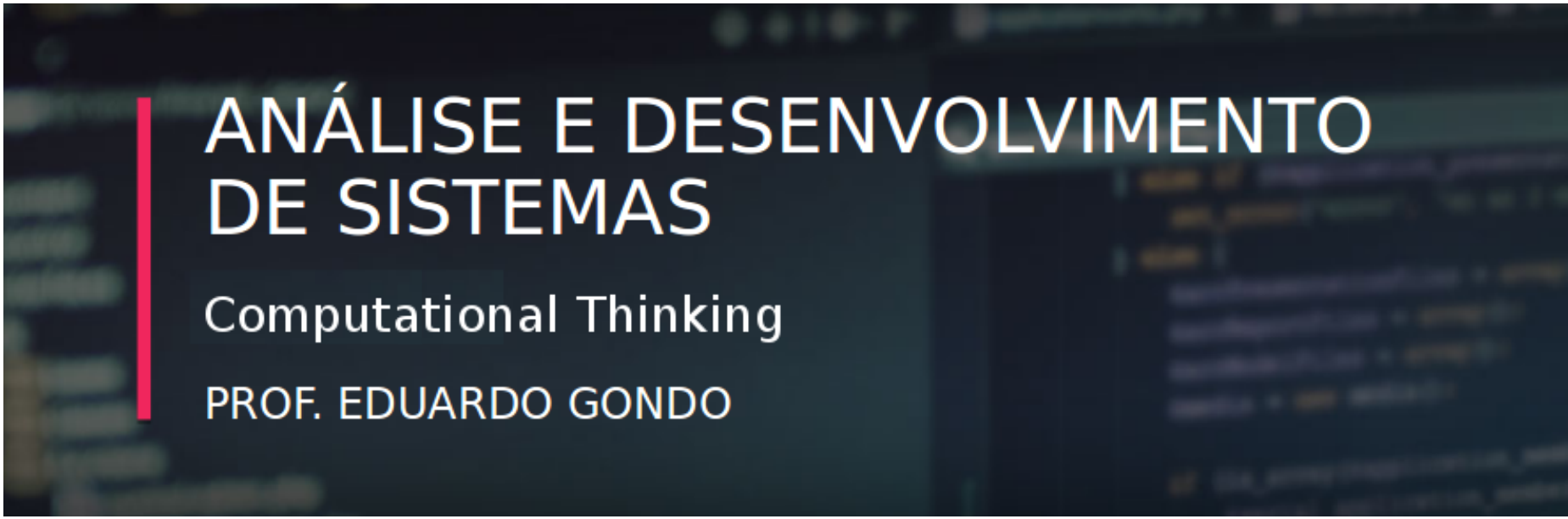


I'm trying to free your mind, Neo. But I can only show you the door. You're the one that has to walk through it.

Morpheus in Matrix (1999)

Agenda - Matrizes

- Motivação
- Definição
- Aplicações
- Utilização
- Exemplos

Motivação

Planilha eletrônica, tela de lcd, tabuleiro de dama, tabela de banco de dados são objetos usados no dia-a-dia que possuem uma característica em comum:

Motivação

Planilha eletrônica, tela de lcd, tabuleiro de dama, tabela de banco de dados são objetos usados no dia-a-dia que possuem uma característica em comum:
todos são compostos por elementos arrumados na forma de tabela, ou seja, em linhas e colunas.
No caso da planilha eletrônica os elementos são células, na tela de lcd são pixels, no tabuleiro são casas e na tabela de banco é o dado do registro.
Todos esses objetos podem ser representados, dentro da linguagem de programação, por uma **matriz**. A matriz é utilizada quando precisamos armazenar informações na forma de tabelas.

