Computação I - Python
Aula 9 - Matrizes e loop aninhado

Apresentado por: Bernardo F. Costa

Produção DCC-UFRJ

Metodologia de referência https://doi.org/10.5753/wei.2016.9683

Matriz

- Matrizes podem ser representadas por lista de listas
- Como organizar os elementos numa lista de listas?
- Organização por linhas permite acesso conhecido: **matriz[linha][coluna]**

$$matriz = \begin{bmatrix} 4 & 10 & 7.3 & 16 \\ 0.5 & 0 & 5 & 102 \\ 1 & 3.6 & 9 & 40.8 \end{bmatrix}$$

```
#organização por linhas
>>> matriz = [ [4, 10, 7.3, 16], [0.5,
0, 5, 102], [1, 3.6, 9, 40.8] ]
>>> matriz[0][1]
10
>>> matriz[1][2]
5
>>> matriz[2][3]
40.8
#organização por colunas
>>> matriz = [ [4, 0.5, 1], [10, 0,
3.6], [7.3, 5, 9], [16, 102, 40.8] ]
>>> matriz[0][1]
0.5
>>> matriz[1][2]
3.6
>>> matriz[2][3]
9
```

Matriz

- Como acessar todos os elementos de uma matriz?
  - A função abaixo usa índices para calcular a densidade de uma matriz
  - Dois índices: linhas (**i**) e colunas (**j**)
  - Dois laços: um para linha, outro para coluna
  - Laços aninhados: *loop* = *linhas* × *colunas*
  - Quatro contadores: total de elementos (**total**), quantidade de zeros (**zeros**), linhas (**i**) e colunas (**j**)

```
def densidadeMatriz1(matriz):
    '''Retorna a densidade de uma matriz, ou seja,
    a proporcao de elementos diferentes de zero.
    list(list) -> float'''
    total = 0
    zeros = 0
    for i in range(len(matriz)):
        for j in range(len(matriz[i])):
            total = total + 1
            if matriz[i][j] == 0:
                zeros = zeros + 1
    return (total - zeros)/total
```

Matriz

- Melhoria possível
  - Quantidade de elementos (**total**) pode ser calculada multiplicando-se total de linhas por colunas
  - Três contadores: quantidade de zeros (**zeros**), linhas (**i**) e colunas (**j**)

```
def densidadeMatriz2(matriz):
    '''Retorna a densidade de uma matriz, ou seja,
    a proporcao de elementos diferentes de zero.
    list(list) -> float'''
    zeros = 0
    for i in range(len(matriz)):
        for j in range(len(matriz[i])):
            if matriz[i][j] == 0:
                zeros = zeros + 1
    total = len(matriz)*len(matriz[0])
    return (total - zeros)/total
```

Matriz

- Outra melhoria possível
  - Usando **for**, não precisamos de índices para acessar elementos
  - Dois contadores: total de elementos (**total**) e quantidade de zeros (**zeros**)

```
def densidadeMatriz3(matriz):
    '''Retorna a densidade de uma matriz, ou seja,
    a proporcao de elementos diferentes de zero.
    list(list) -> float'''
    total = 0
    zeros = 0
    for linha in matriz:
        for aij in linha:
            total = total + 1
            if aij == 0:
                zeros = zeros + 1
    return (total - zeros)/total
```

Matriz

- Combinando as melhorias possíveis
  - Ficamos apenas com o contador necessário
  - Um contador: quantidade de zeros (**zeros**)

```
def densidadeMatriz4(matriz):
    '''Retorna a densidade de uma matriz, ou seja,
    a proporcao de elementos diferentes de zero.
    list(list) -> float'''
    zeros = 0
    for linha in matriz:
        for aij in linha:
            if aij == 0:
                zeros = zeros + 1
    total = len(matriz)*len(matriz[0])
    return (total - zeros)/total
```

Matriz

- Revisão
  - Matrizes podem ser representadas por lista de listas
  - Organização por linhas é mais intuitiva e preferível
    - Índice de linhas vem antes do índice de colunas
  - Precisamos de laços aninhados para acessar todos os elementos de uma matriz
  - Usando **for** podemos acessar os elementos sem índices

Autores

- João C. P. da Silva ▸ Lattes
- Carla Delgado ▸ Lattes
- Ana Luisa Duboc ▸ Lattes

Colaboradores

- Anamaria Martins Moreira ▸ Lattes
- Fabio Mascarenhas ▸ Lattes
- Leonardo de Oliveira Carvalho ▸ Lattes
- Charles Figueiredo de Barros ▸ Lattes
- Fabrício Firmino de Faria ▸ Lattes