BCM0505-22 – Processamento da Informação

Matrizes - Parte 1

Maycon Sambinelli
m.sambinelli@ufabc.edu.br
http://professor.ufabc.edu.br/~m.sambinelli/

Outline

Introdução

Operações Elementares

Exemplos

Exercícios

Introdução

Objetivo Didático

Apreender a manipular matrizes:

- criar uma matriz $p \times q$;
- acessar uma entrada da matriz;
- alterar uma entrada da matriz;
- carregar uma matriz a partir dos dados fornecidos pelo usuário;
- imprimir o conteúdo de uma matriz.

Matriz

Uma matriz é uma estrutura matemática que organiza elementos (geralmente números) em forma de tabela, com linhas e colunas.

$$\begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

Matriz em Python

Em Python, e na maioria das linguagens de programação, representamos uma matriz como um **vetor de vetores**.

$$M = \begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

```
1  M = [[10, 9, -2, 3],
2       [4, 5, 6, 7],
3       [2, 0, -1, 7]]
```

Matriz em Python

Em Python, e na maioria das linguagens de programação, representamos uma matriz como um **vetor de vetores**.

$$M = \begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

```
M = [[10, 9, -2, 3],
     [4, 5, 6, 7],
     [2, 0, -1, 7]]
```

Note que:

```
print(type(M))      # => <class 'list'>
print(type(M[1]))   # => <class 'list'>
print(M[1])         # => [4, 5, 6, 7]
```

Acessando uma entrada da Matriz

Dada uma "matriz do Python" `M`, podemos acessar a entrada da linha `i` e coluna `j` escrevendo `M[i][j]`.

- No restante da apresentação, diremos simplesmente que `M` é uma matriz ao invés de "Matriz do Python".

$$M = \begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

```
1  M = [[10, 9, -2, 3],
2       [4, 5, 6, 7],
3       [2, 0, -1, 7]]
```

Acessando uma entrada da Matriz

Dada uma "matriz do Python" `M`, podemos acessar a entrada da linha `i` e coluna `j` escrevendo `M[i][j]`.

- No restante da apresentação, diremos simplesmente que `M` é uma matriz ao invés de "Matriz do Python".

$$M = \begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

```
1  M = [[10, 9, -2, 3],
2       [4, 5, 6, 7],
3       [2, 0, -1, 7]]
```

- `M[0][1] -> 9`

Acessando uma entrada da Matriz

Dada uma "matriz do Python" `M`, podemos acessar a entrada da linha `i` e coluna `j` escrevendo `M[i][j]`.

- No restante da apresentação, diremos simplesmente que `M` é uma matriz ao invés de "Matriz do Python".

$$M = \begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

```
1  M = [[10, 9, -2, 3],
2       [4, 5, 6, 7],
3       [2, 0, -1, 7]]
```

- `M[0][1] -> 9`
- `M[2][3] -> 7`

Acessando uma entrada da Matriz

Dada uma "matriz do Python" `M`, podemos acessar a entrada da linha `i` e coluna `j` escrevendo `M[i][j]`.

- No restante da apresentação, diremos simplesmente que `M` é uma matriz ao invés de "Matriz do Python".

$$M = \begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

```
1  M = [[10, 9, -2, 3],
2       [4, 5, 6, 7],
3       [2, 0, -1, 7]]
```

- `M[0][1] -> 9`
- `M[2][3] -> 7`
- `M[1][2] -> 6`

Modificando entradas da Matriz

Podemos modificar uma entrada de uma matriz fazendo uma atribuição de um novo valor àquela entrada.

- `M[0][1] = 15`

$$M = \begin{bmatrix} 10 & 9 & -2 & 3 \\ 4 & 5 & 6 & 7 \\ 2 & 0 & -1 & 7 \end{bmatrix}$$

Modificando entradas da Matriz

Podemos modificar uma entrada de uma matriz fazendo uma atribuição de um novo valor àquela entrada.

