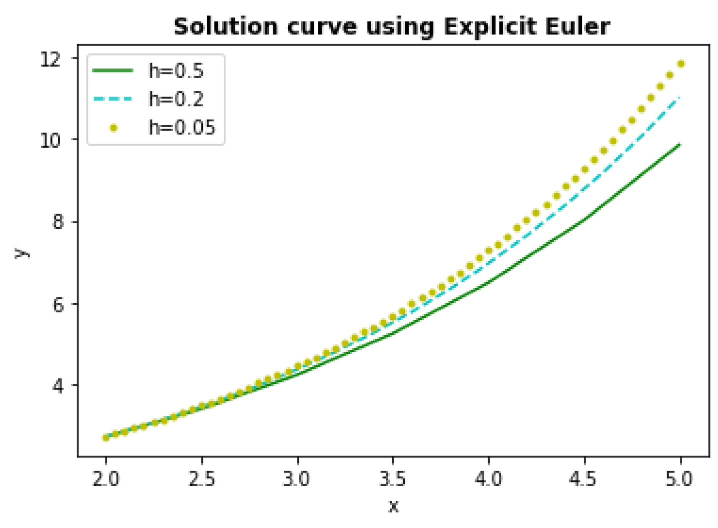
In [49]:
```python
%run my_functions_library.ipynb                                     #running my library here
import math
import numpy as np
```

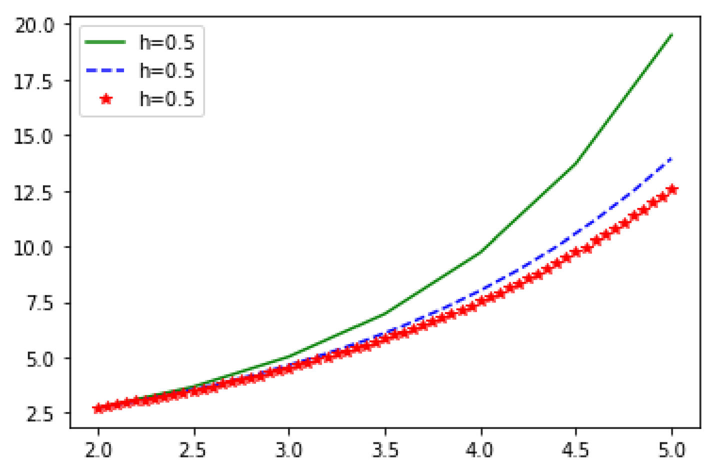
QUESTION 1

Use Explicit Euler And Predictor Collector

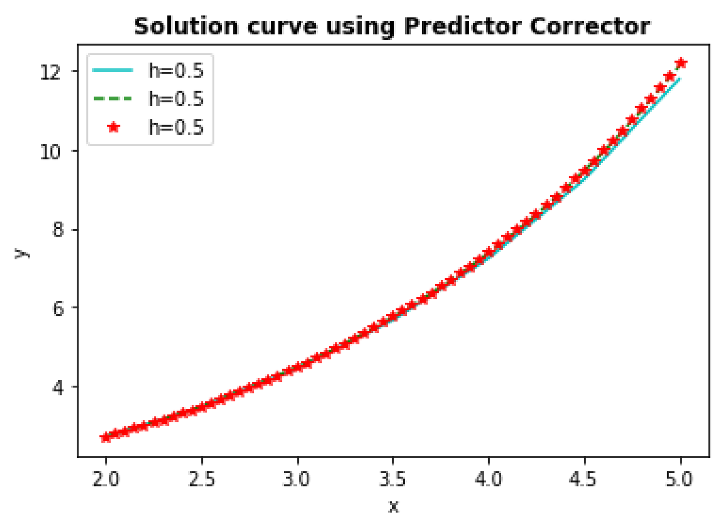
In [59]:
```python
func = lambda x,y: (y*math.log(y))/x                              # using explicit euler method with 3 h values

lists1=euler_forward_yours(func,0.5,2,2.71828,5)

lists2 =euler_forward_yours(func,0.2,2,2.71828,5)

lists3 =euler_forward_yours(func,0.05,2,2.71828,5)


plt.plot(lists1[0],lists1[1],'g-',label='h=0.5')                        # plotting the solution obtained
plt.plot(lists2[0],lists2[1],'c--',label='h=0.2')                       # using euler method on 1 graph
plt.plot(lists3[0],lists3[1],'y.',label='h=0.05')
plt.title('Solution curve using Explicit Euler',fontweight="bold")
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```



In [41]:
```python
func = lambda x,y: (y*math.log(y))/x

