In [1]:
```
from numpy import *
import matplotlib.pyplot as plt
```

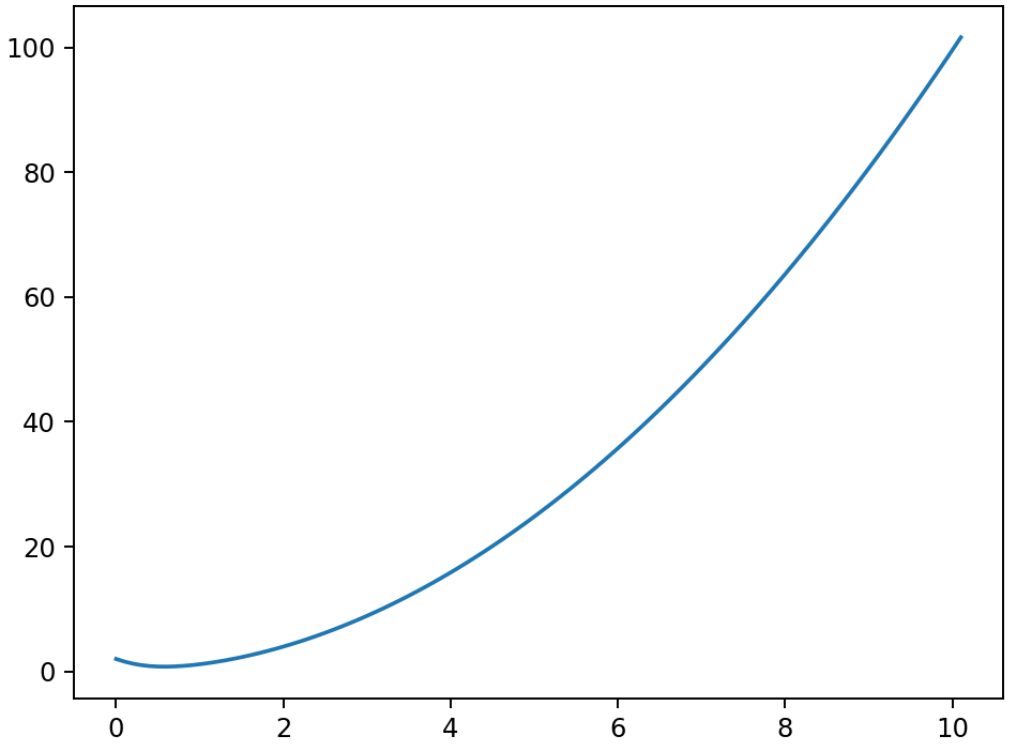
Example 1: y' = 5t - 3sqrt(y), y(0)=2 Using Euler's Method, estimate y(10)) with stepsize h=0.1

In [2]:
```
#Step 1: define function fofty using the def command
def fofty(t,y):
    return 5*t-3*y**0.5
#Step 2: input t0 and y0. Using tplot and yplot for the lists
tplot=[0]
yplot=[2]
#Step 3: define stepsize h and number of steps n
h=.1
n=101  # n=(tf-t0)/h = (10-0)/0.1 = 100
#Step 4: Loop to create tplot and yplot
for j in range(0,n): #range(a,b) is from a INCLUSIVE to b EXCLUSIVE
    fj=fofty(tplot[j],yplot[j])
    yplot.append(yplot[j]+h*fj)
    tplot.append(tplot[j]+h)
#Step 5: output y-value at the end
print('y(',tplot[n-1],') is approximately',yplot[n-1])
```

```
y( 9.99999999999998 ) is approximately 99.6054479142069
```

