TÉCNICO DE INFORMÁTICA - DE 2024

# PYTHON

EXPLORANDO, CODIFICANDO E SUPERANDO - JORNADA DE ESTUDO PARA DOMINAR OS PRINCÍPIOS DAS ESTRUTURAS EM PYTHON.

MAYSA ARRUDA

***Explorando Cardinalidades em Bancos de Dados***

*Iniciamos uma empolgante jornada de estudo focada na maestria dos fundamentos da linguagem de programação Python. Este projeto é meticulosamente elaborado para proporcionar uma imersão prática e desafiadora, centrada no exercício e aprimoramento das habilidades em Python.*

*Nossa abordagem, concentrada no entendimento profundo da linguagem, será acompanhada por uma série de exercícios estruturados. Estes desafios práticos visam não apenas consolidar os conceitos fundamentais, mas também desenvolver a proficiência necessária para aplicar eficazmente o Python em contextos específicos de bancos de dados.*

*Ao avançarmos neste projeto, antecipamos uma experiência dinâmica e envolvente, na qual cada exercício servirá como um degrau para o domínio crescente de Python. Esteja preparado(a) para mergulhar nos exercícios, explorar casos de uso reais e, acima de tudo, desafiar-se a alcançar novos patamares de competência em programação Python. Vamos iniciar essa jornada rumo à excelência técnica e ao domínio da linguagem Python!*

*Maysa Arruda*

*GitHub:Maysa502 (https://github.com/Maysa502)*

*“Era necessário deixar-se iluminar, é na clareza do vazio que vemos o tamanho real do que somos”*

*Time after time*

# Jornada Python: Desafios de Algoritmos

## *Python: Uma Visão Geral*

Python foi concebido por Guido van Rossum durante um projeto no Instituto de Pesquisa CWI, na Holanda, no final dos anos 80. A primeira versão, Python 0.9.0, foi lançada em fevereiro de 1991. A linguagem evoluiu ao longo dos anos, passando por diversas versões, com destaque para as séries Python 2.x e Python 3.x. O foco na legibilidade, simplicidade e filosofia "Zen of Python" (um conjunto de princípios orientadores para escrever código Python) contribuíram para a popularidade contínua da linguagem.

**Características Principais:**

1. **Legibilidade:**
    - A filosofia de design enfatiza a clareza e a legibilidade do código.
    - O uso de indentação em vez de chaves torna a estrutura do código visualmente mais intuitiva.
2. **Dinâmica:**
    - Python é dinâmico, permitindo que as variáveis sejam reatribuídas a diferentes tipos durante a execução.
    - Não é necessário declarar explicitamente o tipo de uma variável, simplificando a escrita do código.
3. **Multiparadigma:**
    - Suporta programação orientada a objetos, onde tudo é um objeto.
    - Oferece elementos da programação imperativa e funcional, proporcionando flexibilidade no estilo de programação.
4. **Interpretada:**

- Python é uma linguagem interpretada, o que significa que o código-fonte é executado diretamente, sem a necessidade de compilação prévia.
- Facilita o desenvolvimento e a depuração, tornando o ciclo de desenvolvimento mais rápido.

5. **Biblioteca Padrão Rica:**
    - Inclui uma extensa biblioteca padrão que abrange áreas como manipulação de strings, operações matemáticas, desenvolvimento web, entrada/saída, entre outras.
    - A comunidade Python contribui constantemente para o ecossistema, criando módulos e pacotes adicionais.

**Aplicações e Reconhecimento:** Python é amplamente utilizado em diversas áreas, desde desenvolvimento web e automação de tarefas até ciência de dados e inteligência artificial. Sua comunidade ativa, documentação robusta e versatilidade tornam Python uma escolha poderosa para programadores em todo o mundo. A simplicidade de sintaxe e a ênfase na legibilidade fazem dela uma excelente escolha para iniciantes e profissionais experientes.

**Python no mundo corporativo:** Python é uma linguagem de programação amplamente adotada por diversas empresas em todo o mundo devido à sua versatilidade, legibilidade e vasta comunidade de desenvolvedores. Algumas empresas notáveis que utilizam Python em seus projetos incluem:

1. **Google:**
    - Python é uma das linguagens de programação preferidas no Google. Muitos serviços e produtos internos, bem como partes do Google App Engine, foram implementados em Python.

