TAREA 4

CRISTINA MORENO

1. Utilice el método de bisección para encontrar soluciones precisas dentro de 10^{-6} para la ecuación:

$x^3 - 7x^2 + 14x - 6 = 0$

```
f   : función de la que se busca la raíz
a   : extremo izquierdo del intervalo
b   : extremo derecho del intervalo
TOL : tolerancia (criterio de convergencia)
N   : máximo número de iteraciones
```

In [ ]:
```
def f(x):
    return x**3 - 7*x**2 + 14*x - 6

def biseccion(f, a, b, TOL, N=1000):
    FA = f(a)
    FB = f(b)


    if abs(FA) < TOL:
         return a

    if abs(FB) < TOL:
        return b

    if FA * FB > 0:
        print("La función no tiene raíces o tiene múltiples raíces en el intervalo dado.")
        return None

    for i in range(1, N + 1):
        p = (a + b) / 2
        FP = f(p)

        if abs(b - a) / 2 < TOL or abs(FP) < TOL :
            return p

        if FA * FP > 0:
            a = p
            FA = FP
        else:
            b = p

    print(f"El método fracasó después de {N} iteraciones.")
    return None
resultado_A=biseccion(f,0,1,10**-2)
resultado_B=biseccion(f,1,3.2,10**-2)
resultado_C=biseccion(f,3.2,4,10**-2)

print(f"Raiz en [0,1] {resultado_A}")
print(f"Raiz en [1,3.2] {resultado_B}")
print(f"Raiz en [3.2,4] {resultado_C}")
```

```
Raiz en [0,1] 0.5859375
Raiz en [1,3.2] 2.9937500000000004
Raiz en [3.2,4] 3.4125000000000005
```

2. Definir las graficas de x=y y sen(x)=y

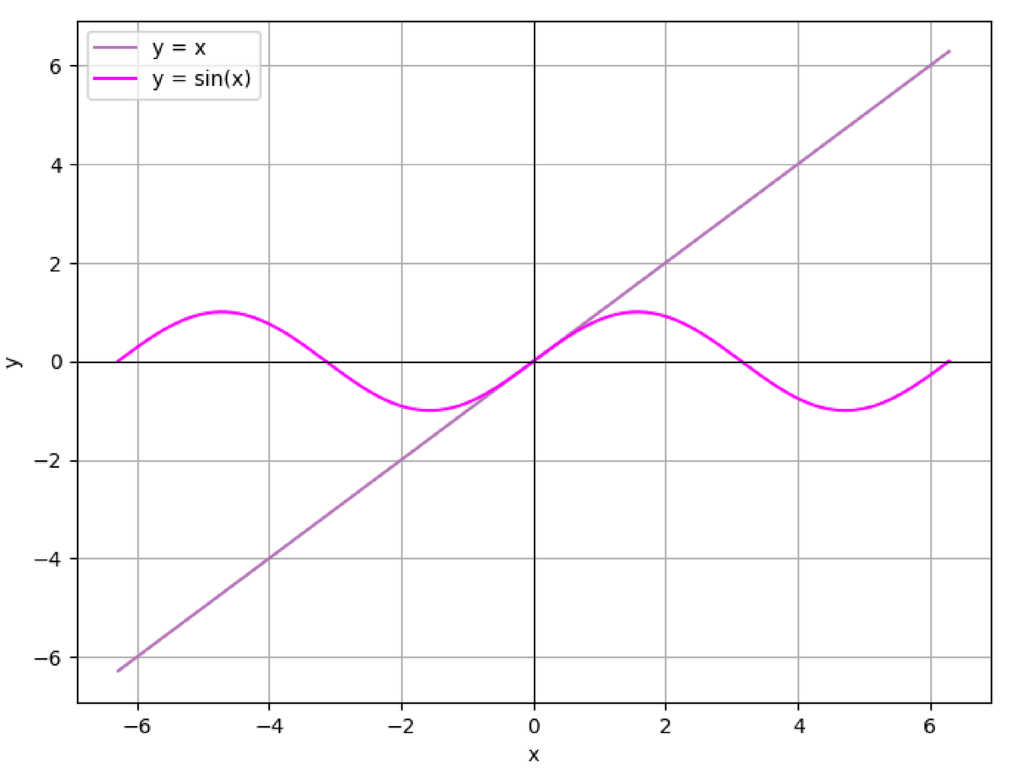
In [6]:
```
import numpy as np
import matplotlib.pyplot as plt


x = np.linspace(-2*np.pi, 2*np.pi, 400)


y1 = x
y2 = np.sin(x)


plt.figure(figsize=(8, 6))
plt.plot(x, y1, label='y = x', color="#B676B9")
plt.plot(x, y2, label='y = sin(x)', color='#FF00FF')


plt.axhline(0, color='black', linewidth=0.7)
plt.axvline(0, color='black', linewidth=0.7)
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```



b._ Definir método de bisección para encontrar soluciones precisas dentro de 10^-5 para el primer valor positivo de x con x = 2 sin x.

El seno se encutra entre [0, 1]

$0 < sen(x) < 1$

$0 < 2sen(x) < 2$

Intervalo [0,2]

Definir la función $y = x - 2sin(x)$

In [8]:
```
import numpy as np

