Integration

In [8]:
```
from scipy.integrate import quad, dblquad, tplquad
import numpy as np
```

Integration

In [16]:
```
fx = lambda x: np.exp(-x**2)

lx = 0
ux = 4

val, err = quad(fx, lx, ux)
print(f'Value: {val}')
print(f'Error: {err}')
```

```
Value: 0.8862269117895689
Error: 1.3180149476235 46e-08
```

Double Integration

In [17]:
```
fx = lambda y, x: (x**2)+y

lx = 0
ux = 2

ly = lambda x: 0
uy = lambda x: x

val, err = dblquad(fx, lx, ux, ly, uy)
print(f'Value: {val}')
print(f'Error: {err}')
```

```
Value: 5.333333333333333
Error: 1.1034865012825346e-13
```

Triple integration

In [7]:
```
fx = lambda y,z,x : (x*y**2) + z
ly = lambda x, z: z
uy = lambda x, z: (1- 2*x)+z

lz = lambda x: -x
uz = lambda x: x

lx = 0
ux = 0.5

val, err = tplquad(fx, lx, ux, lz, uz, ly, uy)
print(f'Value: {val}')
print(f'Error: {err}')
```

```
Value: 0.0020833333333333337
Error: 1.7561035485227292e-15
```

Interpolation

In [25]:
```
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
```

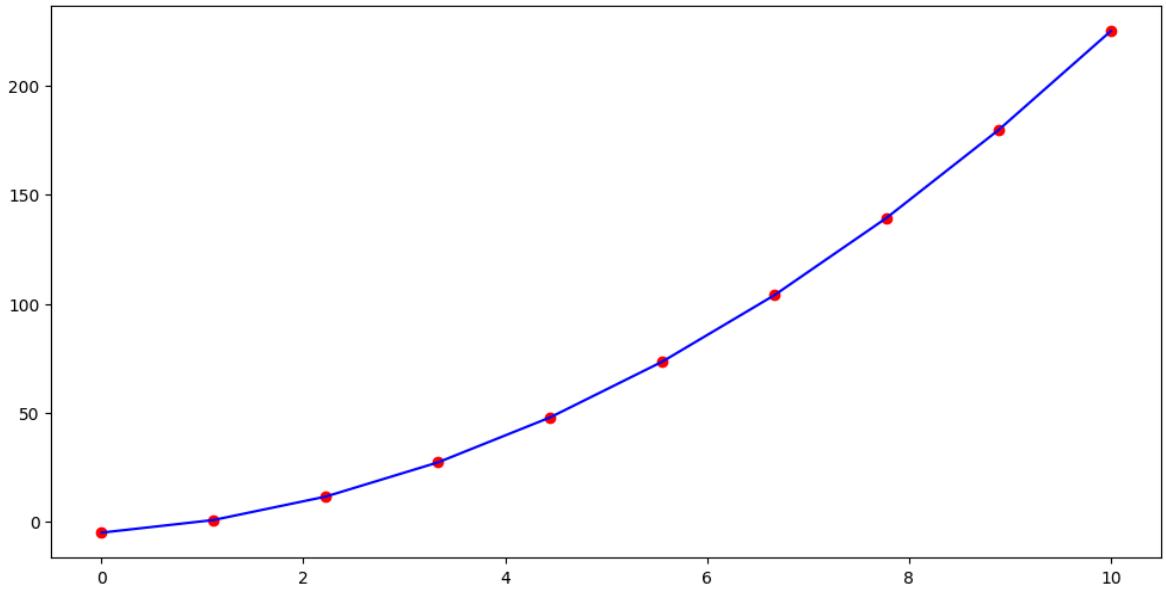
In [24]:
```
x = np.linspace(0, 10, 10)
y = 2*x**2 + 3*x- 5
y
```

Out[24]:
```
array([ -5.        ,   0.80246914,  11.54320988,  27.22222222,
        47.83950617,  73.39506173, 103.88888889, 139.32098765,
       179.69135802, 225.        ])
```

In [39]:
```
lp = interp1d(x,y, kind='linear')

fig, axs = plt.subplots(figsize = (12,6))
axs.plot(x,y, 'or')
axs.plot(x, lp(x), '-b')
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x29738e89a30>]
```



In [45]:
```
def allOddDigits(n):
    for i in list(str(n)):
        if int(i)%2 == 0:
            msg = 'There is a even number it has'
            break
    else:
        msg = 'All are ODD digits'

    return msg

allOddDigits(347)
```

Out[45]:
```
'There is a even number it has'
```

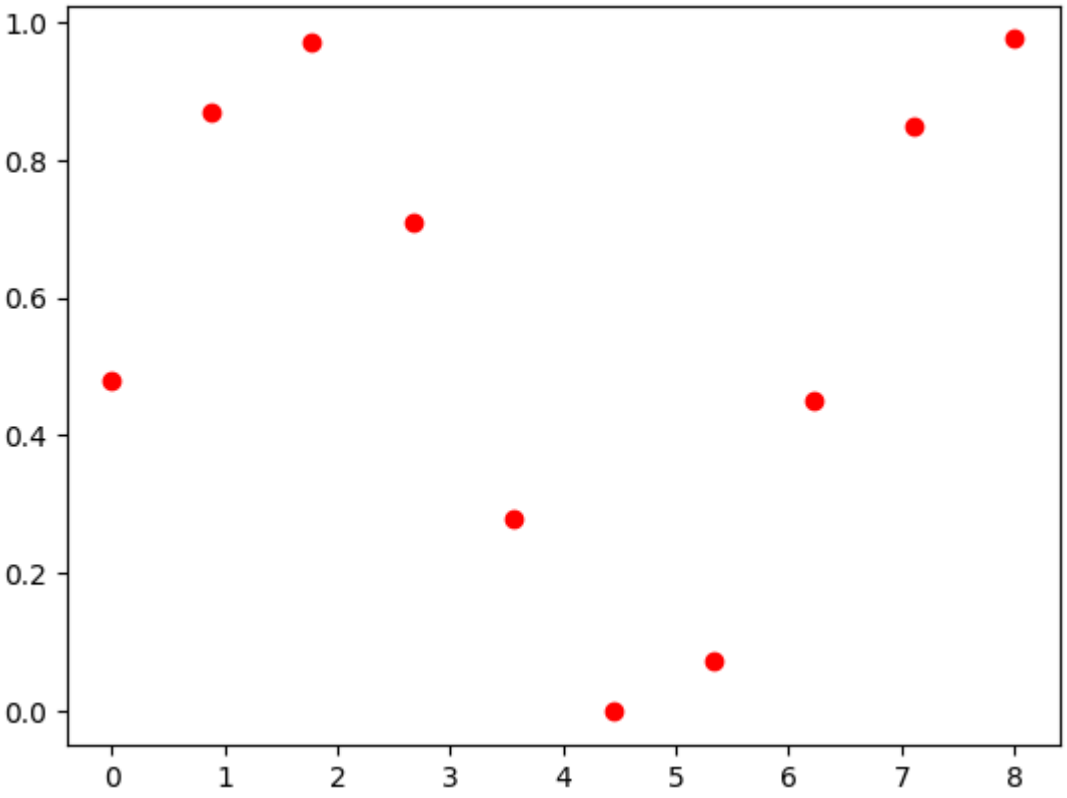
Optimization

In [47]:
```
from scipy.optimize import curve_fit

x = np.linspace(0,8,10)
y= np.array([0.48077, 0.86895, 0.97010, 0.70940, 0.27965, -0.00138, 0.07410, 0.4
fx = lambda x,a,b,c:a*np.sin(b*x) + c


fig, axs = plt.subplots()
axs.plot(x, y, 'or')
plt.show
```

Out[47]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [ ]:
```
popt, pcov = curve_fit(fx, x, y)
popt
```

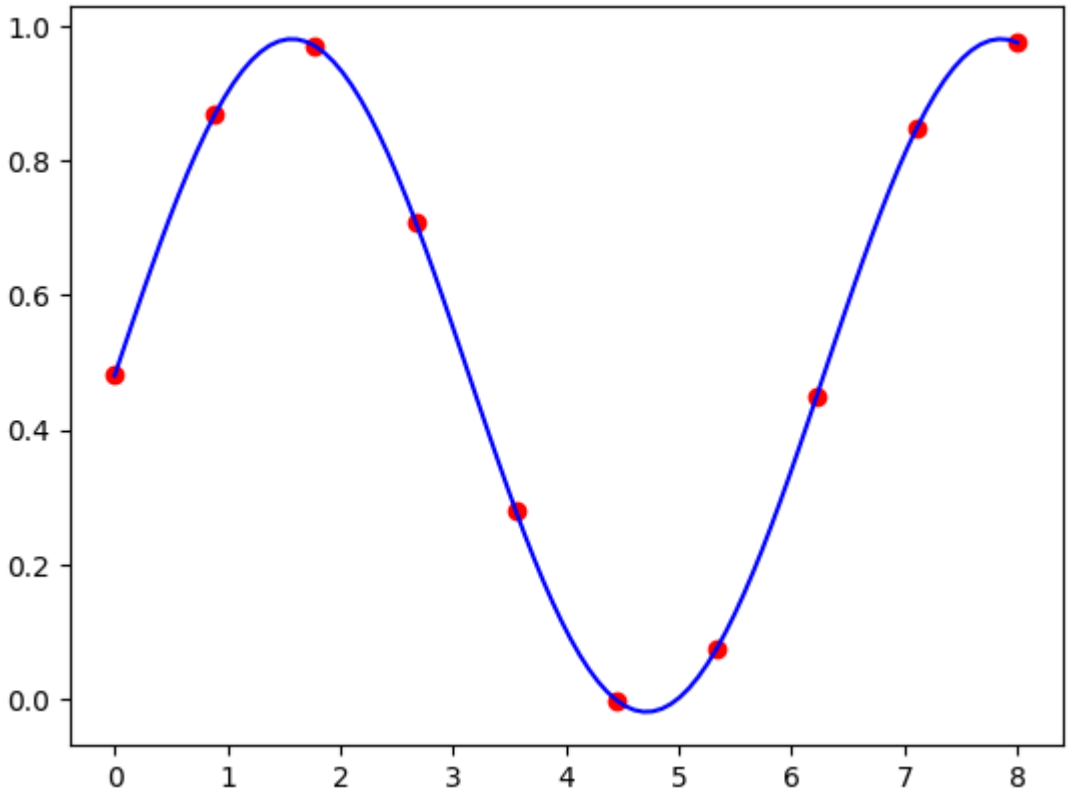
Out[ ]:
```
array([[ 5.31652724e-12,  1.86370054e-14, -9.08331202e-13],
       [ 1.86370054e-14,  1.16735529e-12, -3.73860945e-13],
       [-9.08331202e-13, -3.73860945e-13,  2.80328450e-12]])
```

In [51]:
```
def fxn(x, *popt):
    return popt[0]*np.sin(popt[1]*x) + popt[2]
```

In [57]:
```
xn = np.linspace(0,8, 100)

fig, axs = plt.subplots()
axs.plot(x, y, 'or')
axs.plot(xn, fxn(xn, *popt), '-b')
plt.show
```

Out[57]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Linear Algebra

In [74]:
```
from scipy.linalg import solve
arr1 = np.array([[1,2,5],[2,-5,1],[2,-3,8]])
arr2 = np.array([9,8,2])
sol = solve(arr1, arr2)
sol
```

Out[74]:
```
array([11.71111111,  2.75555556, -1.64444444])
```

In [78]:
```
from scipy.linalg import det, eig

val, vec = eig(arr1)
print(det(arr1))
print(f'Eigen Value: {val}')
print(f'Eigen Vectors: {vec}')
```

```
-44.99999999999999
Eigen Value: [-5.73596985+0.j  0.88652191+0.j  8.84944794+0.j]
Eigen Vectors: [[ 0.44279391 -0.94449629  0.55704311]
 [-0.86037046 -0.29707312  0.13955505]
 [-0.2523811   0.14026517  0.81867415]]
```

In [79]:
```
from scipy.linalg import svd

u, s, vh = svd(arr1)
print(f'U = {u}')
print(f'S = {s}')
print(f'VH = {vh}')
```

```
U = [[-0.40969745  0.61283657 -0.67570654]
 [-0.29972726 -0.79003578 -0.53479625]
 [-0.86157505 -0.01657699  0.50735947]]
S = [10.17406569  5.73529876  0.77119096]
VH = [[-0.26855558  0.32081241 -0.9082716 ]
 [-0.17442665  0.91112656  0.37339487]
 [-0.94734009 -0.25870405  0.18872988]]
```

In [ ]: