Tarea 12 - ODE Método de Euler

Nombre: Alexis Bautista
Fecha de entrega: 05 de febrero de 2025
Curso: GR1CC
Enlace de GitHub: https://github.com/alexis-bautista/Tarea12-MN

Conjunto de Ejercicios

Ejercicio 3

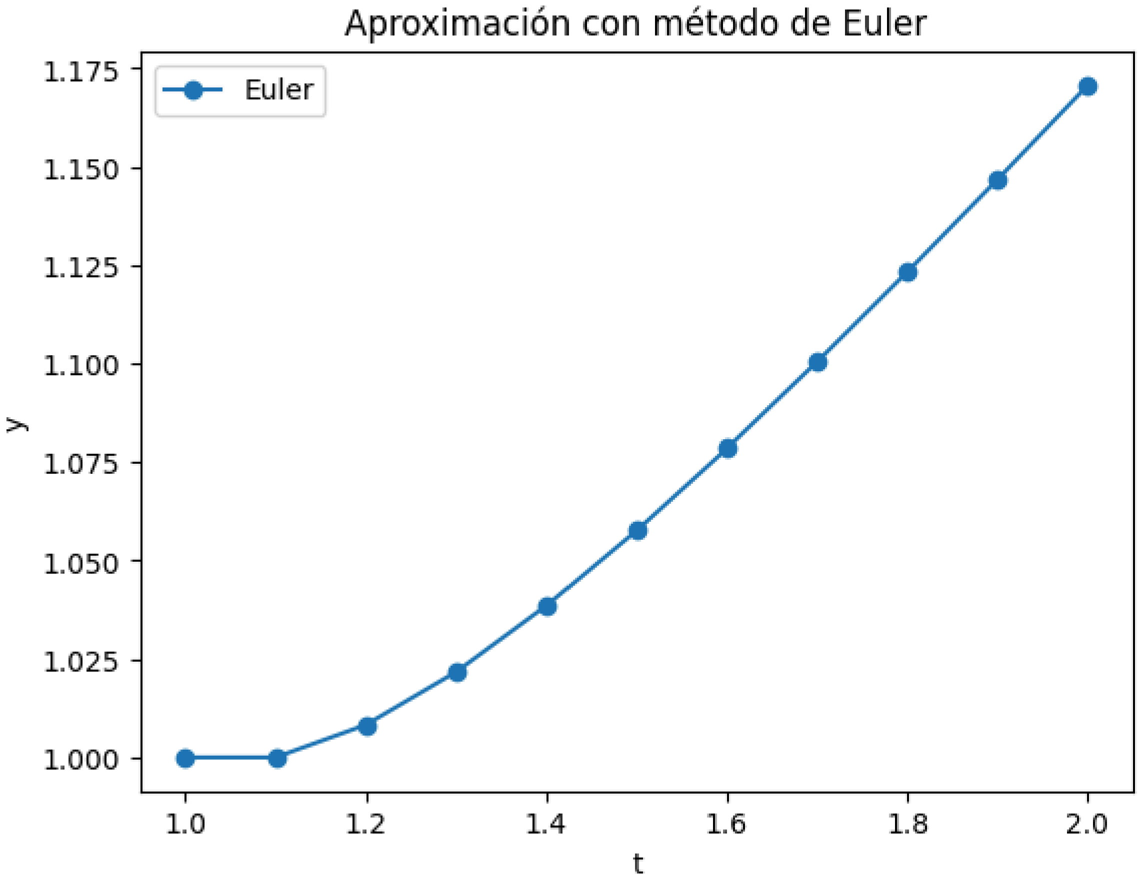
Utilice el método de Euler para aproximar las soluciones para cada uno de los siguientes problemas de valor inicial.

In [29]:
```
import matplotlib.pyplot as plt

def metodo_Euler(f, t0, y0, tf, h):
    n_pasos = int((tf - t0) / h)
    t_vals = [t0]
    y_vals = [y0]

    for _ in range(n_pasos):
        y_siguiente = y_vals[-1] + h * f(t_vals[-1], y_vals[-1])
        t_siguiente = t_vals[-1] + h
        t_vals.append(t_siguiente)
        y_vals.append(y_siguiente)

    print("Solución aproximada:")
    for t_val, y_val in zip(t_vals, y_vals):
        print(f"t = {t_val}, y = {y_val}")

    # Graficar la solución aproximada
    plt.plot(t_vals, y_vals, 'o-', label='Euler')
    plt.xlabel('t')
    plt.ylabel('y')
    plt.title('Aproximación con método de Euler')
    plt.legend()
    plt.show()

    return t_vals, y_vals
```

a. $y' = \frac{y}{t} - \left(\frac{y}{t}\right)^2, \quad 1 \leq t \leq 2, \quad y(1) = 1, \quad \text{con } h = 0.1$

In [30]:
```
def f_a(t, y):
    return (y / t) - (y / t)**2

t_aprox_3a, y_aprox_3a = metodo_Euler(f_a, t0=1, y0=1, tf=2, h=0.1)
```

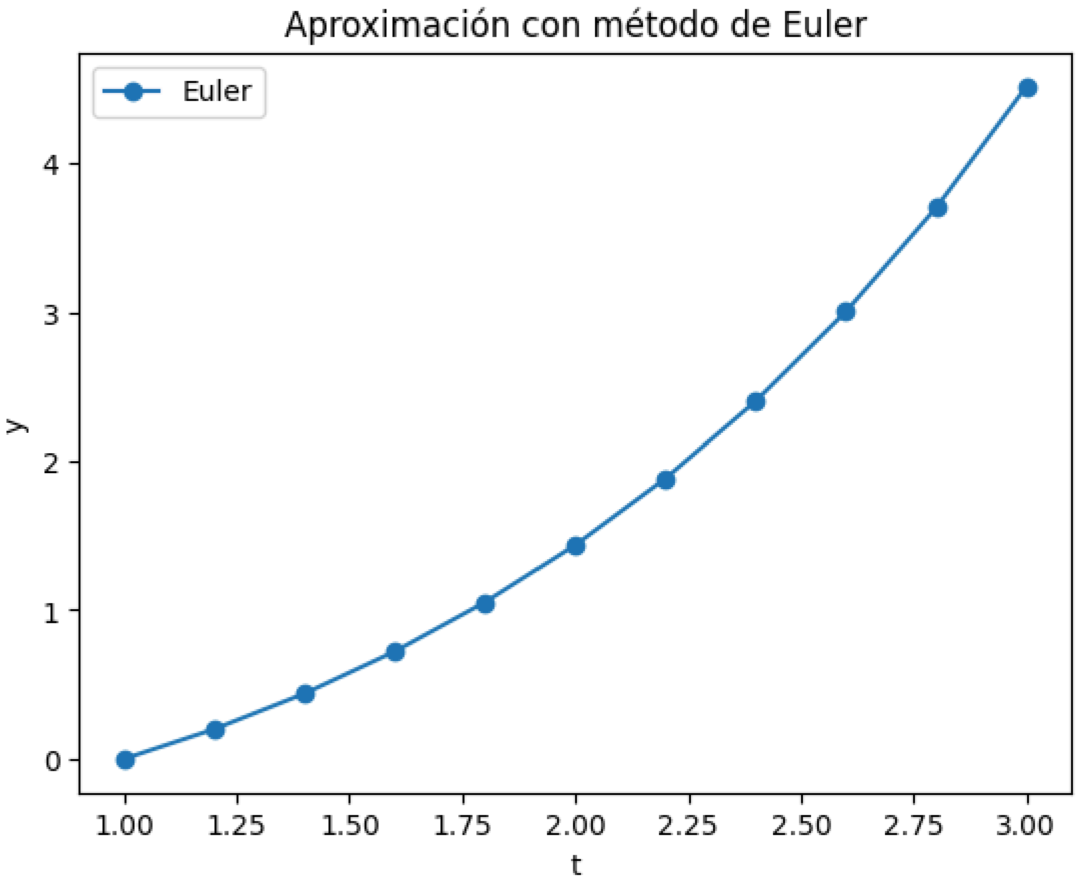
```
Solución aproximada:
t = 1, y = 1
t = 1.1, y = 1.0
t = 1.2000000000000002, y = 1.0082644628099173
t = 1.3000000000000003, y = 1.0216894717270375
t = 1.4000000000000004, y = 1.038514734248178
t = 1.5000000000000004, y = 1.0576681921408762
t = 1.6000000000000005, y = 1.0784610936317547
t = 1.7000000000000006, y = 1.100432164699466
t = 1.8000000000000007, y = 1.1232620515812632
t = 1.9000000000000008, y = 1.1467235965295264
t = 2.000000000000001, y = 1.1706515695646647
```



b. $y' = 1 + \frac{y}{t} + \left(\frac{y}{t}\right)^2, \quad 1 \leq t \leq 3, \quad y(1) = 0, \quad \text{con } h = 0.2$