def biseccion(f, a, b, TOL, N=1000):
    FA = f(a)
    FB = f(b)

    if abs(FA) < TOL:
         return a

    if abs(FB) < TOL:
        return b

    if FA * FB > 0:
        print("La función no tiene raíces o tiene múltiples raíces en el intervalo dado.")
        return None

    for i in range(1, N + 1):
        p = (a + b) / 2
        FP = f(p)

        if abs(b - a) / 2 < TOL or abs(FP) < TOL:
            return p

        if FA * FP > 0:
            a = p
            FA = FP
        else:
            b = p

    print(f"El método fracasó después de {N} iteraciones.")
    return None


def f2(x):
    return x - 2 * np.sin(x)

resultado_D = biseccion(f2, 1, 5, 10**-5);
print(f"Raíz aproximada: {resultado_D}")
```

```
Raíz aproximada: 1.8954925537109375
```

3. Dibuje las gráficas para y = x y y = tan x.

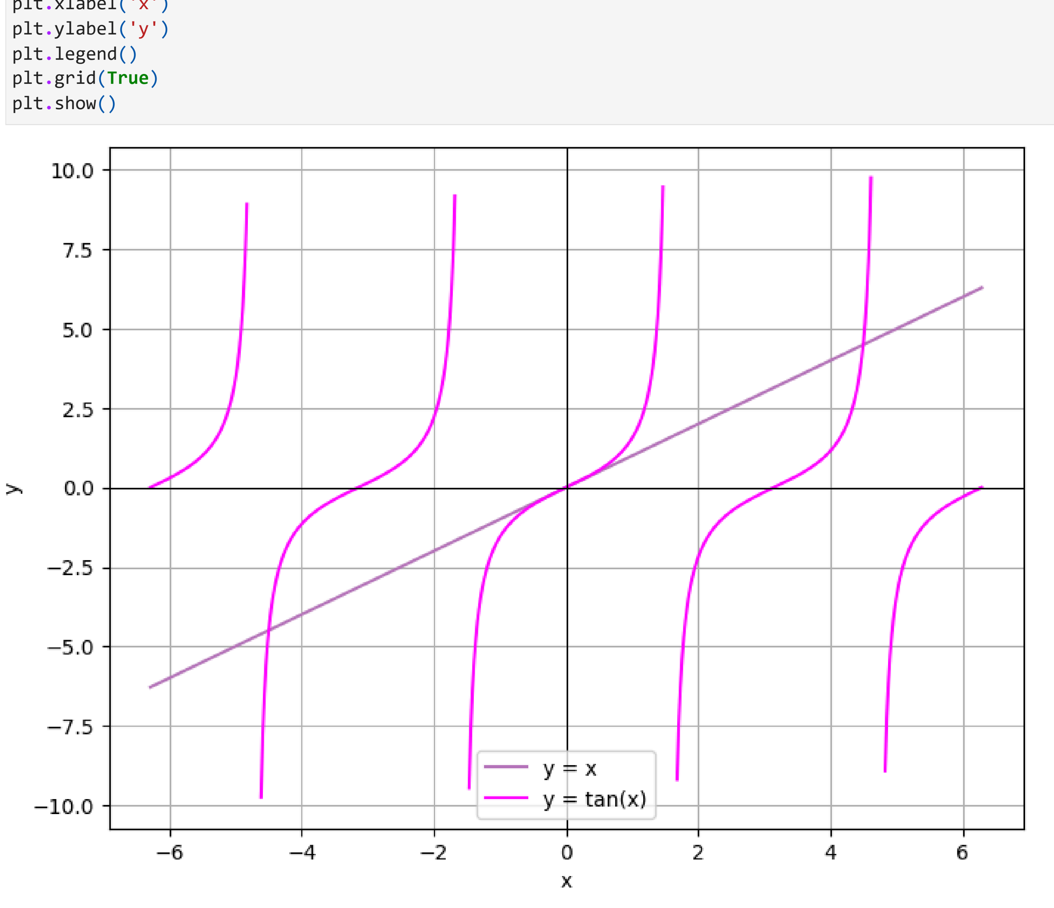
In [20]:
```
import numpy as np
import matplotlib.pyplot as plt


x = np.linspace(-2*np.pi, 2*np.pi, 1000)


y1 = x
y2 = np.tan(x)
y2[np.abs(y2) > 10] = np.nan

plt.figure(figsize=(8, 6))
plt.plot(x, y1, label='y = x', color="#B676B9")
plt.plot(x, y2, label='y = tan(x)', color='#FF00FF')


plt.axhline(0, color='black', linewidth=0.7)
plt.axvline(0, color='black', linewidth=0.7)
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```



b._ Use el método de bisección para encontrar una aproximación dentro de 10^-3 para el primer valor positivo de x con x = tan(x)

In [24]:
```
def f4(x):
    return x-np.tan(x)

resultado_F=biseccion(f4,4.6,4,10**-3)
print(resultado_F)
```

```
4.493945312499999
```

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

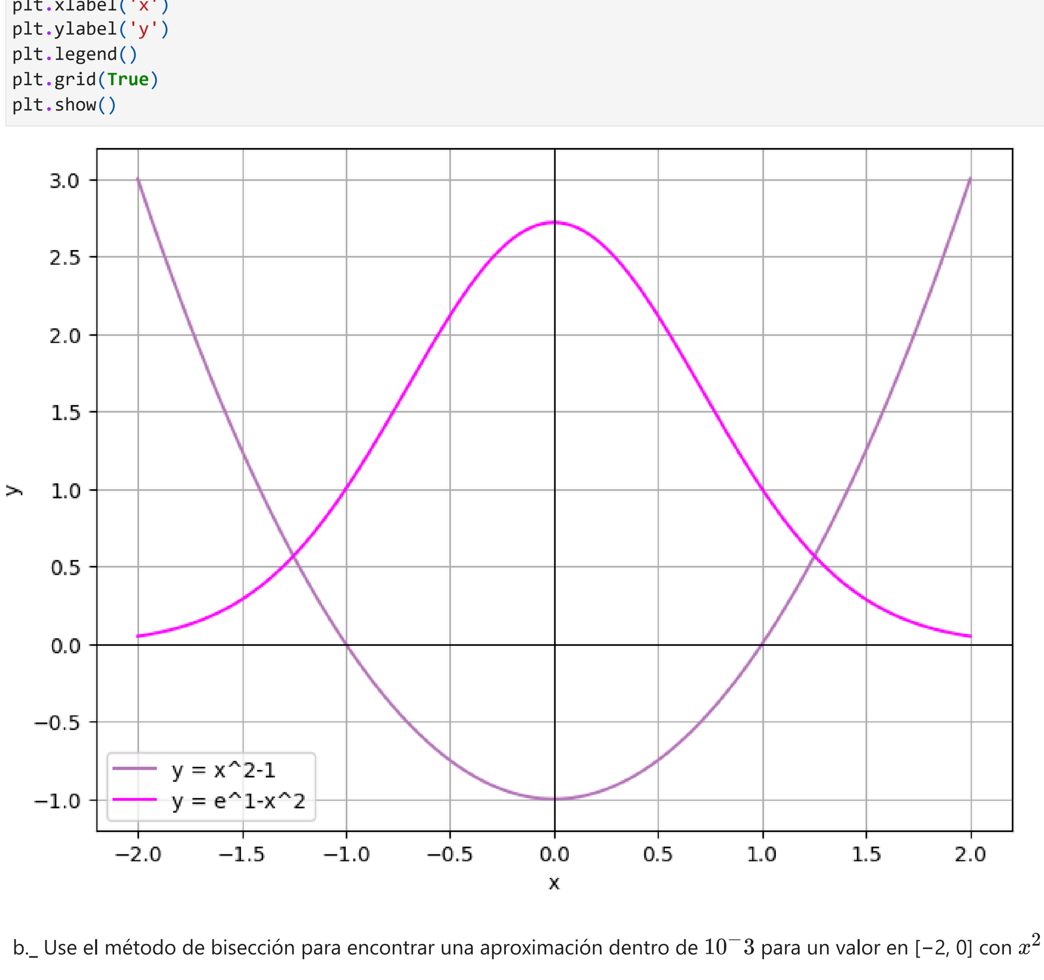
In [14]:
```
import numpy as np
import matplotlib.pyplot as plt



x = np.linspace(-2,2,400)
y1 = x**2-1
y2 = np.exp(1 - x**2)


plt.figure(figsize=(8, 6))
plt.plot(x, y1, label='y = x^2-1', color="#B676B9")
plt.plot(x, y2, label='y = e^1-x^2', color='#FF00FF')


plt.axhline(0, color='black', linewidth=0.7)
plt.axvline(0, color='black', linewidth=0.7)
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```



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

In [17]:
```
def f3(x):
    return x**2 - 1 - np.exp(1 - x**2)