In [3]:
```
matplotlib notebook
```

In [4]:
```
plt.plot(tplot,yplot)
```



Out[4]:
```
[<matplotlib.lines.Line2D at 0x24655c4e908>]
```

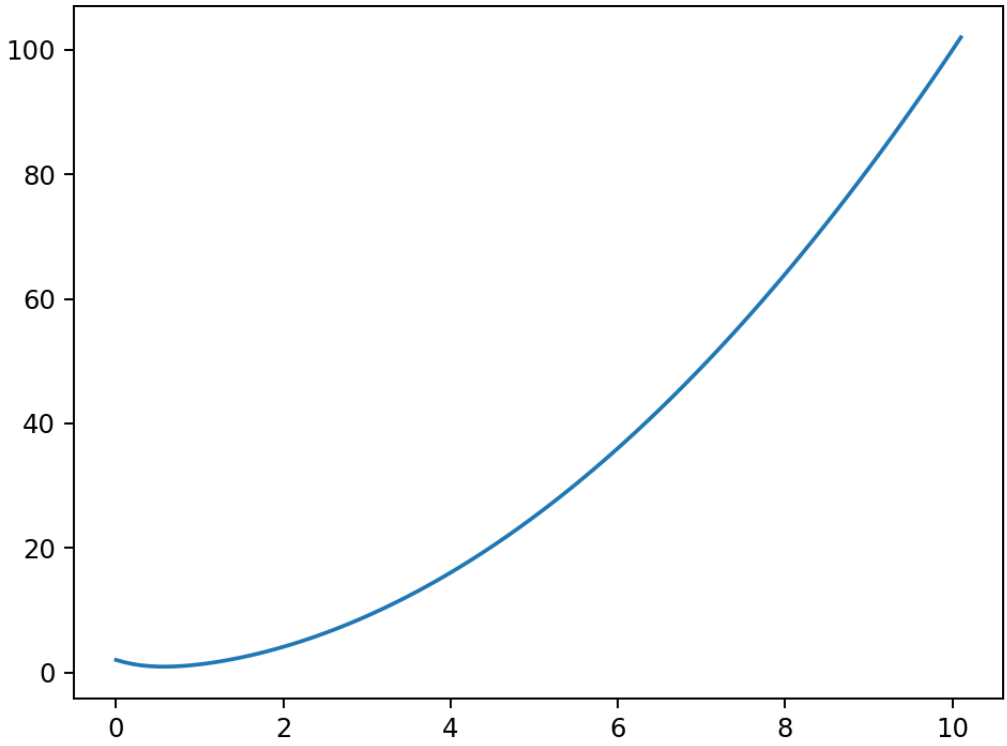
Example 2: y' = 5t - 3sqrt(y), y(0)=2 Using Improved Euler's Method, estimate y(10)) with stepsize h=0.1, then h=0.001

In [5]:
```
#Step 1: define function fofty using the def command
def fofty(t,y):
    return 5*t-3*y**0.5
#Step 2: input t0 and y0. Using tplot and yplot for the lists
tplot=[0]
yplot=[2]
#Step 3: define stepsize h and number of steps n
h=.1
n=101  # n=(tf-t0)/h = (10-0)/0.1 = 100
#Step 4: Loop to create tplot and yplot
for j in range(0,n): #range(a,b) is from a INCLUSIVE to b EXCLUSIVE
    k1=fofty(tplot[j],yplot[j])
    k2=fofty(tplot[j]+h,yplot[j]+h*k1)
    yplot.append(yplot[j]+h/2*(k1+k2))
    tplot.append(tplot[j]+h)
#Step 5: output y-value at the end
print('y(',tplot[n-1],') is approximately',yplot[n-1])
```

```
y( 9.99999999999998 ) is approximately 100.01441066269453
```

In [6]:
```
matplotlib notebook
```

In [7]:
```
plt.plot(tplot,yplot)
```



Out[7]:
```
[<matplotlib.lines.Line2D at 0x2465976a4c8>]
```

In [8]:
```
#Step 1: define function fofty using the def command
def fofty(t,y):
    return 5*t-3*y**0.5
#Step 2: input t0 and y0. Using tplot and yplot for the lists
tplot=[0]
yplot=[2]
#Step 3: define stepsize h and number of steps n
h=.001
n=10001  # n=(tf-t0)/h = (10-0)/0.001 = 10000
#Step 4: Loop to create tplot and yplot
for j in range(0,n): #range(a,b) is from a INCLUSIVE to b EXCLUSIVE
    k1=fofty(tplot[j],yplot[j])
    k2=fofty(tplot[j]+h,yplot[j]+h*k1)
    yplot.append(yplot[j]+h/2*(k1+k2))
    tplot.append(tplot[j]+h)
#Step 5: output y-value at the end
print('y(',tplot[n-1],') is approximately',yplot[n-1])
```

```
y( 9.999999999999897 ) is approximately 100.00928285968205
```