In [31]:
```
def f_b(t, y):
    return 1+y/t+(y/t)**2

t_aprox_3b, y_aprox_3b = metodo_Euler(f_b, t0=1, y0=0, tf=3, h=0.2)
```

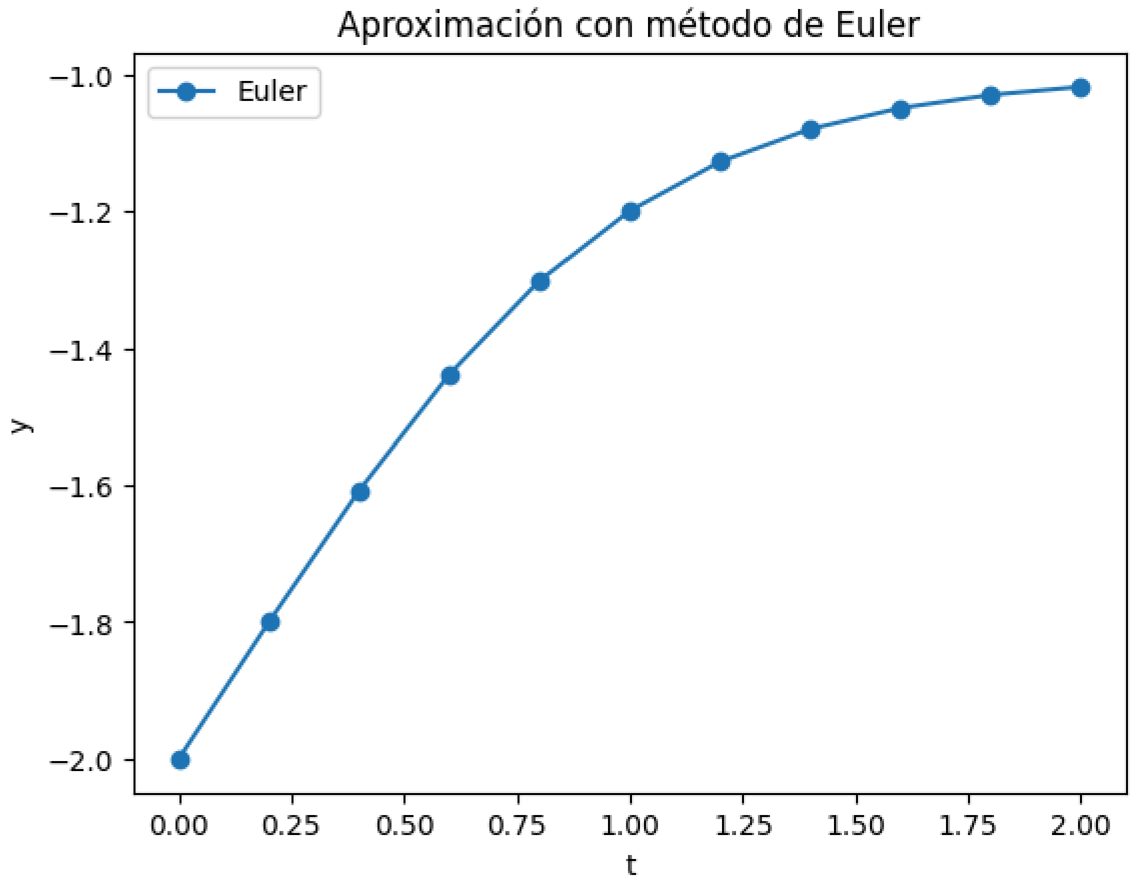
```
Solución aproximada:
t = 1, y = 0
t = 1.2, y = 0.2
t = 1.4, y = 0.4388888888888889
t = 1.5999999999999999, y = 0.721242756361804
t = 1.7999999999999998, y = 1.0520380316573712
t = 1.9999999999999998, y = 1.4372511475238394
t = 2.1999999999999997, y = 1.8842608053291532
t = 2.4, y = 2.402269588561542
t = 2.6, y = 3.0028371645572136
t = 2.8000000000000003, y = 3.7006007049327985
t = 3.0000000000000004, y = 4.5142774281767
```



c. $y' = -(y+1)(y+3), \quad 0 \le t \le 2, \quad y(0) = -2, \quad \text{con } h = 0.2$

In [32]:
```
def f_c(t, y):
    return -(y+1)*(y+3)

t_aprox_3c, y_aprox_3c = metodo_Euler(f_c, t0=0, y0=-2, tf=2, h=0.2)
```

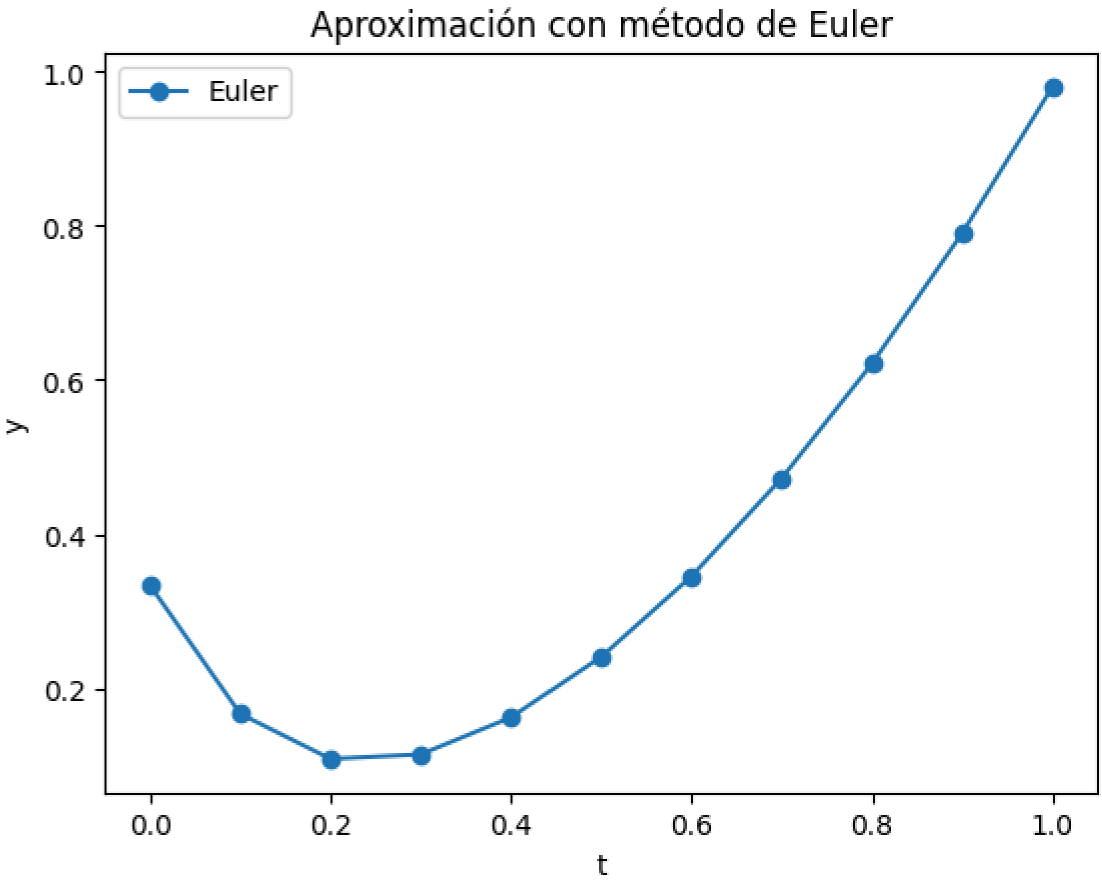
```
Solución aproximada:
t = 0, y = -2
t = 0.2, y = -1.8
t = 0.4, y = -1.608
t = 0.6000000000000001, y = -1.4387328000000001
t = 0.8, y = -1.3017369739591682
t = 1.0, y = -1.199251224666308
t = 1.2, y = -1.1274909449059896
t = 1.4, y = -1.079745355150198
t = 1.5999999999999999, y = -1.0491190774237251
t = 1.7999999999999998, y = -1.0299539832076265
t = 1.9999999999999998, y = -1.0181518381465764
```



d. $y' = -5y + 5t^2 + 2t, \quad 0 \leq t \leq 1, \quad y(0) = \frac{1}{3}, \quad \text{con } h = 0.1$