resultado_E= biseccion(f3,-2,0,10**-3)

print(resultado_E)
```

```
-1.251953125
```

5. A que raiz converge la siguiente funcion

In [25]:
```
def f6(x):
    return (x+3)*(x+1)**2*x*(x-1)**3*(x-3)

resultado_G=biseccion(f6,-1.5,2.5,10**-3)
print(f"Raiz en el [-1.5,2.5]: {resultado_G}")
resultado_H=biseccion(f6,-0.5,2.4,10**-3)
print(f"Raiz en el [-0.5,2.4]: {resultado_H}")
resultado_I=biseccion(f6,-0.5,3,10**-3)
print(f"Raiz en el [-0.5,3]: {resultado_I}")
resultado_J=biseccion(f6,-3,-0.5,10**-3)
print(f"Raiz en el [-3,-0.5]: {resultado_J}")
```

```
La función no tiene raíces o tiene múltiples raíces en el intervalo dado.
Raiz en el [-1.5,2.5]: None
La función no tiene raíces o tiene múltiples raíces en el intervalo dado.
Raiz en el [-0.5,2.4]: None
Raiz en el [-0.5,3]: 3
Raiz en el [-3,-0.5]: -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 Suponga que L = 10 cm, r = 1 cm y V = 12.4 cm . Encuentre la profundidad del agua en el abrevadero dentro de 0.01 cm.

In [29]:
```
def f(h,l=10,r=1):
    return (l * ((0.5 * np.pi * r**2)
                - (r**2 * np.arcsin(h / r))
                - h * (r**2 - h**2)**0.5)
            - 12.4)

resultado_K=biseccion(f,0,1,10**-2)

print(f"la profundiad es: {resultado_K}")
```

```
la profundiad es: 0.1640625
```

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 y que la altura del objeto después de t segundos es donde $g = 9.81m/s$ y k representa el coeficiente de la resistencia del aire en N/sm. Suponga $s_0 = 300m$, $m = 0.25kg$ y $k = 0.1Ns/m$. Encuentre, dentro de 0.01 $segundos$, el tiempo que tarda un cuarto de kg en golpear el piso.

In [31]:
```
def f(t, so=300,m=0.25,g=9.81,k=0.1):
    return  so - (m * g / k) * t + (m**2 * g / k**2) * (1 - np.exp(-k * t / m))

resultado_L=biseccion(f,0,50,10**-2)
print(f"El tiempo es: {resultado_L}")
```

```
El tiempo es: 14.727783203125
```

Ejercicio Teoricos

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.

Teorema

$$|p_n - p| <= \frac{b-a}{2^n}$$

In [32]:
```
def f(x):
    return x**3 - x - 1

tolerancia = 1e-4
a = 1
b = 2
ni = np.ceil(np.log((b - a) / tolerancia) / np.log(2))
print(ni)
```

```
14.0
```

La función definida por $f(x)$ = sin π x tiene ceros en cada entero. Muestre cuando −1 < a < 0 y 2 < b < 3, el método de bisección converge a Describa cuál rango es más eficiente

In [45]:
```
def f9(x):
    return np.sin(x*np.pi)

resultado_M=biseccion(f9,-0.5,0.5,10**-4)
print(f"Cuando {-0.5+0.5} la raiz converge a {resultado_M}")

restulado_N=biseccion(f9,2.8,3,10**-4)
print(f"Cuando {2.2+3} la raiz converge a {restulado_N}")

resultado_P=biseccion(f9,-2,4,10**-4)
print(f"Cuando {1+1} la raiz converge a {resultado_P}")
```

```
Cuando 0.0 la raiz converge a 0.0
Cuando 5.2 la raiz converge a 3
Cuando 2 la raiz converge a -2
```