

Departamento de Engenharia de Sistemas e Computação

Implementação 2

Listas e dicionários, arquivos e tratamento de erros

João Araujo

Algoritmos Computacionais I

Aula 32

Objetivo da aula

- Completar as funcionalidades essenciais e acrescentar as desejáveis
- Uso **consistente** de listas e dicionários
- **Persistência**: gravar e carregar dados em arquivos
- **Tratamento de erros**: o programa não pode "quebrar" por culpa do usuário
- Feedback contínuo para garantir a qualidade do código

Persistência em CSV: carregar e salvar

```python
import csv

def carrega_estoque(caminho):
    """Lê o CSV e devolve o dicionário de insumos (vazio se não existir)."""
    estoque = {}
    try:
        with open(caminho, newline="", encoding="utf-8") as f:
            for linha in csv.DictReader(f):
                estoque[linha["nome"]] = {
                    "unidade": linha["unidade"],
                    "preco": float(linha["preco"]),
                    "qtd": float(linha["qtd"]),
                    "minimo": float(linha["minimo"]),
                }
    except FileNotFoundError:
        print("Arquivo não encontrado: começando com estoque vazio.")
    return estoque
```

Persistência em CSV: salvar

```python
def salva_estoque(estoque, caminho):
    """Grava o estoque em CSV."""
    campos = ["nome", "unidade", "preco", "qtd", "minimo"]
    with open(caminho, "w", newline="", encoding="utf-8") as f:
        escritor = csv.DictWriter(f, fieldnames=campos)
        escritor.writeheader()
        for nome, dados in estoque.items():
            escritor.writerow({"nome": nome, **dados})
```

- **dados *desempacota* o dicionário interno junto com o nome
- **Regra:** carregue ao iniciar e salve ao sair *e* após cada alteração importante
- Teste: salve, feche o programa, abra de novo: os dados voltaram?
- Se usam Pandas: `pd.read_csv` e `df.to_csv(caminho, index=False)`

Validando entradas

```python
def le_float(mensagem, minimo=0.0):
    """Pede um número até o usuário digitar um valor válido (>= minimo)."""
    while True:
        try:
            valor = float(input(mensagem).replace(",", "."))
        except ValueError:
            print("Digite um número, por exemplo 12.5.")
            continue
        if valor < minimo:
            print(f"O valor deve ser pelo menos {minimo}.")
            continue
        return valor

preco = le_float("Preço unitário (R$): ")
```

Uma só função de leitura segura, reutilizada em todo o programa (ver Aula 13). Faça também `le_inteiro` e `le_opcao`.

Funções que sinalizam erros

```python
def baixa_estoque(estoque, nome, qtd):
    """Retira qtd do estoque. Levanta KeyError ou ValueError."""
    if nome not in estoque:
        raise KeyError(f"Insumo '{nome}' não cadastrado")
    if qtd > estoque[nome]["qtd"]:
        raise ValueError("Estoque insuficiente")
    estoque[nome]["qtd"] -= qtd

# no menu:
try:
    baixa_estoque(estoque, nome, qtd)
    print("Saída registrada.")
except (KeyError, ValueError) as erro:
    print("Não foi possível:", erro)
```

`raise` sinaliza o problema *onde* ele é detectado; quem chama decide *como* tratar (mensagem, nova tentativa...). A função de cálculo não conversa com o usuário.

Relatórios e gráficos

```python
import matplotlib.pyplot as plt

def grafico_mais_caros(estoque, n=5, arquivo="mais_caros.png"):
    """Barras dos n insumos de maior preço unitário."""
    itens = sorted(estoque.items(), key=lambda par: par[1]["preco"],
                   reverse=True)[:n]
    nomes = [nome for nome, _ in itens]
    precos = [dados["preco"] for _, dados in itens]
    fig, ax = plt.subplots()
    ax.bar(nomes, precos)
    ax.set_ylabel("Preço unitário (R$)")
    ax.set_title(f"Os {len(nomes)} insumos mais caros")
    fig.savefig(arquivo, dpi=150, bbox_inches="tight")
    plt.close(fig)
```

A função *devolve o arquivo* em vez de abrir uma janela: assim pode ser usada sem interação e testada.

Com Pandas: análise de um CSV

```python
import pandas as pd

def resumo_por_maquina(caminho):
    """Tabela com contagem, tempo médio e taxa de defeitos por máquina."""
    df = pd.read_csv(caminho)
    df["com_defeito"] = df["status"] != "Aprovado"
    return (df.groupby("maquina")
              .agg(pecas=("id_peca", "count"),
                   tempo_medio=("tempo_ciclo_seg", "mean"),
                   taxa_defeitos=("com_defeito", "mean"))
              .round(2))

print(resumo_por_maquina("dadosproducao.csv"))
```

`agg(nome=(coluna, função))` produz colunas com nomes escolhidos por você. Separe *calcular* (esta função) de *apresentar* (`print`, gráfico, arquivo).

Erros que o programa deve tratar

Situação	Como tratar
entrada que não é número	`ValueError` → pedir de novo
opção de menu inexistente	mensagem e volta ao menu
arquivo ausente ou vazio	`FileNotFoundError`; começar vazio
linha malformada no CSV	ignorar com aviso (`try` por linha)
divisão por zero (média sem dados)	verificar antes ou `ZeroDivisionError`
chave inexistente no dicionário	`in`, `get` ou `KeyError`

Teste cada caso digitando propositalmente o valor errado.

Atividade e entrega parcial

1. Complete as funcionalidades essenciais; sem alterá-las por enquanto, deixem o "extra" para depois
2. Implemente **carregar e salvar** dados; verifique a *persistência*
3. Use funções de leitura **validadas** para toda entrada do usuário
4. Registre os **erros tratados** e os que ainda falta tratar
5. Mostre o programa ao professor: faça uma sessão de uso completa

Atenção ao prazo

Na Aula 33 o foco muda: **refatoração e testes**. Funcionalidades novas só se sobrar tempo.

Checklist de implementação 2

- ☐ Os dados persistem entre execuções?
- ☐ Toda entrada do usuário é validada?
- ☐ Arquivos são abertos com `with` e erros de arquivo são tratados?
- ☐ Listas/dicionários são usados de forma consistente (mesma estrutura em todo o código)?
- ☐ Há ao menos 1 gráfico ou relatório gerado automaticamente?
- ☐ Cálculo, entrada/saída e arquivos estão em funções/módulos separados?