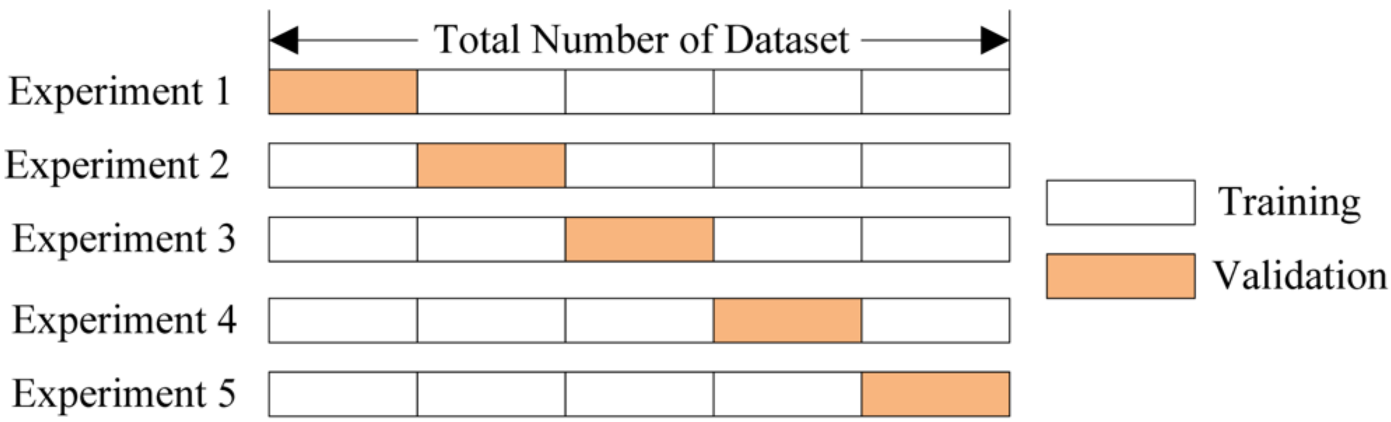
Cross validation

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.



Trying cross validation

Imports

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from fit_plot import fit_plot
```

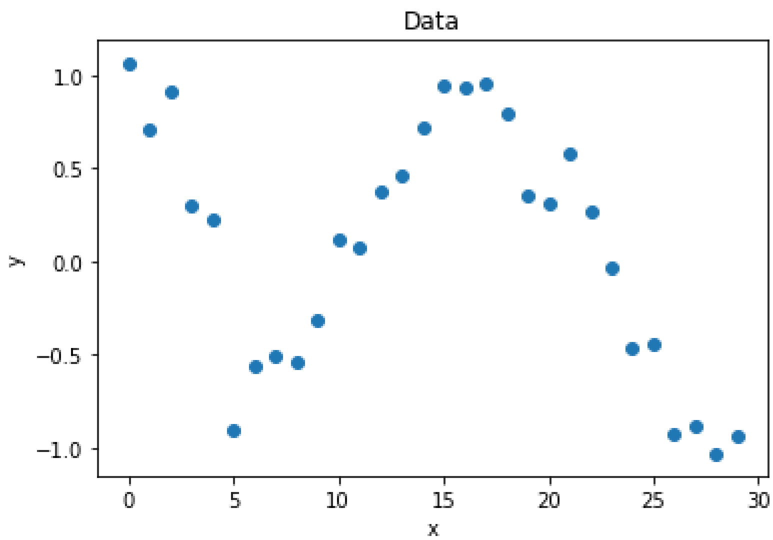
Let's try it to see if it can catch the overfitting problem we saw earlier.

In [2]:
```
def true_fun(X):
    return np.cos(1.5 * np.pi * X)

np.random.seed(0)
n_samples = 30

X = np.sort(np.random.rand(n_samples)*2)
y = true_fun(X) + np.random.randn(n_samples) * 0.1
```

Visualizing the data

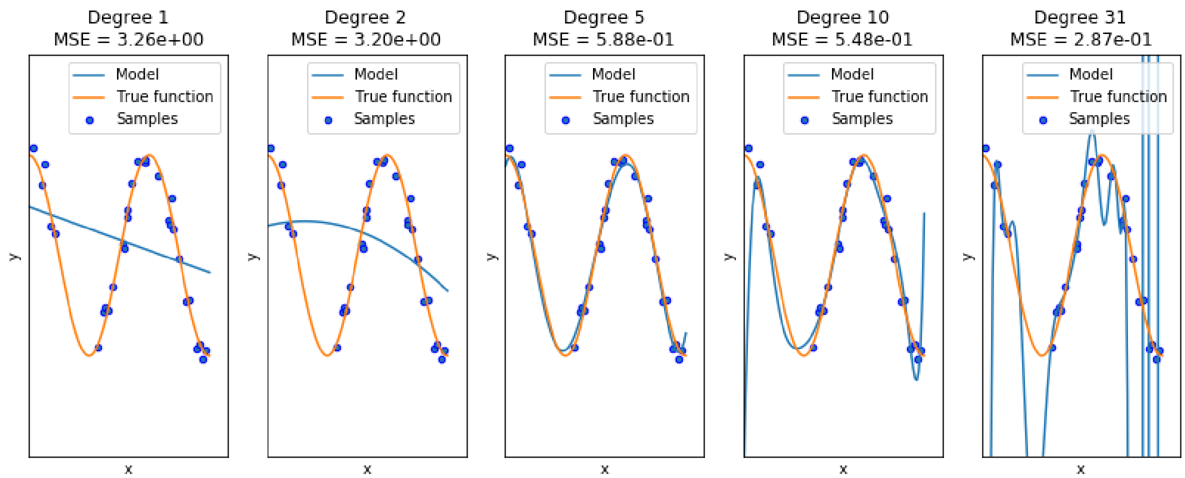
In [3]:
```
plt.title('Data')
plt.scatter(np.arange(n_samples), y)
plt.xlabel('x')
plt.ylabel('y')
```

Out[3]: Text(0,0.5,u'y')



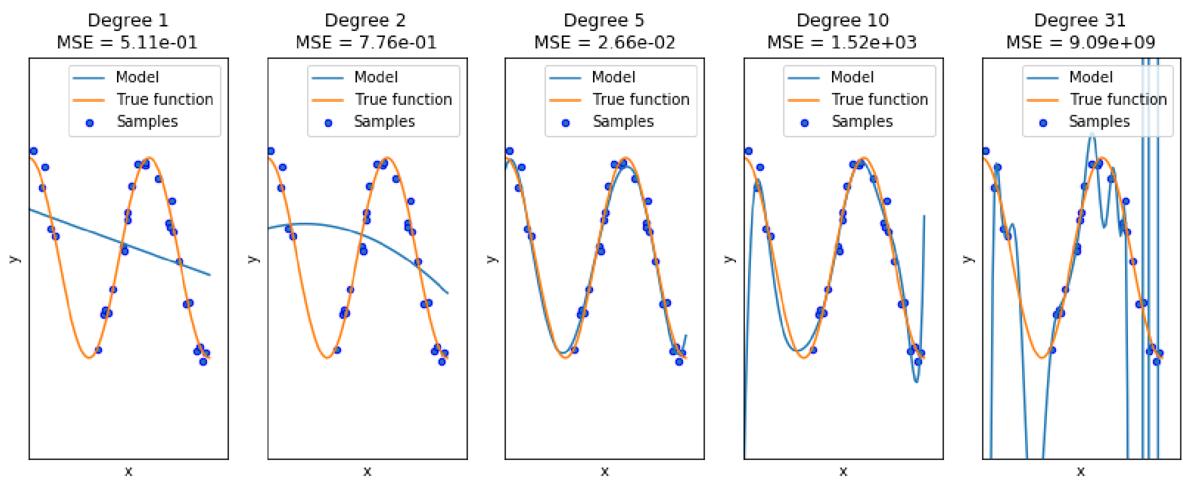
Without cross validation:

```
In [5]: degrees = [1, 2, 5, 10, 31]
        fit_plot(X, y, degrees)
```



Using cross validation as a score:

```
In [6]: degrees = [1, 2, 5, 10, 31]
        fit_plot(X, y, degrees, score='cv')
```



The error used here is the mean of the errors of each fold.

We notice how the error diverges for degree 5 and 31.

According to cross validation, the best model here is for degree 5.