Aula 06

Programação Orientada a Objetos

Matrizes

Yuri Max

Natal - RN
15 de setembro de 2022

Conteúdo

- Matrizes
- NumPy
- Pandas (introdução)
- Exercícios

Matriz

- É uma coleção ordenada de listas;
- Cada lista armazenada possui uma posição na memória;
- É possível acessar diretamente uma ou várias posições com o uso de [][];
- A primeira posição é [0][0];

```
matriz = [[1, 2, 3],[4, 5, 6]]

print(matriz[0])
print(matriz[1])
print(type(matriz))
print(matriz[1][2])
```

```
[1, 2, 3]
[4, 5, 6]
<class 'list'>
6
```

Matriz

- Como é uma matriz de listas, é possível utilizar as funções de listas;
- .append() adiciona uma nova linha no final da matriz;
- .reverse() inverte as posições das linhas;

```
matriz = [[1, 2, 3],[4, 5, 6]]

matriz.append([i for i in range(7,10)])

for i in range(len(matriz)):
    print(matriz[i])

matriz.reverse()

for i in range(len(matriz)):
    print(matriz[i])
```

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
[7, 8, 9]
[4, 5, 6]
[1, 2, 3]
```

Matriz

- .pop(n) remove a linha n da matriz;
- .remove() remove a linha caso igual ao argumento;
- len() retorna a quantidade de linhas da matriz;
- Use o módulo copy e o método deepcopy quando quiser copiar matrizes

```
matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

matriz.pop(0)

for i in range(len(matriz)):
    print(matriz[i])

print(len(matriz))

matriz.remove([7, 8, 9])

for i in range(len(matriz)):
    print(matriz[i])
```

```
[4, 5, 6]
[7, 8, 9]
2
[4, 5, 6]
```

Matriz

- + adiciona uma matriz ao final de outra;
- * multiplica as linhas de uma matriz, adicionando-as ao final da matriz;
- max() e min() retornam o valor máximo e mínimo do primeiro elemento da matriz;

```
matriz = [[1, 2, 200], [4, 5, 6], [7, 8, 9]]

matriz += 2*[[10, 11, 12]]

for i in range(len(matriz)):
    print(matriz[i])

print('\n', min(matriz), sep='')

print('\n', max(matriz), sep='')
```

```
[1, 2, 200]
[4, 5, 6]
[7, 8, 9]
[10, 11, 12]
[10, 11, 12]

[1, 2, 200]

[10, 11, 12]
```

Matriz

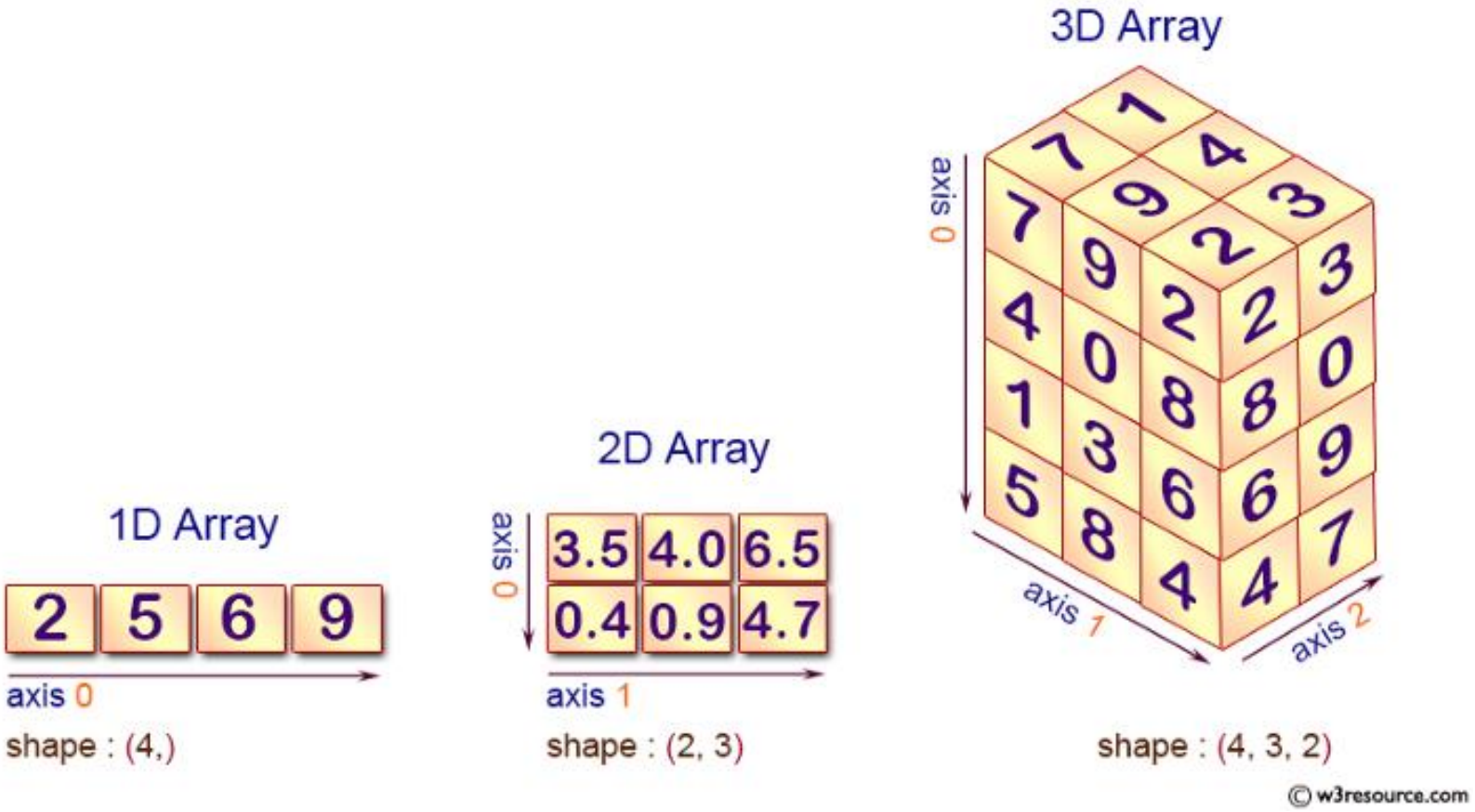
- Exemplo:

- Crie uma matriz 3x3 com valores aleatórios de 1 a 10 e coloque a matriz em ordem decrescente a partir das linhas.

NumPy

- Numerical Python
- É uma biblioteca especializada para trabalhar com arrays (vetores e matrizes);
- É atualizada frequentemente;
- Permite operações matemáticas entre vetores e matrizes;
- Indicada quando for necessário muitas operações com matrizes;
- As dimensões são referidas como eixos (shape), representadas como tuplas;

NumPy


1D Array
2 5 6 9
axis 0
shape : (4,)
2D Array
axis 0
3.5 4.0 6.5
0.4 0.9 4.7
axis 1
shape : (2, 3)
3D Array
axis 0
axis 1
axis 2
shape : (4, 3, 2)
© w3resource.com

NumPy

```
>>> a[0,3:5]
array([3,4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```

NumPy

- np.array() recebe uma lista ou matriz e transforma em array;
- np.arange() cria um array similarmente ao comando range();
- np.empty() cria um array vazio;
- np.one() cria um array de 1s;

```python
import numpy as np

arr1 = np.array([[1, 2, 3], [4, 5, 6]])
arr2 = np.arange(6)
arr3 = np.empty([2, 3], dtype = int)
arr4 = np.ones([2, 3])

print(arr1)
print(arr2)
print(arr3)
print(arr4)
```

```
[[1 2 3]
 [4 5 6]]
[0 1 2 3 4 5]
[[ 1572917   -65396 20381695]
 [  -65418 18153471   -65262]]
[[1. 1. 1.]
 [1. 1. 1.]]
```

NumPy

- np.linspace() cria um array com início-fim e quantidade de pontos;
- np.eye() cria um array identidade;
- np.zeros() cria uma array de zeros;
- np.diag() cria uma array diagonal;

```
import numpy as np

arr1 = np.linspace(0, 10, 3)
arr2 = np.eye(3)
arr3 = np.zeros([3, 3])
arr4 = np.diag([2, 3, 1])

print(arr1)
print(arr2)
print(arr3)
print(arr4)
```

```
[ 0.  5. 10.]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[2 0 0]
 [0 3 0]
 [0 0 1]]
```

NumPy

- np.random.randint() cria uma array aleatório;
- np.fix() np.trunc() cria um array sem a parte decimal;
- np.around() cria um array com limitação de casas decimais;
- np.floor() arredonda para o número inteiro mais baixo;
- np.ceil() arredonda para o número inteiro mais baixo;

```
import numpy as np

arr1 = np.random.randint(0,10, size=(2,5))
arr2 = np.fix([3.166,3.666])
arr3 = np.around([3.166, 3.666],2)
arr4 = np.floor([3.166,3.666])
arr5 = np.ceil([3.166,3.666])

print(arr1)
print(arr2)
print(arr3)
print(arr4)
print(arr5)
```

```
[[5 4 7 1 0]
 [6 4 1 3 5]]
[3. 3.]
[3.17 3.67]
[3. 3.]
[4. 4.]
```

NumPy

- np.concatenate() une dois ou mais arrays;
- np.split() separa um array em várias partes;
- np.where() retorna o local caso condição seja verdadeira;
- np.flip() inverte as posições do array

```python
import numpy as np

arr1 = np.random.randint(0, 10, size=(2, 2))
arr2 = np.random.randint(0, 10, size=(2, 2))

print(arr1)
print(arr2)
print(np.concatenate((arr1, arr2)))
print(np.concatenate((arr1, arr2), 1))
print(np.split(arr1, 2, axis = 0))
print(np.where(arr1 > 4))
```

```
[[4 7]
 [5 1]]
[[5 0]
 [9 3]]
[[4 7]
 [5 1]
 [5 0]
 [9 3]]
[[4 7 5 0]
 [5 1 9 3]]
[array([[4, 7]]), array([[5, 1]])]
(array([0, 1], dtype=int64), array([1, 0],
```

NumPy

- np.poly1d() cria um polinômio onde cada elemento representa quem acompanha a variável;
- .roots mostra as raízes e .order o grau do polinômio;
- np.cos() cria um array com o cosseno de cada elemento;
- ps.: sin, cosh, sinh, tan, arccos, arccsin, arctan, etc.;

```
import numpy as np

arr1 = np.poly1d([2,1,3])
arr2 = np.linspace(0,1,5)

print(arr1)
print(arr1.roots, arr1.order)
print(np.around(np.cos(arr2), 2))
```

```
   2
2 x + 1 x + 3
[-0.25+1.19895788j -0.25-1.19895788j] 2
[1.   0.97 0.88 0.73 0.54]
```

NumPy

- .shape retorna uma tupla com linhas e colunas do array;
- .ndim retorna interio com a dimensão do array;
- .dtype retorna o tipo do array;
- len() retorna a quantidade de linhas do array;
- .reshape() modifica as linhas e colunas de um array;

```
import numpy as np

arr1 = np.random.randint(0, 10, size = (2, 3))

print(arr1)
print(arr1.shape)
print(arr1.ndim)
print(arr1.dtype)
print(len(arr1))
print(arr1.reshape(3, 2))
```

```
[[3 1 7]
 [0 0 3]]
(2, 3)
2
int32
2
[[3 1]
 [7 0]
 [0 3]]
```

NumPy

- .mean() retorna a média aritmética de todos os elementos;
- .max()/min() retorna o elemento máximo/mínimo do array;
- .ravel() transforma um array matriz em array linha;
- ps: None: todos os elementos, 0: entre linhas diferentes, 1: na mesma linha

```
import numpy as np

arr1 = np.random.randint(0,10, size=(2,3))

print(arr1)
print(arr1.mean())
print(arr1.max(0))
print(arr1.min(1))
print(arr1.ravel())
```

```
[[9 9 3]
 [8 9 0]]
6.333333333333333
[9 9 3]
[3 0]
[9 9 3 8 9 0]
```

NumPy

- .argmin()/argmax() retorna a coluna do max/min coluna;
- .std() retorna o desvio padrão;
- .sum() retorna a soma dos elementos;
- .sort() coloca cada linha em ordem crescente;
- .sort(0) coloca cada coluna em ordem;

```python
import numpy as np

arr1 = np.random.randint(0, 10, size=(2, 3))

print(arr1)
print(arr1.argmin(), arr1.argmax())
print(arr1.std())
print(arr1.sum())
arr1.sort()
print(arr1)
arr1.sort(0)
print(arr1)
```

```
[[3 6 0]
 [2 4 0]]
2 1
2.140872096444188
15
[[0 3 6]
 [0 2 4]]
[[0 2 4]
 [0 3 6]]
```

NumPy

- + produz a soma entre elementos do array;
- * ou / produz a multiplicação entre cada elemento do array;
- .dot() faz a multiplicação entre arrays;
- / ou * divide ou multiplica cada elemento por um inteiro;
- .T produz a transposta do array;

```python
import numpy as np

arr1 = np.random.randint(0,3, size=(2,3))
arr2 = np.random.randint(0,3, size=(2,3))

print(arr1,'\n', arr2)
print(arr1+arr2)
print(arr1*arr2)
print(arr1.dot(arr2.T))
print(arr1/2)
```

```
[[0 1 1]
 [2 0 2]]
 [[1 1 1]
 [0 0 0]]
[[1 2 2]
 [2 0 2]]
[[0 1 1]
 [0 0 0]]
[[2 0]
 [4 0]]
[[0.  0.5 0.5]
 [1.  0.  1. ]]
```

NumPy

- Suporta operadores relacionais (>, <, ==, !=, ...);
- np.all() retorna verdadeiro se todos os elementos forem verdadeiros;
- np.any() retorna verdadeiros se algum elemento for verdadeiro;

```python
import numpy as np

arr1 = np.random.randint(0, 10, size=(2, 3))
arr2 = np.random.randint(0, 10, size=(2, 3))

print(arr1)
print(arr2)
print(arr1 > arr2)
print(arr1 == arr2)
print(arr1[arr1 < arr2])
```

```
[[9 6 5]
 [2 3 6]]
[[3 6 6]
 [2 4 7]]
[[ True False False]
 [False False False]]
[[False  True False]
 [ True False False]]
[5 3 6]
```

NumPy

- Exemplo:

- Crie uma matriz 3x3 com valores aleatórios de 1 a 10 e coloque a matriz em ordem decrescente a partir das linhas.
- Crie uma função seno de –pi a pi e plote o gráfico.
- Arredonde todos os elementos de uma matriz 3x3 com zero casas decimais.

Pandas

- Biblioteca muito utilizada para análise de dados;
- Eficiente para matrizes com muitas linhas e colunas (500k+)
- Utiliza numpy na sua programação;
- Possui dois tipos de dados: DataFrame e Series;
- Permite leitura de arquivos de dados como excel, .csv, etc.;

```python
import pandas as pd

data = {
    'Comodos': ['Quarto', 'Cozinha', 'Sala'],
    'cadeiras': [2, 6, 10]
}

datapanda = pd.DataFrame(data)
seriespanda = pd.Series([2, 6, 10])

print(datapanda)
print(seriespanda)
```

```
    Comodos  cadeiras
0    Quarto         2
1  Cozinha          6
2     Sala         10
0      2
1      6
2     10
```

Exercícios

1. Crie um programa que receba uma matriz 3x3 do usuário.
2. Crie duas matrizes aleatórias com tamanhos iguais, então faça a soma e subtração entre elas;
3. Crie um código que transforme uma matriz 4x4 de caracteres e mostre o equivalente de cada posição em ascii.
4. Crie uma matriz aleatória 2x2 e mostre o quadrado de cada posição.
5. Crie uma matriz aleatória 10x10 e mostre apenas os elementos da diagonal principal.
6. Crie uma matriz aleatória 3x3 e calcule sua determinante.
7. Crie uma matriz 3x3 a partir da multiplicação de dois vetores aleatórios (3x1 e 1x3).
8. Crie uma matriz aleatória 3x3 e some suas colunas;