DOMINANDO O PYTHON EM 1 ANO

Caderno de Exercícios

52 aulas de prática, do #01 ao #52

Python 3.13 · 9 módulos · 80 questões · 1 desafio integrador

Autor Desconhecido · theknowledgeisle.com

Nome do aluno: Início: Término:

Como usar este caderno

Cada módulo reúne um bloco de aulas do curso, com a lista dos artigos que ele cobre. As questões seguem a ordem dos artigos; as marcadas como desafio, no fim de cada módulo, juntam vários assuntos. Escreva e rode cada programa: o interpretador é o melhor corretor.

Tente sempre resolver sozinho antes de abrir o **Caderno de Respostas**. É no erro que o conhecimento fixa.

Sumário

Módulo	Tema	Aulas	Questões
MÓDULO 1	Fundamentos da Linguagem	#01–#06	Q1–Q10
MÓDULO 2	Estruturas de Dados e Algoritmos	#07–#12	Q11–Q20
MÓDULO 3	Orientação a Objetos	#13–#18	Q21–Q30
MÓDULO 4	Arquivos, I/O e Banco de Dados	#19–#24	Q31–Q40
MÓDULO 5	Python para Web	#25–#30	Q41–Q50
MÓDULO 6	Automação e Scripts	#31–#34	Q51–Q56
MÓDULO 7	Data Science e Machine Learning	#35–#42	Q57–Q66
MÓDULO 8	Testes, Qualidade e Boas Práticas	#43–#47	Q67–Q74
MÓDULO 9	Projetos Reais e Carreira	#48–#52	Q75–Q80
DESAFIOS	Projetos integradores	todas	D1

Legenda: `assim` indica código. As linhas pautadas são para a sua resposta; nas questões de código, use também o computador: **todo programa deve ser executado**. As dicas ajudam a começar, não entregam a resposta.

MÓDULO 1

Fundamentos da Linguagem

Artigos 01 a 06

#01 A História do Python e os Primeiros Passos
#02 Variáveis e Tipos de Dados
#03 Operadores e Expressões
#04 Estruturas de Controle: if, elif e else
#05 Laços de Repetição: for e while
#06 Funções: definição, parâmetros e escopo

Q1 Explique com suas próprias palavras o que é o Zen do Python e cite 3 de seus princípios.

Dica: Execute: python3 -c "import this"

Q2 Qual a diferença entre tipagem dinâmica e tipagem estática? Dê um exemplo em Python.

Dica: Pense em como Python atribui tipos automaticamente.

Q3 O que o código abaixo imprime? Justifique.

```
x = 10
y = 3
print(x // y)
print(x % y)
print(x ** y)
print(x / y)
```

Dica: Analise cada linha antes de executar.

Q4 Sem executar, classifique cada valor abaixo como *truthy* ou *falsy*. Depois rode e explique os que você errou.

```
0    0.0    ""    "0"    " "    []    [0]    [[]]    None    set()    "False"
float("nan")
```

Dica: Só coleções vazias, zeros, `None` e `False` são falsy.

Q5 O que o trecho abaixo imprime, e por quê? Em seguida, explique o `else` do `for` e use-o para escrever `eh_primo(n)`, listando os primos menores que 30.

```
for i in range(3):
    pass
print(i)
```

Dica: O `else` do laço roda quando ele termina sem `break`.

Q6 A função abaixo tem um dos bugs mais famosos do Python. Mostre o que acontece ao chamá-la duas vezes seguidas, explique a causa e corrija.

```
def adicionar(item, lista=[]):
    lista.append(item)
    return lista
```

Dica: Quando o valor padrão de um parâmetro é criado?

Q7 O que é escopo LEGB? Escreva um exemplo que demonstre um conflito de escopo.

Dica: L=Local, E=Enclosing, G=Global, B=Built-in

Q8 Qual a diferença entre '==' e 'is'? Quando usar cada um? Dê exemplos.

Dica: is compara identidade de objeto na memória.

Q9 Escreva uma função com *args e **kwargs que receba qualquer número de argumentos e os imprima formatados.

Dica: Explore args como tupla e kwargs como dicionário.

Q10

Implemente a sequência de Fibonacci usando recursão e compare com a versão iterativa. Qual é mais eficiente? Por quê?