In [33]:
```
def f_d(t, y):
    return -5*y+5*t**2+2*t

t_aprox_3d, y_aprox_3d = metodo_Euler(f_d, t0=0, y0=1/3, tf=1, h=0.1)
```

```
Solución aproximada:
t = 0, y = 0.3333333333333333
t = 0.1, y = 0.16666666666666666
t = 0.2, y = 0.10833333333333334
t = 0.30000000000000004, y = 0.11416666666666667
t = 0.4, y = 0.16208333333333336
t = 0.5, y = 0.2410416666666667
t = 0.6, y = 0.3455208333333333
t = 0.7, y = 0.4727604166666667
t = 0.7999999999999999, y = 0.6213802083333333
t = 0.8999999999999999, y = 0.7906901041666666
t = 0.9999999999999999, y = 0.9803450520833332
```



Ejercicio 4

Aquí se dan las soluciones reales para los problemas de valor inicial en el ejercicio 3. Calcule el error real en las aproximaciones del ejercicio 3.

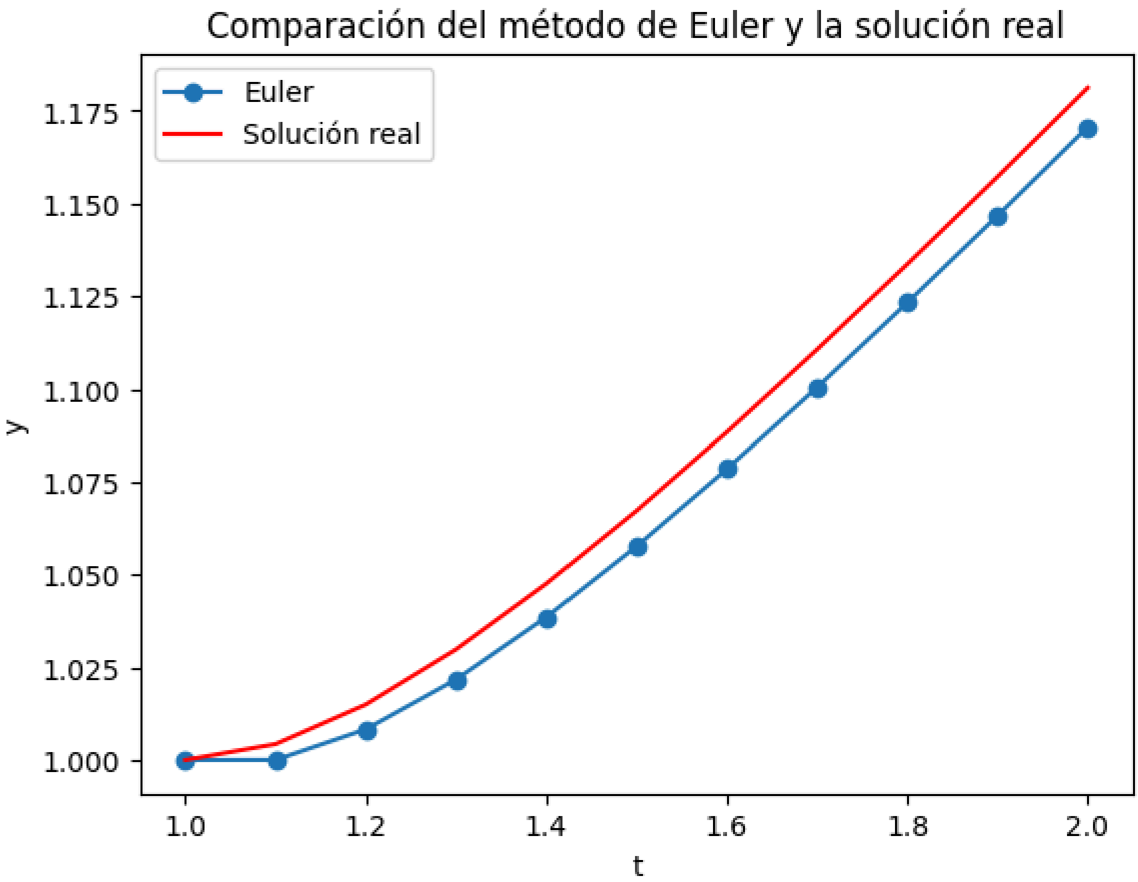
In [34]:
```
def euler_con_error(f, real_func, t0, y0, tf, h):
    n_pasos = int((tf - t0) / h)
    t_vals = [t0]
    y_vals = [y0]

    for _ in range(n_pasos):
        y_siguiente = y_vals[-1] + h * f(t_vals[-1], y_vals[-1])
        t_siguiente = t_vals[-1] + h
        t_vals.append(t_siguiente)
        y_vals.append(y_siguiente)

    # Cálculo de la solución real
    y_real = [real_func(t) for t in t_vals]
    # Cálculo del error
    errores = [abs(ya - yr) for ya, yr in zip(y_vals, y_real)]

    print("Error real para cada t:")
    for t_val, err_val in zip(t_vals, errores):
        print(f"t = {t_val}, Error = {err_val}")

    # Gráfica de comparación
    plt.plot(t_vals, y_vals, 'o-', label='Euler')
    plt.plot(t_vals, y_real, 'r-', label='Solución real')
    plt.xlabel('t')
    plt.ylabel('y')
    plt.title('Comparación del método de Euler y la solución real')
```

```python
    plt.legend()
    plt.show()

    return t_vals, y_vals, y_real, errores
```

a. $y(t) = \frac{t}{1+\ln t}$

In [35]:
```python
import numpy as np

def y_real_a(t):
    return t / (1 + np.log(t))

t_aprox_3a, y_aprox_3a, y_real_a, err_a = euler_con_error(f_a, y_real_a, t0=1, y
```

```
Error real para cada t:
t = 1, Error = 0.0
t = 1.1, Error = 0.004281727936202406
t = 1.2000000000000002, Error = 0.006687851223824204
t = 1.3000000000000003, Error = 0.00812421723094725
t = 1.4000000000000004, Error = 0.009019185004341734
t = 1.5000000000000004, Error = 0.009594162040996945
t = 1.6000000000000005, Error = 0.009971593314036298
t = 1.7000000000000006, Error = 0.010222887446798445
t = 1.8000000000000007, Error = 0.010391505152042235
t = 1.9000000000000008, Error = 0.010504836525143224
t = 2.000000000000001, Error = 0.010580648734618059
```

