Disciplina BCM0505-15
Processamento da Informação

Matrizes em Python

Profa. Carla Negri Lintzmayer

carla.negri@ufabc.edu.br

http://professor.ufabc.edu.br/~carla.negri

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Agenda

Matrizes em Python

Operações básicas

Mais exemplos

Pratique!

Matrizes em Python

Matrizes em Python

- A forma mais simples de representar uma matriz em Python é usando também **listas**.

Matrizes em Python

- A forma mais simples de representar uma matriz em Python é usando também **listas**.
- Uma matriz bidimensional nada mais é do que uma lista que contém outras listas dentro dela.

Matrizes em Python

- A forma mais simples de representar uma matriz em Python é usando também **listas**.
- Uma matriz bidimensional nada mais é do que uma lista que contém outras listas dentro dela.
- Uma matriz d-dimensional é uma lista que contém listas $d-1$-dimensionais dentro dela.

Matrizes em Python

```
1  vogais = [["a", "e", "i", "o", "u"], ["A", "E", "I", "O", "U"]]
2  print(vogais)
3  print(vogais[0])
4  print(vogais[0][0])
5
6  magico = [[2, 7, 6], [9, 5, 1], [4, 3, 8]]
```

Criando matrizes

Temos uma forma de criar matrizes (bidimensionais) em Python:

```
1  dim = input().split()
2  nlin, ncol = int(dim[0]), int(dim[1])
3  matriz = []
4  for l in range(nlin):
5      linha = []
6      for c in range(ncol):
7          linha.append(float(input()))
8      matriz.append(linha)
```

Criando matrizes

CUIDADO! A forma a seguir está MUITO errada!

```
1  dim = input().split()
2  nlin, ncol = int(dim[0]), int(dim[1])
3  matriz = [[0] * ncol] * nlin
4  for l in range(nlin):
5      for c in range(ncol):
6          matriz[l][c] = float(input())
```

Percorrendo matrizes

Temos algumas formas de percorrer as matrizes em Python:

```
1  for l in range(nlin):
2      for c in range(ncol):
3          print(matriz[l][c])
```

```
1  for l in range(len(matriz)):
2      for c in range(len(matriz[0])):
3          print(matriz[l][c])
```

```
1  for linha in matriz:
2      for elem in linha:
3          print(elem)
```

Operações básicas

Leitura

"A entrada consiste em uma linha que contém dois inteiros n e m, seguida de $n \times m$ linhas que contêm um inteiro cada."

Leitura

"A entrada consiste em uma linha que contém dois inteiros n e m, seguida de $n \times m$ linhas que contêm um inteiro cada."

```
1  dim = input().split()
2  nlin, ncol = int(dim[0]), int(dim[1])
3  matriz = []
4  for l in range(nlin):
5      linha = []
6      for c in range(ncol):
7          linha.append(int(input()))
8      matriz.append(linha)
```

Leitura

"A entrada consiste em uma linha que contém dois inteiros n e m, seguida de n linhas que contêm, cada uma, m inteiros separados por espaço."

Leitura

"A entrada consiste em uma linha que contém dois inteiros n e m, seguida de n linhas que contêm, cada uma, m inteiros separados por espaço."

```
1  dim = input().split()
2  nlin, ncol = int(dim[0]), int(dim[1])
3  matriz = []
4  for l in range(nlin):
5      linha = input().split()
6      for c in range(ncol):
7          linha[c] = int(linha[c])
8      matriz.append(linha)
```

Leitura

"A entrada consiste em uma linha que contém dois inteiros n e m, seguida de uma linha linha que contem $n \times m$ inteiros separados por espaço."

Leitura

"A entrada consiste em uma linha que contém dois inteiros n e m, seguida de uma linha linha que contem $n \times m$ inteiros separados por espaço."

```
1  dim = input().split()
2  nlin, ncol = int(dim[0]), int(dim[1])
3  tudo = input().split()
4  i = 0
5  matriz = []
6  for l in range(nlin):
7      linha = []
8      for c in range(ncol):
9          linha.append(int(tudo[i]))
10         i = i + 1
11     matriz.append(linha)
```

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

```
1 for l in range(n):
2     for c in range(m):
3         print(matriz[l][c], end=" ")
```

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

```
1 for l in range(n):
2     for c in range(m):
3         print(matriz[l][c], end=" ")
```

CERTAMENTE ERRADO

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

```
1  for l in range(n):
2      for c in range(m):
3          print(matriz[l][c], end=" ")
4      print()
```

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

```
1  for l in range(n):
2      for c in range(m):
3          print(matriz[l][c], end=" ")
4      print()
```

POSSIVELMENTE ERRADO

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

```
1  for l in range(n):
2      for c in range(m-1):
3          print(matriz[l][c], end=" ")
4      print(matriz[l][m-1])
```

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

```
1  for l in range(n):
2      for c in range(m-1):
3          print("%d " % (matriz[l][c]), end="")
4      print("%d" % (matriz[l][m-1]))
```

Escrita

"Escreva n linhas, cada uma contendo m inteiros separados por espaço."

```
1  for l in range(n):
2      print("%d" % (matriz[l][0]), end="")
3      for c in range(1, m):
4          print(" %d" % (matriz[l][c]), end="")
5      print()
```

Busca

Dada uma matriz com $n \times m$ elementos e um elemento k, descubra se k está armazenado na matriz.

```
1  def busca(M: [[int]], k: int) -> (int,int):
2      nlin = len(M)
3      ncol = len(M[0])
4      for l in range(nlin):
5          for c in range(ncol):
6              if M[l][c] == k:
7                  return l,c
8      return -1,-1
```

Mais exemplos

Matriz transposta

Faça um programa que calcule a transposta de uma matriz.

Matriz transposta

Faça um programa que calcule a transposta de uma matriz.

```
1  def gera_transposta(M: [[int]]) -> [[int]]:
2      nlin = len(M)
3      ncol = len(M[0])
4      MT = cria_matriz(ncol, nlin)
5      for l in range(nlin):
6          for c in range(ncol):
7              MT[c][l] = M[l][c]
8      return MT
```

Criando matrizes

```
1  def cria_matriz(nlin: int, ncol: int) -> [[int]]:
2      M = []
3      for l in range(nlin):
4          linha = []
5          for c in range(ncol):
6              linha.append(0)
7          M.append(linha)
8      return M
```

Criando matrizes

```python
def cria_matriz(nlin: int, ncol: int) -> [[int]]:
    M = []
    for l in range(nlin):
        linha = []
        for c in range(ncol):
            linha.append(0)
        M.append(linha)
    return M
```

```python
def cria_matriz(nlin: int, ncol: int) -> [[int]]:
    M = []
    for l in range(nlin):
        M.append([0] * ncol)
    return M
```

Matriz transposta (outra resposta)

Faça um programa que calcule a transposta de uma matriz.

```
1  def gera_transposta(M: [[int]]) -> [[int]]:
2      nlin = len(M)
3      ncol = len(M[0])
4      MT = []
5      for c in range(ncol):
6          linha = []
7          for l in range(nlin):
8              linha.append(M[l][c])
9          MT.append(linha)
10     return MT
```

Matriz identidade

Faça um programa que cria uma matriz identidade de ordem n.

Matriz identidade

Faça um programa que cria uma matriz identidade de ordem n.

```
1  def gera_identidade(n: int) -> [[int]]:
2      M = []
3      for l in range(n):
4          linha = []
5          for c in range(n):
6              if l == c:
7                  linha.append(1)
8              else:
9                  linha.append(0)
10         M.append(linha)
11     return M
```

Matriz identidade (outra solução)

Faça um programa que cria uma matriz identidade de ordem n.

Matriz identidade (outra solução)

Faça um programa que cria uma matriz identidade de ordem n.

```
1  def gera_identidade(n: int) -> [[int]]:
2      M = cria_matriz(n, n)
3      for i in range(n):
4          M[i][i] = 1
5      return M
```

Soma de duas matrizes

Faça um programa que calcula a soma de duas matrizes.

Soma de duas matrizes

Faça um programa que calcula a soma de duas matrizes.

```
1  def soma_matrizes(A: [[int]], B: [[int]]) -> [[int]]:
2      nlin = len(A) # == len(B)
3      ncol = len(A[0]) # == len(B[0])
4      C = cria_matriz(nlin, ncol)
5      for l in range(nlin):
6          for c in range(ncol):
7              C[l][c] = A[l][c] + B[l][c]
8      return C
```

Multiplicação de duas matrizes

Faça um programa que multiplique duas matrizes.

Multiplicação de duas matrizes

Faça um programa que multiplique duas matrizes.

```
1  def multiplica_matrizes(A: [[int]], B: [[int]]) -> [[int]]:
2      nlin = len(A)
3      ncol = len(B[0])
4      C = cria_matriz(nlin, ncol)
5
6      ncolA = len(A[0]) # == nlinB
7      for l in range(nlin):
8          for c in range(ncol):
9              for k in range(ncolA):
10                 C[l][c] = C[l][c] + (A[l][k] * B[k][c])
11     return C
```

Pratique!

Exercício

Faça um programa que constrói um triângulo de Pascal de ordem n.

Abaixo temos o triângulo de Pascal de ordem 6:

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5  10  10   5   1
```

Os elementos extremos de cada linha são iguais a 1. Os outros elementos são obtidos somando-se os dois elementos que aparecem imediatamente acima e à esquerda na linha anterior.

Exercício

```
1  def gera_triang_pascal(n: int) -> [[int]]:
2      T = cria_matriz(n, n)
3
4      for l in range(n):
5          T[l][0] = 1
6          T[l][l] = 1
7          for c in range(1,l):
8              T[l][c] = T[l-1][c] + T[l-1][c-1]
9      return T
```