MÓDULO 2

Estruturas de Dados e Algoritmos

Artigos 07 a 12

#07 Listas: criação, manipulação e métodos
#08 Tuplas e Sets: imutabilidade e unicidade
#09 Dicionários: chave, valor e as estruturas do mundo real
#10 Strings em Profundidade: métodos, formatação e expressões regulares
#11 Algoritmos de Ordenação e Busca
#12 Recursão e Estruturas de Dados Avançadas: pilhas, filas e árvores

Q11 Qual a diferença entre list.sort() e sorted()? Em quais situações você preferiria cada um?

Dica: Uma modifica a lista in-place, a outra retorna nova lista.

Q12 Reescreva o código abaixo usando list comprehension em uma única linha.

```
resultado = []
for n in range(1, 21):
    if n % 2 == 0:
        resultado.append(n ** 2)
```

Q13 O que acontece ao tentar modificar uma tupla? Por que esse comportamento é útil?

Dica: Tente: t = (1,2,3); t[0] = 10

Q14 Use um dicionário para contar a frequência de cada caractere na string abaixo.

```
frase = "python e uma linguagem incrivel"
```

Dica: Use dict.get() com valor padrão 0.

Q15 Explique com um exemplo prático a diferença entre set.remove() e set.discard().

Dica: O que acontece quando o elemento não existe?

Q16 Implemente a busca binária iterativa. Qual sua complexidade de tempo e por quê?

Dica: Divida o espaço de busca pela metade a cada iteração.

Q17 Escreva uma função que use uma pilha (list) para verificar se uma expressão tem parênteses balanceados.

Dica: Push ao abrir, pop ao fechar. Compara pares.

Q18 Escreva uma expressão regular que valide e-mails no formato usuario@dominio.com.

Dica: Use o módulo re. Considere pontos, hífens e underscores.

Q19 Explique o funcionamento do Merge Sort com um exemplo passo a passo para a lista [5, 2, 8, 1, 9].

Dica: Divisão e conquista recursiva.

Q20 Implemente uma Árvore Binária de Busca com os métodos inserir(), buscar() e percurso_em_ordem(). Demonstre com os valores [10, 5, 15, 3, 7, 20].

MÓDULO 3

Orientação a Objetos

Artigos 13 a 18

#13 Classes e Objetos: os fundamentos da Orientação a Objetos
#14 Herança e Polimorfismo
#15 Interfaces, Protocolos e Composição
#16 Decoradores e Metaprogramação
#17 Tratamento de Exceções e Erros
#18 Módulos, Pacotes e Organização de Projetos

Q21 O que é encapsulamento? Qual a diferença entre atributo com _ e com __ em Python?

Dica: Name mangling: __attr vira _Classe__attr.

Q22 Crie uma classe ContaBancaria com saldo privado, métodos depositar(), sacar() e saldo (property). Valide valores negativos.

Dica: Use @property para controle de acesso ao saldo.

Q23 O que é o método __repr__ e quando ele é chamado? Como difere de __str__?

Dica: repr() vs str() — quando cada um é usado?

Q24 Explique o princípio de Liskov e mostre um exemplo de violação com herança em Python.

Dica: Subclasse deve poder substituir a superclasse.

Q25 Implemente um decorador @medir_tempo que calcula e imprime o tempo de execução de qualquer função.

Dica: Use time.perf_counter() e @wraps.

Q26 Qual a diferença entre @classmethod e @staticmethod? Crie um exemplo com cada um na mesma classe.

Dica: classmethod recebe cls; staticmethod não recebe self nem cls.

Q27 Crie uma hierarquia de exceções customizadas para um sistema de pagamentos: ErroBase, ErroValidacao, ErroSaldoInsuficiente, ErroCartaoBloqueado.

Dica: Herde de Exception. Adicione atributos relevantes.

Q28 Quando usar composição ao invés de herança? Reescreva o exemplo abaixo usando composição.

```
class Motor(Veiculo):  # errado!
    def ligar(self): ...
    def desligar(self): ...
```

Dica: Relação 'tem um' → composição. 'é um' → herança.

Q29 Use @dataclass para criar uma classe Produto com nome, preco e estoque. Adicione um método desconto() e valide preco >= 0 em __post_init__.

Dica: from dataclasses import dataclass, field

Q30

Implemente o padrão Singleton usando um decorator. Em seguida, crie uma classe ConfiguracaoApp que use o decorator e demonstre que múltiplas instâncias são o mesmo objeto.

MÓDULO 4

Arquivos, I/O e Banco de Dados

Artigos 19 a 24

#19 Leitura e Escrita de Arquivos
#20 Trabalhando com Datas e Horas
#21 Banco de Dados com SQLite e SQLAlchemy
#22 MySQL, MariaDB, PostgreSQL e MongoDB
#23 Consumindo APIs REST com requests e httpx
#24 Concorrência: threads, processos e async/await

Q31 Qual a vantagem de usar 'with open(...)' em vez de open()/close() manualmente? O que acontece se uma exceção ocorrer?

Dica: O context manager garante fechamento mesmo com erros.

Q32 Escreva um script que leia um arquivo CSV de alunos (nome, nota) e gere um relatório JSON com média, maior e menor nota.

Dica: Use csv.DictReader e json.dump com indent=2.

Q33 O que é um datetime 'naive' vs 'aware'? Por que essa distinção importa em sistemas distribuídos?

Dica: Pense em fusos horários e servidores em diferentes regiões.

Q34 Escreva uma query SQL que liste, **para cada turma**, os 5 alunos com maior média de notas, com a posição de cada um na turma.

Dica: Com `LIMIT 5` a lista inteira para em 5 linhas; para cortar dentro de cada turma, use uma função de janela.

Q35 O que é SQL Injection? Como o uso de parâmetros (?) previne esse ataque? Dê um exemplo do ataque e da prevenção.

Dica: Nunca concatene strings para montar SQL!

Q36 Explique a diferença entre threading, multiprocessing e asyncio. Para cada um, cite um caso de uso ideal.

Dica: I/O vs CPU vs alta concorrência.

Q37 Escreva uma função assíncrona que busque dados de 5 URLs em paralelo usando asyncio.gather() e httpx.

Dica: Use async with httpx.AsyncClient() as client.

Q38 O que é o padrão Repository? Escreva uma interface (Protocol) para um repositório de alunos com métodos salvar(), buscar(), listar() e deletar().

Dica: from typing import Protocol

Q39 Qual a diferença entre MongoDB e PostgreSQL? Em que situações você escolheria cada um?

Dica: Documental vs relacional — pense em esquema e queries.

Q40 Implemente um gerenciador de contexto personalizado usando a classe __enter__/__exit__ que mede o tempo de execução de um bloco de código e registra em arquivo de log.

MÓDULO 5

Python para Web

Artigos 25 a 30

#25 Introdução ao Desenvolvimento Web com Flask
#26 FastAPI: APIs modernas com tipagem e documentação automática
#27 Autenticação e Segurança em APIs Python
#28 Templates HTML com Jinja2 e Deploy com Docker
#29 WebSockets e Comunicação em Tempo Real em Python
#30 Celery: Filas de Tarefas e Processamento Assíncrono

Q41 Quais são os princípios REST? Uma API que usa POST para tudo é RESTful? Justifique.

Dica: GET=buscar, POST=criar, PUT=substituir, PATCH=atualizar, DELETE=remover.

Q42 Crie um endpoint FastAPI GET /alunos/{id} que retorne 200 com o aluno ou 404 com mensagem de erro. Use Pydantic para o schema de resposta.

Dica: HTTPException(status_code=404, detail=...)

Q43 O que é JWT? Quais são suas três partes? Por que não armazenar dados sensíveis no payload?

Dica: Header.Payload.Signature — o payload é apenas codificado, não criptografado.

Q44 Explique o que é Dependency Injection no FastAPI. Escreva uma dependência get_db() que fornece uma sessão SQLAlchemy.

Dica: Depends() injeta a dependência automaticamente.

Q45 Qual a diferença entre Flask e FastAPI? Cite 3 vantagens de cada um.

Dica: Performance, async, documentação automática, maturidade...

Q46 O que é CORS? Por que uma API precisa configurar CORS? Escreva a configuração para permitir apenas http://localhost:3000.

Dica: Cross-Origin Resource Sharing — segurança do browser.

Q47 O que são WebSockets e em que diferem do HTTP convencional? Cite 3 casos de uso.

Dica: Full-duplex vs request-response.

Q48 Explique o que é o Celery e qual problema ele resolve. Qual a diferença entre broker e backend?

Dica: Processamento assíncrono fora do ciclo request-response.

Q49 Escreva um Dockerfile multi-stage para uma aplicação FastAPI. Explique por que multi-stage reduz o tamanho da imagem.

Dica: FROM ... AS builder / FROM ... AS runtime

Q50 Projete uma API REST completa para um sistema de biblioteca: endpoints para livros, usuários e empréstimos. Desenhe os schemas Pydantic, as rotas e o modelo de banco. Implemente autenticação JWT.

MÓDULO 6

Automação e Scripts

Artigos 31 a 34

#31 Automação com Python: arquivos, pastas e sistema operacional
#32 Web Scraping com BeautifulSoup e Selenium
#33 Automação de Planilhas e PDFs
#34 Envio de E-mails e Integração com APIs de Comunicação

Q51 Escreva um script que organize arquivos de uma pasta por extensão, movendo para subpastas (Imagens/, Documentos/, etc.). Adicione modo de simulação.

Dica: Use pathlib e shutil. simular=True só exibe o que faria.

Q52 Qual a diferença entre BeautifulSoup e Selenium para web scraping? Quando usar cada um?

Dica: HTML estático vs JavaScript dinâmico.

Q53 Escreva um script de scraping que colete título e preço dos 5 primeiros livros de books.toscrape.com.

Dica: Use requests + BeautifulSoup. Respeite robots.txt.

Q54 Crie um script com argparse que aceite --entrada (pasta), --saida (pasta) e --formato (csv ou json) e gere relatório de arquivos.

Dica: Use argparse com argumentos obrigatórios e opcionais.

Q55 Escreva o código para enviar um e-mail HTML com anexo usando smtplib. Quais cuidados de segurança são necessários?

Dica: Use MIMEMultipart, MIMEText e MIMEBase. Senha em variável de ambiente.

Q56 Crie um monitor de preços que: (1) coleta preços de 3 produtos em sites reais, (2) salva histórico em JSON, (3) envia alerta via e-mail quando o preço cai abaixo de um threshold configurável.

MÓDULO 7

Data Science e Machine Learning

Artigos 35 a 42

#35 Introdução ao Data Science com Python
#36 Machine Learning com scikit-learn
#37 Redes Neurais com TensorFlow e Keras
#38 Processamento de Linguagem Natural (NLP)
#39 Visão Computacional com OpenCV e PIL
#40 Análise de Dados com Pandas Avançado e Plotly
#41 Séries Temporais e Previsão com Prophet e statsmodels
#42 MLOps: Versionamento de Modelos, MLflow e Deploy de ML

Q57 Qual a diferença entre média, mediana e moda? Em qual situação a mediana é mais representativa do que a média?

Dica: Pense em distribuições com outliers.

Q58 O código abaixo tem um erro clássico de pandas. Identifique e corrija.

```
df_alunos = df[df['nota'] >= 6]
df_alunos['aprovado'] = True  # bug!
```

Dica: SettingWithCopyWarning — use .loc para atribuição.

Q59 Explique overfitting e underfitting. Como cross-validation ajuda a detectar overfitting?

Dica: Bias-variance tradeoff.

Q60 O que é TF-IDF? Por que é superior a simples contagem de palavras para classificação de texto?

Dica: TF=frequência no doc, IDF=raridade no corpus.

Q61 Escreva um pipeline scikit-learn completo: StandardScaler + RandomForestClassifier + GridSearchCV para um dataset de classificação.

Dica: Pipeline([('scaler', ...), ('clf', ...)]) encadeia pré-processamento e modelo.

Q62 O que é a Curva ROC e o AUC? Por que AUC é melhor que acurácia para datasets desbalanceados?

Dica: Considera todos os limiares de classificação.

Q63 Qual a diferença entre Batch Normalization e Dropout em redes neurais? Quando usar cada um?

Dica: BN normaliza ativações; Dropout desativa neurônios aleatoriamente.

Q64 O que é Transfer Learning? Explique Fine-tuning. Quando faz sentido congelar as camadas da base?

Dica: Modelos pré-treinados como MobileNet, BERT, GPT-2.

Q65 O que é MLflow? Cite 5 coisas que ele rastreia automaticamente. O que é o Model Registry?

Dica: Experimentos, parâmetros, métricas, artefatos, modelos.

Q66

Crie um pipeline completo de análise de sentimentos: (1) escreva 20 reviews de produtos, metade positivas e metade negativas; (2) pré-processe o texto (minúsculas, sem acentos, sem pontuação e sem *stopwords*); (3) vetorize com TF-IDF; (4) treine uma regressão logística; (5) avalie com F1 e matriz de confusão. O que o resultado diz sobre o tamanho do conjunto de dados?

MÓDULO 8

Testes, Qualidade e Boas Práticas

Artigos 43 a 47

#43 Testes com pytest: fixtures, mocks e cobertura
#44 Qualidade de Código: linting, type hints e mypy
#45 CI/CD com GitHub Actions
#46 Logging, Monitoramento e Observabilidade
#47 Design Patterns em Python

Q67 O que é o padrão AAA em testes? Escreva um teste pytest para a função calcular_media() que testa o caso de lista vazia.

Dica: Arrange / Act / Assert.

Q68 O que são fixtures no pytest? Qual a diferença de escopo function vs session? Escreva um exemplo.

Dica: scope='function' recria a cada teste; 'session' cria uma vez.

Q69 O que é Mock? Escreva um teste que usa mocker.patch para simular o envio de e-mail em uma função de cadastro.

Dica: Isola dependências externas dos testes unitários.

Q70 Escreva 5 type hints diferentes usando recursos modernos do Python 3.10+: union com |, list[] sem List, dict[] sem Dict.

Dica: from __future__ import annotations pode ajudar.

Q71 O que detecta cada ferramenta: ruff, black, mypy e bandit? Qual a ordem recomendada para rodar em um pipeline CI?

Dica: Linting, formatação, tipos, segurança.

Q72 Escreva um workflow GitHub Actions que rode em pull_request: instale dependências, execute ruff, mypy e pytest com cobertura.

Dica: jobs > steps > run.

Q73 O que são os três pilares da observabilidade? Qual a diferença entre log estruturado (JSON) e log em texto puro?

Dica: Logs, métricas e traces.

Q74

Implemente os padrões Strategy, Observer e Command em um sistema de processamento de pedidos: Strategy para cálculo de frete, Observer para notificações, Command para histórico com undo.

MÓDULO 9

Projetos Reais e Carreira

Artigos 48 a 52

#48 Projeto 1: API REST Completa com FastAPI, PostgreSQL e Docker
#49 Projeto 2: Pipeline de Data Science do Zero ao Deploy
#50 Projeto 3: Bot de Automação com Scraping e Notificações
#51 Projeto 4: Dashboard Analítico com Dash e Banco de Dados Real
#52 Carreira: portfólio, GitHub, open source e o mercado Python

Q75 Projete a arquitetura de uma API REST de e-commerce: entidades, endpoints principais, autenticação e banco de dados.

Dica: Pense em Produto, Carrinho, Pedido, Usuário.

Q76 O que é o padrão Repository e por que ele facilita testes? Qual a diferença entre Repository e DAO?

Dica: Repository fala a linguagem do domínio; DAO fala SQL.

Q77 Escreva um README.md profissional para um projeto fictício 'task-api'. Inclua badges, instalação, uso e arquitetura.

Dica: Siga a estrutura do artigo 52: badges, demo, quick start, arquitetura.

Q78 Quais são as 5 perguntas técnicas mais comuns em entrevistas Python? Responda cada uma em 2-3 frases.

Dica: GIL, mutabilidade, decoradores, generators, *args/**kwargs.

Q79 Implemente o algoritmo Two Sum em Python com complexidade O(n) usando hash map. Explique a solução.

Dica: Para cada número, verifique se o complemento já foi visto.

Q80

1. FastAPI com CRUD completo (Criar, Listar, Atualizar, Deletar tarefas)
2. PostgreSQL via SQLAlchemy com migrations Alembic
3. Autenticação JWT (registro, login, refresh token)
4. Testes com pytest cobrindo todos os endpoints (mínimo 80%)
5. Dockerfile + docker-compose com PostgreSQL
6. GitHub Actions executando lint + testes a cada push
7. README.md profissional com instruções de instalação e uso

Projete e implemente um sistema completo de gestão de tarefas (TODO API) com:

Este é o exercício que vai para o seu portfólio.

PARA IR ALÉM

Desafios integradores

Projetos maiores, que juntam vários módulos. Não há uma única resposta certa.

D1

Uma calculadora, duas interfaces

Escreva uma calculadora com sete operações: soma, subtração, multiplicação, divisão, potência (expoente inteiro), porcentagem (`15 % 200` é 15% de 200) e raiz quadrada. Ela deve: (a) fazer as contas com `decimal.Decimal`, para que `0,1 + 0,2` dê exatamente `0,3`, aceitando vírgula ou ponto e limitando o resultado a 16 dígitos significativos; (b) guardar o último resultado em `ans`, que pode ser usado na conta seguinte; (c) recusar divisão por zero, raiz de negativo, expoente fracionário e texto que não é número com uma exceção própria e uma mensagem clara, **sem encerrar o programa**. Escreva o núcleo num módulo que não sabe nada de entrada e saída, e reaproveite-o em duas interfaces: uma de **terminal** (com os comandos `hist` e `sair`) e uma **gráfica** com `tkinter`, com teclado na tela, visor e histórico.

Sua solução (arquitetura, decisões e código principal):

Bom estudo e bons exercícios!