

Three interlocking cubes, one purple and two green, are arranged in a row. The text 'Apresentação da Disciplina' is overlaid on them.

# Apresentação da Disciplina

# Sumário

- Sobre Mim
- Repositório *Git*
- Objetivo Geral da Matéria
- Encontros
- Ementa



# Sumário





- Conteúdo Programático
- Critérios de Avaliação
- Ferramentas
- Referências Bibliografias Obrigatórias
- Referências Bibliografias Complementares

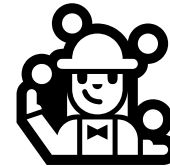
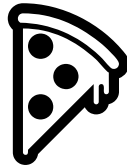


# Sobre Mim



ALEXANDRA RAIBOLT

-  Professora — UNIFESO.
-  Mestra em Ciência em Sistemas e Computação pelo Instituto Militar de Engenharia — IME.
-  Aluna de Doutorado no Programa de Pós-graduação em Engenharia de Defesa — PGEDD do Instituto Militar de Engenharia — IME.
-  Especialista Pesquisadora Plena I na Fu2re Smart Solutions.



ESTRUTURA DE DADOS E PARADIGMAS  
APRESENTAÇÃO DA DISCIPLINA

# Sobre Mim



ALEXANDRA RAIBOLT

- Contatos:

@ whoisraibolt

✉ [alexandra.raibolt@gmail.com](mailto:alexandra.raibolt@gmail.com)

[whoisraibolt.com.br](http://whoisraibolt.com.br) 🌐


[github.com/whoisraibolt](https://github.com/whoisraibolt) 💻

[linkedin.com/in/whoisraibolt](https://linkedin.com/in/whoisraibolt) 💼

[stackoverflow.com/users/8233320/whoisraibolt](https://stackoverflow.com/users/8233320/whoisraibolt) 😊




# Repositório *Git*

- [github.com/whoisraibolt/UNIFESO-CCOMP-EDP](https://github.com/whoisraibolt/UNIFESO-CCOMP-EDP)
-  Repositório com o conteúdo da disciplina **Estrutura de Dados e Paradigmas** do curso **Ciência da Computação** da **UNIFESO** — Centro Universitário Serra dos Órgãos.



# Objetivo Geral da Matéria

-  Os estudantes devem conhecer os principais **Paradigmas de Linguagens de Programação**, comparando suas características, sendo capazes de identificar as diferenças e aplicabilidade dos Paradigmas. Além disso, os estudantes devem conhecer as principais **Estruturas de Dados**, sendo capazes de identificar a estrutura mais adequada a ser utilizada para resolver problemas reais.



# Encontros

## Dia e Horário:

- **Dia:**
  - Segundas-feiras.
- **Horário:**
  - 19:00 às 22:00.
- **Local:**
  - Laboratório I.

## Feriados:

- **28/02/2022:**
  - Recesso de Carnaval.
- **13/06/2022:**
  - Santo Antônio.





# Ementa

## Paradigma de Programação Imperativo:

- Programação Estruturada/Procedural.
- Programação Orientada a Objetos.

## Paradigma de Programação Declarativo:

- Programação Funcional.
- Programação Lógica.



# Ementa

## Paradigma de Programação Concorrente e Paralela:

- Programação Concorrente.
- Programação Paralela.

## Paradigma de Programação Baseada em Eventos:

- Programação Baseada em Eventos.



# Ementa

## Estruturas de Dados Lineares:

- Listas Lineares.
- Pilhas.
- Filas.

## Estruturas de Dados Não-Lineares:

- Árvores:
  - Árvores Binárias.
  - Árvores A.V.L.
  - Árvore de Fenwick.
- Grafos.



# Ementa

Algoritmo de Divisão e  
Conquista:

- Busca Binária.

Lista de  
Prioridades:

- Heap.



# Ementa

O problema do caminho  
mais curto/mínimo:

- Algoritmo de Dijkstra.
- Algoritmo de Bellman-Ford.
- Algoritmo de Floyd-Warshall.

Estruturas de Dados  
Disjuntas:

- Algoritmo Union-Find.



# Ementa

Árvore Geradora Mínima:

Tabelas de Dispersão:

- Algoritmo de Kruskal.
- Algoritmo de Prim.
- Tabelas HASH.



# Conteúdo Programático

- Paradigmas de Linguagens de Programação:
  - Os estudantes devem conhecer os diferentes **Paradigmas de Linguagens de Programação**, comparando suas características, sendo capazes de identificar as diferenças e aplicabilidade dos **Paradigmas**.



# Conteúdo Programático

- Listas Lineares:
  - Os estudantes devem conhecer a **Estrutura de Dados** do tipo **Lista**, compreendendo seu funcionamento interno e sabendo aplicá-la na resolução de problemas.





# Conteúdo Programático

- Pilhas e Filas:
  - Os estudantes devem conhecer as **Estruturas de Dados** dos tipos **Pilhas** e **Filas**, suas diferentes formas de implementação sabendo aplica-las na resolução de problemas.



# Conteúdo Programático

- Busca Binária:

- Os estudantes devem conhecer a técnica de **Busca Binária**, compreendendo o quanto ela pode acelerar as buscas, suas limitações, formas de implementação, e saber aplicá-la na resolução de problemas reais.



# Conteúdo Programático

- Árvores Binárias:

- Os estudantes devem conhecer o conceito de **Árvores** como uma **Estrutura de Dados hierárquica**, seu uso para estruturar informações para acelerar buscas, inserção de elementos, e saber aplicá-la na resolução de problemas.



# Conteúdo Programático

- Árvores A.V.L.:
- Os estudantes devem conhecer o conceito de **Árvores Balanceadas**, compreender as rotações das **Árvores A.V.L.** e como elas mantêm uma árvore balanceada após operações de inserção de nós, para que consigam utilizá-la na resolução de problemas.



# Conteúdo Programático

- Heaps:

- Os estudantes devem conhecer o conceito de **Heaps**, compreendendo seu funcionamento para manipular filas de prioridades, sua complexidade, e como um **Heap** pode ser utilizado para ordenar elementos, através do estudo da implementação do clássico **algoritmo Heapsort**.



# Conteúdo Programático

- Avaliação 1 (AV1):
  - Avaliar o desempenho dos estudantes relativo aos objetivos educacionais da **primeira parte** do componente curricular.



# Conteúdo Programático

- Grafos e o problema do caminho mais curto:
  - Os estudantes devem conhecer os principais conceitos sobre **Grafos**, suas representações, entender o problema do caminho mais curto e saber resolvê-lo utilizando o **algoritmo de Dijkstra**.



# Conteúdo Programático

- Algoritmo de Bellman-Ford:
  - Os estudantes devem conhecer o **algoritmo de Bellman-Ford** para encontrar o caminho mais curto entre nós de um **Grafo**, compreendendo, sua vantagem em relação ao **algoritmo de Dijkstra**, e saber implementá-lo para a resolução de problemas.





# Conteúdo Programático

- Algoritmo de Floyd-Warshall:
  - Os estudantes devem conhecer o **algoritmo de Floyd-Warshall** para encontrar o caminho mais curto entre todos os pares de nós em um **Grafo**, compreendendo, suas vantagens e desvantagens em relação aos **algoritmos de Dijkstra** e **Bellman-Ford**, e saber implementá-lo para a resolução de problemas.



# Conteúdo Programático

- Estruturas de Dados Disjuntas:
  - Os estudantes devem conhecer o conceito de **Estruturas de Dados Disjuntas**, implementadas através do **algoritmo Union-Find** sabendo implementá-lo para a resolução de problemas.



# Conteúdo Programático

- Árvore Geradora Mínima - Algoritmo de Kruskal:
  - Os estudantes devem conhecer o conceito de **Árvore Geradora Mínima**, suas aplicações, e compreender o **algoritmo de Kruskal**, que usa o **método Union-Find** para resolver o problema de gerar a árvore, sabendo aplica-lo na resolução de problemas.



# Conteúdo Programático

- Árvore Geradora Mínima - Algoritmo de Prim:
  - Os estudantes devem conhecer o **algoritmo de Prim**, que usa **Heaps** para resolver o problema de gerar a árvore geradora mínima de um **Grafo**, sabendo aplicá-lo na resolução de problemas.



# Conteúdo Programático

- Tabelas de Dispersão:

- Os estudantes devem conhecer o conceito de **tabelas HASH**, ou **tabelas de dispersão**, compreendendo seu uso, complexidade, critérios para a criação de funções **HASH** e soluções para o problema das colisões, como endereçamento aberto e o uso de encadeamento, sendo capazes de aplicá-las na resolução de problemas.



# Conteúdo Programático

- Árvore de Fenwick:
  - Os estudantes devem conhecer o problema da soma de um sub-array, que pode ser resolvido através da **Árvore de Fenwick**, um algoritmo que permite resolver esse problema em tempo logarítmico  $O(\log_2 n)$ , sendo capazes de aplica-la na resolução de problemas.



# Conteúdo Programático

- Avaliação 2 (AV2):
  - Avaliar o desempenho dos estudantes relativo aos objetivos educacionais da segunda parte do componente curricular.



# Conteúdo Programático

- 2ª Chamada:

- Avaliar o desempenho dos estudantes relativo aos objetivos educacionais da **primeira** ou **segunda parte** do componente curricular, considerando a ausência do estudante em **AV1** ou **AV2**.






# Conteúdo Programático

- Reavaliação do Conhecimento:
  - Avaliar o desempenho dos estudantes relativo aos objetivos educacionais do componente curricular, considerando o **não alcance da média** necessária à aprovação em **AV1** e **AV2**.



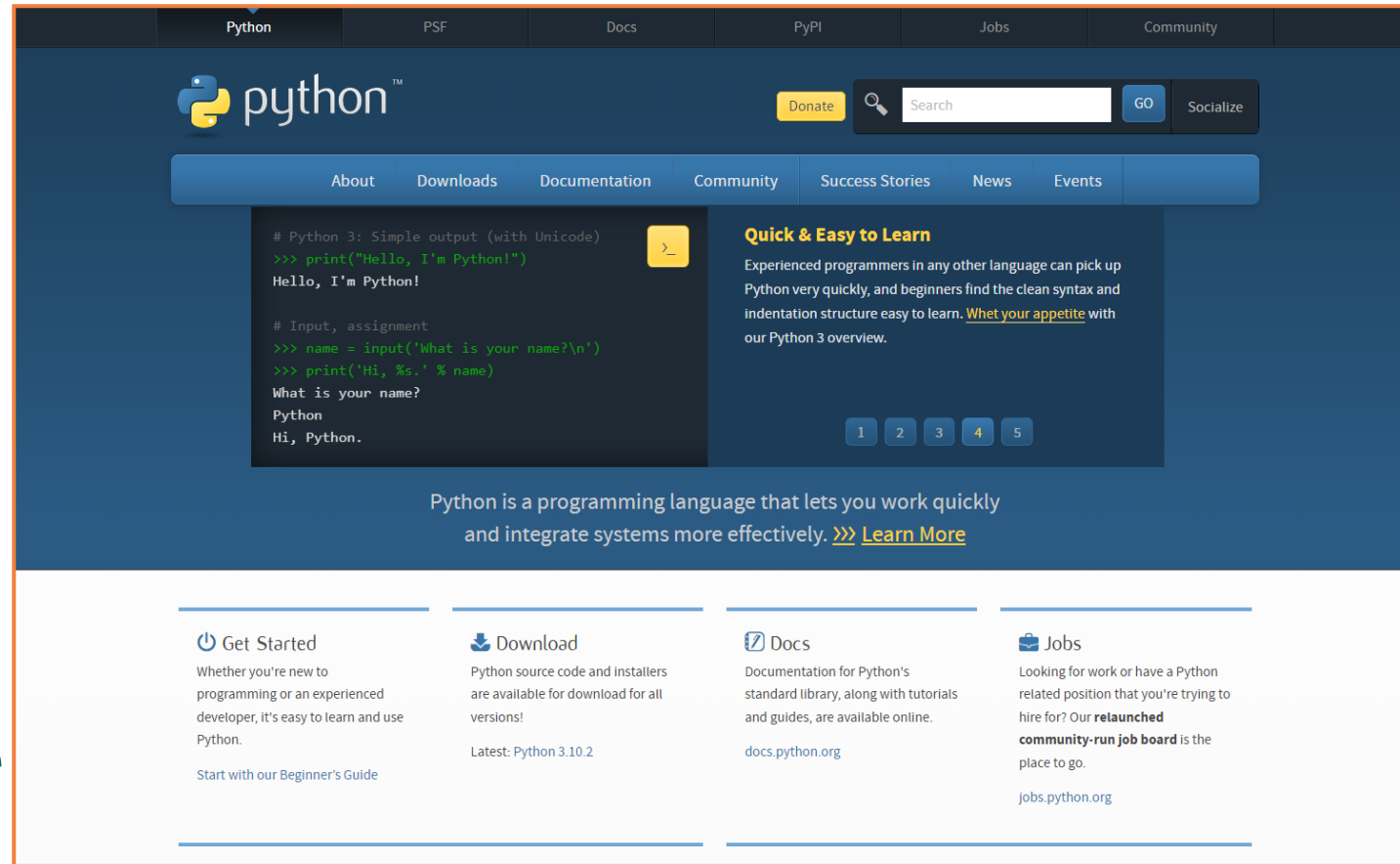
# Critérios de Avaliação

-  A nota de cada **AV** (**AV1** e **AV2**) se dará pela aplicação de um instrumento avaliativo valendo **10 pontos**, que corresponderá à **40%** da nota da respectiva **AV**. Os **60%** restantes serão compostos pelas notas das atividades apresentadas nas aulas, compostas por **exercícios práticos** e/ou **mini questionários de avaliação**.



# Ferramentas

# Python



The screenshot shows the Python.org homepage with a dark blue header and navigation bar. The main content area features a code snippet on the left, a 'Quick & Easy to Learn' section on the right, and a footer with four columns: 'Get Started', 'Download', 'Docs', and 'Jobs'.

**Navigation Bar:** Python, PSF, Docs, PyPI, Jobs, Community

**Header:** python™, Donate, Search, GO, Socialize

**Secondary Navigation:** About, Downloads, Documentation, Community, Success Stories, News, Events

**Code Snippet:**

```
# Python 3: Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!

# Input, assignment
>>> name = input('What is your name?\n')
>>> print('Hi, %s.' % name)
What is your name?
Python
Hi, Python.
```

**Quick & Easy to Learn:** Experienced programmers in any other language can pick up Python very quickly, and beginners find the clean syntax and indentation structure easy to learn. [Whet your appetite](#) with our Python 3 overview.

**Footer:**

- Get Started:** Whether you're new to programming or an experienced developer, it's easy to learn and use Python. Start with our [Beginner's Guide](#).
- Download:** Python source code and installers are available for download for all versions! Latest: [Python 3.10.2](#).
- Docs:** Documentation for Python's standard library, along with tutorials and guides, are available online. [docs.python.org](#)
- Jobs:** Looking for work or have a Python related position that you're trying to hire for? Our **relaunched community-run job board** is the place to go. [jobs.python.org](#)



ESTRUTURA DE DADOS E PARADIGMAS  
APRESENTAÇÃO DA DISCIPLINA

# Ferramentas

## Google Colab.

laboratory Features

Inserir Ambiente de execução Ferramentas Ajuda

+ Código + Texto Copiar para o Drive

### Cells

A notebook is a list of cells. Cells contain either explanatory text or executable code and its output. Click a cell to select it.

### Code cells

Below is a **code cell**. Once the toolbar button indicates CONNECTED, click in the cell to select it and execute the contents in the following ways:

- Click the **Play icon** in the left gutter of the cell;
- Type **Cmd/Ctrl+Enter** to run the cell in place;
- Type **Shift+Enter** to run the cell and move focus to the next cell (adding one if none exists); or
- Type **Alt+Enter** to run the cell and insert a new code cell immediately below it.

There are additional options for running some or all cells in the **Runtime** menu.

```
[ ] a = 10
a
10
```

### Text cells

This is a **text cell**. You can **double-click** to edit this cell. Text cells use markdown syntax. To learn more, see our [markdown guide](#).

You can also add math to text cells using [LaTeX](#) to be rendered by [MathJax](#). Just place the statement within a pair of **\$** signs. For example `$$\sqrt{3x-1}+(1+x)^2$` becomes  $\sqrt{3x-1} + (1+x)^2$ .

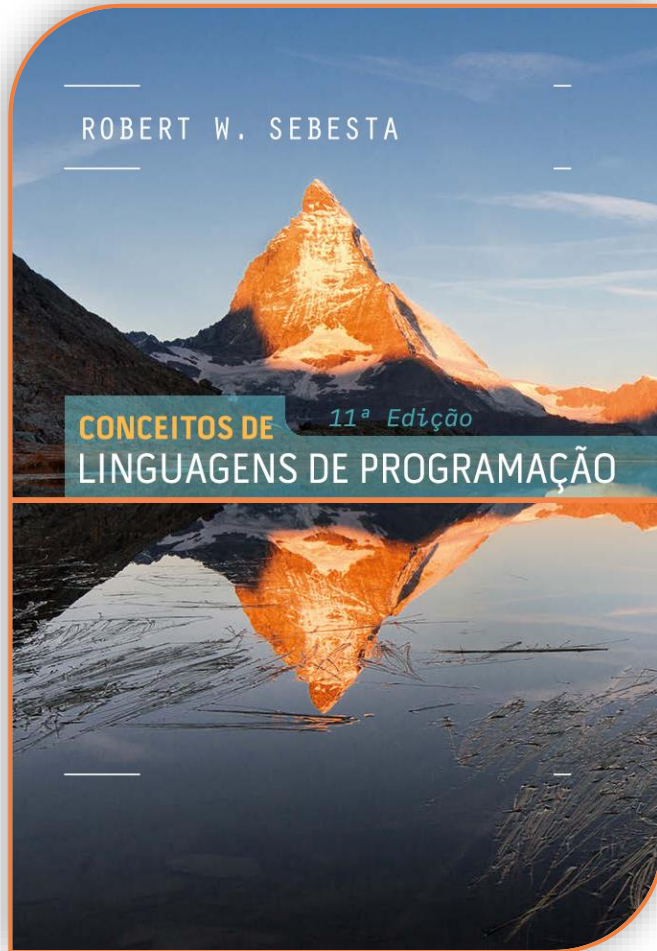
### Adding and moving cells


You can add new cells by using the **+ CODE** and **+ TEXT** buttons that show when you hover between cells. These buttons are also in the toolbar above the notebook where they can be used to add a cell below the currently selected cell.

You can move a cell by selecting it and clicking **Cell Up** or **Cell Down** in the top toolbar.



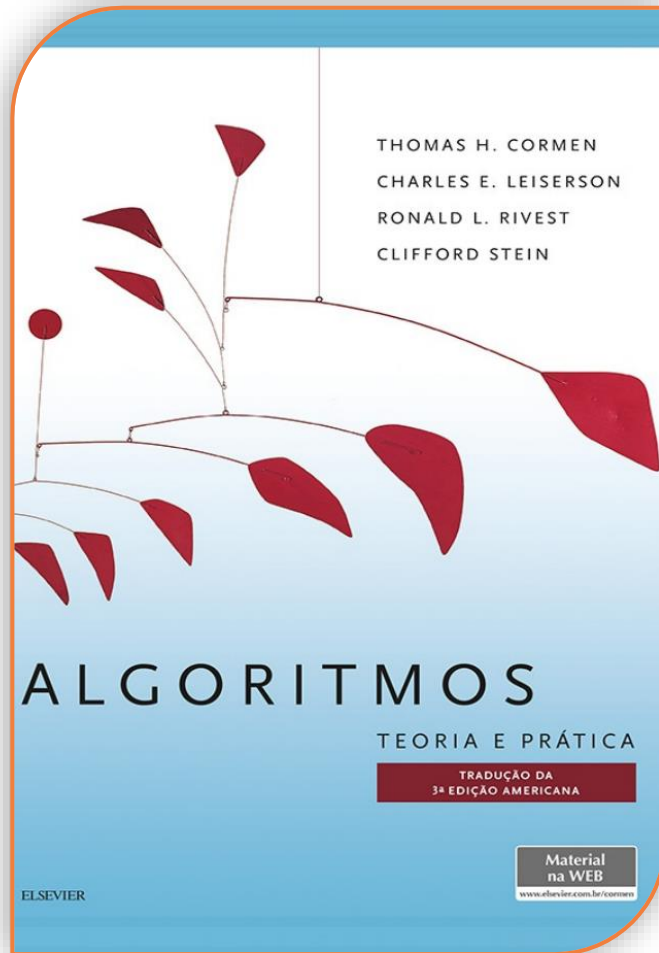
# Referências Bibliografias Obrigatórias




-  SEBESTA, R. W.  
**Conceitos de Linguagens de Programação.** 11<sup>a</sup> ed.  
Porto Alegre: Bookman,  
2018.



# Referências Bibliografias Obrigatórias

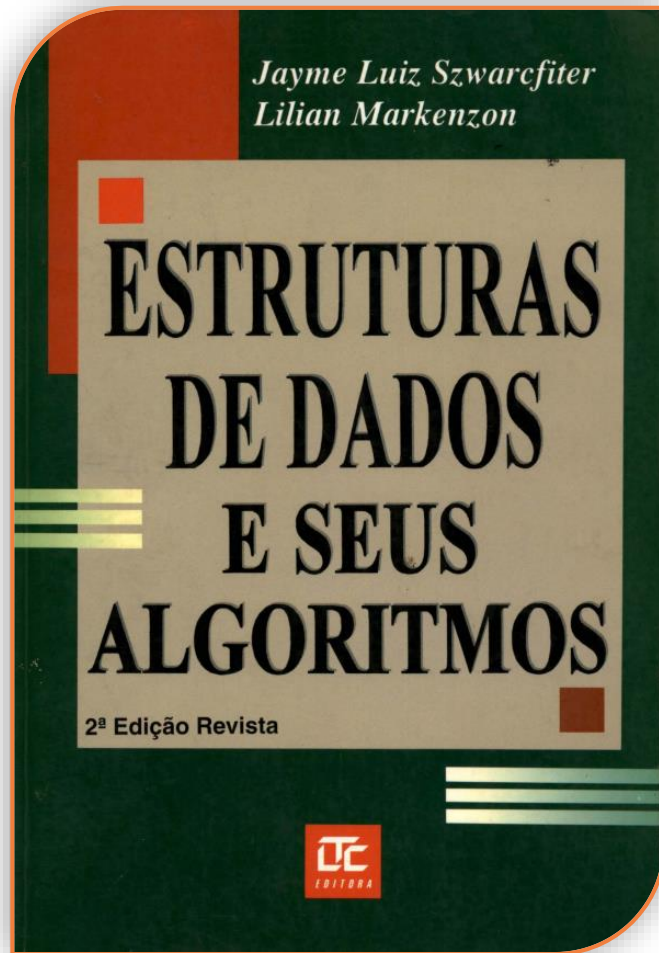


-  CORMEN, T.H. et al. **Algoritmos: Teoria e Prática**. 3ª ed. Rio de Janeiro: Elsevier, 2012.



ESTRUTURA DE DADOS E PARADIGMAS  
APRESENTAÇÃO DA DISCIPLINA

# Referências Bibliografias Complementares



-  SZWACTFITER, J.L. et al. **Estruturas de Dados e Seus Algoritmos**. 2ª ed. Editora LTC, 1997.







# Apresentação da Disciplina

Até a próxima!