Get All Imports

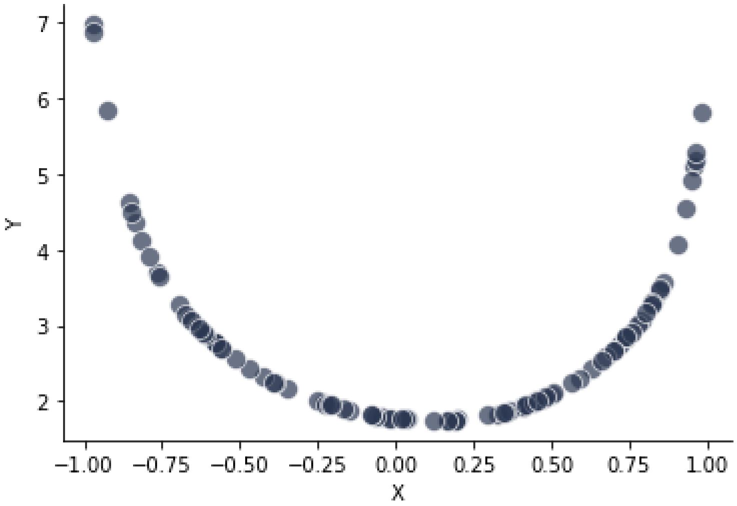
In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Generate Data Points

In [ ]:

```
poly_degree = 15
np.random.seed(27)

X = np.random.uniform(-1, 1, 100)
coeff = np.random.uniform(-2, 2, poly_degree+1)
Y = [sum([coeff[j] * i**j for j in range(1, poly_degree+1)]) + coeff[0] for i in X]

# Plot the data points
sns.scatterplot(x=X, y=Y, marker='o', s=100, color="#2B3751", edgecolors="#E5E5E5", alph
sns.despine()
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



Define the linear regression model using Pseudo Inverse.

Also define the function for calling the model and plotting

In [ ]:

```
class LinearRegression():
    # Constructor of this class
    def __init__(self):
        self.coeff = list()

    # This function is used to find the coefficients for the line of best fit
    def fit(self, A, Y):
        # Add Bias
```

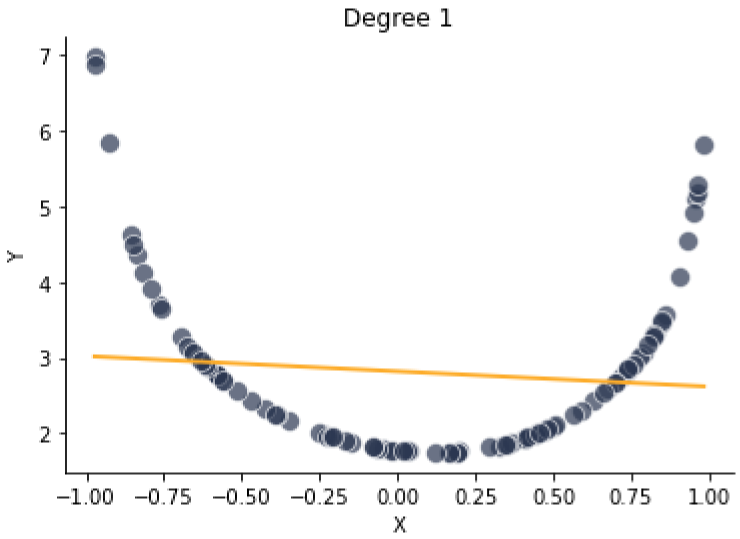
```python
        A = np.concatenate((np.ones((len(A), 1)), A), axis=1)
        # Find Pseudo Inverse
        pseudo_inv = np.matmul(np.linalg.inv(np.matmul(np.transpose(A), A)), np.transpos
        # Finally get the coefficients
        self.coeff = np.matmul(pseudo_inv, np.reshape(Y, (-1, 1)))

    # This function uses the found coefficients to deliver predictions
    def predict(self, A):
        A = np.concatenate((np.ones((len(A), 1)), A), axis=1)
        return np.squeeze(np.matmul(A, self.coeff))


# This function converts a given input to required number of degrees.
# Then uses the best fit line and plots the predictions
def fit_curve(X, Y, degree):
    if degree == 1:
        X_new = np.reshape(X, (-1, 1))
    else:
        X_new = np.transpose(np.asarray([X**i for i in range(1, degree+1)]))

    model = LinearRegression()
    model.fit(X_new, Y)
    preds = model.predict(X_new)
    print("RMSE:", np.sqrt(np.sum((preds-Y)**2)))

    # Plot the predicted points
    sns.scatterplot(x=X, y=Y, marker='o', s=100, color="#2B3751", edgecolors="#E5E5E5",
    sns.lineplot(x=X, y=preds, color="#FDAC29", linewidth=2)
    sns.despine()
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.title("Degree "+str(degree))
    plt.show()
```

Fit curves with varying degrees

In [ ]:

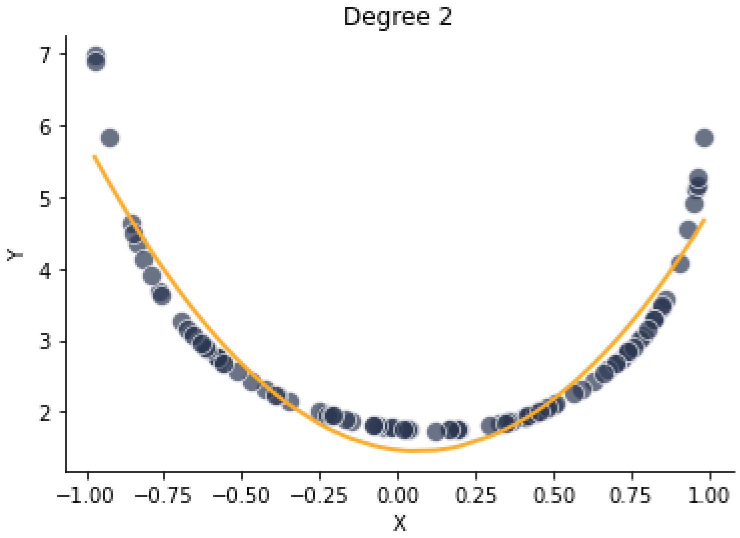
```python
fit_curve(X, Y, 1)
print("\n\n")
fit_curve(X, Y, 2)
print("\n\n")
fit_curve(X, Y, 4)
print("\n\n")
fit_curve(X, Y, 7)
print("\n\n")
fit_curve(X, Y, 11)
print("\n\n")
fit_curve(X, Y, 15)
```

```
RMSE: 11.443395392758626
```



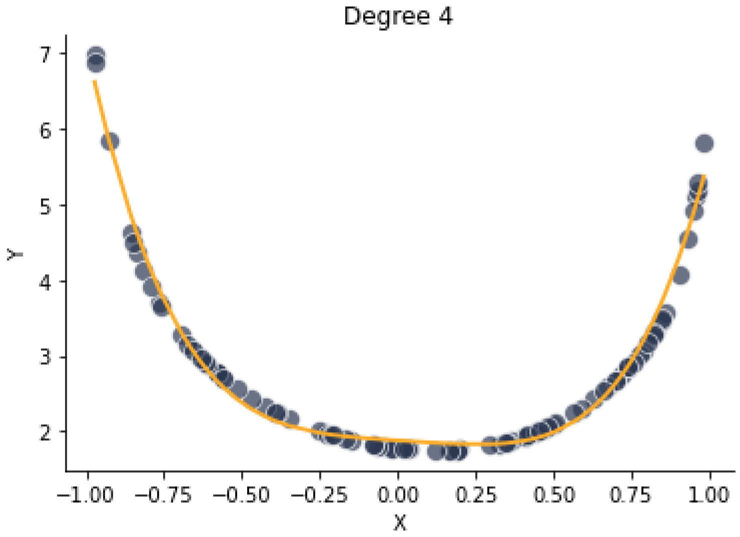
Degree 1

```
RMSE: 3.6070947302665872
```



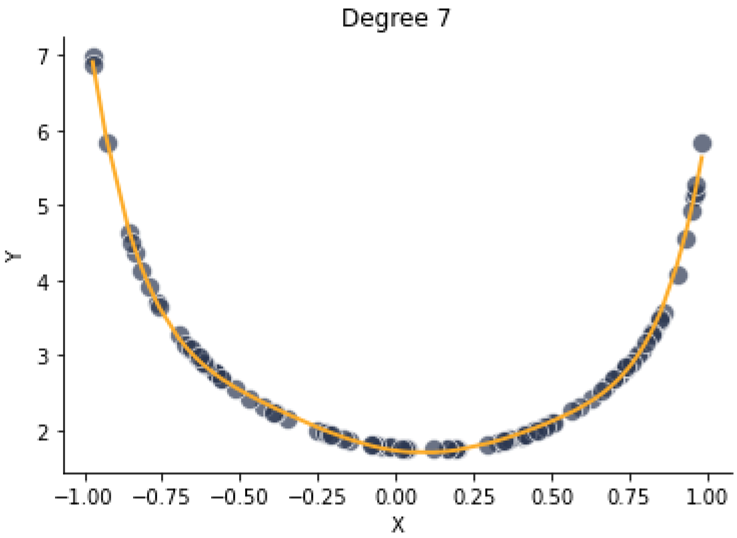
Degree 2

```
RMSE: 1.306961108467238
```



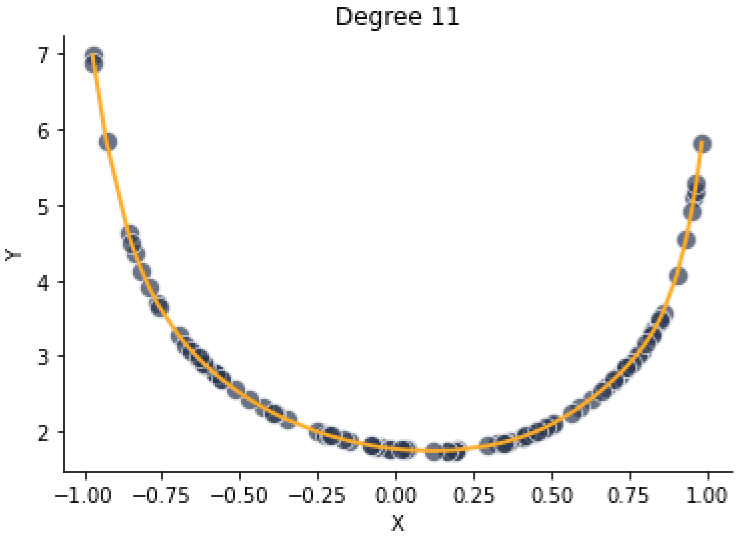
Degree 4

```
RMSE: 0.4669204777459218
```



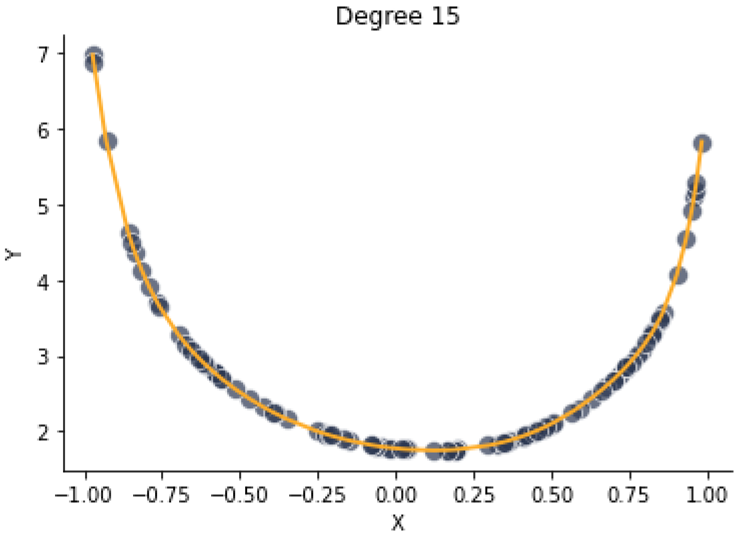
Degree 7

```
RMSE: 0.013350990802565669
```



Degree 11

```
RMSE: 1.2720922305679981e-06
```



Degree 15

```
In [ ]:
```