Matrizes (arrays bidimensionais)

Sumário

Matrizes

Uma matriz é definida como uma lista onde cada elemento também é uma lista:

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

Matrizes

Uma matriz é definida como uma lista onde cada elemento também é uma lista:

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

Muitas vezes é mais intuitivo definirmos uma matriz alinhando as suas colunas:

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

Matrizes

Acessamos o elemento na linha i e coluna j utilizando a sintaxe

```
valor = valores[i][j]
```

Matrizes

- Em Python, a seguinte sintaxe é utilizada para acessar a lista correspondente a uma linha da matriz:

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

# linha será a lista [4, 5, 6]
linha = valores[1]
```

- Não é possível acessar diretamente uma coluna inteira da matriz

Matrizes

- Como uma matriz é uma lista de contendo outras listas, podemos encontrar o número de linhas e colunas de uma matriz através dos comandos:

```
# Número de linhas
nl = len(valores)
# O número de colunas é dado pelo tamanho de uma das linhas
nc = len(valores[0])
```

Exemplo de código utilizando matrizes – Soma dos elementos de uma matriz

```
def soma_matriz(valores):
    """Soma os valores de uma matriz.
    Parâmetros:
        valores: Matriz de valores
    Retorna:
        soma: A soma dos valores"""

    # Número de linhas
    nl = len(valores)
    # O número de colunas é dado pelo tamanho de uma das linhas
    nc = len(valores[0])
    soma = 0
    for i in range(0, nl):
        for j in range(0, nc):
            soma += valores[i][j]

    return soma
```

Alocação de valores em Python

Nos próximos exercícios, utilizaremos a seguinte sintaxe para inicializar os valores de listas e matrizes:

```
n = 10
# Cria lista possuindo n valores 0
valores = [0]*n
```

```
nl = 3
nc = 8
# Cria matriz de valores 0 possuindo nl linhas e nc colunas
valores = [[0]*nc for i in range(nl)]
```

Exercício 1 - Coluna

Faça uma função que receba como entrada uma matriz e o índice de uma coluna. A função retorna uma lista contendo os elementos da coluna da matriz

Por exemplo, se a função receber como entrada o índice 1 e a matriz

$$\begin{bmatrix} 1 & 4 & 1 \\ 1 & 2 & 3 \\ 2 & 1 & 1 \end{bmatrix}$$

A saída da função será a lista [4, 2, 1]

Solução

```
def coluna(matriz, indice):
    """Retorna uma coluna da matriz.
    Parâmetros
        matriz: Matriz de valores
        indice: O índice da coluna desejada
    Retorna:
        vals_col: Coluna da matriz no índice requisitado
    """

    # Número de linhas da matriz
    nl = len(matriz)
    # Cria lista de resultado contendo nl valores 0
    vals_col = [0]*nl

    for i in range(0, nl):
        vals_col[i] = matriz[i][indice]

    return vals_col

def testes():

    matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
    print(coluna(matriz, 1))

testes()
```

Exercício 2 - Soma linhas

Faça uma função que receba como entrada uma matriz e retorne uma lista possuindo a soma de cada linha da matriz

Por exemplo:

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 3 \\ 2 & 2 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 4 \\ 6 \\ 5 \end{bmatrix}$$

Entrada:
[[1, 2, 1],[1, 2, 3],[2, 2, 1]]
Saída:
[4, 6, 5]

Solução

```
def soma_linhas(matriz):
    """Soma as linhas de uma matriz
    Parâmetros
        matriz: Matriz de valores
    Retorna:
        somas: Lista contendo as somas"""

    # Número de linhas
    nl = len(matriz)
    # Número de colunas
    nc = len(matriz[0])
    somas = [0]*nl
    for i in range(0, nl):
        # As 4 linhas abaixo somam os valores da i-ésima linha
        da matriz e armazenam o resultado em somas[i]
        soma = 0
        for j in range(0, nc):
            soma += matriz[i][j]
        somas[i] = soma

    return somas
```

Exercício 3 - Diagonal

Faça uma função que receba uma matriz e retorne True se a matriz for diagonal e False caso contrário. Considere que a matriz sempre é quadrada (mesmo número de linhas e colunas)

*Lembrando que uma matriz é diagonal se apenas os elementos da sua diagonal forem diferentes de zero. Por exemplo:

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

Solução

```
def diagonal(matriz):
    """Verifica se uma matriz é diagonal.
    Parâmetros:
        matriz: Matriz de valores
    Retorna:
        True se a matriz for diagonal e False caso contrário"""

    nl = len(matriz)
    nc = len(matriz[0])
    for i in range(0, nl):
        for j in range(0, nc):
            if i!=j and matriz[i][j]!=0:
                return False

    return True

def testes():

    # Matriz não diagonal
    matriz = [[1, 0, 1], [0, 2, 0], [0, 0, 3]]
    print(diagonal(matriz))

    # Matriz diagonal
    matriz = [[1, 0, 0], [0, 2, 0], [0, 0, 3]]
    print(diagonal(matriz))

testes()
```

Exercício 4 - Produto matricial

Faça uma função que realize o produto matricial entre duas matrizes. O produto matricial é calculado como

$$\begin{bmatrix} 2 & 3 & 1 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 1 & 0 & 2 \\ 0 & 1 & 3 & 3 \\ 2 & 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 5 & 10 & 15 \\ 4 & 3 & 7 & 10 \end{bmatrix}$$

Valor do resultado na linha 0 e coluna 2:

$$2 * 0 + 3 * 3 + 1 * 1 = 10$$

* Para criar a matriz de resultado:

```
resultado = [[0]*nc2 for i in range(nl1)]
```

onde nl1 é o número de linhas da matriz 1 e nc2 é o número de colunas da matriz 2.

Etapa 1 do exercício 4

Vamos primeiro fazer uma função que realize o cálculo de um elemento da matriz de reasultado.

Faça uma função chamada produto_listas que receba como entrada duas listas de valores e retorne o produto escalar entre as duas listas.

O produto escalar é dado pela soma dos produtos elemento-a-elemento das duas listas:

$$s = \sum_{i=0}^{n-1} l1[i] * l2[i]$$

Exemplo:

$$[2, 3, 1] * [0, 3, 1] = 2 * 0 + 3 * 3 + 1 * 1 = 10$$

Solução

```
def produto_listas(lista1, lista2):
    """Calcula a soma dos produtos dos elementos de duas listas.
    Parâmetros
        lista1: Lista de valores
        lista2: Lista de valores
    Retorna
        soma: lista1[0]*lista2[0] + lista1[1]*lista[1] + ...
    """

    n = len(lista1)
    soma = 0
    for i in range(0, n):
        soma += lista1[i]*lista2[i]

    return soma
```

Exercício 4 - Produto matricial

Faça uma função que realize o produto matricial entre duas matrizes. O produto matricial é calculado como

$$\begin{bmatrix} 2 & 3 & 1 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 1 & 0 & 2 \\ 0 & 1 & 3 & 3 \\ 2 & 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 5 & 10 & 15 \\ 4 & 3 & 7 & 10 \end{bmatrix}$$

Valor do resultado na linha 0 e coluna 2:

$$2 * 0 + 3 * 3 + 1 * 1 = 10$$

* Para criar a matriz de resultado:

```
resultado = [[0]*nc2 for i in range(nl1)]
```

onde nl1 é o número de linhas da matriz 1 e nc2 é o número de colunas da matriz 2.

Solução

```
def produto_matricial(matriz1, matriz2):
    """Realiza o produto matricial entre duas matrizes.
     ..."""

    # Tamanhos das matrizes
    nl1 = len(matriz1)
    nc1 = len(matriz1[0])
    nl2 = len(matriz2)
    nc2 = len(matriz2[0])

    if nc1!=nl2:
        print("Tamanhos incompatíveis")
        return None

    resultado = [[0]*nc2 for i in range(nl1)]
    for i in range(0, nl1):
        for j in range(0, nc2):
            vals_linha = matriz1[i]
            # Função feita em outro exercício. Retorna
            # a coluna j da matriz
            vals_col = coluna(matriz2, j)
            resultado[i][j] = produto_listas(vals_linha, vals_col)

    return resultado
```

Testes da solução

```
def testes():

    matriz1 = [[1, 2],
               [2, 1]]
    matriz2 = [[3, 1],
               [2, 2]]
    # Esperado: [[7, 5], [8, 4]]
    print(produto_matricial(matriz1, matriz2))

    # Matrizes possuindo 1 linha e 1 coluna
    matriz1 = [[2]]
    matriz2 = [[3]]
    # Esperado: [[6]]
    print(produto_matricial(matriz1, matriz2))

    # matriz1 de tamanho 1x3 e matriz2 de tamanho 3x2
    matriz1 = [[1, 2, 3]]
    matriz2 = [[3, 1],
               [2, 2],
               [0, 1]]
    # Esperado: [7, 8]
    print(produto_matricial(matriz1, matriz2))
```

Exercício 5 - Compara colunas

Faça uma função que receba uma matriz e retorne True se a matriz possuir ao menos duas colunas idênticas, e False caso contrário.

Início da solução

Podemos simplificar a função requisitada se primeiro criarmos uma função especificamente para comparar duas listas:

```
def compara_listas(lista1, lista2):
    """Compara os elementos de duas listas.
    Parâmetros
        lista1: Primeira lista
        lista2: Segunda lista
    Retorna
        True se as listas forem iguais e False
        caso contrário
    """

    n = len(lista1)
    for i in range(0, n):
        if lista1[i]!=lista2[i]:
            return False

    return True
```

Solução

```
def compara_colunas(matriz):
    """Identifica se duas colunas de uma matriz são iguais.
    Parâmetros:
        matriz: Matriz de valores
    Retorna:
        True se a matriz possuir ao menos duas colunas iguais.
        False caso contrário."""

    nl = len(matriz)
    nc = len(matriz[0])
    for ind_col1 in range(0, nc-1):
        for ind_col2 in range(ind_col1+1, nc):
            col1 = coluna(matriz, ind_col1)
            col2 = coluna(matriz, ind_col2)
            iguais = compara_listas(col1, col2)
            if iguais:
                return True

    return False
```

Testes da solução

```
def testes():

     # Duas colunas iguais
    valores = [[1, 1],
               [2, 2],
               [3, 3]]
    print(compara_colunas(valores))

    # Colunas distintas
    valores = [[1, 1],
               [2, 3],
               [3, 3]]
    print(compara_colunas(valores))

    # Uma única coluna
    valores = [[1],
               [2],
               [3]]
    print(compara_colunas(valores))

    # Várias colunas, duas iguais
    valores = [[1, 2, 3, 4, 2],
               [6, 7, 8, 9, 7],
               [2, 3, 4, 5, 3]]
    print(compara_colunas(valores))
```