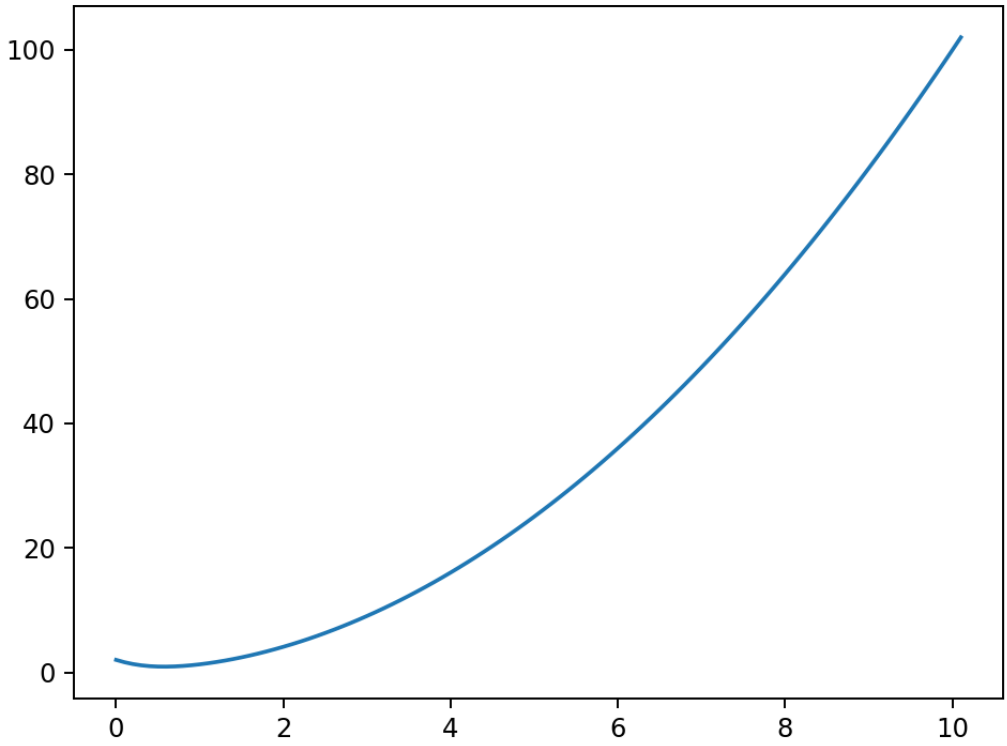
Same example using Runge-Kutta with h=0.1 and h=0.001

In [9]:
```
#Step 1: define function fofty using the def command
def fofty(t,y):
    return 5*t-3*y**0.5
#Step 2: input t0 and y0. Using tplot and yplot for the lists
tplot=[0]
yplot=[2]
#Step 3: define stepsize h and number of steps n
h=.1
n=101  # n=(tf-t0)/h = (10-0)/0.1 = 100
#Step 4: Loop to create tplot and yplot
for j in range(0,n): #range(a,b) is from a INCLUSIVE to b EXCLUSIVE
    k1=fofty(tplot[j],yplot[j])
    k2=fofty(tplot[j]+h/2,yplot[j]+h/2*k1)
    k3=fofty(tplot[j]+h/2,yplot[j]+h/2*k2)
    k4=fofty(tplot[j]+h,yplot[j]+h*k3)
    yplot.append(yplot[j]+h/6*(k1+2*k2+2*k3+k4))
    tplot.append(tplot[j]+h)
#Step 5: output y-value at the end
print('y(',tplot[n-1],') is approximately',yplot[n-1])
```

```
y( 9.99999999999998 ) is approximately 100.00928300820146
```

In [10]:
```
matplotlib notebook
```

In [11]:
```
plt.plot(tplot,yplot)
```



Out[11]:
```
[<matplotlib.lines.Line2D at 0x24659f37048>]
```

In [12]:
```
#Step 1: define function fofty using the def command
def fofty(t,y):
    return 5*t-3*y**0.5
#Step 2: input t0 and y0. Using tplot and yplot for the lists
tplot=[0]
yplot=[2]
#Step 3: define stepsize h and number of steps n
h=.001
n=10001  # n=(tf-t0)/h = (10-0)/0.001 = 10000
#Step 4: Loop to create tplot and yplot
for j in range(0,n): #range(a,b) is from a INCLUSIVE to b EXCLUSIVE
    k1=fofty(tplot[j],yplot[j])
    k2=fofty(tplot[j]+h/2,yplot[j]+h/2*k1)
    k3=fofty(tplot[j]+h/2,yplot[j]+h/2*k2)
    k4=fofty(tplot[j]+h,yplot[j]+h*k3)
    yplot.append(yplot[j]+h/6*(k1+2*k2+2*k3+k4))
    tplot.append(tplot[j]+h)
#Step 5: output y-value at the end
print('y(',tplot[n-1],') is approximately',yplot[n-1])
```

```
y( 9.999999999999897 ) is approximately 100.00928235253296
```

In [ ]:
```

```