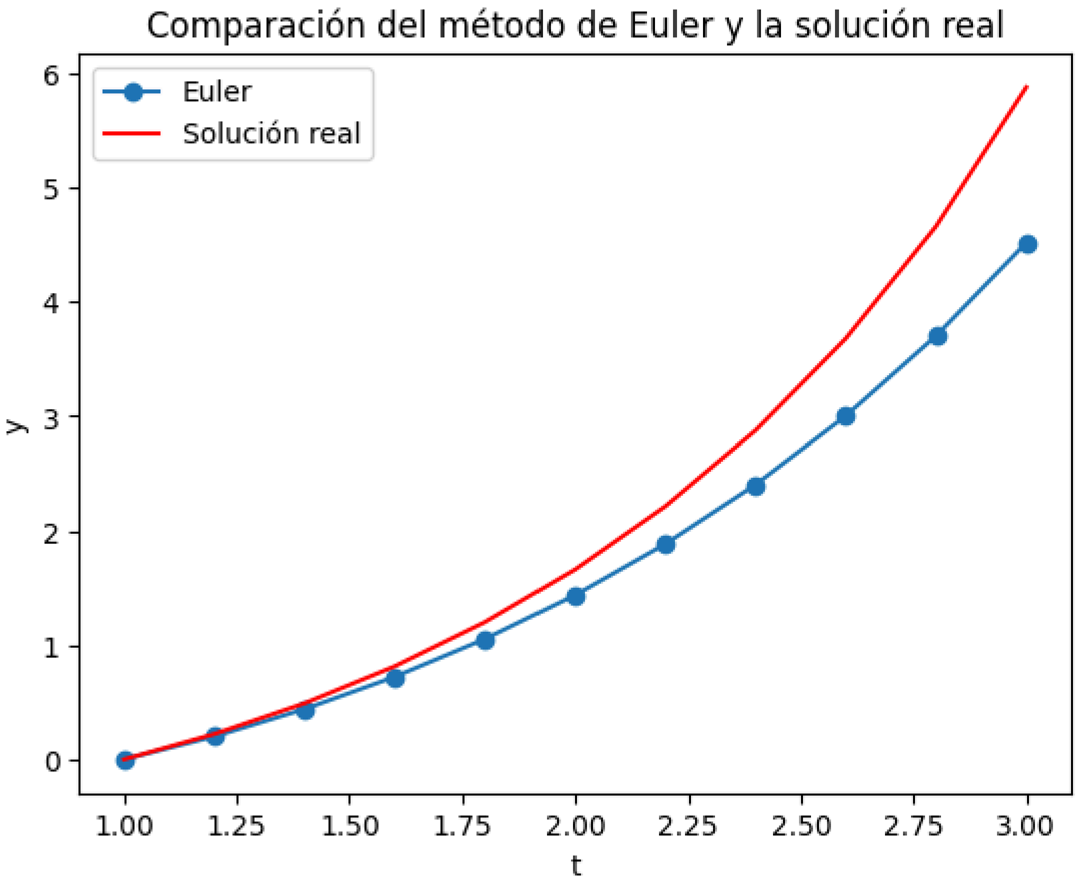


b. $y(t) = t\tan(\ln t)$

In [36]:
```python
def y_real_b(t):
    return t*np.tan(np.log(t))
```

```
t_aprox_3b, y_aprox_3b, y_real_b, err_b = euler_con_error(f_b, y_real_b, t0=1, y
```

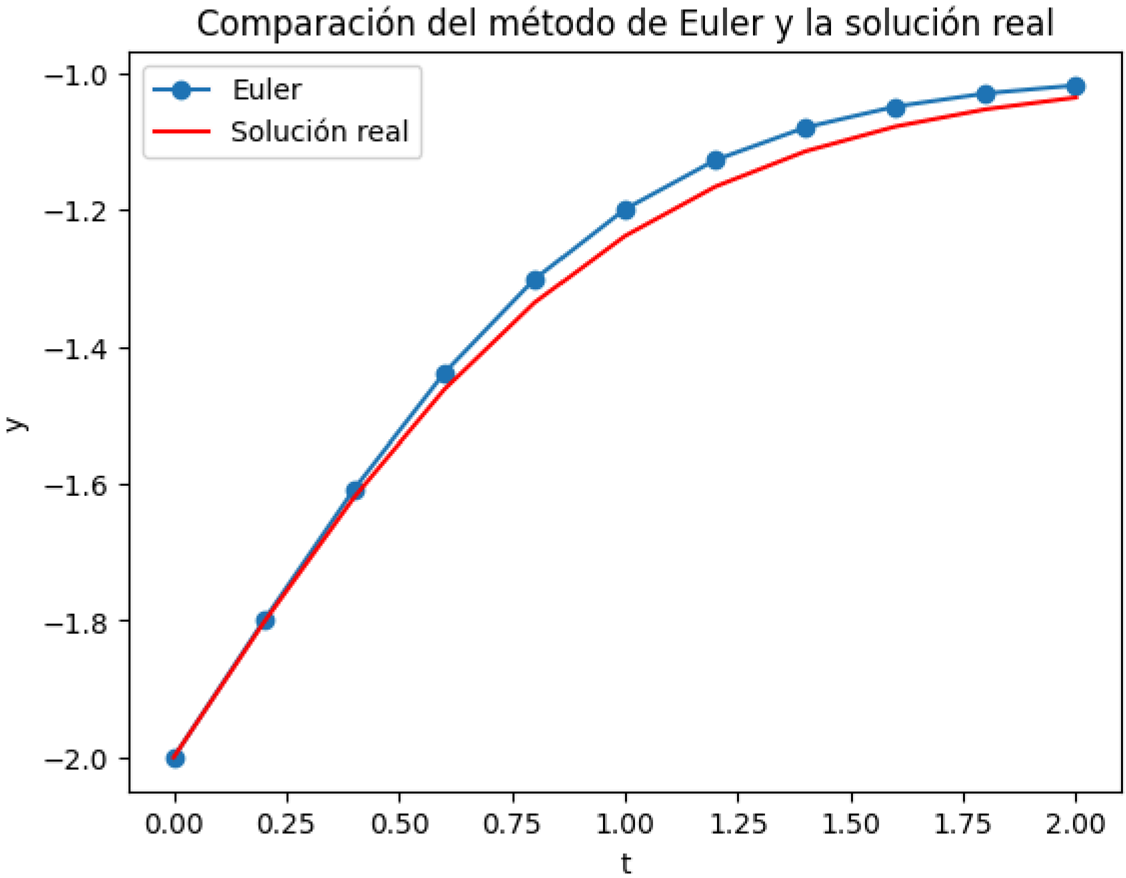
```
Error real para cada t:
t = 1, Error = 0.0
t = 1.2, Error = 0.021242772757631118
t = 1.4, Error = 0.05079277486205375
t = 1.5999999999999999, Error = 0.09150998419973799
t = 1.7999999999999998, Error = 0.14740060866856886
t = 1.9999999999999998, Error = 0.2240306081977277
t = 2.1999999999999997, Error = 0.32924100815147983
t = 2.4, Error = 0.4742818314333004
t = 2.6, Error = 0.6756381662946311
t = 2.8000000000000003, Error = 0.9580643533067188
t = 3.0000000000000004, Error = 1.359822550007471
```



c. $y(t) = -3 + \frac{2}{1+e^{-2t}}$

In [46]:
```
def y_real_c(t):
    return -3+2/(1+np.exp(-2*t))

t_aprox_3c, y_aprox_3c, y_real_c, err_c = euler_con_error(f_c, y_real_c, t0=0, y
```

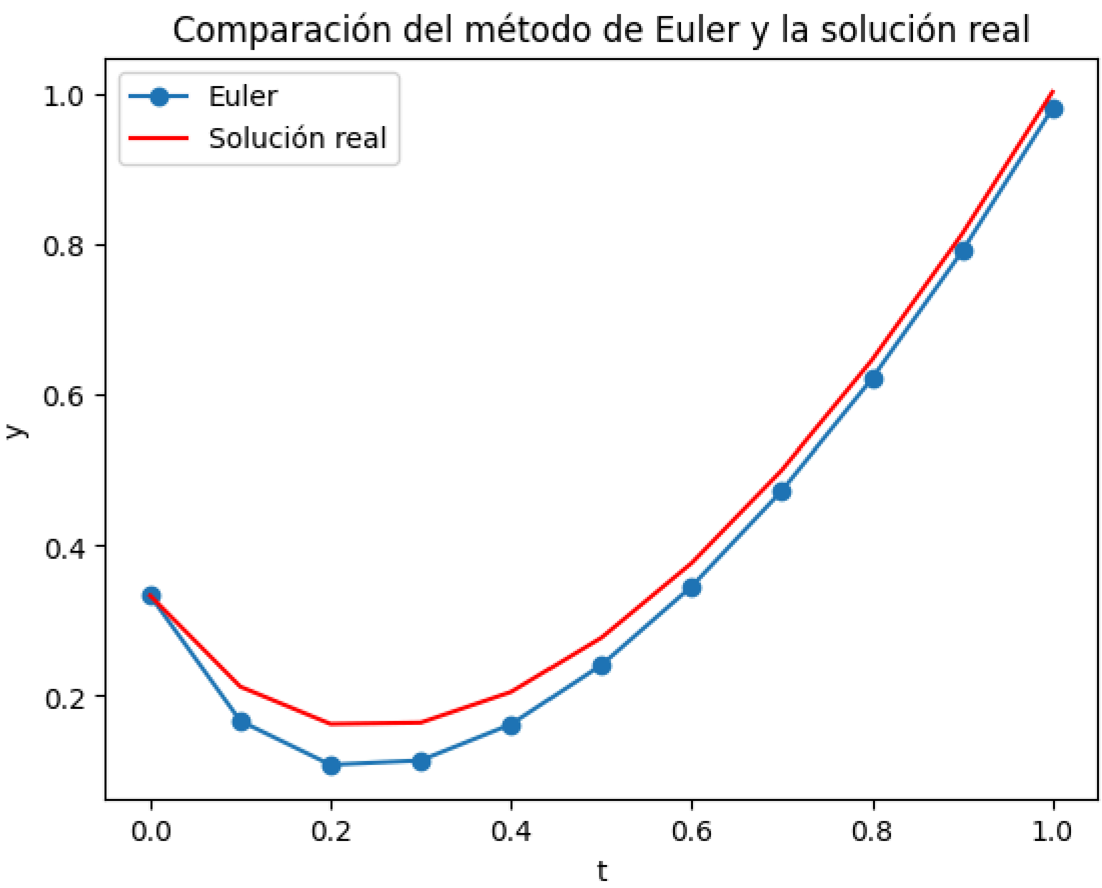
```
Error real para cada t:
t = 0, Error = 0.0
t = 0.2, Error = 0.0026246797750959505
t = 0.4, Error = 0.012051037744774895
t = 0.6000000000000001, Error = 0.024217633001964334
t = 0.8, Error = 0.0342262557729828
t = 1.0, Error = 0.0391546193779273
t = 1.2, Error = 0.03885444808185512
t = 1.4, Error = 0.034902996647539375
t = 1.5999999999999999, Error = 0.029212368169803555
t = 1.7999999999999998, Error = 0.023240003946105537
t = 1.9999999999999998, Error = 0.017820581777606703
```



d. $y(t) = t^2 + \frac{1}{3}e^{-5t}$

In [38]:
```python
def y_real_d(t):
    return t**2+1/3*np.exp(-5*t)

t_aprox_3d, y_aprox_3d, y_real_d, err_d = euler_con_error(f_d, y_real_d, t0=0, y
```

```
Error real para cada t:
t = 0, Error = 0.0
t = 0.1, Error = 0.04551021990421114
t = 0.2, Error = 0.05429314705714744
t = 0.30000000000000004, Error = 0.050210053382809955
t = 0.4, Error = 0.043028427745537556
t = 0.5, Error = 0.03631999954129955
t = 0.6, Error = 0.031074856122621286
t = 0.7, Error = 0.027305377807439468
t = 0.7999999999999999, Error = 0.02472500462957805
t = 0.8999999999999999, Error = 0.02301289467941392
t = 0.9999999999999999, Error = 0.021900930249695083
```



Ejercicio 5

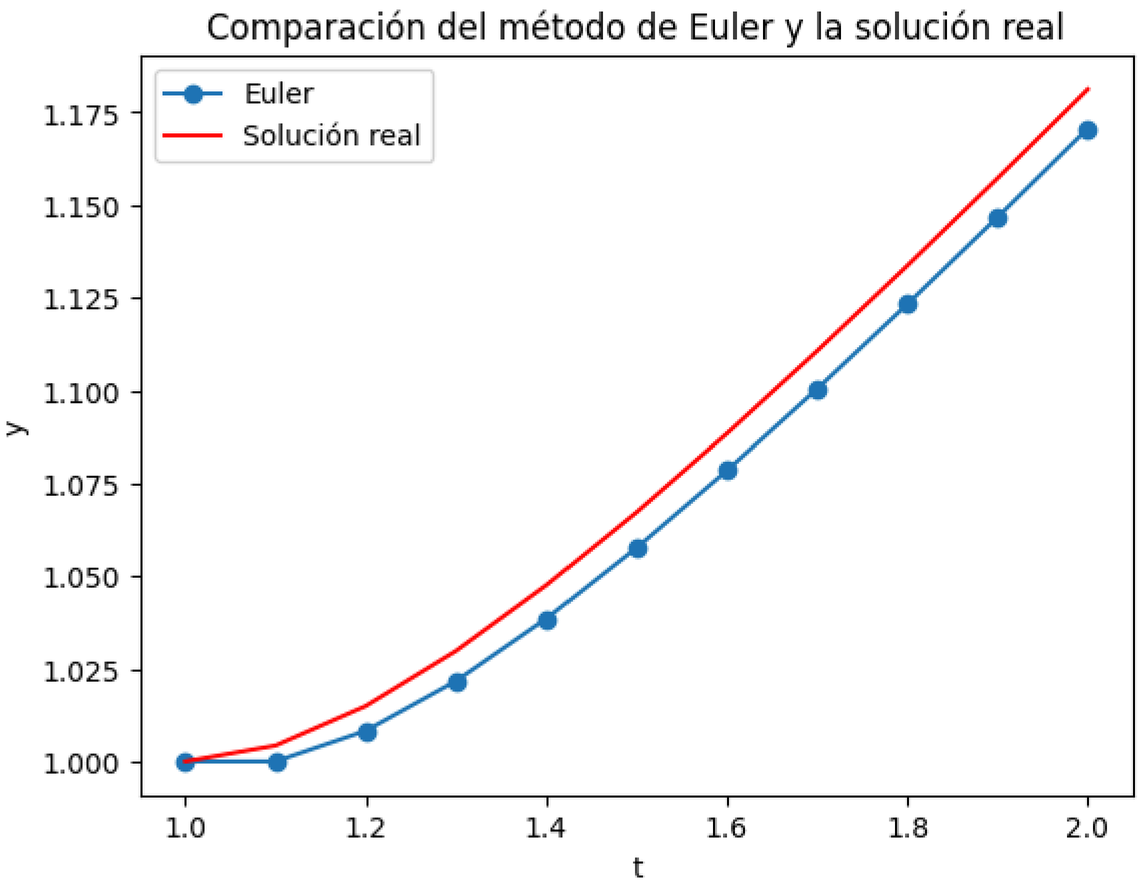
Utilice los resultados del ejercicio 3 y la interpolación lineal para aproximar los siguientes valores de $y(t)$. Compare las aproximaciones asignadas para los valores reales obtenidos mediante las funciones determinadas en el ejercicio 4.

In [39]:
```
def euler_con_interpolacion(f, real_func, t0, y0, tf, h, t_evaluar):
    # Obtenemos los resultados del método de Euler y la solución real
    t_vals, y_vals, y_real, errores = euler_con_error(f, real_func, t0, y0, tf,

    # Interpolación lineal para aproximar y(t) en los puntos de interés
    y_aprox_interpolados = np.interp(t_evaluar, t_vals, y_vals)
    y_real_interpolados = [real_func(ti) for ti in t_evaluar]

    for ti, ya, yr in zip(t_evaluar, y_aprox_interpolados, y_real_interpolados):
        print(f"t = {ti}, y_aprox = {ya}, y_real = {yr}, error = {abs(ya - yr)}"

    # Graficar resultados
    plt.figure()
    plt.plot(t_vals, y_vals, 'o-', label='Euler')
    t_fino = np.linspace(t0, tf, 100)
    y_fino = [real_func(ti) for ti in t_fino]
    plt.plot(t_fino, y_fino, 'r-', label='Solución real')
    # Puntos interpolados
    plt.plot(t_evaluar, y_aprox_interpolados, 's', label='Interpolación')
    plt.xlabel('t')
    plt.ylabel('y')
    plt.title('Interpolación lineal de la aproximación de Euler')
    plt.legend()
    plt.show()
```

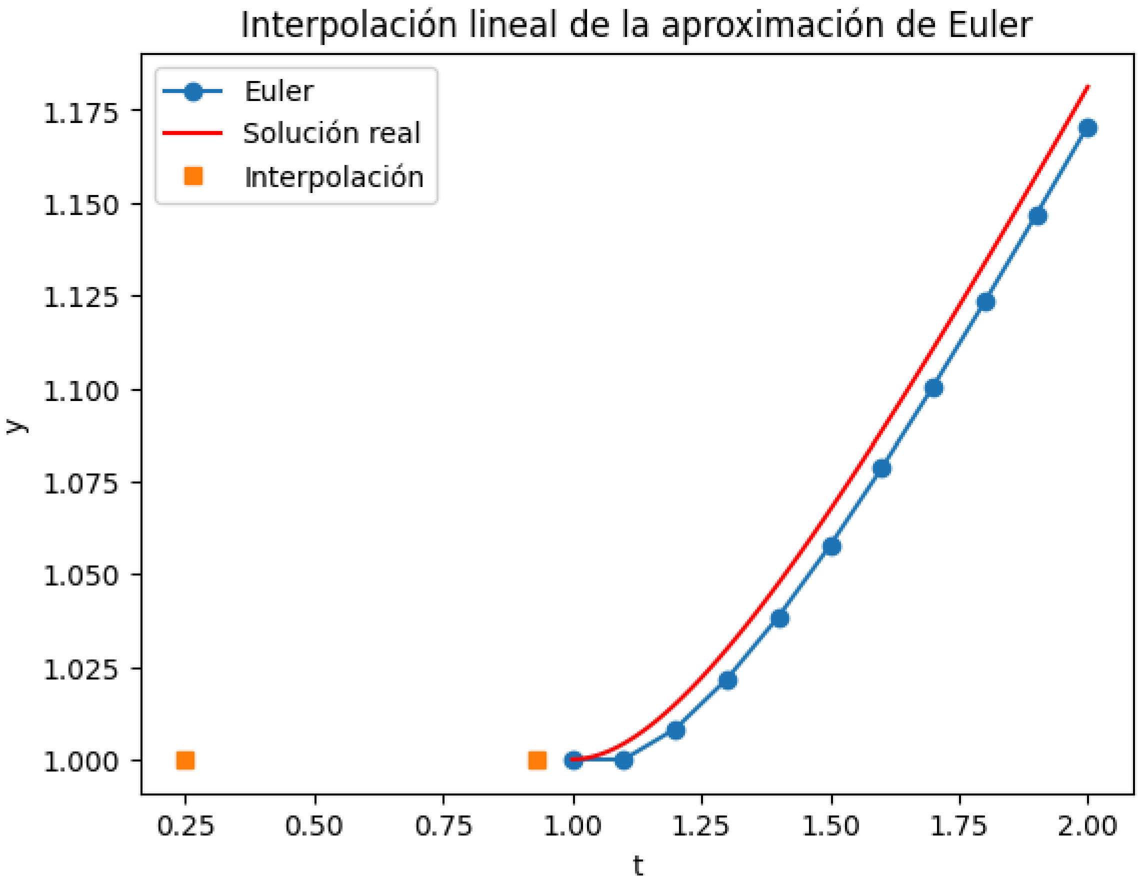
a. $y(0.25)$ y $y(0.93)$

In [42]:
```
def f_a(t, y):
    return (y / t) - (y / t)**2

def y_real_a(t):
    return t / (1 + np.log(t))

euler_con_interpolacion(f_a, y_real_a, t0=1, y0=1, tf=2, h=0.1, t_evaluar=[0.25,
```

```
Error real para cada t:
t = 1, Error = 0.0
t = 1.1, Error = 0.004281727936202406
t = 1.2000000000000002, Error = 0.006687851223824204
t = 1.3000000000000003, Error = 0.00812421723094725
t = 1.4000000000000004, Error = 0.009019185004341734
t = 1.5000000000000004, Error = 0.009594162040996945
t = 1.6000000000000005, Error = 0.009971593314036298
t = 1.7000000000000006, Error = 0.010222887446798445
t = 1.8000000000000007, Error = 0.010391505152042235
t = 1.9000000000000008, Error = 0.010504836525143224
t = 2.000000000000001, Error = 0.010580648734618059
```



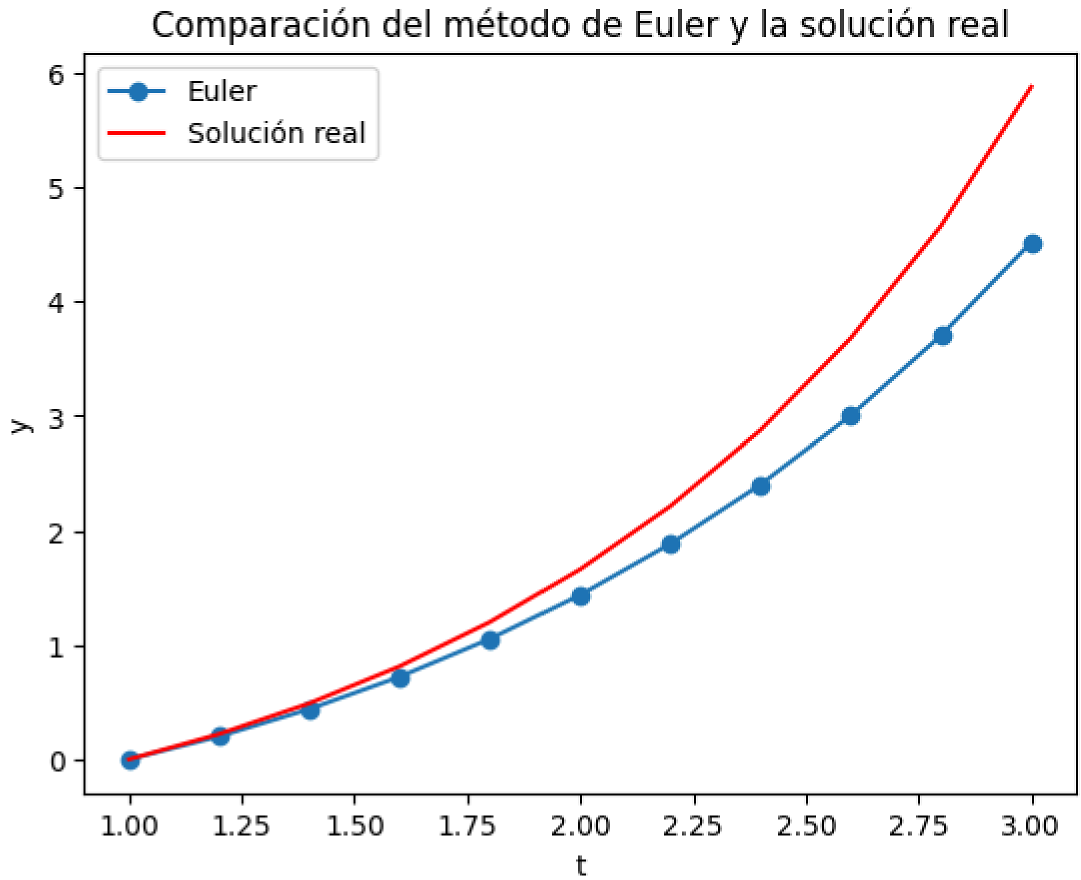

```
t = 0.25, y_aprox = 1.0, y_real = -0.6471748623905226, error = 1.6471748623905227
t = 0.93, y_aprox = 1.0, y_real = 1.0027718477462106, error = 0.00277184774621064
28
```



