

Departamento de Engenharia de Sistemas e Computação

Revisão e Exercícios

Modularização, boas práticas e estilo de código (Cap. 4 a 6 e Cap. 9)

João Araujo

Algoritmos Computacionais I

Aula 24

Roteiro

- Um pouco de estilo (Cap. 9): pensar, escrever para o leitor, simplicidade, testes
- Ferramentas: PEP 8, `pylint`, formatadores
- Testando funções: `assert`
- Revisão dos capítulos 4 a 6 e orientações para a Prova 2

9.1 Pense antes de programar

1. **Entenda** o problema; explique-o a um colega
2. **Pesquise**: já existe um caminho conhecido?
3. **Esboce** no papel: algoritmo, estruturas de dados
4. **Liste os casos-limite**: lista vazia, zero, entrada inválida
5. Escolha uma solução *razoável* e avance (o ótimo é inimigo do bom)

Brainstorming

Primeiro gere todas as ideias, *sem censura*; só depois elimine, com um critério.

9.2 Preocupe-se com seu leitor

- Código é lido muito mais vezes do que escrito: o leitor pode ser você daqui a três semanas
- **Nomes significativos** e coerentes: `media_final`, não `mf` ou `x2`
- Siga o estilo da linguagem: `snake_case` para variáveis e funções, MAIÚSCULAS para constantes
- Funções **curtas**: se não cabe na tela, divida
- Linhas com no máximo **79 caracteres**; 4 espaços de indentação
- Comente o *porquê*, não o óbvio; um bom nome dispensa o comentário

Antes e depois

Difícil de ler

```
def f(l):
    r=0
    for i in l:
        if i>=0:r=r+i
    return r/len(l)
x=[3,-1,4]
print(f(x))
```

Legível

```
def media_dos_positivos(valores):
    """Média dos valores >= 0."""
    positivos = [v for v in valores if v >= 0]
    if not positivos:
        return 0.0
    return sum(positivos) / len(positivos)

medidas = [3, -1, 4]
print(media_dos_positivos(medidas))
```

Observe: nomes claros, docstring, tratamento da lista vazia (caso-limite), e a divisão só pelos positivos (o original divide por *todos*: comportamento diferente!).

Seja simples

- Evite "truques" e códigos espertos
- Não otimize antes da hora
- Muitos comentários podem indicar código obscuro

Seja desapegado

- Código de 3 semanas atrás parece escrito por outra pessoa
- Se uma solução dá muita dor de cabeça, *recomece*
- Código ruim se apaga

Exemplo "esperto" demais

O `code/Cap5/prog.5.c_lifegame.py` faz o Jogo da Vida em uma compreensão gigante. Funciona, mas ninguém consegue ler. Reescreva-o em funções (Exercício de refatoração).

PEP 8 e ferramentas

Ferramenta	Para que serve
`pylint`	analisa o código e aponta problemas de estilo e erros prováveis (nota de 0 a 10)
`pycodestyle`	verifica a conformidade com o guia PEP 8
`yapf`, `black`	*formatam* automaticamente o arquivo

Uso

```
pip install pylint yapf
pylint programa.py      yapf -i programa.py
```

Nem todo conselho do `pylint` precisa ser seguido, mas entender cada reclamação ensina muito (Cap. 9.2).

O Zen do Python

Em um terminal Python digite `import this`:

- Belo é melhor que feio.
- Explícito é melhor que implícito.
- Simples é melhor que complexo.
- Legibilidade conta.
- Erros nunca devem passar silenciosamente.

Python foi projetado para criar bons hábitos de programação. (Cap. 9.6)

9.5 Teste seu programa: `assert`

```python
def eh_bissexto(ano):
    return (ano % 4 == 0 and ano % 100 != 0) or ano % 400 == 0

# testes: casos típicos e casos-limite
assert eh_bissexto(2024)
assert not eh_bissexto(2023)
assert not eh_bissexto(1900)     # divisível por 100, não por 400
assert eh_bissexto(2000)         # divisível por 400
print("Todos os testes passaram!")
```

- `assert condição`: se for falsa, o programa para com `AssertionError`
- Testes escritos não se esquecem; rode-os a cada modificação
- Teste *desde o início*, e pense nas condições-limite

Teste de funções com listas e strings

```python
def inverte(s):
    return s[::-1]

def media(lista):
    if not lista:
        return 0.0
    return sum(lista) / len(lista)

assert inverte('abc') == 'cba'
assert inverte('') == ''
assert media([2, 4, 6]) == 4
assert media([]) == 0.0
assert abs(media([0.1, 0.2]) - 0.15) < 1e-12   # float: com tolerância
```

Revisão: Capítulo 4 (funções e módulos)

- Funções: `def`, parâmetros, `return`, valores padrão, retorno múltiplo
- Escopo: local *versus* global; `UnboundLocalError`
- Módulos: `import`, criar o seu, `__name__ == "__main__"`
- Exceções: `try/except/else/finally`

Pergunta de prova

Escreva `le_float(msg, minimo, maximo)` que repete a leitura até o usuário digitar um número válido no intervalo.

Revisão: Capítulo 5 (listas, matrizes, dicionários, tuplas, NumPy)

- Listas: índices, fatias `[i:j:k]`, compreensão, métodos, *alias* × cópia
- Matrizes: lista de listas, o erro de `[[0]*c]*l`
- Dicionários: `d[chave]`, `get`, `items`, padrão de contagem
- Tuplas: imutáveis, desempacotamento, chaves de dicionário
- NumPy: operações vetorizadas, máscaras booleanas, `axis`

Pergunta de prova

Dado um dicionário `{'M1': [31.6, 34.5], 'M2': [24.9]}`, produza o tempo médio de cada máquina.

Revisão: Capítulo 6 (strings e arquivos)

- Strings: imutáveis, fatias, `split`/`join`, `strip`, f-strings
- Arquivos: modos `'r'`, `'w'`, `'a'`; `with`; `rstrip`; CSV

Pergunta de prova

Leia um arquivo de texto e grave em outro as linhas invertidas (última primeira), sem as linhas em branco.

Questão "leia o código"

```
s = 'engenharia'; print(s[::2], s[-3:], s.upper().count('E'))
```

Resposta: `egnai ria 2`

Orientações para a Prova 2 (Aula 25)

- **Conteúdo:** capítulos 4 a 6 (funções, módulos, exceções, estruturas de dados, strings e arquivos) + estilo (Cap. 9)
- Programas completos e funções; leitura de trechos de código
- Estude refazendo exercícios do site *sem* olhar a solução
- Treine escrever à mão: a prova não tem computador

Depois da Prova 2

Começa o projeto final (Aula 26). Pense desde já em um problema da sua área que possa ser resolvido com um programa!

Para praticar: exercícios do livro

Site do livro – Capítulo 5

- Capítulo 4: **4.7, 4.8, 4.11, 4.12, 4.13, 4.16**
- Capítulo 5: **5.7 a 5.16, 5.25, 5.26, 5.27, 5.29, 5.33**
- Capítulo 6: **6.4 a 6.9, 6.13, 6.14**

Capítulo 9 não tem exercícios no site: aplique as recomendações aos seus próprios programas.

Enunciados e soluções comentadas em
`https://www.araujo.eng.uerj.br/livro/exercicios/cap5/`

Exercícios adicionais (sem solução): `https://www.araujo.eng.uerj.br/livro/extras/`