Unyleya
EDUCACIONAL

A biblioteca Numpy

A biblioteca Numpy

Essa é uma biblioteca que apoia muito o trabalho com dados, especialmente para a realização de operações matemáticas sobre arrays.

Seus recursos são projetados para permitir a realização de cálculos numéricos de uma forma mais simples e direta.

Contém muitas funcionalidades para a manipulação de imagens.

ndarray

O array do Numpy tem características e funcionalidades diferentes, sendo especialmente chamado de ndarray.

Utiliza um esquema diferente para mapeamento dos dados do ndarray na memória. Essa indexação é responsável por fazer o manuseio desse tipo de objeto ser executado de forma rápida, mesmo quando possui uma grande quantidade de dados.

ndarray

A criação de um ndarray é bastante simples e pode ser realizada logo depois da biblioteca ter sido importada.

```
import numpy as np
numeros = np.array([6, 7, 2, 1, 9])
```

Verificando a estrutura

A estrutura de ndarray pode ser verificada através da função *shape* que retorna uma tupla com as dimensões do objeto, linhas e colunas. Para o objeto “numeros” apresentado anteriormente o retorno da função *shape* seria o seguinte:

```
numeros.shape
(5,)
```

Array com duas dimensões

Muitas ocasiões necessitarão de arrays com múltiplas dimensões para representar determinadas informações. A criação de um array com múltiplas dimensões é muito simples:

```
numeros = np.array([[6, 7, 2, 1, 9], [6, 7, 2, 1, 9]])
```

Array com duas dimensões

Se a função *shape* fosse utilizada sobre esse novo array, o resultado seria diferente. Na tupla resultante está indicado que o array possui duas dimensões com 5 valores cada.

```
numeros = np.array([[6, 7, 2, 1, 9], [6, 7, 2, 1, 9]])
numeros.shape
(2, 5)
```

Acessando o ndarray

A indexação do ndarray começa da posição zero, sendo possível acessar os valores do objeto através de seus índices.

```
numeros[1]
```

Alternado elemento ndarray

A alteração de um elemento dentro de ndarray é feita de forma simples, basta referenciar o índice que se deseja modificar e fazer a atribuição direta do valor.

```
numeros[1] = 7
```

Criando ndarray vazio

Um array vazio pode se útil em várias situações. Com os recursos do numpy isso pode ser feito de forma fácil. A função *empty* é adequada para essa situação.

```
numeros = np.empty([3,2], dtype = int)
```

Criando ndarray com zeros

A criação de um vetor com valores 0 pode ser interessante em muitas situações dentro da programação. Para isso, deve-se utilizar uma função específica, a *zeros*. No exemplo será criado ndarray com 5 zeros.

```
numeros = np.zeros((5))
```

Criando ndarray com valores aleatórios

A função random pode ser utilizada para a geração de valores aleatórios, podendo os valores ser diretamente alocados em ndarray.

```
numeros  = np.random.random((5))
```

Manipulando ndarray

São muitas as operações matemáticas que podem ser executadas em um ndarray e algumas delas funcionam muito diferente do que acontece em grande parte das linguagens de programação.

Vejamos que a operação de adição de ndarrays pode ser executadas de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])
numeros = numeros1 + numeros2
```

```
Adição =
[[ 8.  10.]
[13. 17.]]
```

Manipulando ndarray

Vejamos que a operação de subtração de ndarrays pode ser executadas de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])
numeros = numeros1 - numeros2

Subtração =
[[ -4.  -4.]
[7. -1.]]
```

Manipulando ndarray

Vejamos que a operação de multiplicação por elementos de ndarrays pode ser executadas de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])
numeros = numeros1 * numeros2

Subtração =
[[ 12.  21.]
[30. 72.]]
```

Manipulando ndarray

Vejamos que a operação de divisão de ndarrays pode ser executadas de forma direta.

```
numeros1 = np.array([[2.0, 3.0], [10.0, 8.0]])
numeros2 = np.array([[6.0, 7.0], [3.0, 9.0]])
numeros = numeros1 / numeros2
```

```
Subtração =
[[ 0.3333  0.4285]
[3.3333 0.8888]]
```

Obrigada!

Ana Laurentino