p
            f_a = f_p

        i = i + 1

    print("Se alcanzo el maximo de iteraciones.")

TOLERANCIA = 0.01
a = 10.0
b = 20.0

metodo_biseccion(a, b, TOLERANCIA)
```

```
Tiempo aproximado para golpear el piso: 14.716796875
```

EJERCICIOS TEÓRICOS

1. Use el teorema 2.1 para encontrar una cota para el número de iteraciones necesarias para lograr una aproximación con precisión de 10^{-4} para la solución de $x^3 - x - 1 = 0$ que se encuentra dentro del intervalo $[1, 2]$. Encuentre una aproximación para la raíz con este grado de precisión.

```
import math

a = 1.0
b = 2.0
TOL = 0.0001

print("--- Parte 1: Cota de Iteraciones (Teorema 2.1) ---")

n_calculado = (math.log(b - a) - math.log(TOL)) / math.log(2)
n_requerido = math.ceil(n_calculado)

print("Iteraciones necesarias (n):", n_requerido)


print("\n--- Parte 2: Metodo de Biseccion ---")

def f(x):
    return (x * x * x) - x - 1

f_a = f(a)

i = 0
max_iteraciones = 100

p = 0.0
terminado = False

while i < max_iteraciones:

    p = a + (b - a) / 2

    if (b - a) / 2 < TOL:
        print("Solucion aproximada:", p)
        print("Iteraciones realizadas:", i + 1)
```

```
        terminado = True
        break

    f_p = f(p)

    if f_p == 0:
        print("Solucion exacta:", p)
        print("Iteraciones realizadas:", i + 1)
        terminado = True
        break

    if f_a * f_p < 0:
        b = p
    else:
        a = p
        f_a = f_p

    i = i + 1

if terminado == False:
     print("Se alcanzo el maximo de iteraciones.")
```

```
--- Parte 1: Cota de Iteraciones (Teorema 2.1) ---
Iteraciones necesarias (n): 14

--- Parte 2: Metodo de Biseccion ---
Solucion aproximada: 1.32476806640625
Iteraciones realizadas: 14
```

2. La función definida por $f(x) = \sin \pi x$ tiene ceros en cada entero. Muestre cuando $-1 < a < 0$ y $2 < b < 3$, el método de bisección converge a

a. 0, si $a + b < 2$

b. 2, si $a + b > 2$

c. 1, si $a + b = 2$

```
import math

def f(x):
    return math.sin(math.pi * x)
```

```
def metodo_biseccion_demo(a, b):

    print("Intervalo inicial:", a, b)

    f_a = f(a)

    i = 0
    p = 0.0

    while i < 30:

        p = a + (b - a) / 2

        f_p = f(p)

        if f_p == 0:
            print("Solucion exacta:", p)
            return

        if (b - a) / 2 < 0.0000001:
            break

        if f_a * f_p < 0:
            b = p
        else:
            a = p
            f_a = f_p

        i = i + 1

    print("Converge a:", p)


print("--- Caso a: a + b < 2 ---")
a1 = -0.5
b1 = 2.4
print("a = -0.5, b = 2.4. (a + b = 1.9)")
metodo_biseccion_demo(a1, b1)


print("\n--- Caso b: a + b > 2 ---")
a2 = -0.5
```

```
b2 = 2.6
print("a = -0.5, b = 2.6. (a + b = 2.1)")
metodo_biseccion_demo(a2, b2)


print("\n--- Caso c: a + b = 2 ---")
a3 = -0.5
b3 = 2.5
print("a = -0.5, b = 2.5. (a + b = 2.0)")
metodo_biseccion_demo(a3, b3)
```

```
--- Caso a: a + b < 2 ---
a = -0.5, b = 2.4. (a + b = 1.9)
Intervalo inicial: -0.5 2.4
Converge a: 8.940696700995172e-09

--- Caso b: a + b > 2 ---
a = -0.5, b = 2.6. (a + b = 2.1)
Intervalo inicial: -0.5 2.6
Converge a: 1.9999999254941943

--- Caso c: a + b = 2 ---
a = -0.5, b = 2.5. (a + b = 2.0)
Intervalo inicial: -0.5 2.5
Converge a: -2.9802322387695312e-08
```