2. **Facebook:**
    - Partes do código do Facebook são escritas em Python, e a empresa oferece a linguagem como uma opção para desenvolvimento em seu framework web Django.

3. **Instagram:**
    - Adquirido pelo Facebook, o Instagram utiliza Python em sua base de código, principalmente por meio do framework Django.

4. **Spotify:**
    - A equipe de backend do Spotify utiliza Python em vários componentes do serviço de streaming de música.

5. **Dropbox:**
    - Python é uma linguagem chave para o Dropbox, sendo utilizado tanto no backend quanto em ferramentas internas.

6. **Reddit:**
    - O Reddit utiliza Python em seu código fonte e no desenvolvimento do site, e o framework web Pylons (uma alternativa ao Django) foi utilizado no passado.

7. **NASA:**
    - Python é amplamente utilizado na NASA para várias tarefas, incluindo simulações, análise de dados e automação de tarefas.

8. **Netflix:**
    - Python é utilizado para diversas tarefas na Netflix, incluindo automação, ferramentas internas e parte do
    - backend.

9. **Microsoft:**
    - A Microsoft incorporou suporte a Python em muitos de seus produtos, como Azure, Visual Studio, e a linguagem é usada para desenvolvimento interno.

10. **Quora:**
    - O Quora utiliza Python em seus servidores e em várias partes de seu código, aproveitando a facilidade de uso da linguagem.

11. **Uber:**
    - Python é utilizado no desenvolvimento de ferramentas internas, automação e na construção de microsserviços na arquitetura do Uber.

# *Comandos Importantes:*

1. **Print:**

```
print("Olá, mundo!")
```

2. **Variáveis:**

```
nome = "Python"
numero = 42
```

3. **Estruturas Condicionais:**

```
if condicao: # código se a condição for verdadeira
else: # código se a condição for falsa
```

4. **Laços de Repetição:**

```
for elemento in lista: # código a ser repetido
```

5. **Funções:**

```
def saudacao(nome):
    return "Olá, " + nome + "!"
```

6. **Listas:**

```
numeros = [1, 2, 3, 4, 5]
```

7. **Dicionários:**

```
aluno = {"nome": "João", "idade": 20}
```

8. **Importação de Módulos:**

```
import math
raiz_quadrada = math.sqrt(25)
```

9. **Manipulação de Arquivos:**

```
with open("arquivo.txt", "r") as arquivo:
conteudo = arquivo.read()
```

10. **Expressões Lambda:**

```
quadrado = lambda x: x**2
```

# *Lista de Exercícios: Construção de um Sistema de Registro de Tarefas em Python*

1. Imprimir "Olá, mundo!" no console.
2. Calcular a soma de dois números inteiros.
3. Verificar se um número é par ou ímpar.
4. Calcular a média de três números.
5. Converter temperatura de Celsius para Fahrenheit.
6. Calcular o fatorial de um número.
7. Encontrar o maior número em uma lista.
8. Verificar se uma string é um palíndromo.
9. Calcular a sequência de Fibonacci até o enésimo termo.
10. Contar o número de vogais em uma string.
11. Ordenar uma lista de números em ordem crescente.
12. Encontrar todos os números primos até um dado limite.
13. Calcular o produto escalar de duas listas.
14. Inverter uma lista.
15. Implementar a busca binária em uma lista ordenada.
16. Verificar se uma lista é uma sub-sequência de outra.
17. Contar o número de ocorrências de cada palavra em um texto.
18. Calcular a raiz quadrada de um número sem usar bibliotecas.
19. Implementar um algoritmo de ordenação (Bubble Sort, Selection Sort, etc.).
20. Validar um endereço de e-mail.
21. Implementar uma árvore binária e realizar travessias (in-order, pre-order, post-order).

