Python'da NumPy nedir?

NumPy, matematiksel, bilimsel, mühendislik ve veri bilimi programlamasına yardımcı olan Python'da bulunan açık kaynaklı bir kütüphanedir. Python'da matematiksel ve istatistiksel işlemleri gerçekleştirmek için çok kullanışlı bir kütüphanedir. Çok boyutlu diziler ve matris çarpımı için mükemmel çalışır. C/C++ ve Fortran ile entegre edilmesi kolaydır.

Herhangi bir bilimsel proje için NumPy bilinmesi gereken bir araçtır.

N boyutlu dizi, doğrusal cebir, rastgele sayı, Fourier dönüşümü vb. ile çalışmak üzere inşa edilmiştir.

NumPy, çok boyutlu diziler ve matrislerle ilgilenen bir programlama dilidir. Diziler ve matrislerin yanı sıra NumPy çok sayıda matematiksel işlemi de destekler.

Neden NumPy Kullanmalı?

Python'da dizilerin amacına hizmet eden listelerimiz var, ancak işlemeleri yavaş.

NumPy, geleneksel Python listelerinden 50 kata kadar daha hızlı bir dizi nesnesi sağlamayı amaçlamaktadır.

NumPy'deki dizi nesnesi ndarray olarak adlandırılır, ndarray ile çalışmayı çok kolaylaştıran birçok destekleyici işlev sağlar.

Diziler, hızın ve kaynakların çok önemli olduğu veri biliminde çok sık kullanılır.

Veri Bilimi: Bilgisayar biliminin bir dalıdır ve veriden bilgi elde etmek için verinin nasıl depolanacağını, kullanılacağını ve analiz edileceğini inceler.

NumPy Neden Listelerden Daha Hızlıdır?

NumPy dizileri, listelerden farklı olarak bellekte sürekli bir yerde saklanır, bu nedenle işlemler bunlara çok verimli bir şekilde erişebilir ve bunları değiştirebilir.

Bu davranış bilgisayar bilimlerinde referansın yerelliği olarak adlandırılır.

NumPy'nin listelerden daha hızlı olmasının ana nedeni budur. Ayrıca en son CPU mimarileri ile çalışmak üzere optimize edilmiştir.

NumPy Hangi Dilde Yazılmıştır?

NumPy bir Python kütüphanesidir ve kısmen Python'da yazılmıştır, ancak hızlı hesaplama gerektiren kısımların çoğu C veya C++'da yazılmıştır.

NumPy Kod Tabanı Nerede?

NumPy için kaynak kodu bu github deposunda yer almaktadır
https://github.com/numpy/numpy

NumPy Nasıl Kurulur

NumPy kütüphanesini yüklemek için lütfen eğitimimize bakın. NumPy, Anaconda ile varsayılan olarak yüklenir.

pip install numpy as np

Python NumPy Dizisi Nedir?

NumPy dizileri biraz Python listelerine benzer, ancak yine de aynı zamanda çok farklıdır. Konuya yeni başlayanlar için, tam olarak ne olduğunu ve ne işe yaradığını açıklığa kavuşturalım.

Adından da anlaşılacağı üzere NumPy dizisi, numpy kütüphanesinin merkezi veri yapısıdır. Kütüphanenin adı aslında "Numeric Python" veya "Numerical Python" kelimelerinin kısaltmasıdır.

NumPy Dizisi Oluşturma

Numpy'de bir dizi oluşturmanın en basit yolu Python List kullanmaktır.

```
myPythonList = [1,9,8,3]
```

np.array nesnesini kullanarak python listesini bir numpy dizisine dönüştürmek için.

```
numpy_array_from_list = np.array(myPythonList)
```

Listenin içeriğini görüntülemek için

```
numpy_array_from_list
```

https://github.com/ekremtnckr35/DataScience.git

Output:

```
array([1, 9, 8, 3])
```

Pratikte, bir Python Listesi bildirmeye gerek yoktur. İşlem birleştirilebilir.

```
a  = np.array([1,9,8,3])
```

NOT: Numpy dokümantasyonu, bir dizi oluşturmak için np.ndarray kullanımını belirtir. Ancak, bu önerilen yöntemdir.

Ayrıca bir Tuple'dan bir numpy dizisi de oluşturabilirsiniz.

Dizi Üzerinde Matematiksel İşlemler

Bir dizi üzerinde toplama, çıkarma, bölme ve çarpma gibi matematiksel işlemler gerçekleştirebilirsiniz. Sözdizimi, dizi adı ve ardından işlem (+.-,*,/) ve ardından işlenen şeklindedir

Example:

```
numpy_array_from_list + 10
```

Output:

```
array([11, 19, 18, 13])

Bu işlem numpy dizisinin her bir elemanına 10 ekler.
```

Shape of Array(Dizi Şekli)

Dizinin şeklini, dizinin adından önce gelen nesne şekli ile kontrol edebilirsiniz. Aynı şekilde, dtypes ile türünü kontrol edebilirsiniz.

```
import numpy as np
a  = np.array([1,2,3])
print(a.shape)
print(a.dtype)

(3,)
int64
```

Tamsayı, ondalık içermeyen bir değerdir. Ondalıklı bir dizi oluşturursanız, tür float olarak değişecektir.

```
#### Different type
b  = np.array([1.1,2.0,3.2])
print(b.dtype)

float64
```

2 Dimension Array(2 Boyutlu Dizi)

", "koma ile bir boyut ekleyebilirsiniz

Parantez içinde olması gerektiğine dikkat edin []

```
### 2 dimension
c = np.array([(1,2,3),
              (4,5,6)])
print(c.shape)
(2, 3)
```

3 Dimension Array(3 Boyutlu Dizi)

Daha yüksek boyut aşağıdaki gibi oluşturulabilir:

```
### 3 dimension
d = np.array([
    [[1, 2,3],
        [4, 5, 6]],
    [[7, 8,9],
        [10, 11, 12]]
])
print(d.shape)
(2, 2, 3)
```

Objective	Code
Create array	array([1,2,3])
print the shape	array([.]).shape

numpy.zeros() nedir?

numpy.zeros() veya np.zeros Python fonksiyonu, sıfırlarla dolu bir matris oluşturmak için kullanılır. numpy.zeros() Python'da, TensorFlow ve diğer istatistik görevlerinde ilk yineleme sırasında ağırlıkları başlattığınızda kullanılabilir.

numpy.zeros() function Syntax

```
numpy.zeros(shape, dtype=float, order='C'
```

Python numpy.zeros() Parametreler

- **Shape**: numpy sıfır dizisinin şeklidir
- **Dtype**: numpy sıfırları cinsinden veri türüdür. İsteğe bağlıdır. Varsayılan değer float64'tür
- **Order**: Varsayılan değer, Python'da numpy.zeros() için temel bir satır stili olan C'dir.

- **Python numpy.zeros() Example**

```
- import numpy as np
- np.zeros((2,2))
```

- **Output:**

```
array([[0., 0.],
           [0., 0.]])
```

Example of numpy zero with Datatype

```
import numpy as np
np.zeros((2,2), dtype=np.int16)
```

Output:

```
array([[0, 0],
         [0, 0]], dtype=int16)
```

numpy.ones() nedir?

np.ones() fonksiyonu birlerle dolu bir matris oluşturmak için kullanılır. numpy.ones() Python'da TensorFlow ve diğer istatistik görevlerinde ilk iterasyon sırasında ağırlıkları başlattığınızda kullanılabilir.

Python numpy.ones() Syntax

```
numpy.ones(shape, dtype=float, order='C')
```

Python numpy.ones() Parameters

- **Shape**: np.ones Python Dizisinin şekli
- **Dtype**: numpy olanlarda veri türüdür. Opsiyoneldir. Varsayılan değer float64'tür
- **Order**: Varsayılan değer, temel bir satır stili olan C'dir.

Python numpy.ones() Veri Tipi ile 2D Dizi Örneği

```
import numpy as np
np.ones((1,2,3), dtype=np.int16)
```

Output:

```
array([[[1, 1, 1],
        [1, 1, 1]]], dtype=int16)
```

numpy.reshape() function in Python

Python NumPy Reshape fonksiyonu, bir diziyi verilerini değiştirmeden şekillendirmek için kullanılır. Bazı durumlarda, verileri genişten uzuna yeniden şekillendirmeniz gerekebilir. Bunun için np.reshape fonksiyonunu kullanabilirsiniz.

Syntax of np.reshape()

```
numpy.reshape(a, newShape, order='C')
```

a: Yeniden şekillendirmek istediğiniz dizi

newShape: Yeni arzular şekilleniyor

Order: Varsayılan değer, temel bir satır stili olan C'dir.

Example of NumPy Reshape

```
import numpy as np
e  = np.array([(1,2,3), (4,5,6)])
print(e)
e.reshape(3,2)
```

Output:

```
 // Before reshape
[[1 2 3]
 [4 5 6]]
//After Reshape
array([[1, 2],
   [3, 4],
   [5, 6]])
```

https://github.com/ekremtnckr35/DataScience.git

numpy.flatten() in Python

Python NumPy Flatten fonksiyonu dizinin tek boyutlu bir kopyasını döndürmek için kullanılır. Convnet gibi bazı sinir ağları ile uğraşırken, diziyi düzleştirmeniz gerekir. Bunun için np.flatten() fonksiyonlarını kullanabilirsiniz.

Syntax of np.flatten()

```
numpy.flatten(order='C'
```

Order: Varsayılan değer, temel bir satır stili olan C'dir.

Example of NumPy Flatten

```
e.flatten()
```

Output:

```
array([1, 2, 3, 4, 5, 6])
```

Generate Random Numbers using NumPy

Gauss dağılımı için rastgele sayılar üretmek için şunu kullanın:

```
numpy.random.normal(loc, scale, size)
```

Here,

- **Loc:** ortalama. Dağılımın merkezi
- **Scale:** standard deviation.(standart sapma.)
- **Size:** number of returns 9iade sayısı)

Example:

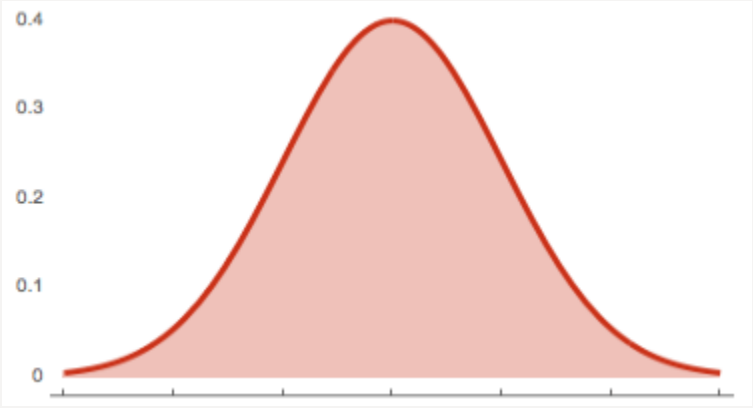
```
## Generate random nmber from normal distribution
(Normal dağılımdan rastgele nmber oluşturun)

normal_array = np.random.normal(5, 0.5, 10)
print(normal_array)
```

```
[5.56171852 4.84233558 4.65392767 4.946659   4.85165567
5.61211317 4.46704244 5.22675736 4.49888936 4.68731125]
```

If plotted the distribution will be similar to following plot

(Çizilirse dağılım aşağıdaki grafiğe benzer olacaktır)



Example to Generate Random Numbers using NumPy

What is numpy.arange()?

numpy.arange() is an inbuilt numpy function that returns an ndarray object containing evenly spaced values within a defined interval. For instance, you want to create values from 1 to 10; you can use np.arange() in Python function.

(numpy.arange(): tanımlanmış bir aralık içinde eşit aralıklı değerler içeren bir ndarray nesnesi döndüren dahili bir numpy işlevidir. Örneğin, 1'den 10'a kadar değerler oluşturmak istiyorsunuz; Python işlevinde np.arange() kullanabilirsiniz.

Syntax:

```
numpy.arange(start, stop, step, dtype)
```

Python NumPy arange Parameters:

- **Start:** Start of interval for np.arange in Python function.
- **Stop:** End of interval.
- **Step:** Spacing between values. Default step is 1.

- **Dtype:** Is a type of array output for NumPy arange in Python.

Example:

```
import numpy np
np.arange(1, 11)
```

Output:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

Example:

If you want to change the step in this NumPy arange function in Python example, you can add a third number in the parenthesis. It will change the step.

(Python örneğindeki bu NumPy arange fonksiyonundaki adımı değiştirmek isterseniz, parantez içine üçüncü bir sayı ekleyebilirsiniz. Bu, adımı değiştirecektir.)

```
import numpy np
np.arange(1, 14, 4)
```

Output:

```
array([ 1,  5,  9, 13])
```

NumPy Linspace Function

Linspace gives evenly spaced samples.

(Linspace eşit aralıklı örnekler verir.)

Syntax:

```
numpy.linspace(start, stop, num, endpoint)
```

Here,

- **Start:** Starting value of the sequence
- **(Dizinin başlangıç değeri)**
- **Stop:** End value of the sequence
- (Dizinin son değeri)
- **Num:** Number of samples to generate. Default is 50
-
- **Endpoint:** If True (default), stop is the last value. If False, stop value is not included.

Example:

For instance, it can be used to create 10 values from 1 to 5 evenly spaced.

```
import numpy as np
np.linspace(1.0, 5.0, num=10)
```

Output:

```
array([1.        , 1.44444444, 1.88888889, 2.33333333,
2.77777778,       3.22222222, 3.66666667, 4.11111111,
4.55555556, 5.        ])
```

If you do not want to include the last digit in the interval, you can set endpoint to false

```
np.linspace(1.0, 5.0, num=5, endpoint=False)
```

Output:

```
array([1. , 1.8, 2.6, 3.4, 4.2])
```

LogSpace NumPy Function in Python

LogSpace returns even spaced numbers on a log scale. Logspace has the same parameters as np.linspace.

Syntax:

```
numpy.logspace(start, stop, num, endpoint)
```

Example:

```
np.logspace(3.0, 4.0, num=4)
```

Output:

```
array([ 1000. ,  2154.43469003,  4641.58883361, 10000.
])
```

Finaly, if you want to check the memory size of an element in an array, you can use itemsize

```
x = np.array([1,2,3], dtype=np.complex128)
x.itemsize
```

Output:

```
16
```

Each element takes 16 bytes.

https://github.com/ekremtnckr35/DataScience.git

NumPy Array Indexing(NumPy Dizi İndeksleme)

Dizi Elemanlarına Erişim

Dizi indeksleme, bir dizi elemanına erişmekle aynı şeydir.

Bir dizi elemanına indeks numarasına başvurarak erişebilirsiniz.

NumPy dizilerindeki indeksler 0 ile başlar, yani ilk elemanın indeksi 0, ikincisinin indeksi 1 vb. olur.

Example

Get the first element from the following array:

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[0])
```

Example

Get the second element from the following array.

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[1])
```

Example

Get third and fourth elements from the following array and add them.

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[2] + arr[3])
```

2 Boyutlu Dizilere Erişim

2-B dizilerdeki elemanlara erişmek için, elemanın boyutunu ve indeksini temsil eden virgülle ayrılmış tamsayılar kullanabiliriz.

2 boyutlu dizileri satırları ve sütunları olan bir tablo gibi düşünün; burada boyut satırı ve dizin sütunu temsil eder.

Example

Access the element on the first row, second column:

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])
```

Example

Access the element on the 2nd row, 5th column:

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('5th element on 2nd row: ', arr[1, 4])
```

Access 3-D Arrays(3 Boyutlu Dizilere Erişim)

3 boyutlu dizilerdeki öğelere erişmek için boyutları ve öğenin indeksini temsil eden virgülle ayrılmış tamsayılar kullanabiliriz.

Example

Access the third element of the second array of the first array:

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])
```

Negative Indexing

Bir diziye sondan erişmek için negatif indeksleme kullanın.

Example

Print the last element from the 2nd dim:

```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('Last element from 2nd dim: ', arr[1, -1])
```

NumPy Array Slicing

Slicing arrays

Python'da dilimleme, verilen bir indeksten verilen başka bir indekse eleman almak anlamına gelir.

İndeks yerine dilimi şu şekilde geçiriyoruz: [start:end].

Adımı şu şekilde de tanımlayabiliriz:
[start:end:step].

Eğer pas geçemezsek 0 sayılır.

Sonu geçmezsek, o boyuttaki dizinin uzunluğu olarak kabul edilir

Eğer adımı geçemezsek 1 olarak kabul edilir.

Example

Slice elements from index 1 to index 5 from the following array:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

Not: Sonuç başlangıç indeksini içerir, ancak bitiş indeksini hariç tutar.

Example

Slice elements from index 4 to the end of the array:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[4:])
```

Example

Slice elements from the beginning to index 4 (not included):

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[:4])
```

Negative Slicing

Sondan bir dizine başvurmak için eksi işlecini kullanın

Example

Slice from the index 3 from the end to index 1 from the end:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[-3:-1])
```

STEP

Sliciling adımını belirlemek için step değerini kullanın:

Example

Return every other element from index 1 to index 5:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5:2])
```

Example

Return every other element from the entire array:

```
import numpy as np

arr=np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[::2])
```

Slicing 2-D Arrays

Example

İkinci elemandan, indeks 1'den indeks 4'e kadar olan elemanları sliciling yapin (dahil değildir)

```
import numpy as np

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])
```

Not: İkinci elemanın indeksinin 1 olduğunu unutmayın.

Example

From both elements, return index 2:

```
import numpy as np

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[0:2, 2])
```

Example

From both elements, slice index 1 to index 4 (not included), this will return a 2-D array:

```
import numpy as np

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[0:2, 1:4])
```

NumPy Data Types

Data Types in Python

strings: metin verilerini temsil etmek için kullanıldığında, metin tırnak işaretleri altında verilir. örneğin "ABCD

integer : tam sayıları temsil etmek için kullanılır. örn. -1, -2, -3

float : gerçek sayıları temsil etmek için kullanılır. örn. 1.2, 42.42

Boolean : Doğru veya Yanlışı temsil etmek için kullanılır. True and False

Complex: karmaşık sayıları temsil etmek için kullanılır. örn. 1.0 + 2.0j, 1.5 + 2.5j

Bir Dizinin Veri Türünü Kontrol Etme

NumPy dizi nesnesi, dizinin veri türünü döndüren dtype adlı bir özelliğe sahiptir:

Example

Get the data type of an array object:

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr.dtype)
```

Example

Get the data type of an array containing strings:

```
import numpy as np

arr = np.array(['apple', 'banana', 'cherry'])

print(arr.dtype)
```

NumPy Sorting Arrays(NumPy Sıralama Dizileri)

Sorting Arrays(Dizileri Sıralama)

Sıralama, öğeleri sıralı bir diziye koymak anlamına gelir.

Sıralı dizi, sayısal veya alfabetik, artan veya azalan gibi öğelere karşılık gelen bir sıraya sahip herhangi bir dizidir.

NumPy ndarray nesnesi, belirtilen bir diziyi sıralayacak sort() adlı bir işleve sahiptir.

Example

Sort the array:

```
import numpy as np

arr = np.array([3, 2, 0, 1])

print(np.sort(arr))
```

Not: Bu yöntem, orijinal diziyi değiştirmeden dizinin bir kopyasını döndürür.Dizeleri veya başka herhangi bir veri türünü de sıralayabilirsiniz:

Example

Sort the array alphabetically:

```
import numpy as np

arr = np.array(['banana', 'cherry', 'apple'])

print(np.sort(arr))
```

Statistical Functions in Python(Python'da İstatistiksel Fonksiyonlar)

NumPy, dizide verilen elemanlardan minimum, maksimum, yüzdelik standart sapma ve varyans vb. bulmak için oldukça kullanışlı birkaç istatistiksel fonksiyona sahiptir. Fonksiyonlar aşağıdaki gibi açıklanmıştır -

Numpy, aşağıda listelenen sağlam istatistiksel fonksiyon ile donatılmıştır

Function	Numpy
Min	np.min()
Max	np.max()
Mean	np.mean()
Median	np.median()
Standard deviation	np.std()

Consider the following Array:

Example:

```
import numpy as np
normal_array = np.random.normal(5, 0.5, 10)
print(normal_array)
```

Output:

```
[5.56171852 4.84233558 4.65392767 4.946659   4.85165567
5.61211317 4.46704244 5.22675736 4.49888936 4.68731125]
```

Example of NumPy Statistical function

```
### Min
print(np.min(normal_array))

### Max
print(np.max(normal_array))

### Mean
print(np.mean(normal_array))

### Median
print(np.median(normal_array))

### Sd
print(np.std(normal_array))
```

Output:

```
4.467042435266913
5.612113171990201
4.934841002270593
4.846995625786663
0.3875019367395316
```

https://github.com/EkremTunckir35