In [19]:

```
from numpy import array
import numpy as np
def taylor (deriv,x,y,xStop,h):
    X=[]
    Y=[]
    X.append(x)
    Y.append(y)
    while x < xStop:
        D= deriv(x,y)
        H=1.0
        for j in range (3):
            H=H*h/(j+1)
            y=y+D[j]*H
        x=x+h
        X.append(x)
        Y.append(y)
    return array(X),array(Y)
def deriv (x,y):
    D=np.zeros((4,1))
    D[0]=[2*y[0]+3*np.exp(x)]
    D[1]=[4*y[0]+9*np.exp(x)]
    D[2]=[8*y[0]+218*np.exp(x)]
    D[3]=[16*y[0]+45*np.exp(x)]
    return D
x=0.0
xStop=0.3
y=array([0.0])
h=0.1
X,Y=taylor(deriv ,x, y, xStop, h)
print("The required values are :at x= %0.2f, y=%0.5f, x=%0.2f, y=%0.5f, x=%0.2f, y=%0.5f,
```

```
The required values are :at x= 0.00, y=0.00000, x=0.10, y=0.38133, x=0.20,
y=0.88717, x= 0.30, y=1.54930
```

In [29]:

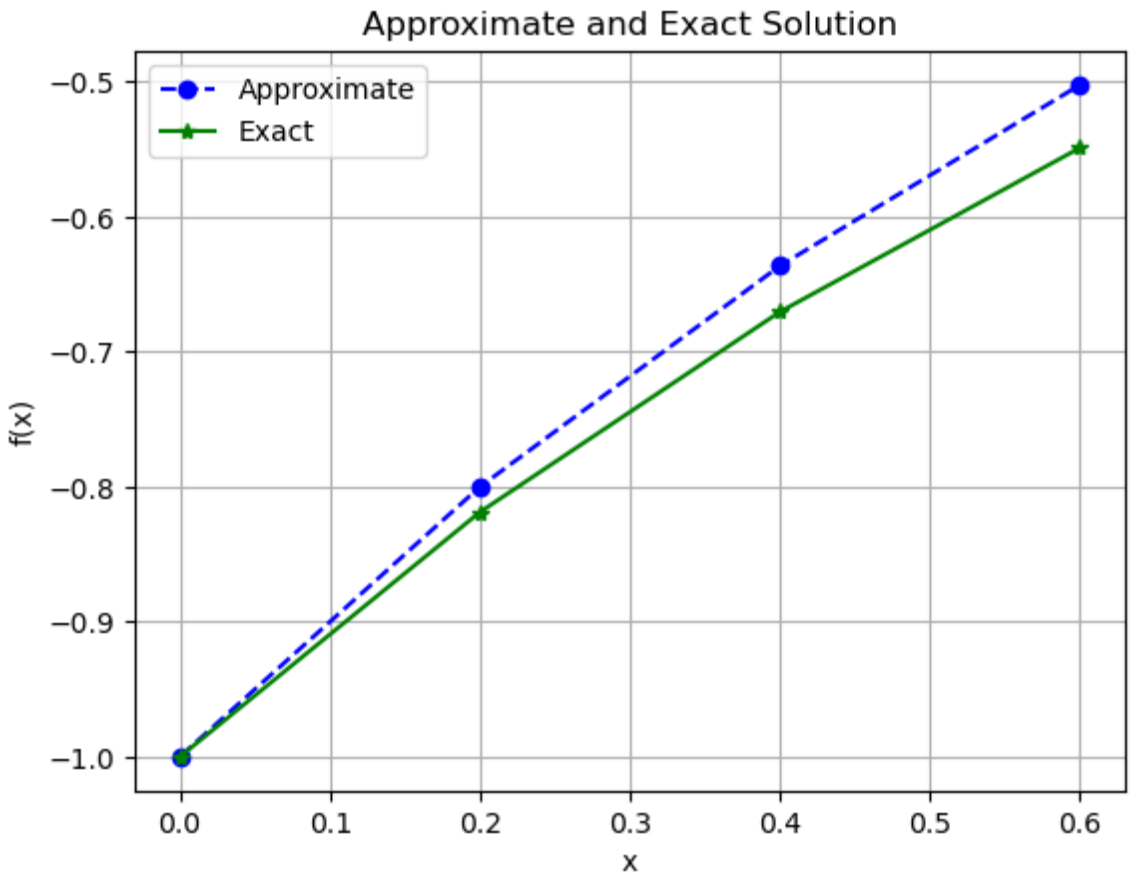
```
from numpy import array
import numpy as np
def taylor (deriv, x, y, xStop, h):
    X=[]
    Y=[]
    X.append(x)
    Y.append(y)
    while x< xStop:
        D= deriv(x,y)
        H=1.0
        for j in range (3):
            H=H*h/(j+1)
            y=y+D[j]*H
        x=x+h
        X.append(x)
        Y.append(y)
    return array(X),array(Y)
def deriv (x,y):
    D=np.zeros((4,1))
    D[0]=[x**2-4*y[0]]
    D[1]=[2*x-4*x**2+16*y[0]]
    D[2]=[2-8*x+16*x**2-64*y[0]]
    D[3]=[-8+32*x-64*x**2+256*y[0]]
    return D
x=0.0
xStop=0.2
y=array([1.0])
h=0.1
X,Y=taylor(deriv,x,y,xStop,h)
print("The required values are :at x=%0.2f, y=%0.5f,x=%0.2f, y=%0.5f,x=%0.2f,y=%0.5f"% (X
```

```
The required values are :at x=0.00, y=1.00000,x=0.10, y=0.66967,x=0.20,y=
0.45026
```

In [33]:

```
import numpy as np
import matplotlib.pyplot as plt
f=lambda x,y:np.exp(-x)
x=np.zeros(n+1)
y=np.zeros(n+1)
h= 0.2
y0 = -1
n=3
y[0]=y0
x[0]=0
for i in range (0,n):
    x[i+1]=x[i]+h
    y[i+1]=y[i]+h*f(x[i], y[i])
print("The required values are :at x= %0.2f, y=%0.5f, x=%0.2f, y=%0.5f, x=%0.2f, y=%0.5f,
print("\n\n")
plt.plot(x,y, 'bo--',label='Approximate')
plt.plot(x, -np.exp(-x), 'g*-', label='Exact')
plt.title("Approximate and Exact Solution")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid()
plt.legend(loc='best')
plt.show()
```

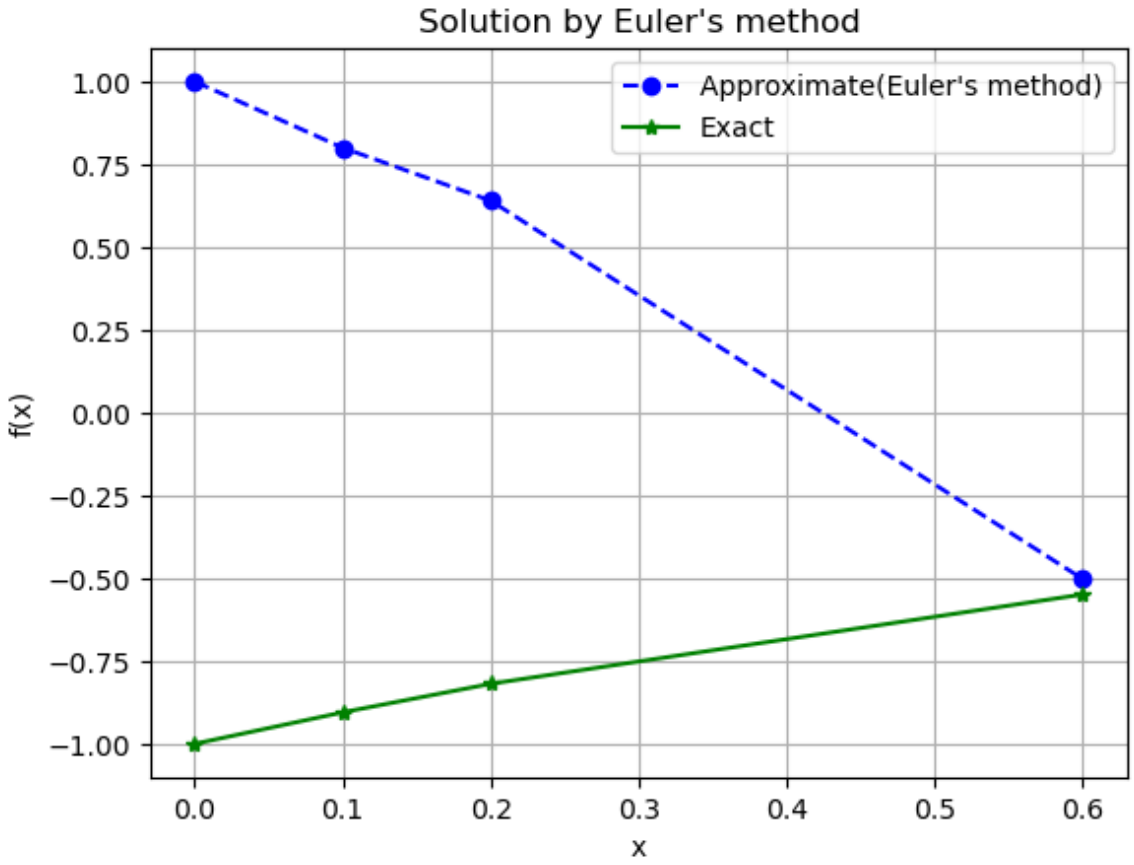
```
The required values are :at x= 0.00, y=-1.00000, x=0.20, y=-0.80000, x=0.4
0, y=-0.63625, x= 0.60, y=-0.50219
```



In [35]:

```
import numpy as np
import matplotlib.pyplot as plt
f=lambda x,y: -2*y+(x**3)*np.exp(-2*x)
h= 0.1
y0 = 1
n=2
y[0]=y0
x[0]=0
for i in range (0,n):
    x[i+1]=x[i]+h
    y[i+1]=y[i]+h*f(x[i], y[i])
print("The required values are :at x= %0.2f, y=%0.5f, x=%0.2f, y=%0.5f, x=%0.2f, y=%0.5f\
plt.plot(x,y, 'bo--',label="Approximate(Euler's method)")
plt.plot(x, -np.exp(-x), 'g*-', label='Exact')
plt.title("Solution by Euler's method")
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid()
plt.legend(loc='best')
plt.show()
```

```
The required values are :at x= 0.00, y=1.00000, x=0.10, y=0.80000, x=0.20,
y=0.64008
```



In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
def modified_euler(f, x0, y0, h,n):
    x=np.zeros(n+1)
    y=np.zeros(n+1)
    x[0]=x0
    y[0]=y0
    for i in range (n):
        x[i+1]=x[i]+h
        k1=h*f(x[i], y[i])
        k2= h*f(x[i+1],y[i])
        y[i+1]=y[i]+0.5* (k1+k2)
    return x,y
```