22. Resolver o problema da mochila (knapsack problem) usando programação dinâmica.
23. Implementar um grafo e encontrar o caminho mais curto usando o algoritmo de Dijkstra.
24. Desenvolver um jogo da forca em que o computador escolhe a palavra aleatoriamente.
25. Implementar um algoritmo genético simples para otimização.
26. Criar um interpretador de expressões matemáticas simples.
27. Implementar um servidor TCP simples para troca de mensagens.
28. Desenvolver um algoritmo de reconhecimento de padrões em uma imagem.
29. Resolver o problema das oito rainhas usando backtracking.
30. Criar um programa que simula o comportamento de um autômato celular.
31. Implementar um algoritmo de aprendizado de máquina simples (regressão linear, k-means, etc.).
32. Desenvolver um jogo de xadrez com inteligência artificial.
33. Criar um sistema de recomendação baseado em filtragem colaborativa.
34. Implementar um algoritmo de compressão de dados.
35. Desenvolver um bot para interagir com APIs de redes sociais.
36. Criar um sistema de processamento de linguagem natural para análise de sentimentos.
37. Implementar um algoritmo de criptografia assimétrica.
38. Desenvolver um sistema de reconhecimento de voz.
39. Criar um jogo de simulação de vida (vida artificial).
40. Implementar um algoritmo de otimização como o algoritmo genético para resolver um problema específico.
41. Registro Inicial:

    Crie uma lista chamada tarefas para armazenar as tarefas do sistema.

42. Adicionar Tarefa:

    Implemente uma função que permita adicionar uma nova tarefa à lista.

43. Listar Tarefas:

    Crie uma função que exiba todas as tarefas presentes na lista.

44. Remover Tarefa:

    Desenvolva uma função para remover uma tarefa específica da lista.

45. Conclusão de Tarefa:

    Implemente uma funcionalidade que marca uma tarefa como concluída.

46. Estatísticas de Tarefas:

    Crie uma função que exiba estatísticas simples, como o número total de tarefas e a quantidade de tarefas concluídas.

47. Priorização de Tarefas:

    Adicione uma funcionalidade para atribuir prioridades (baixa, média, alta) às tarefas.

48. Ordenação por Prioridade:

    Modifique a função de listagem para exibir as tarefas ordenadas por prioridade.

49. Persistência de Dados:

    Implemente a capacidade de salvar e carregar a lista de tarefas de/para um arquivo, para que as tarefas persistam entre execuções.

50. Interface de Usuário (Bônus):

    Desenvolva uma interface de usuário simples utilizando a biblioteca input() para permitir interações mais amigáveis com o sistema.

    *Observações:*

    *Considere a modularização do código, dividindo as funcionalidades em funções separadas para promover a clareza e reusabilidade.*

    *Utilize estruturas de controle de fluxo, listas, funções e manipulação de arquivos conforme necessário.*

51. Datas de Vencimento:

    Adicione a capacidade de atribuir datas de vencimento às tarefas. Implemente funcionalidades relacionadas a datas, como a exibição de tarefas vencidas e a ordenação por prazo de vencimento.

52. Controle de Estoque:

    Crie um sistema simples de controle de estoque. Utilize um dicionário para armazenar produtos e suas quantidades disponíveis.

53. Compra e Venda:

    Implemente funções para realizar compras e vendas de produtos no estoque, atualizando as quantidades disponíveis.

54. Histórico de Transações:

    Mantenha um histórico de todas as transações (compra e venda) realizadas. Utilize uma lista de dicionários para armazenar as informações relevantes.

55. Relatório Financeiro:

    Desenvolva uma função que gere um relatório financeiro, mostrando o saldo atual com base nas transações realizadas.

56. Busca e Filtragem:

    Crie uma função que permite buscar produtos pelo nome e outra função para filtrar transações por tipo (compra ou venda).

57. Operações em Lote:

    Implemente funcionalidades que permitam realizar operações em lote, como adicionar um novo produto ou remover produtos com base em critérios específicos.

58. Ranking de Produtos:

    Gere um ranking dos produtos mais vendidos. Considere a quantidade total vendida de cada produto.

59. Categorias de Produtos:

    Adicione a capacidade de categorizar os produtos e realize operações específicas com base nessas categorias.

60. Análise de Lucratividade:

    Crie uma função que analise a lucratividade de cada produto, considerando o preço de compra e o preço de venda.

61. Exportação de Dados (Bônus):

    Implemente uma funcionalidade para exportar os dados do sistema para um arquivo externo, como um arquivo CSV.

    1. *Considere a utilização de classes para representar produtos e transações, promovendo uma abordagem mais orientada a objetos.*
    2. *Utilize exceções para lidar com situações inesperadas, como tentativas de vender mais unidades do que as disponíveis no estoque.*

62. Desafio Extra

    Sistema de Login:

    Implemente um sistema de login para controlar o acesso às funcionalidades do sistema. Armazene informações de usuários em um dicionário e exija autenticação para realizar operações sensíveis.

63. Sistema de Reservas em Hotel:

    Crie um sistema que gerencie reservas de quartos em um hotel. Utilize um dicionário para representar os quartos e suas condições de reserva.

64. Avaliação de Filmes:

    Implemente um sistema de avaliação de filmes. Utilize um dicionário para armazenar os filmes e suas respectivas avaliações.

65. Agenda de Contatos Avançada:

    Aprimore uma agenda de contatos, permitindo a inclusão de múltiplos números e e-mails associados a cada contato.

66. Simulação de Jogo de Cartas:

    Crie uma simulação simples de um jogo de cartas. Utilize listas para representar os baralhos, cartas na mão dos jogadores e cartas na mesa.

67. Gerenciamento de Projetos:

    Desenvolva um sistema que simule o gerenciamento de projetos. Utilize estruturas de dados apropriadas para armazenar tarefas, responsáveis e datas de conclusão.

68. Estoque de Produtos em Loja Virtual:

    Construa uma estrutura de dados para gerenciar o estoque de produtos em uma loja virtual. Considere características como preço, quantidade disponível e categorias.

69. Análise de Texto:

    Crie um programa que analise um texto, contando a frequência de cada palavra e identificando as palavras mais frequentes.

70. Simulação de Redes Sociais:

    Desenvolva uma simulação simples de uma rede social. Utilize dicionários para representar perfis de usuários, suas postagens e interações.

71. Gerenciamento de Notas em Escola:

    Construa um sistema que gerencie notas de alunos em uma escola. Utilize estruturas de dados para armazenar informações sobre alunos, disciplinas e notas.

72. Sistema de Vendas Online

    Implemente um sistema de vendas online. Utilize listas, dicionários e conjuntos para representar produtos, carrinhos de compras e informações de clientes.

        Ao realizar a implementação, pense em como as estruturas de dados escolhidas podem otimizar a eficiência e clareza do código.

73. Considere a utilização de funções para modularizar e organizar o código.

    Experimente adicionar elementos de tratamento de exceções para lidar com situações inesperadas.

74. Sistema de Recomendação:

    Implemente um sistema de recomendação que sugira produtos, filmes ou outros itens com base no histórico do usuário. Utilize algoritmos simples de recomendação, como filtragem colaborativa.

# *Questões de múltiplas escolhas*

**1. Qual é a sintaxe correta para comentários em Python?**

a) **// Este é um comentário**

b) **/* Este é um comentário */**

c) **# Este é um comentário**

d) **-- Este é um comentário**

**2. Qual é a função do comando print() em Python?**

a) Ler entrada do usuário

b) Exibir dados na tela

c) Calcular uma expressão matemática

d) Definir uma variável

**3. Como você declara uma lista vazia em Python?**

a) **list = []**

b) **lista = {}**

c) **lista = [None]**

d) **list = ()**

**4. O que é uma função lambda em Python?**

a) Uma função que não aceita argumentos

b) Uma função que só pode ser usada uma vez

c) Uma função anônima

d) Uma função que retorna múltiplos valores

**5. Qual é a diferença entre uma lista e uma tupla em Python?**

a) As listas são mutáveis, enquanto as tuplas são imutáveis

b) As listas podem conter apenas números, enquanto as tuplas podem conter qualquer tipo de dado