Aplicações

Dentro da área da computação podemos usar matrizes em:

visão computacional representação de imagens, rotação e expansão de imagens

programação linear muitos problemas se resumem a encontrar um solução para um sistema linear de n equações com n incógnitas

teoria dos grafos representação de grafos como matriz

criptografia uma matriz pode ser a chave no processo de criptografia

banco de dados a tabela representa o conceito de matriz

Tic-tac-toe

O Jogo da velha é um exemplo de aplicação de matriz pois seu tabuleiro é formado por uma matriz:

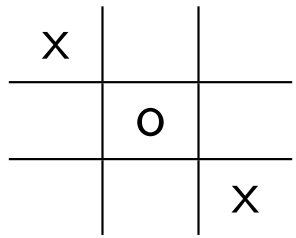


Figura: jogo da velha

vamos imaginar que nossa tarefa é construir um jogo da velha e o primeiro passo é a criação do tabuleiro (a matriz onde serão armazenadas as jogadas).

Matriz em Python - Criação e inserindo dados

Note que a matriz do nosso jogo deverá ter 3 linhas e 3 colunas e armazenar caracteres. Abaixo segue o código para sua criação na linguagem Python:

```
1  tabuleiro = []
2  i = 0
3  while i < 3:
4      tabuleiro.append(['' ] * 3)
5      i = i + 1
6
7  tabuleiro[0][0] = 'x';
8  tabuleiro[1][1] = 'o';
9  tabuleiro[2][2] = 'x';
10 for linha in tabuleiro:
11     print(linha)
```

A matriz em Python é representada como uma lista onde cada posição dela armazena uma outra lista. No próximo eslaide veremos um desenho representando a matriz do jogo da velha com seus índices.

Representação do Jogo da Velha

	0	1	2
0	x		
1		o	
2			x

Figura: matriz 3*x*3 do jogo da velha

Matriz - Colocando dados

O primeiro índice da matriz representa as linhas e o segundo as colunas, assim de acordo com o desenho abaixo, coloque as instruções em Python para criar, preencher e imprimir a matriz:

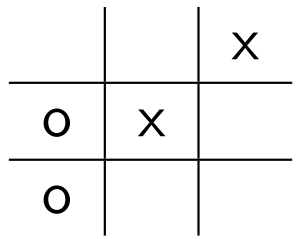


Figura: tabuleiro do jogo da velha

Matriz - Criação

Vamos ver outro exemplo de criação de uma matriz. Vamos criar uma matriz de 4 linhas e 5 colunas preenchida com zeros.

```
1  matriz = []
2  i = 0
3  while i < 4:
4      matriz.append([0] * 5)
5      i = i + 1
```

4 é o número de linhas e 5 o número de colunas, nossa matriz possui uma dimensão 4*x*5. Segue uma figura representando essa matriz:

	0	1	2	3	4
0					
1					
2					
3					

Figura: os zeros foram omitidos para não poluir o desenho

Matriz em Java - Colocando dados

- usamos o comando `while` dentro de outro comando `while` acessar as posições da matriz
- em geral usaremos índices `i` para percorrer as linhas e `j` para as colunas

PROBLEMA: Preencher uma matriz 4x5 de números inteiros com os números de 1 até 20.

Veja abaixo a imagem da matriz resultante desta operação:

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20

Figura: Matriz solução do problema

Solução em Python do Problema

Vamos tentar resolver por partes nosso problema:

- criando a matriz 4x5

```
1  matriz = []
2  i = 0
3  while i < 4:
4      matriz.append([0] * 5)
5      i = i + 1
```

- para preencher a primeira linha da matriz

```
1  num = 1
2  j = 0
3  while j < 5:
4      matriz[0][j] = num
5      num = num + 1
6      j = j + 1
```

- o primeiro argumento da matriz representa as linhas (0)
- o segundo argumento representa as colunas (j)

Solução em Python do Problema

Agora preenchendo a primeira e a segunda linha da matriz:

```
 1  num = 1
 2  j = 0
 3  while j < 5:
 4      matriz[0][j] = num
 5      num = num + 1
 6      j = j + 1
 7
 8  j = 0
 9  while j < 5:
10      matriz[1][j] = num
11      num = num + 1
12      j = j + 1
```