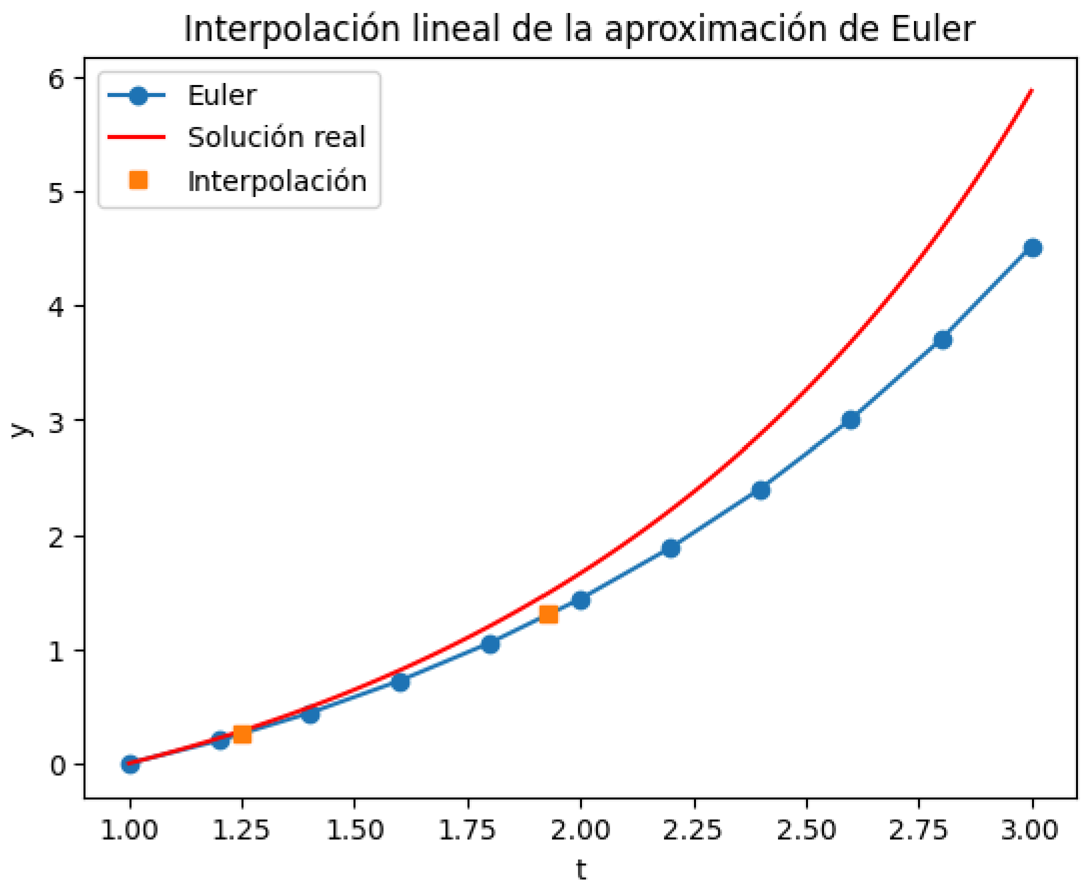
b. $y(t) = y(1.25)$ y $y(1.93)$

```
In [ ]: def f_b(t, y):
            return 1+y/t+(y/t)**2

        def y_real_b(t):
            return t*np.tan(np.log(t))

        euler_con_interpolacion(f_b, y_real_b, t0=1, y0=0, tf=3, h=0.2, t_evaluar=[1.25,
```

```
Error real para cada t:
t = 1, Error = 0.0
t = 1.2, Error = 0.021242772757631118
t = 1.4, Error = 0.05079277486205375
t = 1.5999999999999999, Error = 0.09150998419973799
t = 1.7999999999999998, Error = 0.14740060866856886
t = 1.9999999999999998, Error = 0.2240306081977277
t = 2.1999999999999997, Error = 0.32924100815147983
t = 2.4, Error = 0.4742818314333004
t = 2.6, Error = 0.6756381662946311
t = 2.8000000000000003, Error = 0.9580643533067188
t = 3.0000000000000004, Error = 1.359822550007471
```



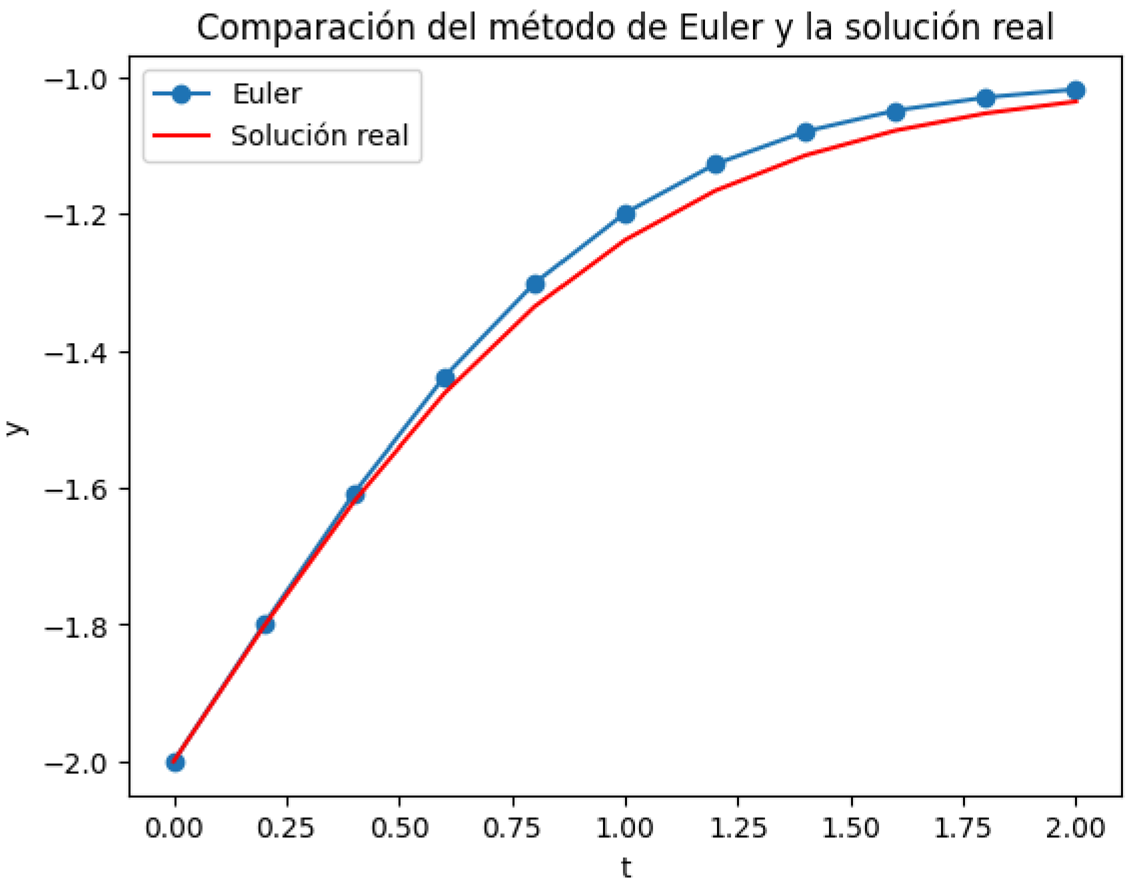

```
t = 1.25, y_aprox = 0.2597222222222223, y_real = 0.2836531261952289, error = 0.02
3930903973006623
t = 1.93, y_aprox = 1.3024265569705757, y_real = 1.4902277738186658, error = 0.18
780121684809004
```