c) As listas têm um tamanho fixo, enquanto as tuplas podem crescer dinamicamente

d) As listas são ordenadas, enquanto as tuplas não possuem ordem definida

**6. Como você remove um elemento específico de uma lista em Python?**

a) **remove(elemento)**

b) **delete(elemento)**

c) **pop(elemento)**

d) **del lista[elemento]**

**7. O que o método join() faz em Python?**

a) Concatena strings

b) Adiciona elementos a uma lista

c) Divide uma string em uma lista

d) Remove espaços em branco de uma string

**8. Qual é a diferença entre == e is em Python?**

a) Ambos são usados para comparação de igualdade

b) == compara valores, enquanto **is** compara identidade de objetos

c) **is** compara valores, enquanto == compara identidade de objetos

d) Não há diferença, ambos são intercambiáveis

**9. O que é um dicionário em Python?**

a) Uma estrutura de dados que armazena dados em uma sequência ordenada

b) Uma função especial que retorna múltiplos valores

c) Uma coleção não ordenada de pares chave-valor

d) Uma lista imutável

**10 . Como você lida com exceções em Python?**

a) Usando a instrução **if-else**

b) Usando a instrução **try-except**

c) Usando a instrução **for-in**

d) Usando a instrução **switch-case**

**11. O que é PEP 8 em Python?**

a) Uma função embutida

b) Um módulo para manipulação de strings

c) Um padrão de estilo de código

d) Um tipo de dado numérico

**12. Qual é a função do operador % em Python?**

a) Divisão

b) Exponenciação

c) Módulo (resto da divisão)

d) Multiplicação

**13. O que é um método em Python?**

a) Uma classe especial que armazena variáveis

b) Uma função associada a um objeto

c) Uma palavra-chave reservada

d) Um tipo de dado

**14. O que é um generator em Python?**

a) Um objeto que pode ser iterado

b) Uma função que retorna um valor único

c) Uma expressão lógica

d) Uma estrutura de controle de fluxo

**15. Como você lida com um arquivo em modo de leitura em Python?**

a) **open('arquivo.txt', 'r')**

b) **open('arquivo.txt', 'w')**

c) **open('arquivo.txt', 'a')**

d) **read('arquivo.txt')**

**16. O que é um decorator em Python?**

a) Uma função para remover elementos de uma lista

b) Um tipo de estrutura de controle de fluxo

c) Uma forma de alterar o comportamento de uma função

d) Uma expressão booleana

**17. Qual é o propósito do comando pass em Python?**

a) Indica o fim de um bloco de código

b) Representa uma exceção

c) É utilizado para criar loops infinitos

d) Não faz nada e é utilizado como espaço reservado

**18. O que é um set em Python?**

a) Uma estrutura de dados que armazena pares chave-valor

b) Uma lista ordenada

c) Uma coleção não ordenada de elementos únicos

d) Um tipo de dado numérico

**19. Como você converte uma string para minúsculas em Python?**

a) **str.lower()**

b) **str.upper()**

c) **str.capitalize()**

d) **str.title()**

**20. O que é o método strip() em Python?**

a) Remove caracteres à direita de uma string

b) Remove caracteres à esquerda de uma string

c) Remove espaços em branco no início e no fim de uma string

d) Substitui caracteres em uma string

# *Atividade Prática: Desenvolvimento de Habilidades em Python com Abordagem Analógica*

Nesta atividade, o objetivo é aprimorar as habilidades em manipulação de Python por meio de uma abordagem analógica, utilizando papel e caneta para resolver uma série de exercícios. A progressão dos exercícios abrange desde conceitos básicos até desafios mais avançados. O foco é promover a compreensão teórica e prática das operações fundamentais em listas.

1. **Troca de Variáveis:**
    - Escreva um algoritmo que troque os valores de duas variáveis sem o uso de uma variável temporária.
2. **Soma de Números Ímpares:**
    - Escreva um algoritmo que calcule a soma dos números ímpares de 1 a 100.
3. **Fatorial de um Número:**
    - Escreva um algoritmo que calcule o fatorial de um número fornecido pelo usuário.
4. **Verificação de Números Primos:**
    - Escreva um algoritmo que verifique se um número fornecido pelo usuário é primo.
5. **Sequência de Fibonacci:**
    - Escreva um algoritmo que gere os primeiros N termos da sequência de Fibonacci.
6. **Inversão de String:**
    - Escreva um algoritmo que inverta uma string fornecida pelo usuário.
7. **Contagem de Dígitos:**
    - Escreva um algoritmo que conte o número de dígitos em um número inteiro fornecido.
8. **Média de uma Lista:**
    - Escreva um algoritmo que calcule a média dos elementos em uma lista de números.

