Naif (Saf) Tahmin (Naive Forecasting)

Bu başlık altında basit bir tahmini ele alalım. Bunu yaparken bir önceki colab dosyasındaki serileri görselleştirmek için kullandığımız, tend, mevsimsellik ve gürültü işlevlerini kullanacağız. Oluşturduğumuz zaman serisinin bir kısmını eğitim ve bir kısmını doğrulama periyodu olarak ayıracağız. Gerekli paketleri ve önceki colab dosyasında oluşturduğumuz fonksiyonları tekrar tanımlayarak başlayalım.

Gerekli Tanımlamaların ve İçeri Aktarmaların Yapılması ¶

In [1]:

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

In [3]:

```
def plot_series(time, series, format="-", start=0, end=None, label=None):
    plt.plot(time[start:end], series[start:end], format, label=label)
    plt.xlabel("Time")
    plt.ylabel("Value")
    if label:
        plt.legend(fontsize=14)
    plt.grid(True)

def trend(time, slope=0):
    return slope * time

def seasonal_pattern(season_time):
    return np.where(season_time < 0.4,
                    np.cos(season_time * 2 * np.pi),
                    1 / np.exp(3 * season_time))

def seasonality(time, period, amplitude=1, phase=0):
    """Her periyotta aynı kalıbı tekrarlar."""
    season_time = ((time + phase) % period) / period
    return amplitude * seasonal_pattern(season_time)

def white_noise(time, noise_level=1, seed=None):
    rnd = np.random.RandomState(seed)
    return rnd.randn(len(time)) * noise_level
```

Tend, Mevsimsellik ve Beyaz Gürültü

In [4]:

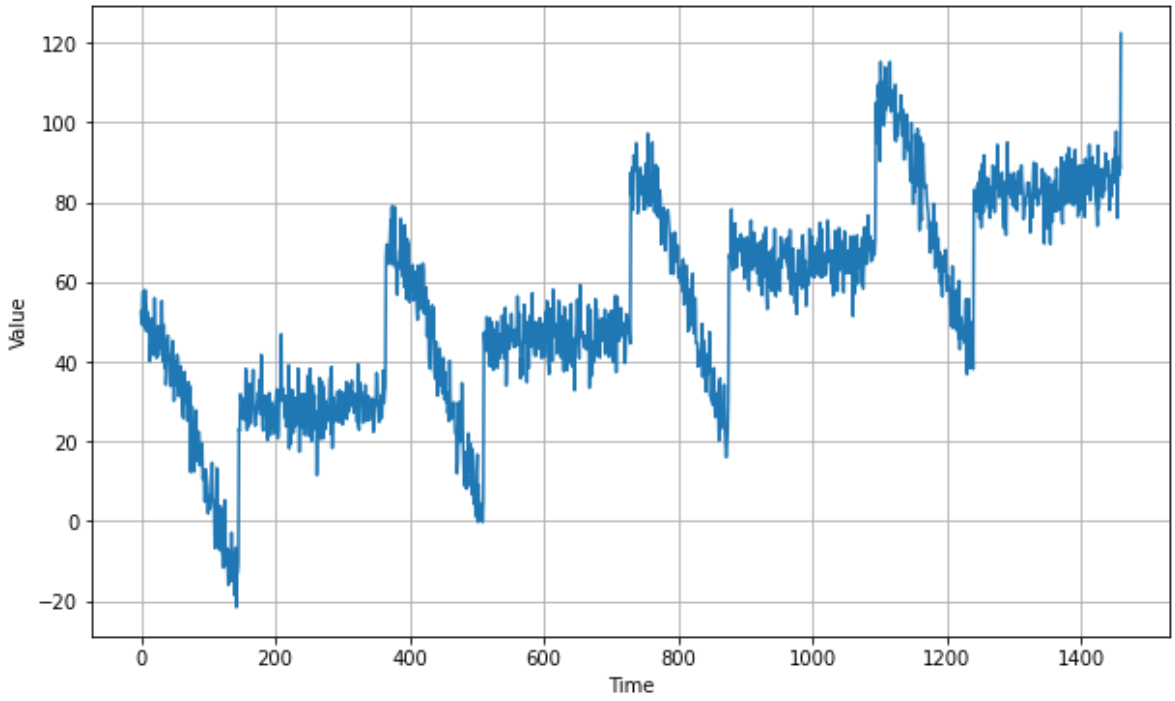
```
time = np.arange(4 * 365 + 1)

slope = 0.05
baseline = 10
amplitude = 40
series = baseline + trend(time, slope) + seasonality(time, period=365, amplitude=amplitude)

noise_level = 5
noise = white_noise(time, noise_level, seed=42)

series += noise

plt.figure(figsize=(10, 6))
plot_series(time, series)
plt.show()
```



Önceki colab dosyasında oluşturduğumuz zaman serisinin aynısı değil mi? Bu hali ile daha gerçekçi duruyor. Bunu tahmin etmeye çalışalım. Bu zaman serisini iki döneme ayıracağız: eğitim dönemi ve doğrulama dönemi. Bölünme zaman adımı olarak 1000 değerini seçiyoruz

In [5]:

```
split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]
```

Naif (Saf) Tahminin Yapılması

Naif tahmin yöntemleri, geçmiş bir dönem için kaydedilen verilere gelecekteki bir döneme ilişkin bir projeksiyona dayanır. Örneğin, naif bir tahmin önceki bir dönemin fiillerine veya belirli önceki dönemlerin gerçeklerinin ortalamasına eşit olabilir.

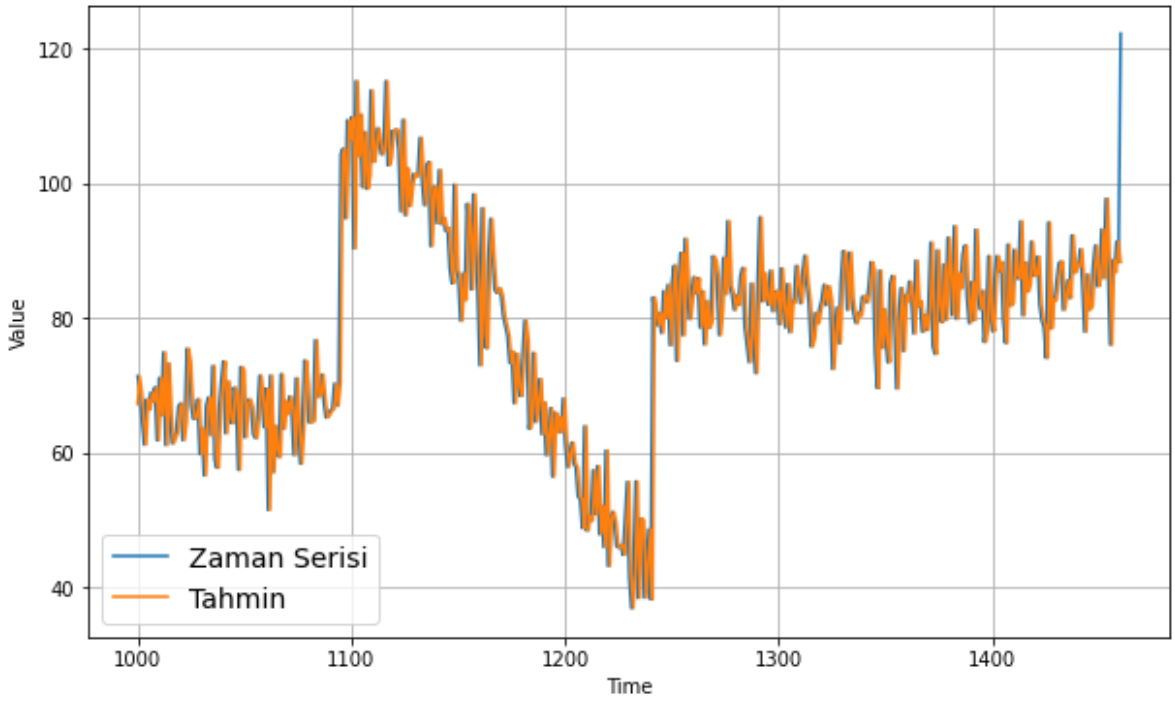
In [6]:

```
naive_forecast = series[split_time - 1:-1]
```

Yukarıdaki kodda yapmak istediğimi aslında değerleri bir birim sola kaydırmaktır. Yani basitçe özetlemek gerekirse bir zaman diliminin tahmini aslında onun bir adım sonraki zaman dilimindeki gerçek değeridir.

In [7]:

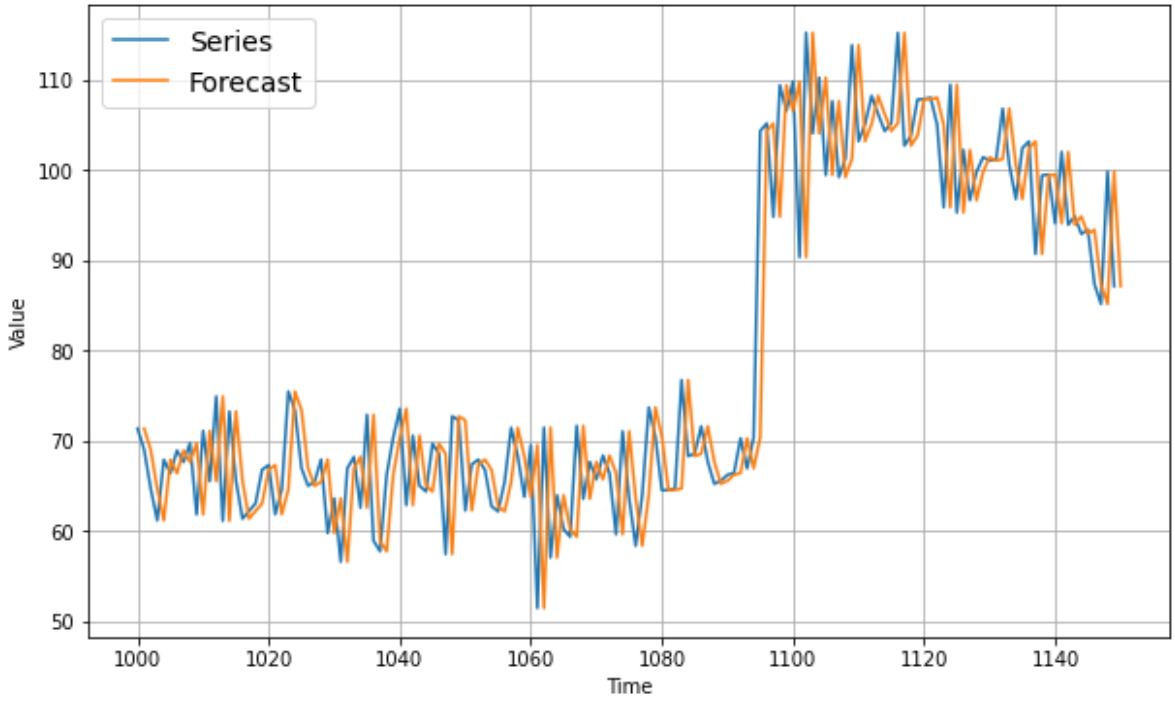
```
plt.figure(figsize=(10, 6))
plot_series(time_valid, x_valid, label="Zaman Serisi")
plot_series(time_valid, naive_forecast, label="Tahmin")
```



Doğrulama periyodunun başlangıcına yakınlaşalım.

In [8]:

```
plt.figure(figsize=(10, 6))
plot_series(time_valid, x_valid, start=0, end=150, label="Series")
plot_series(time_valid, naive_forecast, start=1, end=151, label="Forecast")
```



Naif tahminin zaman serisinin 1 adım gerisinde kaldığını görebilirsiniz.

Şimdi doğrulama periyodundaki tahminleri ve tahminlerin gerçek değer arasındaki ortalama mutlak hatayı (mean_abosolute_error) hesaplayalım.

In [9]:

```
errors = naive_forecast - x_valid
abs_errors = np.abs(errors)
mae = abs_errors.mean()
mae
```

Out[9]:

```
5.9379085153216735
```