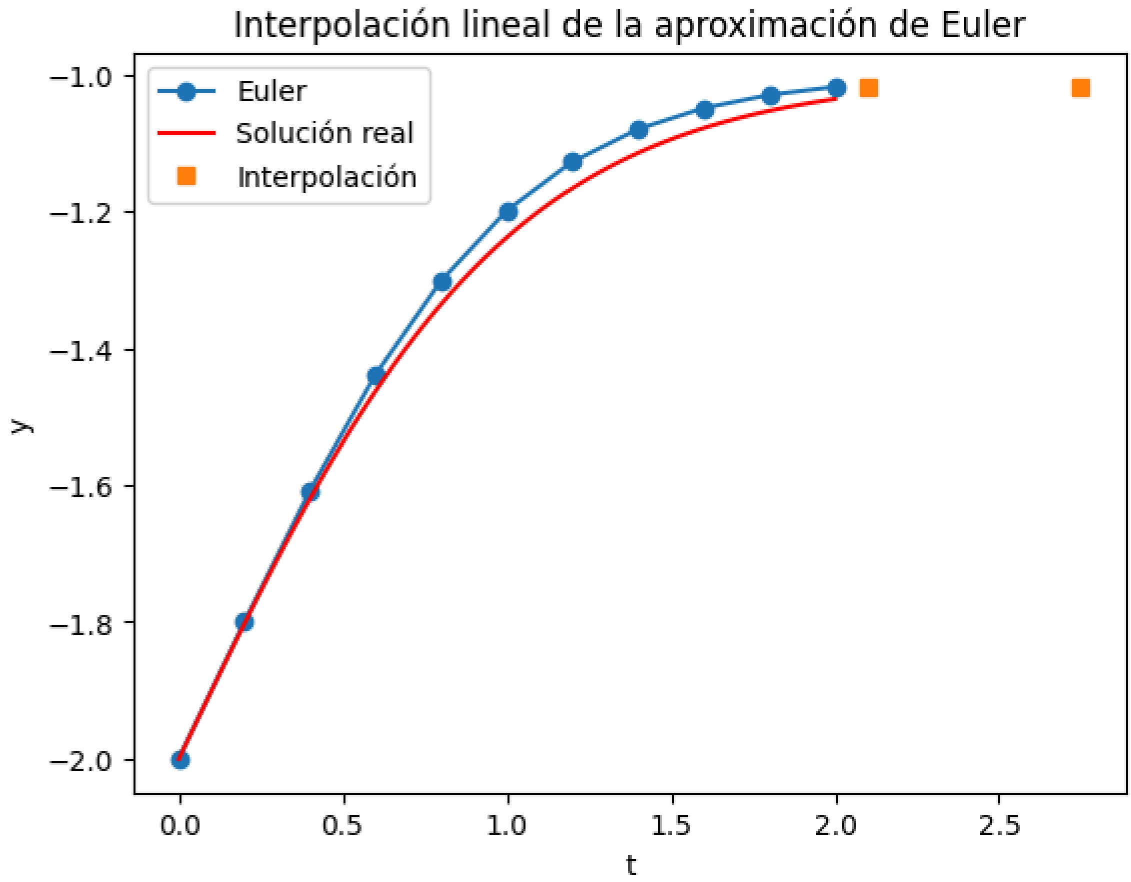
c. $y(2.10)$ y $y(2.75)$

In [47]:
```
def f_c(t, y):
    return -(y+1)*(y+3)

def y_real_c(t):
    return -3+2/(1+np.exp(-2*t))

euler_con_interpolacion(f_c, y_real_c, t0=0, y0=-2, tf=2, h=0.2, t_evaluar=[2.1
```

```
Error real para cada t:
t = 0, Error = 0.0
t = 0.2, Error = 0.0026246797750959505
t = 0.4, Error = 0.012051037744774895
t = 0.6000000000000001, Error = 0.024217633001964334
t = 0.8, Error = 0.03422625577298288
t = 1.0, Error = 0.0391546193779273
t = 1.2, Error = 0.03885444808185512
t = 1.4, Error = 0.034902996647539375
t = 1.5999999999999999, Error = 0.02921236816980355
t = 1.7999999999999998, Error = 0.02324000394610537
t = 1.9999999999999998, Error = 0.017820581777606703
```



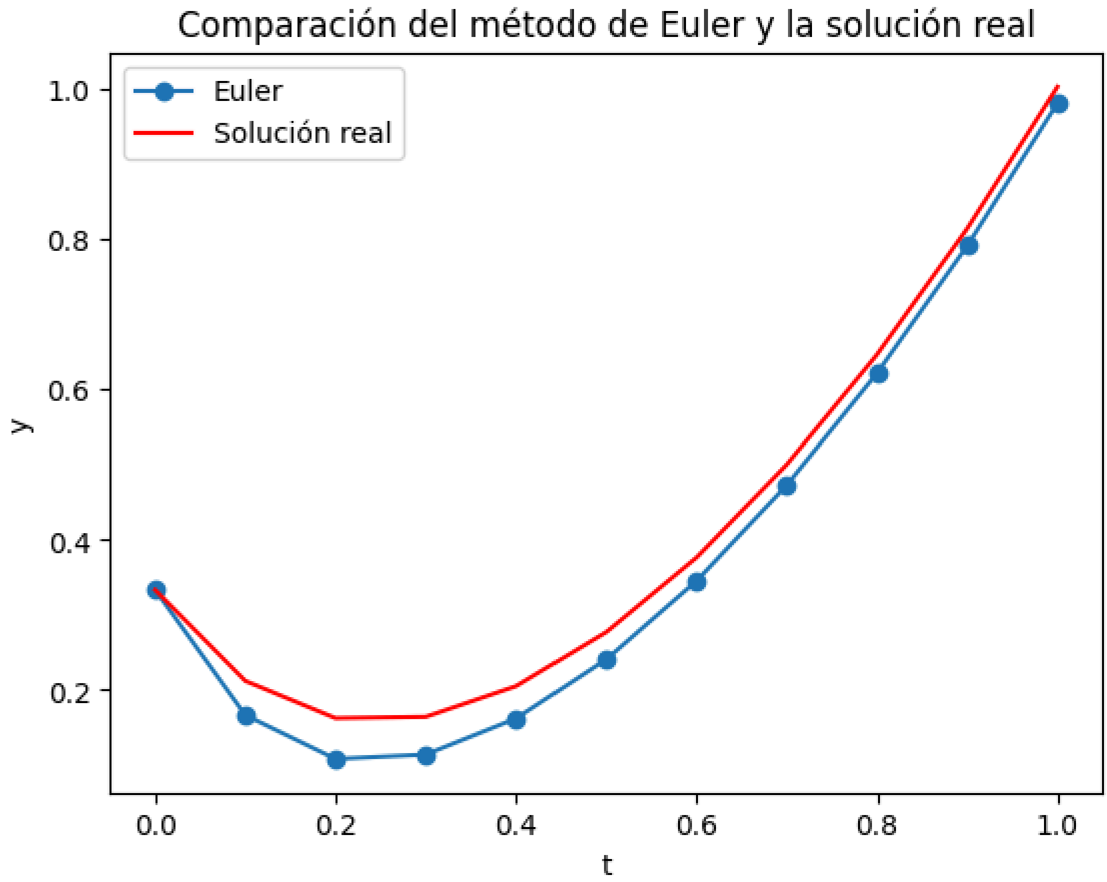

```
t = 2.1, y_aprox = -1.0181518381465764, y_real = -1.0295480633865461, error = 0.0
11396225239969748
t = 2.75, y_aprox = -1.0181518381465764, y_real = -1.008140275431792, error = 0.0
10011562714784317
```



d. $y(t) = y(0.54)$ y $y(0.94)$

In [48]:
```
def f_d(t, y):
    return -5*y+5*t**2+2*t

def y_real_d(t):
    return t**2+1/3*np.exp(-5*t)

euler_con_interpolacion(f_d, y_real_d, t0=0, y0=1/3, tf=1, h=0.1, t_evaluar=[0.5
```

```
Error real para cada t:
t = 0, Error = 0.0
t = 0.1, Error = 0.04551021990421114
t = 0.2, Error = 0.05429314705714744
t = 0.30000000000000004, Error = 0.050210053382809955
t = 0.4, Error = 0.043028427745537556
t = 0.5, Error = 0.03631999954129955
t = 0.6, Error = 0.031074856122621286
t = 0.7, Error = 0.027305377807439468
t = 0.7999999999999999, Error = 0.02472500462957805
t = 0.8999999999999999, Error = 0.02301289467941392
t = 0.9999999999999999, Error = 0.021900930249695083
```




```
t = 0.54, y_aprox = 0.2828333333333334, y_real = 0.3140018375799166, error = 0.03116850424658324
t = 0.94, y_aprox = 0.8665520833333333, y_real = 0.8866317590338986, error = 0.020079675700565236
```