9. **Verificação de Palíndromo:**
    - Escreva um algoritmo que verifique se uma palavra fornecida pelo usuário é um palíndromo.
10. **Multiplicação Matricial:**
    - Escreva um algoritmo que realize a multiplicação de duas matrizes.
11. **Soma dos Dígitos de um Número:**
    - Escreva um algoritmo que calcule a soma dos dígitos de um número inteiro fornecido pelo usuário.
12. **Cálculo do Mínimo e Máximo:**
    - Escreva um algoritmo que encontre o valor mínimo e máximo em uma lista de números.
13. **Conversão de Temperatura:**
    - Escreva um algoritmo que converta uma temperatura em Celsius para Fahrenheit ou vice-versa, dependendo da escolha do usuário.
14. **Busca Binária:**
    - Escreva um algoritmo que implemente a busca binária em uma lista ordenada.
15. **Ordenação de Lista:**
    - Escreva um algoritmo que ordene uma lista de números usando o algoritmo de ordenação de sua escolha (por exemplo, Bubble Sort, Selection Sort).
16. **Cálculo do Quadrado Perfeito:**
    - Escreva um algoritmo que verifique se um número fornecido pelo usuário é um quadrado perfeito.
17. **Contagem de Vogais:**
    - Escreva um algoritmo que conte o número de vogais em uma string fornecida pelo usuário.

18. **Verificação de Ano Bissesto:**
    - Escreva um algoritmo que determine se um ano fornecido pelo usuário é bissexto.
19. **Soma de Números Primos:**
    - Escreva um algoritmo que calcule a soma dos primeiros N números primos.
20. **Adivinhação de Números:**
    - Escreva um algoritmo que permita que o usuário adivinhe um número secreto gerado aleatoriamente.

# *Debugging Python: Desvendando os Mistérios do Código*

A depuração, também conhecida como "debugging", é um aspecto crucial no processo de desenvolvimento de software. Ela refere-se ao processo de identificação, análise e resolução de bugs ou erros em um programa de computador. A depuração não é apenas uma prática técnica, mas uma parte essencial do processo de desenvolvimento de software. Ela desempenha um papel crucial na criação de produtos confiáveis, eficientes e que atendam às expectativas dos usuários, contribuindo para o sucesso a longo prazo de qualquer projeto de software.

Em síntese, a depuração vai além de simplesmente corrigir erros; ela é uma prática holística que permeia todo o ciclo de vida do desenvolvimento de software. Ao incorporar uma abordagem proativa para encontrar e resolver problemas, os desenvolvedores garantem não apenas a estabilidade imediata do software, mas também sua qualidade contínua e adaptabilidade ao longo do tempo.

Abaixo estarei disponibilizando links de sites parar treinar depuração e muitos assuntos referentes a Python, ademais diversos outros assuntos relacionados a programação.

1. **HackerRank (**https://www.hackerrank.com/**):**
    - Oferece uma ampla variedade de desafios de programação em várias linguagens.
    - Inclui seções específicas para prática de algoritmos, estruturas de dados, desenvolvimento web e muito mais.
    - Fornece uma plataforma para entrevistas técnicas.
2. **LeetCode (**https://leetcode.com/**):**
    - Especializado em questões de algoritmos e estruturas de dados.
    - Permite que os usuários pratiquem problemas específicos de empresas de tecnologia.
    - Ótimo para aprimorar habilidades de resolução de problemas e otimização de código.
3. **CodeSignal (**https://codesignal.com/**):**
    - Oferece uma abordagem única com desafios divididos por níveis de dificuldade.
    - Inclui missões específicas para prática em algoritmos, estruturas de dados, e até mesmo entrevistas.
    - Possui uma seção "Arcade" com desafios temáticos.

*Ao praticar em sites como esses, você não só aprimora suas habilidades de programação, mas também ganha experiência valiosa em depuração ao enfrentar desafios diversos*

Este material foi desenvolvido por maysa arruda e você pode encontrar mais conteúdo referente a estudo de Python  no meu repositório do GitHub: : CodeWithPython.

Obrigado por utilizar este projeto!