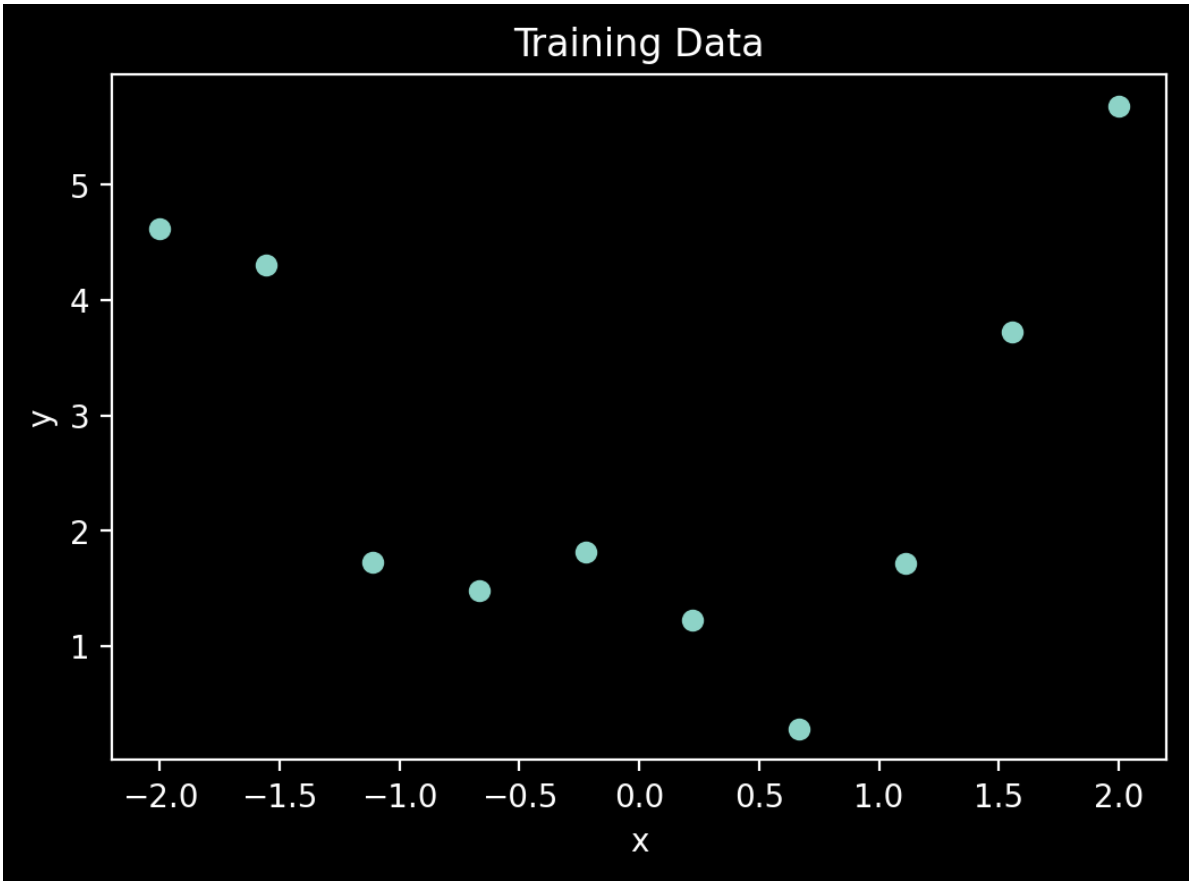
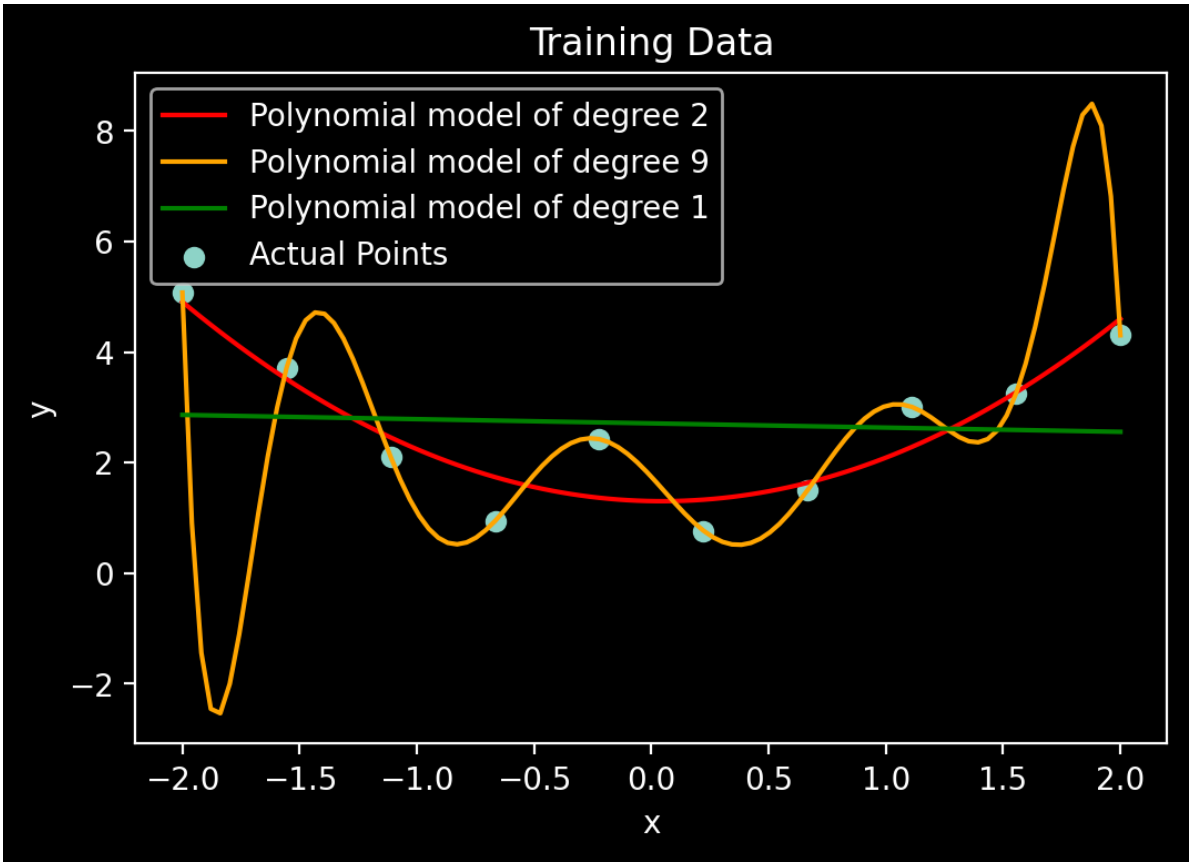
In [2]:

```
import numpy as np # numpy does numbers
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.style.use('dark_background')

# this is the true data-generating function
# n=2 in this case
def f(x, noise):
  return(x**2 + 1 + noise)

# somebody made the first plot with the above polynomial
x_train = np.linspace(-2, 2, 10)
noise_train = np.random.normal(0, 0.5, size = 10)
y_train = f(x_train, noise_train)

# plot the training data
plt.scatter(x_train, y_train)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Training Data')
fig = plt.gcf()
fig.set_dpi(200)
# fig.set_size_inches(4,3, forward=True)
plt.show()
```



In [8]:

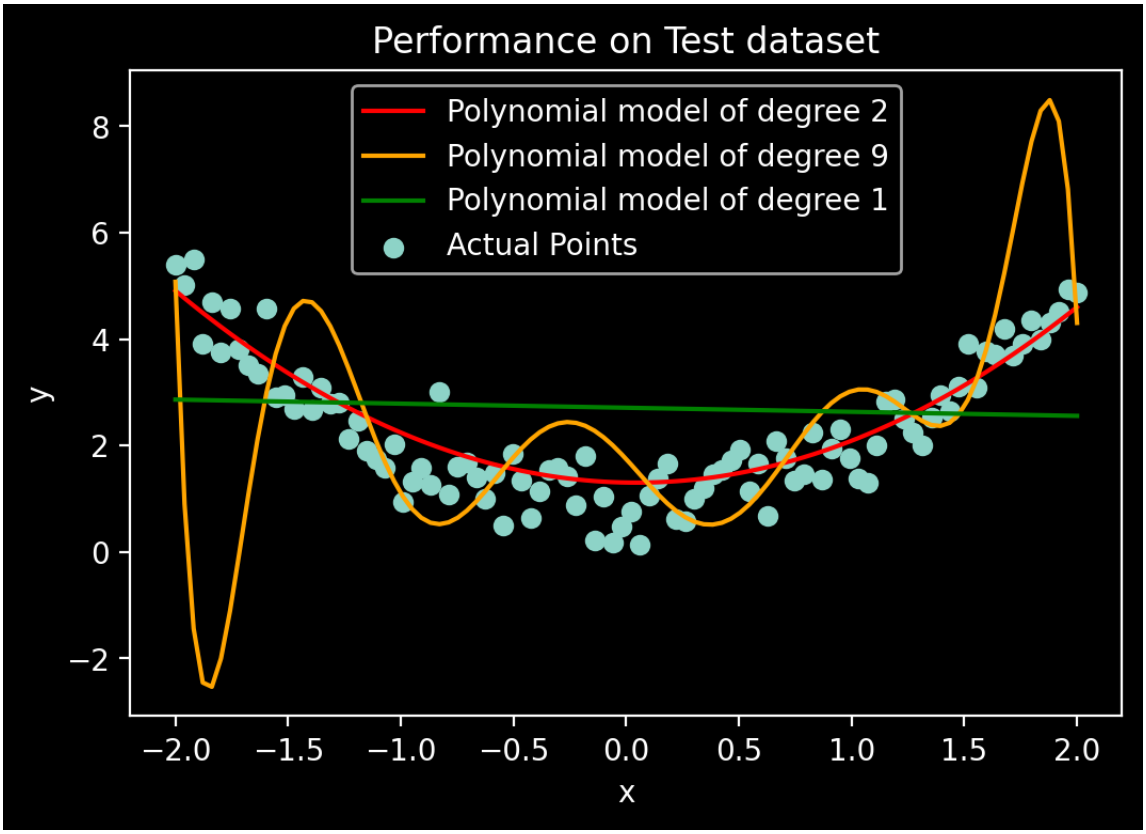
```
# our two polynomial models
model1 = np.poly1d(np.polyfit(x_train, y_train, deg = 2))
model2 = np.poly1d(np.polyfit(x_train, y_train, deg = 9))
model3 = np.poly1d(np.polyfit(x_train, y_train, deg = 1))

plot_x = np.linspace(-2, 2, 100) # so we can plot a curve
plt.scatter(x_train, y_train, label='Actual Points')
plt.plot(plot_x, model1(plot_x), label = 'Polynomial model of degree 2', color = 'red')
plt.plot(plot_x, model2(plot_x), label = 'Polynomial model of degree 9', color = 'orange')
plt.plot(plot_x, model3(plot_x), label = 'Polynomial model of degree 1', color = 'green')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Training Data')
plt.legend()
fig = plt.gcf()
fig.set_dpi(200)
# fig.set_size_inches(4,3, forward = True)
plt.show()
```



In [11]:

```
# Inputs from Test dataset----
x_test = np.linspace(-2, 2, 100) # final exam questions
noise_test = np.random.normal(0, 0.5, size = 100)

# answers to the exam, according to the two models
y1cap = model1(x_test)
y2cap = model2(x_test)
y3cap = model3(x_test)
y_test = f(x_test, noise_test) # final exam answer key

# the penalty from $100
y1_error = np.sum(np.abs(y_test - y1cap))
y2_error = np.sum(np.abs(y_test - y2cap))
y3_error = np.sum(np.abs(y_test - y3cap))

print("Error on Test dataset----")
print("Polynomial model of degree 2:", y1_error)
print("Polynomial model of degree 9:", y2_error)
print("Polynomial model of degree 1:", y3_error)

plt.scatter(x_test, y_test, label = 'Actual Points')
plt.plot(x_test, y1cap, label = 'Polynomial model of degree 2', color = 'red')
plt.plot(x_test, y2cap, label = 'Polynomial model of degree 9', color = 'orange')
plt.plot(x_test, y3cap, label = 'Polynomial model of degree 1', color = 'green')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Performance on Test dataset')
plt.legend()
fig = plt.gcf()
fig.set_dpi(200)
# fig.set_size_inches(4,3, forward=True)
plt.show()
```

```
Error on Test dataset----
Polynomial model of degree 2: 42.81307557993016
Polynomial model of degree 9: 133.88421054998767
Polynomial model of degree 1: 118.74673912707814
```



Overfitting

- The 9th degree polynomial model shown in orange color suffers from overfitting.
- Avoid training complex models when Training data is scarce.
- Question: But how do we know the model suffers from overfitting before we test the model, when it is too late?
- Answer: Cross validation

Undefitting

- The Polynomial model of degree 1 shown in green color suffers from Underfitting
- The capacity of the model is less than the complexity of the problem