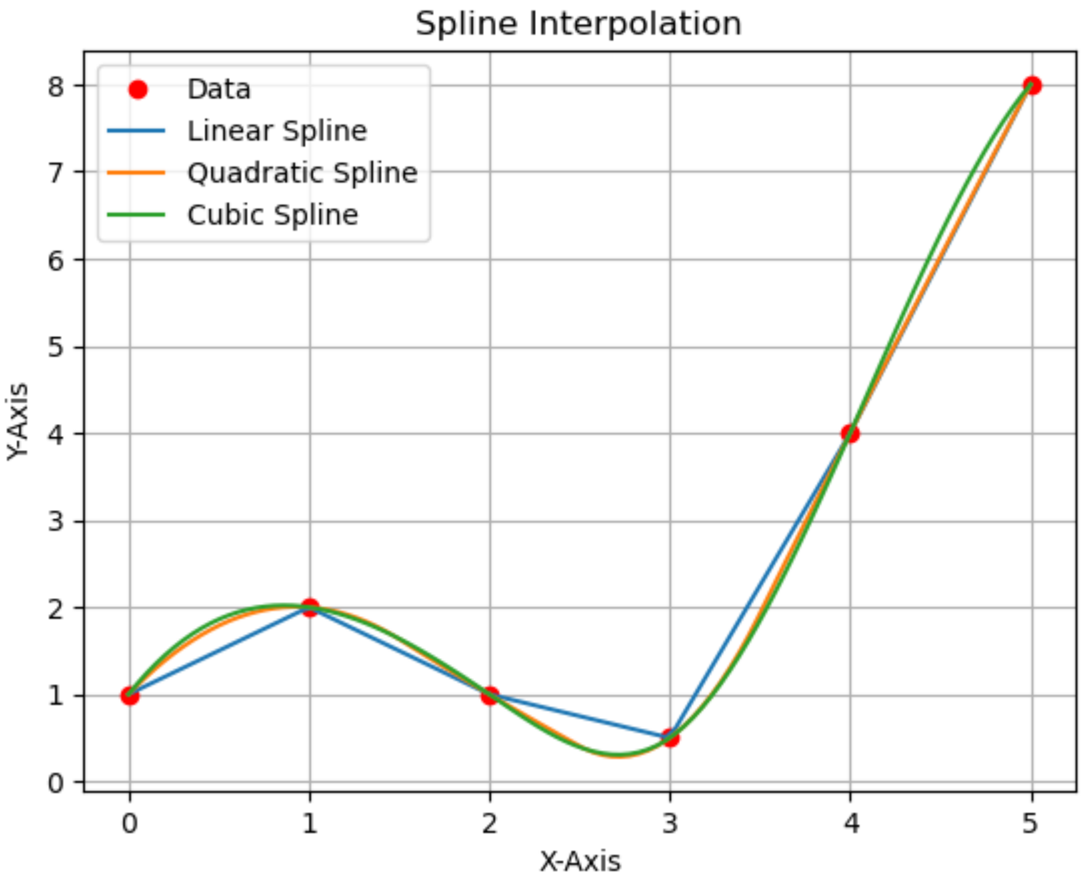
In [2]:
```
import matplotlib.pyplot as plt
from scipy.interpolate import InterpolatedUnivariateSpline as ius
import numpy as np
```

In [3]:
```
x=np.array([0,1,2,3,4,5])
y=np.array([1.0,2.0,1.0,0.5,4.0,8.0])
```

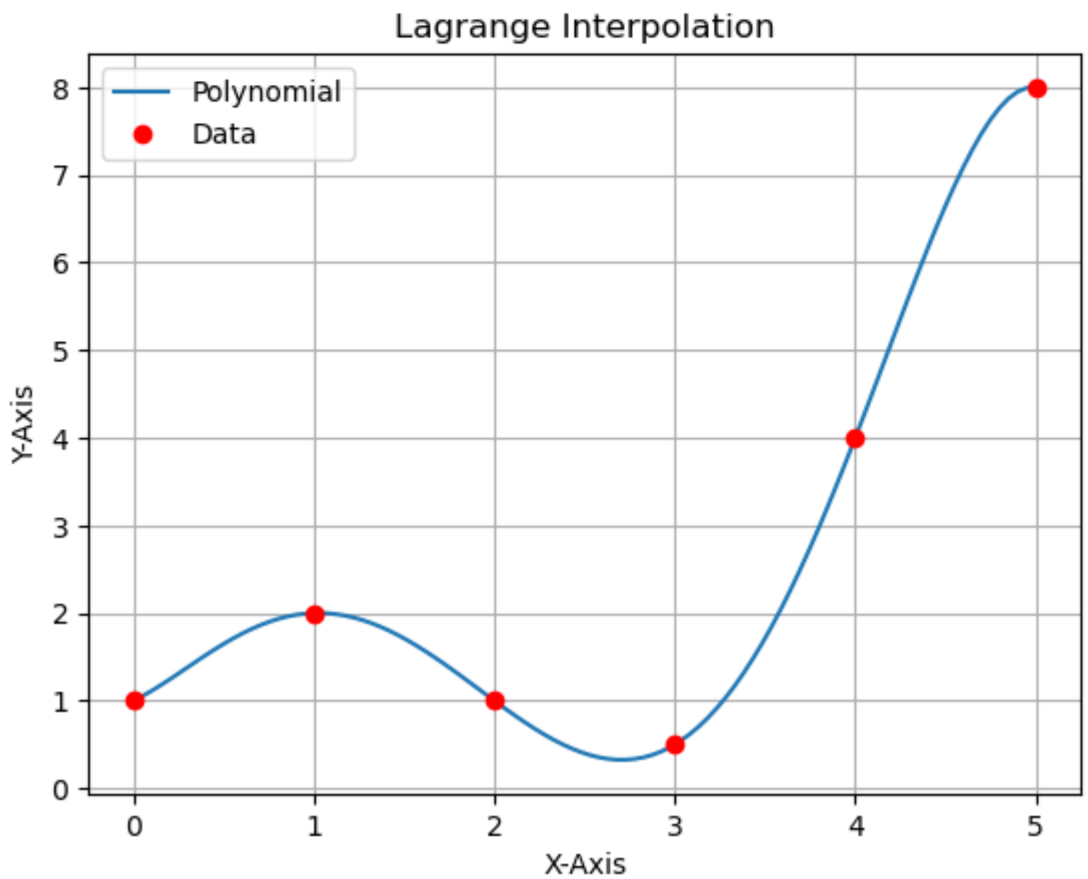
In [10]:
```
#Spline Interpolation
spl_lin=ius(x,y,k=1)
spl_quad=ius(x,y,k=2)
spl_cubic=ius(x,y,k=3)
x_range=np.linspace(0,5,100)
plt.plot(x,y,"or")
plt.plot(x_range,spl_lin(x_range))
plt.plot(x_range,spl_quad(x_range))
plt.plot(x_range,spl_cubic(x_range))
plt.grid()
plt.title("Spline Interpolation")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.legend(["Data","Linear Spline","Quadratic Spline","Cubic Spline"]
```

Out[10]:
```
<matplotlib.legend.Legend at 0x7fd2c0025990>
```



In [39]:
```
from scipy.interpolate import lagrange
from numpy.polynomial.polynomial import Polynomial
poly=lagrange(x,y)
plt.plot(x_range,Polynomial(poly.coef[::-1])(x_range))
plt.plot(x,y,"or")
plt.grid()
plt.title("Lagrange Interpolation")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.legend(["Polynomial","Data"])
Polynomial(poly.coef[::-1])
```

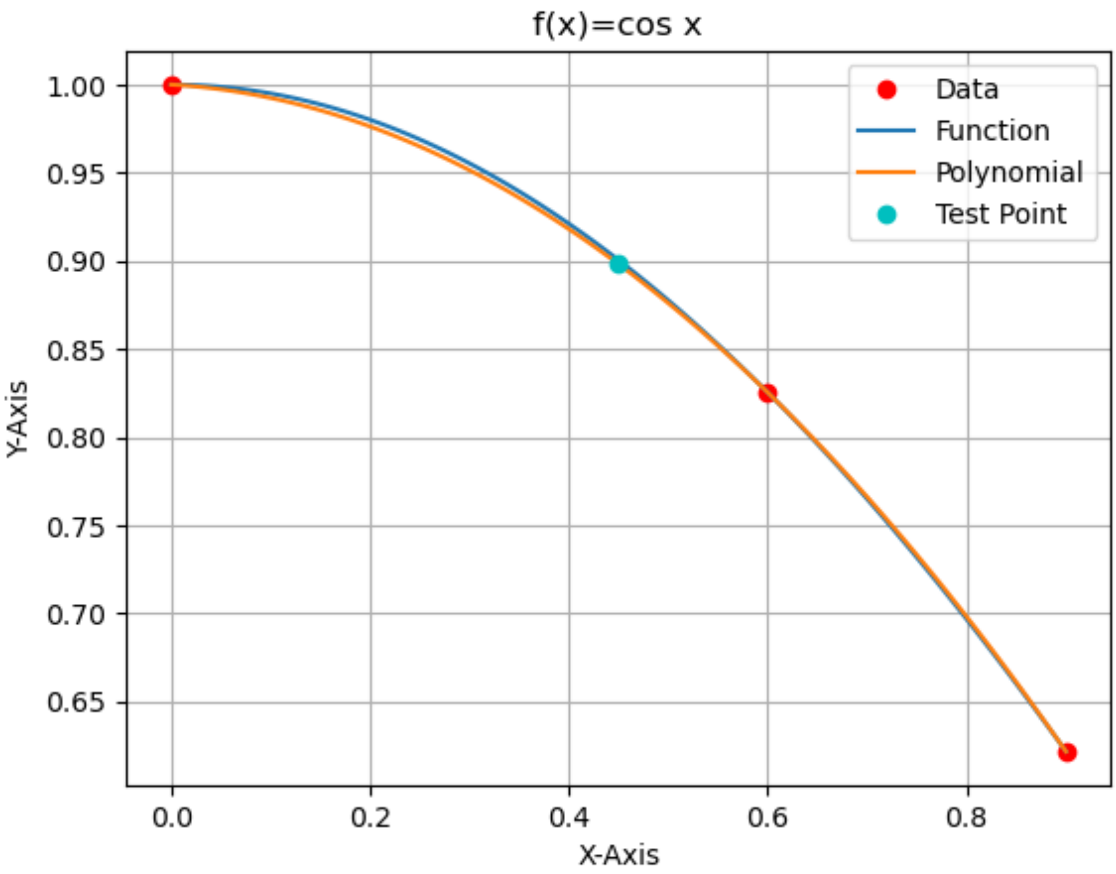
Out[39]: $x \mapsto 1.0 + 0.98333333\,x + 1.54166667\,x^2 - 2.16666667\,x^3 + 0.70833333\,x^4 - 0.06$



In [48]:
```
x=np.array([0,0.6,0.9])
def f1(x):
    return np.cos(x)
def f2(x):
    return np.sqrt(1+x)
def f3(x):
    return np.log(1+x)
def f4(x):
    return np.tan(x)
```

In [58]:
```
#For f1(x)=cos(x)
x_new=np.linspace(0,0.9,100)
y=f1(x)
poly=lagrange(x,y)
res=Polynomial(poly.coef[::-1])(0.45)
err=res-f1(0.45)
plt.plot(x,y,"or")
plt.plot(x_new,f1(x_new))
plt.plot(x_new,Polynomial(poly.coef[::-1])(x_new))
plt.plot(0.45,res,"oc")
plt.grid()
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("f(x)=cos x")
plt.legend(["Data","Function","Polynomial","Test Point"])
print("Result=",res)
print("Error=",err)
```

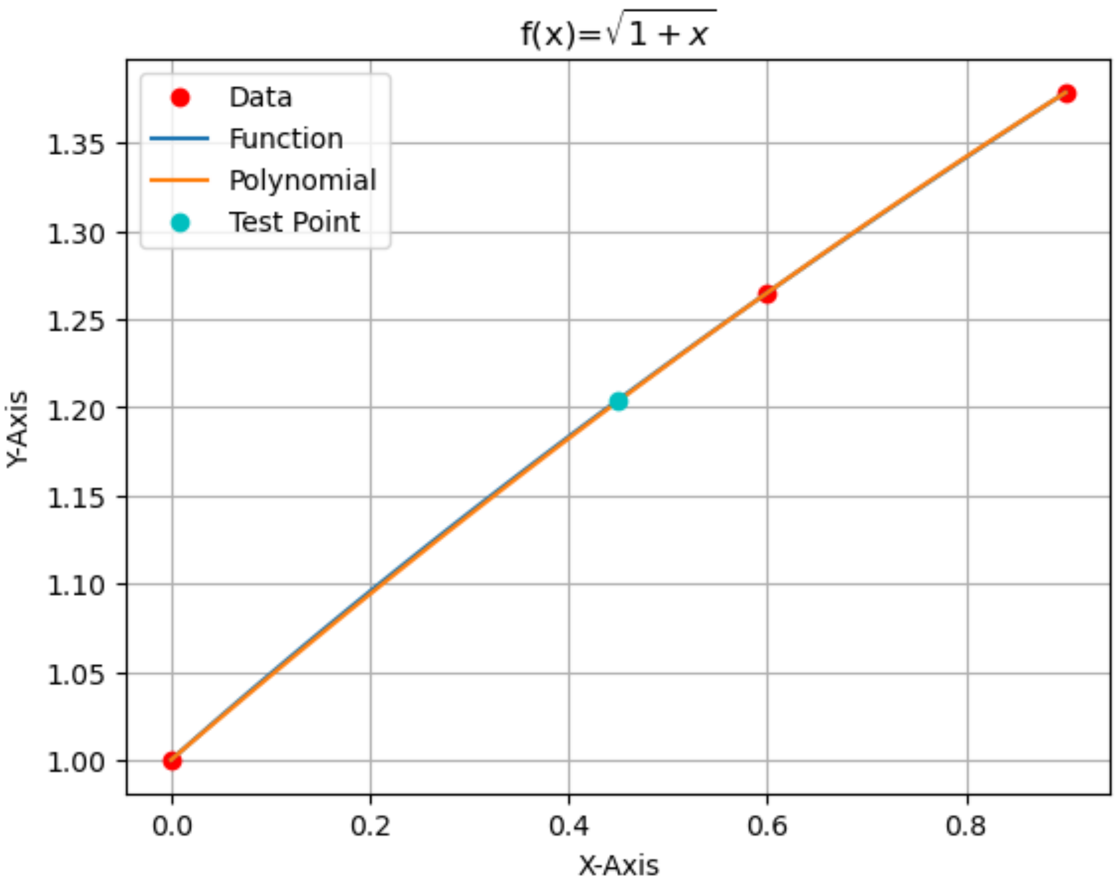
```
Result= 0.898100074705722
Error= -0.0023470276469549356
```



In [61]:
```
#For f2(x)=sqrt(1+x)
y=f2(x)
poly=lagrange(x,y)
res=Polynomial(poly.coef[::-1])(0.45)
err=res-f2(0.45)
print("Result=",res)
print("Error=",err)
plt.plot(x,y,"or")
plt.plot(x_new,f2(x_new))
plt.plot(x_new,Polynomial(poly.coef[::-1])(x_new))
plt.plot(0.45,res,"oc")
plt.grid()
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title(r"f(x)=$\sqrt{1+x}$")
plt.legend(["Data","Function","Polynomial","Test Point"])
```

```
Result= 1.2034237282735154
Error= -0.0007357296057142193
```

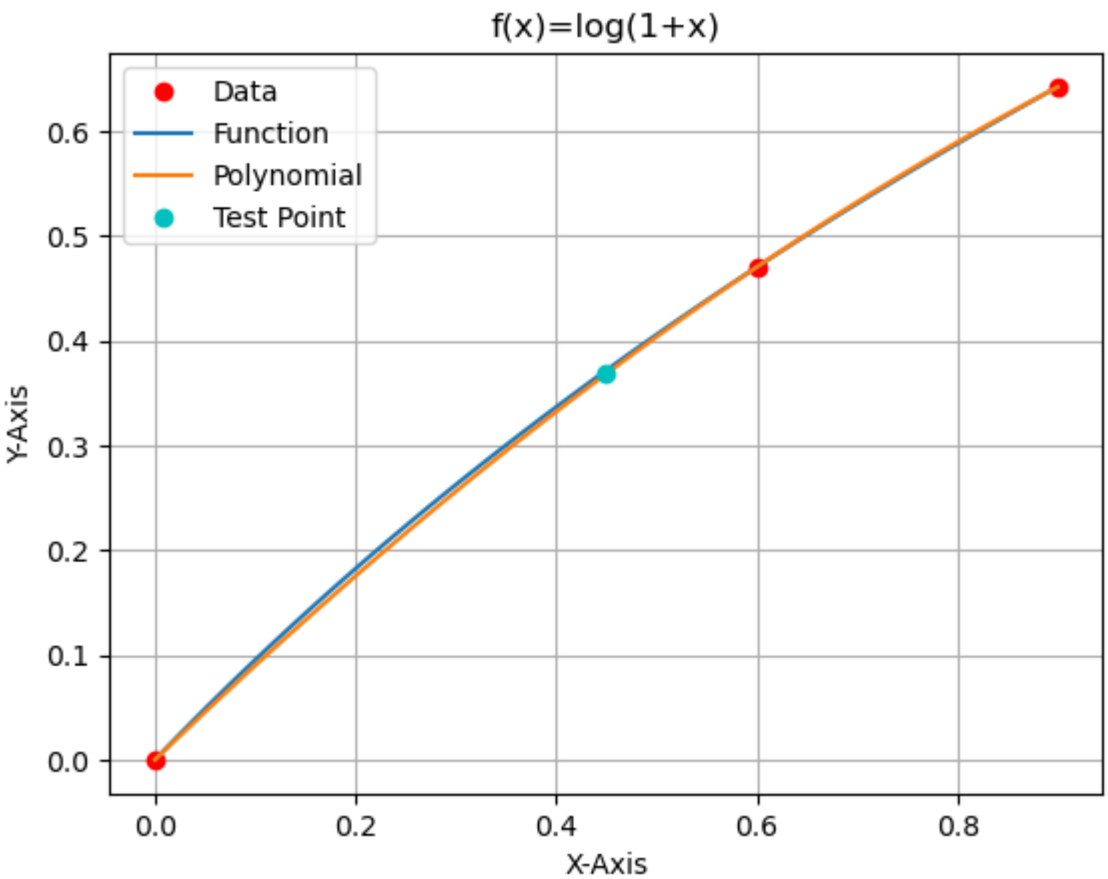
Out[61]: <matplotlib.legend.Legend at 0x7fd2be6eef50>



In [62]:
```
#For f3(x)=log(1+x)
y=f3(x)
poly=lagrange(x,y)
res=Polynomial(poly.coef[::-1])(0.45)
err=res-f3(0.45)
print("Result=",res)
print("Error=",err)
plt.plot(x,y,"or")
plt.plot(x_new,f3(x_new))
plt.plot(x_new,Polynomial(poly.coef[::-1])(x_new))
plt.plot(0.45,res,"oc")
plt.grid()
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("f(x)=log(1+x)")
plt.legend(["Data","Function","Polynomial","Test Point"])
```

```
Result= 0.3682906113583539
Error= -0.003272945074129119
```

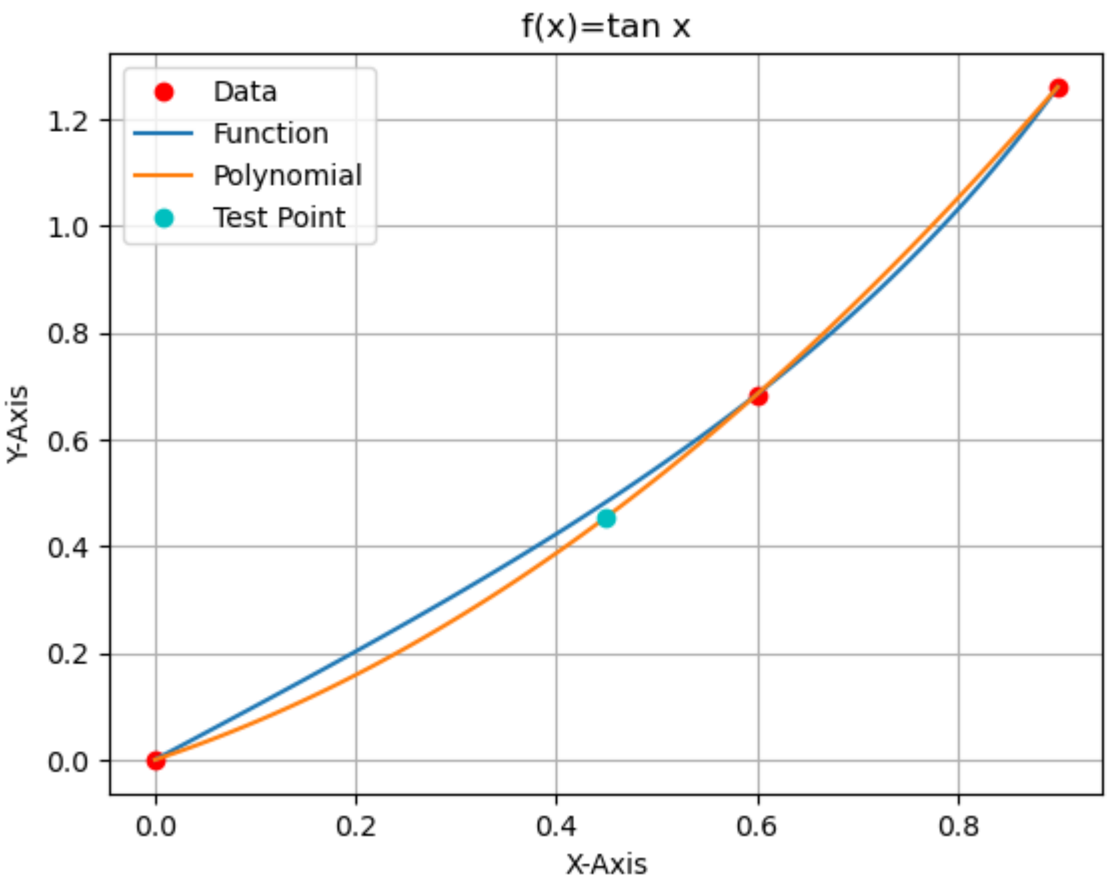
Out[62]:
```
<matplotlib.legend.Legend at 0x7fd2bb8f97d0>
```



In [64]:
```
#For f4(x)=tan x
y=f4(x)
poly=lagrange(x,y)
res=Polynomial(poly.coef[::-1])(0.45)
err=res-f4(0.45)
print("Result=",res)
print("Error=",err)
plt.plot(x,y,"or")
plt.plot(x_new,f4(x_new))
plt.plot(x_new,Polynomial(poly.coef[::-1])(x_new))
plt.plot(0.45,res,"oc")
plt.grid()
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("f(x)=tan x")
plt.legend(["Data","Function","Polynomial","Test Point"])
```

```
Result= 0.4546143549968192
Error= -0.028440710619759224
```

Out[64]: <matplotlib.legend.Legend at 0x7fd2bb53d6d0>



In [ ]: