EAE1106 - Métodos Computacionais para Economia

Fundamentos de Computação

Prof. Arthur Viaro

FEA-USP

Fevereiro, 2026

Plano da Aula

1. Arquivos e Diretórios
2. Interface de Linha de Comando
3. Linguagens de Programação
4. Algoritmos

Arquivos e Diretórios

- No nível do hardware, os dados são armazenados apenas como uma longa sequência de bytes (zeros e uns), sem qualquer organização visível.
- Um **sistema de arquivos** é um conjunto de regras e estruturas que organiza, gerencia e armazena informações em uma estrutura hierárquica (pastas e arquivos).
- Podemos entendê-lo como o “cérebro” do armazenamento: ele garante que os arquivos possam ser localizados e acessados de forma eficiente.
- Ele atua como um bibliotecário que cataloga livros em uma biblioteca:
  - Cada arquivo recebe um “endereço” único;
  - Quando solicitado, o sistema indica rapidamente onde ele está armazenado.

Arquivos e Diretórios

- Um **arquivo** é um conjunto de dados (texto, imagem, código, etc.) que possui um significado próprio e pode ser interpretado por um programa.

- A **extensão do arquivo** é a parte do nome que aparece após o último ponto '(.)':
  - antes do último ponto: nome do arquivo;
  - depois do último ponto: extensão (ex.: `.pdf`, `.jpg`, `.csv`, `.py`, `.ipynb`).

- O sistema operacional utiliza a extensão para associar o arquivo ao programa adequado.

- Uma **pasta (diretório)** é uma estrutura especial do sistema de arquivos que funciona como contêiner lógico para organizar arquivos e outras pastas.

Arquivos e Diretórios

- Os computadores modernos utilizam **sistemas de arquivos hierárquicos**, organizados como uma árvore de diretórios.

- Em vez de armazenar todos os arquivos no mesmo nível, criamos pastas como:
  - `Documentos`, `Imagens`, `Vídeos`, `Downloads`.

- Um arquivo pode então ser localizado por meio de um **caminho (path)**:
  - `C:\Documents\tese.docx` (Windows)
  - `/home/usuario/Documents/tese.docx` (Linux/macOS)

- O caminho é formado pelos nomes das pastas separados por \ (Windows) ou / (Linux/macOS).

Arquivos e Diretórios

Arquivos e Diretórios

- Para acessar arquivos em uma estrutura hierárquica de diretórios, utilizamos duas formas principais: caminho absoluto e caminho relativo.

```
C:\
└── projeto\
    ├── data\
    │   └── dados.csv
    ├── scripts\
    │   └── analise.py
    └── resultados\
        └── figura1.png
```

- O **caminho absoluto** descreve a localização completa do arquivo a partir da raiz do sistema: `C:\projeto\data\dados.csv` (Windows)

- O **caminho relativo** descreve a localização do arquivo em relação ao diretório atual.
  - Se estivermos no diretório `projeto`, basta usar: `data\dados.csv`

Plano da Aula

Interface de Linha de Comando

- Um nível intermediário entre o funcionamento interno do computador e as linguagens de programação é o **terminal** (ou interface de linha de comando).

- O terminal é uma interface textual que permite interagir diretamente com o sistema operacional por meio de comandos.

- Em vez de clicar em ícones ou navegar por menus gráficos, o usuário descreve explicitamente as ações que deseja executar.
  - Essa forma de interação oferece maior controle, precisão e reprodutibilidade.

- Como exemplo prático, utilizaremos o **Prompt de Comando (CMD)** do Windows.

Plano da Aula

1. Arquivos e Diretórios
2. Interface de Linha de Comando
3. Linguagens de Programação
4. Algoritmos

Linguagens de Programação

- Linguagens de programação são **linguagens formais** criadas para comunicar instruções ao computador de forma precisa e sem ambiguidades.

- Diferem das **linguagens naturais** (como português e o inglês), que surgiram naturalmente e são usadas na comunicação humana.

- Principais características:
  - são criadas para ser quase ou completamente inequívocas;
  - são menos redundantes e mais concisas;
  - têm significados exatamente iguais ao que expressam.

- São essenciais para alcançar níveis mais altos de abstração na computação.

- Toda linguagem de programação possui dois níveis fundamentais: sintaxe e semântica.

- **Sintaxe**: regras de escrita da linguagem.

```
print("Hello, World!
```

```
  Cell In[1], line 1
    print("Hello, World!
                       ^
SyntaxError: unterminated string literal (detected at line 1)
```

- **Semântica**: significado das instruções.

```
numeros = [10, 20, 30]
media = sum(numeros) / 2
print(media)
```

```
30.0
```

- As linguagens de programação diferem quanto ao seu nível de abstração.

- **Linguagens de alto nível**
  - Abstraem detalhes do funcionamento interno do computador.
  - Permitem foco na lógica do problema.
  - Mais próximas da linguagem humana.
  - Geralmente ocultam aspectos como gerenciamento de memória.
  - Exemplos: `Python`, `R`, `Java`.

- **Linguagens de baixo nível**
  - Pouca ou nenhuma abstração.
  - Próximas da linguagem de máquina e do hardware.
  - Maior controle, menor portabilidade.
  - Exemplo: `Assembly`.

- Outra distinção importante entre linguagens diz respeito à forma como o código é executado.

- **Linguagens compiladas**
  - O código é traduzido integralmente para código de máquina antes da execução.
  - Exigem uma etapa prévia de compilação (“build”).
  - Maior controle sobre hardware (memória, CPU).
  - Geralmente maior desempenho.
  - Exemplos: `C`, `C++`.

- **Linguagens interpretadas**
  - O código é executado por um intérprete, normalmente linha por linha.
  - Maior flexibilidade e facilidade de experimentação.
  - Geralmente menor desempenho bruto.
  - Exemplos: `Python`, `R`.

- Linguagens de programação podem ser classificadas segundo o paradigma, isto é, o estilo predominante de organização do código.

- **Imperativo**: descreve passo a passo o que o programa deve fazer e em qual ordem.
- **Procedural**: organiza o código em procedimentos ou funções.
- **Funcional**: enfatiza funções como elementos centrais e evita estados mutáveis.
- **Declarativo**: descreve o *que* deve ser feito, não *como*.
- **Orientado a Objetos**: estrutura o programa em objetos que combinam dados e comportamento.

- Muitas linguagens modernas, incluindo Python, são multiparadigma, permitindo combinar diferentes estilos conforme o problema.

- As linguagens diferem quanto à forma como tratam os tipos de dados.

- **Tipagem estática** (ex.: `C`, `Java`)
  - O tipo da variável é declarado explicitamente no código.
  - A verificação ocorre antes da execução (em tempo de compilação).

- **Tipagem dinâmica** (ex.: `Python`)
  - O tipo é inferido automaticamente durante a execução.
  - Não é necessário declarar previamente o tipo da variável.

- O Python é também **fortemente tipado**:
  - Não realiza conversões automáticas entre tipos incompatíveis.
  - Operações inválidas geram erro.

Plano da Aula

Algoritmos

- Um **algoritmo** é um conjunto finito e bem definido de instruções, organizadas passo a passo, para resolver um problema.

- Exemplo: encontrar um nome em uma lista telefônica.

1. Percorrer página por página até encontrar o nome.
2. Examinar múltiplas páginas por vez.
3. Abrir no meio e decidir: o nome está antes ou depois?
   - Descarta-se metade da lista.
   - Repete-se o processo na parte restante.

- Programadores transformam esses algoritmos em código, permitindo execução automática e precisa pelo computador.

Algoritmos

- O **pseudocódigo** descreve algoritmos em uma linguagem próxima da linguagem natural, mas organizada como um programa.

- Não segue a sintaxe rigorosa de uma linguagem específica e serve para:
  - Estruturar e testar a lógica antes da implementação.
  - Comunicar o raciocínio de forma clara.

- No exemplo da lista telefônica, podemos identificar blocos fundamentais da programação:
  - Condicionais
  - Expressões booleanas
  - Laços de repetição
  - Funções

Algoritmos

```
1  Pegue a lista telefônica
2  Abra no meio da lista
3  Observe a página
4  Se a pessoa estiver na página
5     Ligue para a pessoa
6  Senão, se a pessoa estiver antes
7     Abra no meio da metade esquerda
8     Volte para a linha 3
9  Senão, se a pessoa estiver depois
10       Abra no meio da metade direita
11       Volte para a linha 3
12 Senão
13       Pare
```

Algoritmos

```
1  Pegue a lista telefônica
2  Abra no meio da lista
3  Observe a página
4  Se a pessoa estiver na página
5     Ligue para a pessoa
6  Senão, se a pessoa estiver antes
7     Abra no meio da metade esquerda
8     Volte para a linha 3
9  Senão, se a pessoa estiver depois
10    Abra no meio da metade direita
11    Volte para a linha 3
12 Senão
13    Pare
```

Algoritmos

```
1  Pegue a lista telefônica
2  Abra no meio da lista
3  Observe a página
4  Se a pessoa estiver na página
5     Ligue para a pessoa
6  Senão, se a pessoa estiver antes
7     Abra no meio da metade esquerda
8     Volte para a linha 3
9  Senão, se a pessoa estiver depois
10    Abra no meio da metade direita
11    Volte para a linha 3
12 Senão
13    Pare
```

Algoritmos

```
1  Pegue a lista telefônica
2  Abra no meio da lista
3  Observe a página
4  Se a pessoa estiver na página
5     Ligue para a pessoa
6  Senão, se a pessoa estiver antes
7     Abra no meio da metade esquerda
8     Volte para a linha 3
9  Senão, se a pessoa estiver depois
10    Abra no meio da metade direita
11    Volte para a linha 3
12 Senão
13    Pare
```

O que vem pela frente?

- Primeiros Passos no Python
  - Instalação
  - IDEs
  - Bibliotecas
  - Primeiro programa