lists3=euler_backward(func,0.5,2,2.71828,5)
lists4 =euler_backward(func,0.2,2,2.71828,5)
lists5 =euler_backward(func,0.05,2,2.71828,5)

plt.plot(lists3[0],lists3[1],'g-',label='h=0.5')
plt.plot(lists4[0],lists4[1],'c--',label='h=0.5')
plt.plot(lists5[0],lists5[1],'r*',label='h=0.5')
plt.legend()
plt.show()
```



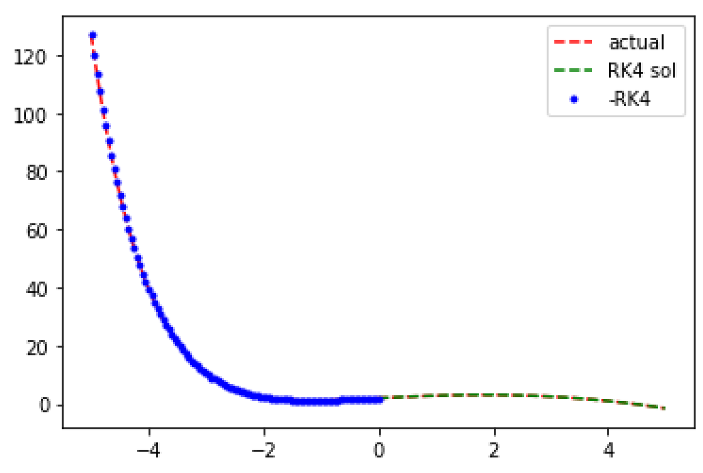
In [57]:
```python
# Predictor Corrector Method


func = lambda x,y: (y*math.log(y))/x

lists6=predictor_corrector(func,0.5,2,2.71828,5)
lists7 =predictor_corrector(func,0.2,2,2.71828,5)
lists8 =predictor_corrector(func,0.05,2,2.71828,5)

plt.plot(lists6[0],lists6[1],'c-',label='h=0.5')
plt.plot(lists7[0],lists7[1],'g--',label='h=0.5')
plt.plot(lists8[0],lists8[1],'r*',label='h=0.5')
plt.title('Solution curve using Predictor Corrector',fontweight="bold")
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```



Question 2

In [53]:
```python
F=[lambda Y,x: Y[1],lambda Y,x:1-x-Y[1]]
x_o=0
Y_o=[[2],[1]]
S=RK4_notsolve_system(F,x_o,Y_o,0.05)
y1=S[0]
x1=S[2]
Y_o=[[2],[1]]
S_=RK4_notsolve_system(F,x_o,Y_o,-0.05)
y_=S_[0]
x_=S_[2]



x=np.arange(-5,5,0.01)
y=1+np.exp(-x)-x**2/2+2*x
plt.plot(x,y,"r--",label='actual')
plt.plot(x1,y1,"g--",label='RK4 sol')
plt.plot(x_,y_,"b.",label='-RK4')
plt.legend()
```

Out[53]:
```
<matplotlib.legend.Legend at 0x26a9bbd6ca0>
```

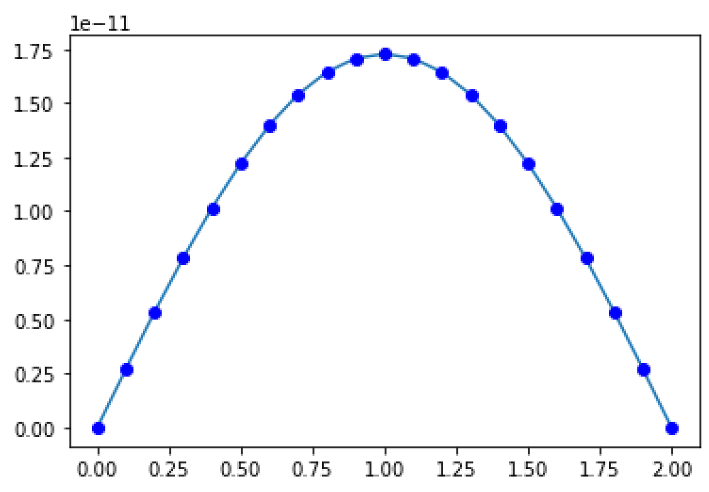


In [48]:
```python
"""Solve y' = f(x,y), intial condition y(x_o) = y_o, x_n is your choice"""
def RK4(fn,step_size,x_0,y_0,x_n):
    x,y = RK4()
```

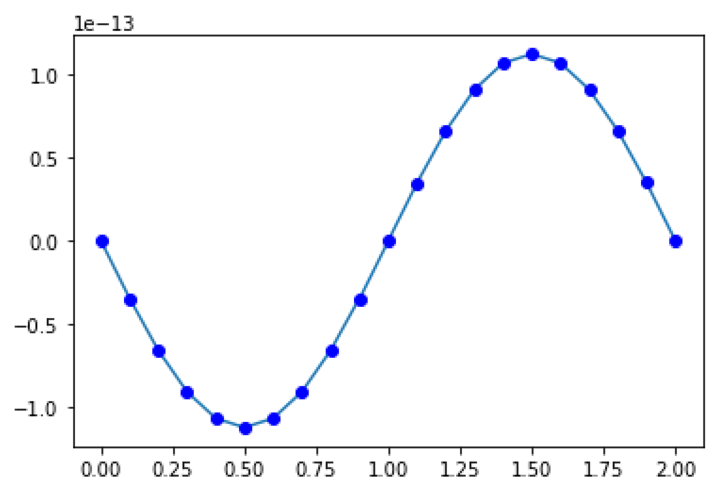
Question 3

In [55]:
```python
# def shooting_method(d2ydx2, dydx, x0, y0, xf, yf, z_guess1, z_guess2, step_size, tol=1e-6):

def d2ydt2(t, y, z):                                              # this is func for d2y/dt2 =func
    return -(math.pi**2)*y/4


def d2ydt3(t, y, z):                                             # this is func for d2y/dt2 =func
    return -(math.pi**2)*y


def dydt(t, y, z):                                                # z = dy/dt
    return z


t_initial = 0                                                      # Define boundary values
t_final = 2
y_initial = 0
y_final = 0

t, y, z = shooting_method(d2ydt2, dydt, t_initial, y_initial, t_final, y_final, -2, 10, step_size=0.1)
plt.plot(t,y)
print(f"velocity at t= 0  is {z[0]}")

plt.plot(t,y,'bo')
plt.show()

t, y, z = shooting_method(d2ydt3, dydt, t_initial, y_initial, t_final, y_final, -2, 10, step_size=0.1)
plt.plot(t,y)
print(f"velocity at t= 0  is {z[0]}")

plt.plot(t,y,'bo')
```

```
velocity at t= 0  is 2.7136959346307776e-11
```



Out[55]:
```
velocity at t= 0  is -3.5349501104064984e-13
[<matplotlib.lines.Line2D at 0x26a9bcf57f0>]
```



In [ ]: