Mantıksal Operatörler (Boolean Masking)

🎓 Öğretmene Notlar:

Öğrencilere, "Excel'de filtreleme yapmak gibi" benzetmesini kullanın.

> , <, ==, !=, &, | gibi karşılaştırmalar, NumPy dizilerine uygulandığında sonuç da bir dizi olur.

Bu teknik, veri bilimi ve makine öğrenmesi öncesinde veri temizleme ve filtreleme için çok önemlidir.

📘 Konu Anlatımı:

```
import numpy as np

veri = np.array([10, 15, 20, 25, 30])
print(veri > 20)  # [False False False  True  True]
```

```
#masking Kullanımı

print(veri[veri > 20])  # [25 30]
```

```
# 🧠 Birden çok koşul:

print(veri[(veri > 10) & (veri < 30)])  # [15 20 25]
```

```
# 📌 Gerçek Hayat Bağlantısı:
# Örneğin bir şehirde sıcaklığı 30 dereceyi geçen günleri bulmak:

sicakliklar = np.array([28, 32, 35, 27, 30])
print(sicakliklar[sicakliklar > 30])  # [32 35]
```

Matematiksel İşlemler

🎓 Öğretmene Notlar:

NumPy, toplama, çıkarma, çarpma gibi işlemleri dizinin tüm elemanlarına vektörel olarak uygular.

Python listelerine göre çok daha hızlı çalışır.

📘 Örnekler:

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(a + b)  # [5 7 9]
```

```
print(a * b)  # [4 10 18]
print(np.mean(a))  # Ortalama
print(np.std(a))   # Standart sapma

# 🧠 İşlemler tüm elemana uygulanır:

print(a + 10)  # [11 12 13]
```

📌 Gerçek Hayat Bağlantısı: Öğrencilerin tüm sınav notlarını 5 puan artırmak:

```
notlar = np.array([60, 70, 80])
print(notlar + 5)  # [65 75 85]
```

```
# Proje 1: Günlük Sıcaklık Takibi ve Ortalama Hesaplama
# Senaryo: 1 haftalık sıcaklık verilerini kaydedin, ortalama sıcaklığı bulun

gunduz_sicaklik = np.array([28, 31, 33, 30, 27, 29, 32])
print("Ortalama:", np.mean(gunduz_sicaklik))
print("Sıcak günler:", gunduz_sicaklik[gunduz_sicaklik > 30])
```

```
# Proje 2: Depo Stok Takibi
# Senaryo: Her üründen kaç adet kaldığını dizi olarak tutun. 50'nin altındak

stoklar = np.array([75, 45, 60, 30])
kritik = stoklar[stoklar < 50]
print("Düşük stoklar:", kritik)
```

```
# Proje 3: Öğrenci Not Analizi
# Senaryo: Sınıftaki öğrencilerin notlarını listeleyin. 50'nin altında kalan

notlar = np.array([70, 55, 40, 65, 80, 45])
print("Ortalama Not:", np.mean(notlar))
print("Kalanlar:", notlar[notlar < 50])
```

🍼 Proje Adı: "Bir Okuldaki 12 Aylık Sınav Performans Analizi"

🎯 **Senaryo:** Bir okulda 4 farklı sınıfın her ay yapılan sınav sonuçları 1 yıl boyunca kaydediliyor. Her sınıf 20 öğrenci içeriyor. Amaç, bu veriler üzerinde analizler, filtrelemeler ve istatistiksel işlemler gerçekleştirmek.

🧱 1. VERİYİ OLUŞTURMA (Array + Random + Reshape)

```
import numpy as np

np.random.seed(42)  # Rastgeleliği sabitlemek için
veri = np.random.randint(30, 100, size=(12, 4, 20))  # 12 ay, 4 sınıf, 20 öğ
```

```
#📌 veri dizisi şu şekilde: (ay, sınıf, öğrenci)
```

🔍 2. GENEL BİLGİLER (shape, size, dtype, ndim)

```
print("Veri Boyutu:", veri.shape)    # (12, 4, 20)
print("Toplam Gözlem:", veri.size)
print("Veri Tipi:", veri.dtype)
print("Boyut Sayısı:", veri.ndim)
```

🗓️ 3. 6. Ayın Tüm Notları (Slicing)

```
haziran = veri[5]
print("Haziran Ayı Notları:\n", haziran)
```

📈 4. Ortalama ve Standart Sapma (mean, std)

```
print("Yıllık Ortalama:", np.mean(veri))
print("Yıllık Std Sapma:", np.std(veri))
```

🎯 **5. Belirli Koşula Göre Filtreleme (Mantıksal Operatörler)** 50'nin altında not alan öğrenci sayısını bulun:

```
print("Başarısız öğrenci sayısı:", np.sum(veri < 50))
```

📊 6. Her Sınıfın Aylık Ortalaması (axis kullanımı)

```
aylik_ortalama = np.mean(veri, axis=2)  # Öğrenciler üzerinden ortalama
print("Aylık Sınıf Ortalamaları:\n", aylik_ortalama)
```

🪓 **7. Split ve Concatenate Kullanımı** Veriyi ilk 6 ay ve son 6 ay olarak ayır:

```
ilk_yaril = np.split(veri, 2, axis=0)[0]
ikinci_yaril = np.split(veri, 2, axis=0)[1]

yeniden_birlesik = np.concatenate((ilk_yaril, ikinci_yaril), axis=0)
```

🖌️ 8. Tüm Notları Tek Listeye Dönüştürme (flatten)

```
tum_notlar = veri.flatten()
```

🔢 9. Sıralama ve En Yüksek Notu Bulma

```
print("En yüksek not:", np.max(veri))
print("En düşük not:", np.min(veri))
print("Notların medyanı:", np.median(veri))
```

🎁 10. Ekstra: En başarılı ayı bul (ortalama ile)

```
ortalama_aylar = np.mean(veri, axis=(1, 2))  # her ay için ortalama
en_basarili_ay = np.argmax(ortalama_aylar)
print("En başarılı ay:", en_basarili_ay + 1, ". ay")
```