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

- `M[0][1] = 15`
- `M[2][3] = -2`

Modificando entradas da Matriz

Podemos modificar uma entrada de uma matriz fazendo uma atribuição de um novo valor àquela entrada.

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

- `M[0][1] = 15`
- `M[2][3] = -2`
- `M[1][2] = M[0][1] + 2 * M[2][0]`

Modificando entradas da Matriz

Podemos modificar uma entrada de uma matriz fazendo uma atribuição de um novo valor àquela entrada.

$$M = \begin{bmatrix} 10 & 15 & -2 & 3 \\ 4 & 5 & 19 & 7 \\ 2 & 0 & -1 & -2 \end{bmatrix}$$

- `M[0][1] = 15`
- `M[2][3] = -2`
- `M[1][2] = M[0][1] + 2 * M[2][0]`

Operações Elementares

Operações elementares

- Nesse ponto vimos toda a sintaxe básica para manipulação de matrizes em Python
  - Você já conhece toda a sintaxe para construir qualquer aplicação envolvendo matrizes
- Nessa seção, vamos criar alguns funções que realizam operações elementares com matrizes para exercitar o que aprendemos

Criando uma matriz $p \times q$ inicializada com 0

Cria uma matriz de dimensão $p \times q$ cujas entradas então todas inicializadas com zero.

Criando uma matriz $p \times q$ inicializada com 0

Cria uma matriz de dimensão $p \times q$ cujas entradas então todas inicializadas com zero.

```
1  def cria_matriz_nula(nlin, ncol):
2      assert(nlin >= 1)
3      assert(ncol >= 1)
4
5      M = []
6      for i in range(nlin):
7          linha = []
8          for j in range(ncol):
9              linha.append(0)
10         M.append(linha)
11     return M
```

Criando uma matriz $p \times q$ inicializada com 0

Uma versão um pouco mais compacta

```
1  def cria_matriz_nula(nlin, ncol):
2      assert(nlin >= 1)
3      assert(ncol >= 1)
4
5      M = []
6      for i in range(nlin):
7          linha = [0] * ncol
8          M.append(linha)
9      return M
```

Imprime uma matriz `M`

Imprime uma matriz

Imprime uma matriz `M`

Imprime uma matriz

```
1  def imprime_matriz(M):
2
3      for lin in range(len(M)):
4          for col in range(len(M[lin])):
5              print(f"{M[lin][col]}\t", end="")
6          print()
```

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas
  - uma matriz 4×5 possui 20 entradas

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas
  - uma matriz 4×5 possui 20 entradas
  - uma matriz 100×500 possui 50000 entradas

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas
  - uma matriz 4×5 possui 20 entradas
  - uma matriz 100×500 possui 50000 entradas
- Pedir o valor de uma entrada por vez ao usuário torna o trabalho de entrada suscetível a erros

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas
  - uma matriz 4×5 possui 20 entradas
  - uma matriz 100×500 possui 50000 entradas
- Pedir o valor de uma entrada por vez ao usuário torna o trabalho de entrada suscetível a erros
  - O mais prático é entrar com todos os valores de uma linha da matriz em uma linha da entrada padrão, separados por espaço

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas
  - uma matriz 4×5 possui 20 entradas
  - uma matriz 100×500 possui 50000 entradas
- Pedir o valor de uma entrada por vez ao usuário torna o trabalho de entrada suscetível a erros
  - O mais prático é entrar com todos os valores de uma linha da matriz em uma linha da entrada padrão, separados por espaço
- Em vez de digitar os dados a cada execução do programa, recomenda-se utilizar o redirecionamento de entrada.

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas
  - uma matriz 4×5 possui 20 entradas
  - uma matriz 100×500 possui 50000 entradas
- Pedir o valor de uma entrada por vez ao usuário torna o trabalho de entrada suscetível a erros
  - O mais prático é entrar com todos os valores de uma linha da matriz em uma linha da entrada padrão, separados por espaço
- Em vez de digitar os dados a cada execução do programa, recomenda-se utilizar o redirecionamento de entrada.

Lendo uma matriz da entrada padrão

- Geralmente uma matriz possui muitas entradas
  - uma matriz 3×4 possui 12 entradas
  - uma matriz 4×5 possui 20 entradas
  - uma matriz 100×500 possui 50000 entradas
- Pedir o valor de uma entrada por vez ao usuário torna o trabalho de entrada suscetível a erros
  - O mais prático é entrar com todos os valores de uma linha da matriz em uma linha da entrada padrão, separados por espaço
- Em vez de digitar os dados a cada execução do programa, recomenda-se utilizar o redirecionamento de entrada.

```
python meu_programa.py < dados_matriz_grande.txt
```

Lendo vários valores de uma só vez (em Python)

- `input()` devolve uma *string* (`str`) com tudo o que o usuário digitou até o caractere `<Enter>` (`\n`)

Lendo vários valores de uma só vez (em Python)

- `input()` devolve uma *string* (`str`) com tudo o que o usuário digitou até o caractere `<Enter>` (`\n`)
- O tipo `str` tem o método `.split(c)`, que "fatia" a string nas ocorrências do caractere `c`, devolvendo um vetor com os pedaços

Lendo vários valores de uma só vez (em Python)

- `input()` devolve uma *string* (`str`) com tudo o que o usuário digitou até o caractere `<Enter>` (`\n`)
- O tipo `str` tem o método `.split(c)`, que "fatia" a string nas ocorrências do caractere `c`, devolvendo um vetor com os pedaços
  - o parâmetro `c` pode ser omitido. Neste caso, o caractere `' '` (espaço) é utilizado para fatiar a string

Lendo vários valores de uma só vez (em Python)

- `input()` devolve uma *string* (`str`) com tudo o que o usuário digitou até o caractere `<Enter>` (`\n`)
- O tipo `str` tem o método `.split(c)`, que "fatia" a string nas ocorrências do caractere `c`, devolvendo um vetor com os pedaços
  - o parâmetro `c` pode ser omitido. Neste caso, o caractere `' '` (espaço) é utilizado para fatiar a string

Lendo vários valores de uma só vez (em Python)

- `input()` devolve uma *string* (`str`) com tudo o que o usuário digitou até o caractere `<Enter>` (`\n`)
- O tipo `str` tem o método `.split(c)`, que "fatia" a string nas ocorrências do caractere `c`, devolvendo um vetor com os pedaços
  - o parâmetro `c` pode ser omitido. Neste caso, o caractere `' '` (espaço) é utilizado para fatiar a string

```
1  "uva;banana;pera".split(";") # => ['uva', 'banana', 'pera']
2  "10 12 5 -3".split()         # => ['10', '12', '5', '-3']
```

Lendo vários valores de uma só vez (em Python)

- `input()` devolve uma *string* (`str`) com tudo o que o usuário digitou até o caractere `<Enter>` (`\n`)
- O tipo `str` tem o método `.split(c)`, que "fatia" a string nas ocorrências do caractere `c`, devolvendo um vetor com os pedaços
  - o parâmetro `c` pode ser omitido. Neste caso, o caractere `' '` (espaço) é utilizado para fatiar a string

```
1 "uva;banana;pera".split(";") # => ['uva', 'banana', 'pera']
2 "10 12 5 -3".split()         # => ['10', '12', '5', '-3']
```

IMPORTANTE: Note que o método `.split` devolve um vetor de `str`!

- Se você espera um vetor de números, é necessário convertê-los!

Lendo vários valores de uma só vez (em Python) - II

- Para ler vários números separados por espaço de uma só vez, precisamos usar o método `split` **e converter** o resultado para um vetor de inteiros

```
1  entrada = input()
2  entrada_fatiada = entrada.split()
3  entrada_fatiada_convertida = []
4  for elem in entrada_fatiada:
5      entrada_fatiada_convertida.append(int(elem))  # ou float(elem)
```

Lendo vários valores de uma só vez (em Python) - II

- Para ler vários números separados por espaço de uma só vez, precisamos usar o método `split` **e converter** o resultado para um vetor de inteiros

```
entrada = input()
entrada_fatiada = entrada.split()
entrada_fatiada_convertida = []
for elem in entrada_fatiada:
    entrada_fatiada_convertida.append(int(elem))  # ou float(elem)
```

- Ou melhor:

```
entrada_fatiada = input().split() # avaliado da esq. p/ dir.
entrada_fatiada_convertida = []
for elem in entrada_fatiada:
    entrada_fatiada_convertida.append(int(elem))  # ou float(elem)
```

Lendo vários valores de uma só vez (em Python) - II

- Para ler vários números separados por espaço de uma só vez, precisamos usar o método `split` **e converter** o resultado para um vetor de inteiros

```
entrada = input()
entrada_fatiada = entrada.split()
entrada_fatiada_convertida = []
for elem in entrada_fatiada:
    entrada_fatiada_convertida.append(int(elem))  # ou float(elem)
```

- Ou melhor:

```
entrada_fatiada = input().split() # avaliado da esq. p/ dir.
entrada_fatiada_convertida = []
for elem in entrada_fatiada:
    entrada_fatiada_convertida.append(int(elem))  # ou float(elem)
```

- Ou melhor ainda....

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função
- Função também é um tipo de dado (`function`) e podemos passá-las como parâmetros!

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função
- Função também é um tipo de dado (`function`) e podemos passá-las como parâmetros!
    - Isso torna a programação bem mais versátil e genérica

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função
- Função também é um tipo de dado (`function`) e podemos passá-las como parâmetros!
  - Isso torna a programação bem mais versátil e genérica
  - Além do escopo do curso de PI

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função
- Função também é um tipo de dado (`function`) e podemos passá-las como parâmetros!
  - Isso torna a programação bem mais versátil e genérica
  - Além do escopo do curso de PI
- `vet` é um vetor (`list`)

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função
- Função também é um tipo de dado (`function`) e podemos passá-las como parâmetros!
  - Isso torna a programação bem mais versátil e genérica
  - Além do escopo do curso de PI
- `vet` é um vetor (`list`)

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função
- Função também é um tipo de dado (`function`) e podemos passá-las como parâmetros!
  - Isso torna a programação bem mais versátil e genérica
  - Além do escopo do curso de PI
- `vet` é um vetor (`list`)

A função `map` devolve um vetor construído pela aplicação de `fun` a cada elemento de `vet`

- Na verdade, `map` devolve um iterador, por isso sempre escreveremos `list(` e `)` ao redor de `map`

Lendo vários valores de uma só vez (em Python) - III

A função `map(fun, vet)` recebe dois parâmetros:

- `fun` é uma função
- Função também é um tipo de dado (`function`) e podemos passá-las como parâmetros!
  - Isso torna a programação bem mais versátil e genérica
  - Além do escopo do curso de PI
- `vet` é um vetor (`list`)

A função `map` devolve um vetor construído pela aplicação de `fun` a cada elemento de `vet`

- Na verdade, `map` devolve um iterador, por isso sempre escreveremos `list(` e `)` ao redor de `map`

```
1  vet = ["10", "25", "3", "-15", "8"]
2  list(map(int, vet)) # => [10, 25, 3, -15, 8]
```

Lendo vários valores de uma só vez - III

Se quisermos ler múltiplos inteiros inseridos em uma só linha, separados por espaços, podemos usar a combinação de `input`, `split` e `map` da seguinte maneira:

```
1  entrada = map(int, input().split())
```

- Se o usuário digitou `10 12 5 -15 3<Enter>`, então o conteúdo da variável entrada será `[10, 12, 5, -15, 3]`
  - Todas as entradas serão do tipo `int`

Lendo vários valores de uma só vez - III

Se quisermos ler múltiplos inteiros inseridos em uma só linha, separados por espaços, podemos usar a combinação de `input`, `split` e `map` da seguinte maneira:

```
1  entrada = map(int, input().split())
```

- Se o usuário digitou `10 12 5 -15 3<Enter>`, então o conteúdo da variável entrada será `[10, 12, 5, -15, 3]`
  - Todas as entradas serão do tipo `int`

Voltando ao problema original de ler uma matriz...

Lendo uma matriz

Crie a função `le_matriz(nlin, ncol)` que cria uma matriz com `nlin` linhas e `ncol` colunas preenchida de inteiros fornecidos pelo usuário. Os dados serão inseridos pelo usuário linha por linha, com os valores separados por espaços.

Lendo uma matriz

Crie a função `le_matriz(nlin, ncol)` que cria uma matriz com `nlin` linhas e `ncol` colunas preenchida de inteiros fornecidos pelo usuário. Os dados serão inseridos pelo usuário linha por linha, com os valores separados por espaços.

```
1  def le_matriz(nlin, ncol):
2      M = []
3      for i in range(nlin):
4          M.append(list(map(int, input().split())))
5      return M
6
7  p, q = list(map(int, input().split()))
8  M = le_matriz(p, q)
```

Lendo uma matriz

```
1  def le_matriz(nlin, ncol):
2      M = []
3      for i in range(nlin):
4          M.append(list(map(int, input().split())))
5      return M
6
7  p, q = list(map(int, input().split()))
8  M = le_matriz(p, q)
```

Na penúltima linha podemos ver um açúcar sintático do Python:

- Faz sentido assumir que antes de fornecer os dados da matriz o usuário digitará os valores de `p` e `q`

Lendo uma matriz

```
1  def le_matriz(nlin, ncol):
2      M = []
3      for i in range(nlin):
4          M.append(list(map(int, input().split())))
5      return M
6
7  p, q = list(map(int, input().split()))
8  M = le_matriz(p, q)
```

Na penúltima linha podemos ver um açúcar sintático do Python:

- Faz sentido assumir que antes de fornecer os dados da matriz o usuário digitará os valores de `p` e `q`
- É conveniente que ele digite os dois valores na mesma linha tbm. Digamos que o usuário digitou `3 4<Enter>`

Lendo uma matriz

```
1  def le_matriz(nlin, ncol):
2      M = []
3      for i in range(nlin):
4          M.append(list(map(int, input().split())))
5      return M
6
7  p, q = list(map(int, input().split()))
8  M = le_matriz(p, q)
```

Na penúltima linha podemos ver um açúcar sintático do Python:

- Faz sentido assumir que antes de fornecer os dados da matriz o usuário digitará os valores de `p` e `q`
- É conveniente que ele digite os dois valores na mesma linha tbm. Digamos que o usuário digitou `3 4<Enter>`
- O lado direito da penúltima linha avalia para `[3,4]`

Lendo uma matriz

```
1  def le_matriz(nlin, ncol):
2      M = []
3      for i in range(nlin):
4          M.append(list(map(int, input().split())))
5      return M
6
7  p, q = list(map(int, input().split()))
8  M = le_matriz(p, q)
```

Na penúltima linha podemos ver um açúcar sintático do Python:

- Faz sentido assumir que antes de fornecer os dados da matriz o usuário digitará os valores de `p` e `q`
- É conveniente que ele digite os dois valores na mesma linha tbm. Digamos que o usuário digitou `3 4<Enter>`
- O lado direito da penúltima linha avalia para `[3,4]`
- Ao fornecer dois valores do lado esquerdo do operador de atribuição, estamos atribuindo `3` à `p` e `4` à `q`.

Lendo uma matriz de forma avançada

```
1  def le_matriz(lin, col, func_conv):
2      M = []
3      for i in range(lin):
4          M.append(list(map(func_conv, input().split())))
5      return M
6
7  p, q = list(map(int, input().split()))
8  M = le_matriz(p, q, int) # le_matriz(p, q, float)
```

Na função acima, passamos a função de conversão como parâmetro. Essa nova função `le_matriz` é capaz de ler matrizes de `int`, `float`, `str` e etc.

- Basta o usuário fornecer a função de conversão que ele deseja aplicar aos dados.

Matrizes como Parâmetros

- Como vimos anteriormente, uma matriz em Python é um vetor cujas entradas são outros vetores

Matrizes como Parâmetros

- Como vimos anteriormente, uma matriz em Python é um vetor cujas entradas são outros vetores
- Já vimos ao longo do curso que modificações realizadas em entradas de um vetor passado como parâmetro de um função são visíveis fora da função

Matrizes como Parâmetros

- Como vimos anteriormente, uma matriz em Python é um vetor cujas entradas são outros vetores
- Já vimos ao longo do curso que modificações realizadas em entradas de um vetor passado como parâmetro de um função são visíveis fora da função
    - Como uma “matriz de Python” é na realidade um vetor, temos que o comportamento é o mesmo

Matrizes como Parâmetros

- Como vimos anteriormente, uma matriz em Python é um vetor cujas entradas são outros vetores
- Já vimos ao longo do curso que modificações realizadas em entradas de um vetor passado como parâmetro de um função são visíveis fora da função
  - Como uma “matriz de Python” é na realidade um vetor, temos que o comportamento é o mesmo
  - Usaremos esse fato no próximo exemplo

Matrizes como Parâmetros

O código abaixo cria uma função que preenche todas as entradas de uma dada matriz `M` com o valor `k`

```
1  def preenche_matriz(M, k):
2      for i in range(len(M)):
3          for j in range(len(M[i])):
4              M[i][j] = k
5
```

Veja um uso:

```
1  M = cria_matriz_nula(4, 4)
2  preenche_matriz(M, 1) # preenchemos com 1's
3  imprime_matriz(M)
```

Matrizes como Parâmetros

O código abaixo cria uma função que preenche todas as entradas de uma dada matriz `M` com o valor `k`

```
1  def preenche_matriz(M, k):
2      for i in range(len(M)):
3          for j in range(len(M[i])):
4              M[i][j] = k
5
```

Veja um uso:

```
1  M = cria_matriz_nula(4, 4)
2  preenche_matriz(M, 1) # preenchemos com 1's
3  imprime_matriz(M)
```

Receita de bolo: uma função consegue modificar as entradas de uma matriz passada por parâmetro

Exemplos

Exemplos

- Na seção anterior, cobrimos o básico da sintaxe de matriz e vimos alguns exemplos triviais
  - Na verdade, não houve nenhuma sintaxe nova, já que matrizes são vetores
- Agora vamos ver alguns exemplos menos triviais

Soma de Duas Matrizes

Crie um programa que leia duas matrizes de mesma dimensão e imprima a matriz resultante da soma dessas.

Soma de Duas Matrizes

Crie um programa que leia duas matrizes de mesma dimensão e imprima a matriz resultante da soma dessas.

```
1  def soma_matriz(M, N):
2      assert(len(M) == len(N))
3      assert(len(M[0]) == len(N[0]))
4
5      Q = cria_matriz_nula(len(M), len(M[0]))
6      for lin in range(len(M)):
7          for col in range(len(M[lin])):
8              Q[lin][col] = M[lin][col] + N[lin][col]
9      return Q
```

Soma de Duas Matrizes

Crie um programa que leia duas matrizes de mesma dimensão e imprima a matriz resultante da soma dessas.

```
1  def soma_matriz(M, N):
2      assert(len(M) == len(N))
3      assert(len(M[0]) == len(N[0]))
4
5      Q = cria_matriz_nula(len(M), len(M[0]))
6      for lin in range(len(M)):
7          for col in range(len(M[lin])):
8              Q[lin][col] = M[lin][col] + N[lin][col]
9      return Q
```

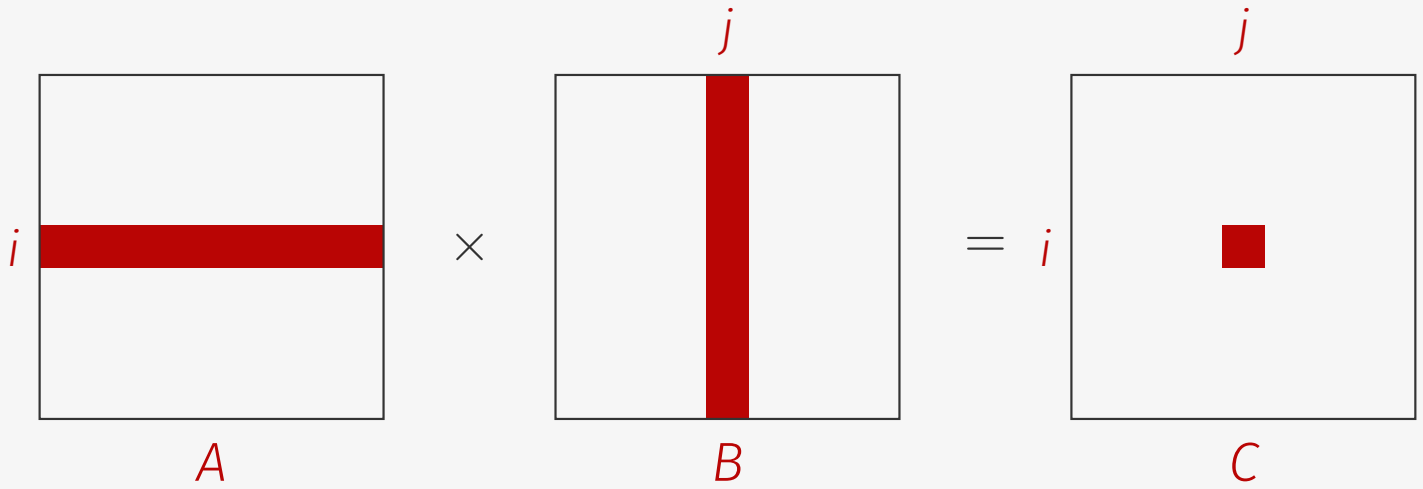
O restante do programa...

```
1  p, q = list(map(int, input().split()))
2  M = le_matriz(p, q, int)
3  N = le_matriz(p, q, int)
4  Q = soma_matriz(M, N)
5  imprime_matriz(Q)
```

Multiplicação de Matrizes Quadradas

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

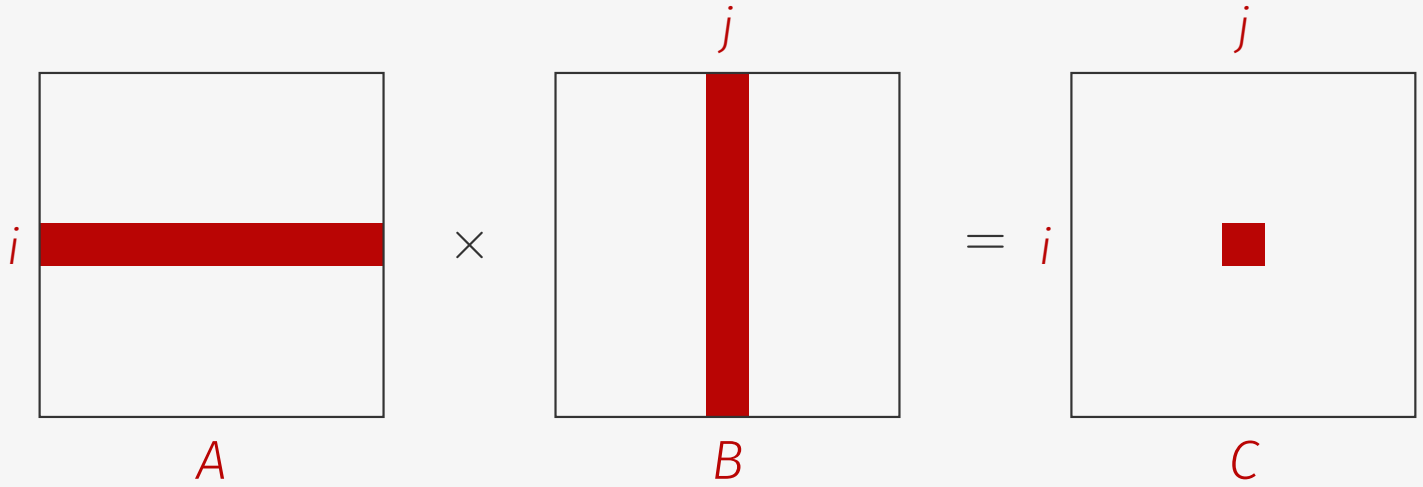
Relembrando...



