Numpy Sum Fonksiyonu

`numpy.sum()`, bir NumPy dizisindeki (array) elemanların toplamını hesaplamak için kullanılan temel ve çok yönlü bir fonksiyondur. Sadece tüm elemanları toplamakla kalmaz, aynı zamanda belirli eksenler (axis) boyunca toplama, veri tipini kontrol etme gibi işlemleri de yapabilir.

Temel Kullanım:

En basit haliyle, fonksiyona bir dizi verdiğinizde, içindeki tüm sayısal elemanları toplar ve tek bir skaler (sayı) değer döndürür.

Python

```
import numpy as np

# Tek boyutlu dizi (vektör)
dizi1d = np.array([1, 2, 3, 4, 5])
toplam1d = np.sum(dizi1d)
print(f"Tek boyutlu dizi: {dizi1d}")
print(f"Tüm elemanların toplamı: {toplam1d}") # Çıktı: 15

# İki boyutlu dizi (matris)
dizi2d = np.array([[1, 2, 3],
                   [4, 5, 6]])
toplam2d = np.sum(dizi2d)
print(f"\nİki boyutlu dizi:\n{dizi2d}")
print(f"Tüm elemanların toplamı: {toplam2d}") # Çıktı: 21
```

Parametreler ve Detaylı Kullanım:

`numpy.sum()` fonksiyonu çeşitli parametreler alarak daha spesifik toplamlar hesaplamanıza olanak tanır:

1. `a`: Toplama işleminin yapılacağı NumPy dizisi (zorunlu parametre).
2. `axis`: Toplama işleminin yapılacağı eksen(ler)i belirtir. Bu, çok boyutlu dizilerde en önemli parametrelerden biridir.Python
   - `axis=None` **(Varsayılan):** Dizideki *tüm* elemanları toplar ve tek bir skaler değer döndürür (yukarıdaki temel kullanım örneğindeki gibi).
   - `axis=0`: Dizinin *ilk* ekseni boyunca toplama yapar. 2 boyutlu bir matris için bu, **sütunlar boyunca aşağı doğru** toplama anlamına gelir. Sonuç, her sütunun toplamını içeren yeni bir dizi olur.
   - `axis=1`: Dizinin *ikinci* ekseni boyunca toplama yapar. 2 boyutlu bir matris için bu, **satırlar boyunca yana doğru** toplama anlamına gelir. Sonuç, her

satırın toplamını içeren yeni bir dizi olur.

- Daha yüksek boyutlu dizilerde `axis=2`, `axis=3` vb. şeklinde devam eder.
- Eksenleri bir tuple içinde belirterek birden fazla eksen boyunca da toplama yapabilirsiniz (örn: `axis=(0, 1)`).

```
dizi2d = np.array([[1, 2, 3],
                   [4, 5, 6]])
print(f"\nİki boyutlu dizi:\n{dizi2d}")

# Sütunların toplamı (axis=0) → [1+4, 2+5, 3+6]
sutun_toplamlari = np.sum(dizi2d, axis=0)
print(f"Sütun toplamları (axis=0): {sutun_toplamlari}") # Çıktı: [5 7 9]

# Satırların toplamı (axis=1) → [1+2+3, 4+5+6]
satir_toplamlari = np.sum(dizi2d, axis=1)
print(f"Satır toplamları (axis=1): {satir_toplamlari}") # Çıktı: [ 6 15]
```

3. **`dtype`**: Toplama işlemi sırasında kullanılacak ve sonucun döndürüleceği veri tipini belirtir. Bu, özellikle büyük tamsayıları toplarken taşma (overflow) sorunlarını önlemek veya daha yüksek hassasiyet elde etmek için kullanışlıdır.Python

```
dizi_int8 = np.array([100, 100, 100], dtype=np.int8) # int8: -128 ile 127 arası
# Normal toplama int8 sınırını aşabilir ve yanlış sonuç verebilir
# print(np.sum(dizi_int8)) # Potansiyel taşma! Sonuç beklenmedik olabilir.

# dtype belirterek daha büyük bir veri tipi kullanma
toplam_int32 = np.sum(dizi_int8, dtype=np.int32)
print(f"\ndtype=np.int32 ile toplam: {toplam_int32}") # Çıktı: 300 (Doğru sonuç)
```

4. **`out`**: Sonucun yazılacağı alternatif bir çıktı dizisi belirtmenizi sağlar. Bu dizi, döndürülecek sonucun şekline ve veri tipine uygun olmalıdır. Genellikle ileri seviye kullanım veya bellek optimizasyonu için tercih edilir.Python

```
dizi = np.array([1, 2, 3])
cikti_dizisi = np.zeros(1, dtype=int) # Sonucun sığacağı bir dizi
np.sum(dizi, out=cikti_dizisi)
print(f"\n'out' parametresi ile sonuç: {cikti_dizisi}") # Çıktı: [6]
```

5. **`keepdims`**: Boolean (True/False) bir değer alır. Varsayılan olarak `False`'tur. Eğer `True` olarak ayarlanırsa, toplama yapılan eksenler sonuç dizisinden kaldırılmaz, bunun yerine boyutları 1 olarak korunur. Bu, orijinal dizinin boyut sayısını korumak istediğinizde (örneğin, broadcasting işlemlerinde) faydalıdır.Python

```
dizi2d = np.array([[1, 2, 3],
                   [4, 5, 6]])
print(f"\nİki boyutlu dizi:\n{dizi2d}")
print(f"Orijinal dizi şekli: {dizi2d.shape}") # Çıktı: (2, 3)

# keepdims=False (Varsayılan)
```

```
sutun_toplam_normal = np.sum(dizi2d, axis=0, keepdims=False)
print(f"axis=0, keepdims=False → Şekil: {sutun_toplam_normal.shape}") # Çıktı: (3,)

# keepdims=True
sutun_toplam_koru = np.sum(dizi2d, axis=0, keepdims=True)
print(f"axis=0, keepdims=True → Şekil: {sutun_toplam_koru.shape}") # Çıktı: (1, 3)
print(f"keepdims=True ile sütun toplamları:\n{sutun_toplam_koru}")
```

6. **initial**: Toplama işlemi için bir başlangıç değeri belirler. Toplama bu değerden başlar. Boş bir dizinin toplamı varsayılan olarak 0'dır, ancak `initial` ile farklı bir değer belirleyebilirsiniz.Python

```
dizi = np.array([1, 2, 3])
toplam_baslangicli = np.sum(dizi, initial=10)
print(f"\ninitial=10 ile toplam: {toplam_baslangicli}") # Çıktı: 16 (10 + 1 + 2 + 3)

bos_dizi = np.array([])
toplam_bos_varsayilan = np.sum(bos_dizi)
toplam_bos_initial = np.sum(bos_dizi, initial=5)
print(f"Boş dizi toplamı (varsayılan): {toplam_bos_varsayilan}") # Çıktı: 0.0
print(f"Boş dizi toplamı (initial=5): {toplam_bos_initial}")  # Çıktı: 5
```

7. **where**: Boolean (True/False) değerlerden oluşan bir maske dizisi alır. Yalnızca maskenin `True` olduğu konumlardaki elemanlar toplama işlemine dahil edilir. Maske, orijinal dizi `a` ile aynı şekle sahip olmalıdır.Python

```
dizi = np.array([1, 2, 3, 4, 5])
maske = np.array([True, False, True, False, True]) # 1., 3. ve 5. elemanlar
toplam_maskeli = np.sum(dizi, where=maske)
print(f"\nMaskeli toplama (True olanlar): {toplam_maskeli}") # Çıktı: 9 (1 + 3 + 5)

# Koşulla maske oluşturma (3'ten büyük elemanları topla)
toplam_kosullu = np.sum(dizi, where=dizi > 3)
print(f"3'ten büyük elemanların toplamı: {toplam_kosullu}") # Çıktı: 9 (4 + 5)
```

Özetle:

`numpy.sum()`, basit bir toplamadan çok daha fazlasını yapabilen güçlü bir fonksiyondur. Özellikle `axis` parametresi, çok boyutlu verilerle çalışırken veri analizinde sıkça kullanılır (örneğin, bir veri setindeki belirli bir özelliğin toplamını veya ortalamasını almak için). `dtype`, `keepdims`, `initial` ve `where` gibi diğer parametreler ise daha özel durumlar ve optimizasyonlar için esneklik sağlar.