Como a matriz possui 4 linhas, teríamos que repetir mais duas vezes o bloco `while` apenas mudando o índice das linhas. E se a matriz tiver mais do que as 4 linhas. Como devemos proceder:

Solução em Python do Problema

A alternativa é colocar o bloco `while` dentro de um outro bloco `while`:

```
1  num = 1
2  i = 0
3  while i < 4:
4      j = 0
5      while j < 5:
6          matriz[i][j] = num
7          num = num + 1
8          j = j + 1
9      i = i + 1
```

Descobrindo as dimensões da matriz

- como as listas, precisamos saber as dimensões de uma matriz
- veja a função abaixo retornando uma tupla contendo a quantidade de linhas e colunas de uma matriz

```
1  def dimensao(matriz):
2      lin = len(matriz)
3      col = len(matriz[0])
4      return (lin, col)
```

- em Python e em outras linguagens, é possível criar matrizes com tamanho de colunas variável, ou seja, para cada linha da matriz temos um número de colunas diferentes
- na nossa disciplina trabalharemos apenas com quantidade de colunas iguais para todas as linhas

Somando um valor em todas as posições da matriz

```
1  def aumento(matriz, valor):
2      i = 0
3      while i < len(matriz):
4          j = 0
5          while j < len(matriz[i]):
6              matriz[i][j] = matriz[i][j] + valor
7              j = j + 1
8          i = i + 1
```

- note que, não precisamos retornar a matriz após a alteração
- como a matriz é uma lista, a passagem de parâmetros para a função é feita por **referência**
- ou seja, toda alteração realizada na matriz dentro da função permanece após o término dela
- escreva um programa usando essa função

Mesmo problema usando for com range

```
1  def aumento(matriz, valor):
2      for i in range(len(matriz)):
3          for j in range(len(matriz[i])):
4              matriz[i][j] = matriz[i][j] + valor
```

- o código ficou bem mais simplificado pois não é necessário inicializar as variáveis i e j
- e nem incrementá-las
- fique à vontade para escolher entre as duas opções para percorrer a matriz

Soma matriz

PROBLEMA: Escreva um método que realiza a soma de duas matrizes. Seu método recebe duas matrizes *a* e *b* de números inteiros e retorna, se possível, uma terceira matriz com a soma de *a* e de *b*.

Antes de escrevermos este método, devemos nos lembrar de como realizar tal soma. Veja abaixo um pequeno exemplo:

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

Soma matriz

PROBLEMA: Escreva um método que realiza a soma de duas matrizes. Seu método recebe duas matrizes *a* e *b* de números inteiros e retorna uma terceira matriz com a soma de *a* e de *b*.

Antes de escrevermos este método, devemos nos lembrar de como realizar tal soma. Veja abaixo um pequeno exemplo:

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

Somando em Python

Note que, a soma só funciona se as duas matrizes tiverem as mesmas dimensões. Mas apenas para efeito da lógica, vamos considerar que as matrizes possuem as mesmas dimensões:

```
1  def soma(matA, matB) {
2      resp = []
3      lin = len(matA)
4      col = len(matA[0])
5      i = 0
6      while i < lin:
7          resp.append([0] * col)
8          j = 0
9          while j < col:
10             resp[i][j] = matA[i][j] + matB[i][j]
11             j = j + 1
12         i = i + 1
13
14     return resp
```

Planilha de notas

Suponha uma matriz de números reais onde em cada linha temos armazenados as notas de um aluno. Sua tarefa é, calcular a média aritmética de cada um dos alunos cujas notas estão armazenadas na matriz.

Note que a solução deve ser sempre imaginar o que deve ser feito para um único aluno!

```
1  //função que retorna uma matriz com as notas
2  notas = Dados.getMatrizNotas();
3
4  for linha in notas:
5      soma = 0;
6      for nota in linha:
7          soma = soma + nota
8      media = soma / len(linha)
9      printf("A média do aluno é", media);
```

Nesse caso, percorremos a matriz usando o equivalente a `for-each`

Jogo da Velha

Agora, iremos fornecer as bases para montar o Jogo da Velha. Vamos pensar nas interações entre os jogadores e o jogo:

- o jogador pode efetuar uma jogada
- essa jogada pode ser feita apenas em uma posição vazia da matriz
- e também só podemos jogar se não houve um ganhador
- a matriz do jogo precisa ser impressa antes de cada jogada

Jogo da Velha

Com base no eslaide anterior, podemos montar as seguintes funções:

```
1  #retorna True se tem espaço em branco para jogar False
       caso contrário
2  def temEspaco(matriz)
3
4  #retorna True se alguém ganhou, False caso contrário
5  def haGanhador(matriz)
6
7  #retorna True se a jogada pode ser realizada
8  def joga(matriz, lin, col, jogador)
9
10 #imprime a matriz do jogo
11 def imprime(matriz)
```

Jogo da Velha - Implementação

```
1  import velha
2
3  tab = velha.criaTabuleiro()
4  player = 'X'
5  while velha.temEspaco(tab) and not velha.haGanhador(tab):
6      velha.imprime(tab)
7      print("Vez do jogador: ", jogador)
8      lin = int(input("Linha:"))
9      col = int(input("Coluna:"))
10     resp = velha.joga(tab, lin, col, player)
11     if resp == True:
12         player = velha.trocaJogador(player)
13     else:
14         print("Jogada inválida, digite outra posição")
15
16
17 player = velha.trocaJogador(player)
18 if velha.haGanhador(tab):
19     print(player + " ganhou")
20 else:
21     print("Deu velha!")
```

Jogo da Velha

Crie um arquivo velha.py com as funções do programa do eslaide anterior

Imagens

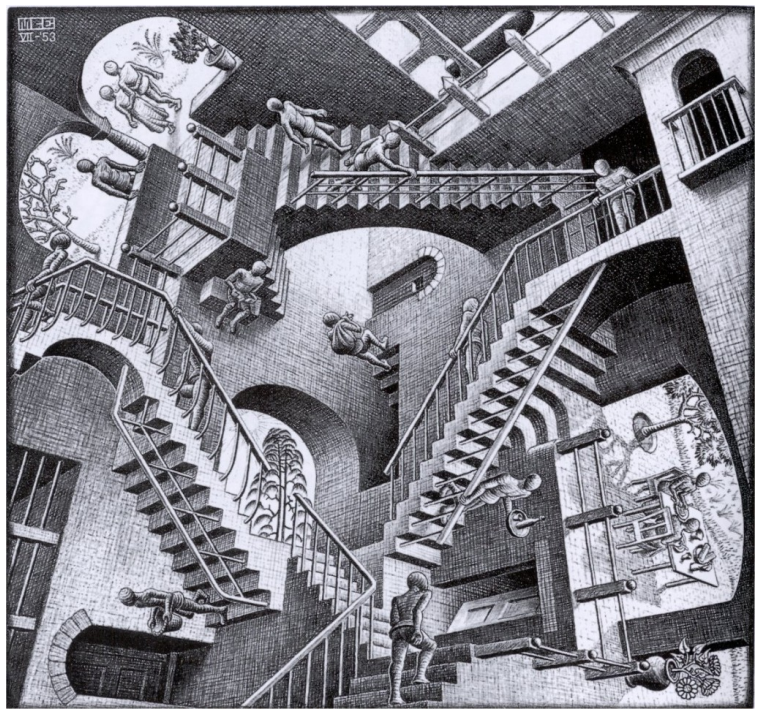
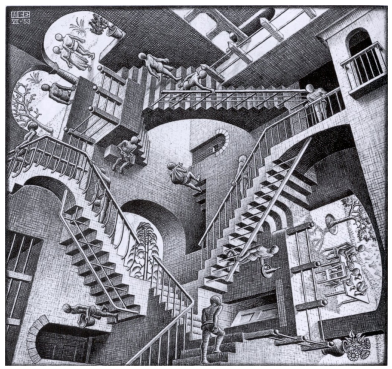


Figura: Escher

Vocês já se perguntaram o que têm por trás das imagens???

Imagens



- as imagens são representadas por uma ou mais matrizes de números inteiros
- cada posição da matriz assume, normalmente, um valor de 0 a 255
- imagens em tons de cinza são representadas por uma única matriz de inteiros
- imagens coloridas são representadas por três ou quatro matrizes de inteiros (RGB-α)

Imagens — Exemplo

$$\begin{bmatrix} 77 & 79 & 85 & 86 & 84 & 88 & 92 & 83 \\ 81 & 78 & 81 & 85 & 86 & 84 & 85 & 86 \\ 81 & 78 & 81 & 86 & 87 & 83 & 79 & 79 \\ 81 & 70 & 61 & 69 & 76 & 62 & 51 & 56 \\ 57 & 62 & 64 & 59 & 48 & 42 & 43 & 47 \end{bmatrix}$$

Figura: Matriz de números inteiros correspondente à área destacada

Imagens — manipulação

- o objetivo é trabalhar com as matrizes, o processamento de imagens será utilizado apenas como motivação
- algumas funções de programas como photoshop consiste apenas em manipular matrizes de números inteiros
- essa manipulação apenas com operações matemáticas e posicionamento dos valores dentro das matrizes, podemos manipular imagens de diversas formas:
  - transformar imagem colorida em tons de cinza
  - rotacionar uma imagem 90 graus
  - transformar uma imagem em preto e branco
  - construir uma animação que faz a imagem desaparecer (efeito Vanish)
  - efeito espelho na imagem

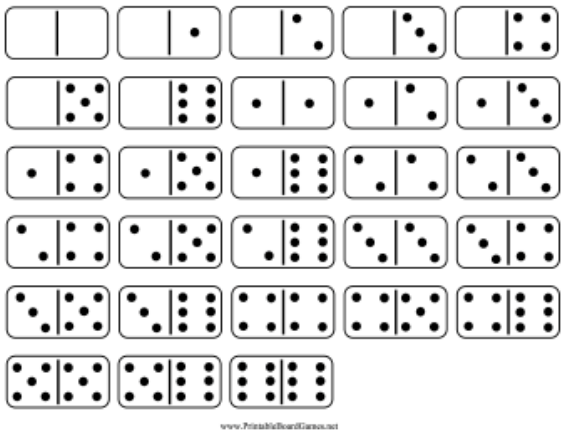
Imagens — Veja alguns exemplos:

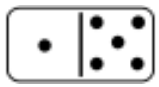
Veja algumas transformações que podemos fazer com as imagems

Figura: escurecendo metade da imagem

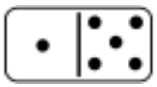
Imagens — Veja alguns exemplos:

Crop de imagem





Rotação de imagem



Imagens — Biblioteca documentação

- será necessário a instalação da biblioteca **PIL (Python Imaging Library)** para executar nossos exemplos
- você pode instalar usando o *pip*, abra um terminal (command prompt) e digite: `pip install Pillow` ou `pip3 install Pillow`
- funciona no Windows e em MAC
- agora, caso não consiga executar o pip você pode baixar a biblioteca diretamente do site `https://pypi.org/project/Pillow/2.2.1/#files`
- crie uma pasta (diretório) no seu computador e descompacte o arquivo imagensPython.zip dentro dela
- esse arquivo está disponível no Teams para download

Como ler as imagens

- criei algumas funções para ler e gravar imagens
- veja o nome das funções e uma pequena descrição do que elas fazem

```
1  #Esta função recebe como parâmetro o nome completo de uma imagem
2  #e retorna uma tupla contendo 3 matrizes representando os pixels
3  #da imagem. A primeira matriz representa a componente R seguida
4  #pela G e depois a B
5  def getMatrizImagemColorida(nome_imagem):
6
7  #Esta função recebe como parâmetro o nome completo de uma imagem
8  #e retorna uma única matriz representando os tons de cinza.
9  def getMatrizImagemCinza(nome_imagem):
10
11 #Esta função recebe como parâmetro o nome da imagem que será criada
12 #e três matrizes com os padrões RGB. No nome da imagem coloque a
13 #extensão png ou jpg.
14 def salvaImagemColorida(nome_imagem, matrizR, matrizG, matrizB):
15
16 def salvaImagemCinza(nome_imagem, matrizC):
```

Imagens — Uso das funções

Coloque os arquivos Imagem.py e as imagens fornecidas dentro de uma pasta.
Abaixo segue um exemplo de utilização da biblioteca:

```
1  import Imagem
2
3  matriz_cinza = Imagem.getMatrizImagemCinza('foto.jpg')
4
5  for i in len(matriz_cinza):
6      for j in len(matriz_cinza[i]):
7          if matriz_cinza[i][j] > 128:
8              matriz_cinza[i][j] = 255
9          else:
10             matriz_cinza[i][j] = matriz_cinza[i][j] - 64
11             if matriz_cinza[i][j] < 0:
12                 matriz_cinza[i][j] = 0
13 Imagem.salvaImagemCinza('foto2.jpg', matriz_cinza)
```

Imagens — Uso das funções

Agora, um exemplo de alteração de imagens coloridas

```
1  import Imagem
2
3  matrizes = Imagem.getMatrizImagemColorida('lago_canada.jpg
       ')
4  matrizR = matrizes[0]
5  matrizG = matrizes[1]
6  matrizB = matrizes[2]
7  for i in range(len(matrizR)):
8      for j in range(len(matrizR[i])):
9          matrizR[i][j] = matrizR[i][j] + 50
10         matrizG[i][j] = matrizG[i][j] + 50
11         matrizB[i][j] = matrizB[i][j] + 50
12
13 Imagem.salvaImagemColorida('lago_canada_claro.jpg',
       matrizR, matrizG, matrizB)
```

Exemplo — Efeito negativo

- aplicado sobre uma imagem em tons de cinza
- seja mat a matriz contendo os pixels originais da imagem
- crie a matriz negmat contendo as mesmas dimensões de mat
- para toda posição i e j de mat fazemos:

$$negmat[i][j] = 255 - mat[i][j]$$

Exemplo — Efeito Vanish

- nosso próximo exemplo, consiste em fazer uma imagem preto e branco desaparecer gradativamente
- o objetivo é encontrar um pixel branco e transformar seus vizinhos em branco também
- veja um exemplo na figura abaixo:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 255 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

antes do vanish

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 255 & 255 & 0 & 0 \\ 0 & 0 & 255 & 255 & 255 & 0 & 0 \\ 0 & 0 & 255 & 255 & 255 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

depois do vanish

Exemplo — Efeito Vanish

o algoritmo pode ser descrito em alto nível da seguinte forma

1. recuperamos a matriz de inteiros associada à imagem, chamamos de `input` essa matriz
2. criamos uma matriz `output` vazia com as mesmas dimensões da matriz `input`
3. para cada posição da matriz (i, j), fazemos

   ```
   output[i][j] = input[i][j]
   ```

4. se `output[i][j]` igual a 255, atribuímos todas as posições vizinhas de i e j com 255
5. gravamos a matriz `output` como uma imagem, atribuímos `output` a `input` e voltamos ao item 3

Referência Bibliográfica

- Puga e Rissetti - Lógica de Programação e Estrutura de Dados
- Ascêncio e Campos - Fundamentos da Programação de Computadores
- Forbelone e Eberspacher - Lógica de programação: a construção de algoritmos e estruturas de dados
- Documentação do Python - https://docs.python.org/3.8/
- Python Programming For Beginners: Learn The Basics Of Python Programming (Python Crash Course, Programming for Dummies) (English Edition). Kindle
- Python: 3 Manuscripts in 1 book: - Python Programming For Beginners - Python Programming For Intermediates - Python Programming for Advanced (English Edition). Kindle

Copyleft