Multiplicação de Matrizes Quadradas

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

Relembrando...



C_{ij} é o produto escalar da linha i de A com a coluna j de B

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Python: Multiplicação de Matrizes

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

Python: Multiplicação de Matrizes

Dadas duas matrizes A e B em $\mathbb{R}^{n \times n}$, calcular $C = A \times B$

```
1  def matriz_produto(A, B):
2      n = len(A)
3      C = cria_matriz_nula(n, n)
4      for i in range(n):
5          for j in range(n)):
6              for k in range(n):
7                  C[i][j] += A[i][k] + B[k][j]
8      return C
```

Verificar se uma dada matriz é simétrica

Escreva a função `matriz_eh_simetrica(M)` que devolve `True` se `M` é uma matriz simétrica e `False`, caso contrário.

Verificar se uma dada matriz é simétrica

Escreva a função `matriz_eh_simetrica(M)` que devolve `True` se `M` é uma matriz simétrica e `False`, caso contrário.

```
1  def matriz_eh_simetrica(M):
2      assert(len(M) == len(M[0]))
3
4      for i in range(len(M)):
5          for j in range(i + 1, len(M)):
6              if M[i][j] != M[j][i]:
7                  return False
8      return True
```

Exercícios

Criação de quaisquer matrizes

Faça uma variação da `cria_matriz_nula(p, q)`, criando uma função que recebe um parâmetro adicional `k`. Esse novo parâmetro deverá ser utilizado para inicializar as entradas da matriz criada.

Multiplicação de quaisquer matrizes

Dadas uma matriz A em $\mathbb{R}^{n \times m}$ e uma matriz B em $\mathbb{R}^{m \times p}$, calcular $C = A \times B$.
Continua valendo que C_{ij} é o produto escalar da linha i de A com a coluna j de B

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$$

Soma da Diagonal Principal

Escreva um programa que receba uma matriz quadrada e calcule a soma dos elementos da diagonal principal.
A *diagonal principal* de uma matriz é formada pelos elementos que vão do canto superior esquerdo ao canto inferior direito da matriz.

Soma da Diagonal Secundária

Escreva um programa que receba uma matriz quadrada e calcule a soma dos elementos da diagonal secundária.
A *diagonal secundária* é formada pelos elementos que vão do canto superior direito ao canto inferior esquerdo da matriz.

Preencher uma Matriz em Espiral

Escreva um programa que leia um número inteiro positivo n e preencha uma matriz quadrada de tamanho $n \times n$ com os números de 1 até n^2 em ordem crescente, seguindo o formato espiral, no sentido horário, começando do canto superior esquerdo.

Assim, se $n = 3$, o resultado deve ser

```
1  2  3
8  9  4
7  6  5
```

E se $n = 4$, o resultado deve ser

```
1   2   3   4
12  13  14  5
11  16  15  6
10   9   8  7
```

Detectar pessoas famosas

Em um conjunto de n pessoas, nomeadas de 0 a $n-1$, sabemos quem conhece quem.

Essa informação é dada por meio de uma matriz de conhecimento `conhece`, $n \times n$, em que `conhece[i][j]` é igual a `1` se a pessoa `i` conhece a pessoa `j`, e `0` caso contrário.

Faça um programa que identifique se existe alguma pessoa famosa no grupo. Uma pessoa famosa é aquela que é conhecida por todos os outros e não conhece ninguém.