Funções Lambdas e Def

Nível Básico

1. **Soma de Números**: Crie uma função chamada soma números que recebe dois números como parâmetros e retorna a soma deles.

2. **Paridade de um Número**: Escreva uma função lambda chamada eh_par que recebe um número como parâmetro e retorna True se for par e False caso contrário. Utilize essa função para verificar se um número inserido pelo usuário é par.

3. **Positivo, Negativo ou Zero**: Implemente um programa que pede ao usuário para inserir um número e, em seguida, imprime se é positivo, negativo ou zero.

4. **Fatorial de um Número**: Crie uma função chamada fatorial que recebe um número como parâmetro e retorna o fatorial desse número. Utilize um loop para calcular o fatorial.

5. **Números Pares de 1 a 10**: Escreva um programa que utilize um loop para imprimir os números pares de 1 a 10.

6. **Multiplicação de Números**: Crie uma função lambda chamada multiplica_numeros que recebe dois números como parâmetros e retorna a multiplicação deles. Utilize essa função em um loop para imprimir as multiplicações de 1 a 5.

7. **Verificação de Palíndromo**: Implemente uma função chamada eh_palindromo que recebe uma string como parâmetro e retorna True se for um palíndromo e False caso contrário.

8. **Maior de Dois Números**: Faça um programa que solicita ao usuário dois números e imprime o maior deles.

9. **Média de Números**: Crie uma função chamada calcula media que recebe uma lista de números como parâmetro e retorna a média deles.

10. **Números de 1 a 10, Exceto Múltiplos de 3**: Escreva um programa que utiliza um loop para imprimir os números de 1 a 10, pulando os múltiplos de 3.

11. **Filtragem de Números Pares**: Implemente uma função chamada filtra pares que recebe uma lista de números como parâmetro e retorna uma lista contendo apenas os números pares.

12. **Potência de um Número**: Crie uma função lambda chamada potencia número que recebe dois números como parâmetros e retorna o primeiro elevado ao segundo. Utilize essa função para calcular a potência de um número.

13. **Verificação de Número Primo**: Faça um programa que pede ao usuário um número e imprime se é primo ou não.

14. **Contagem de Palavras que Começam com 'A'**: Escreva uma função chamada conta_palavras_A que recebe uma lista de palavras como parâmetro e retorna a quantidade de palavras que começam com a letra 'A'.

15. **Soma dos Quadrados de 1 a 5**: Implemente um programa que calcula a soma dos quadrados dos números de 1 a 5.

16. **Dobro de um Número**: Crie uma função lambda chamada dobro que recebe um número como parâmetro e retorna o dobro dele.

17. **Tabuada de um Número**: Faça um programa que imprime a tabuada de um número fornecido pelo usuário.

18. **Ordenação de Números**: Escreva uma função chamada ordena_numeros que recebe uma lista de números como parâmetro e retorna uma lista com os números ordenados de forma crescente.

19. **Intervalo entre Dois Números**: Implemente um programa que solicita ao usuário dois números e imprime todos os números no intervalo entre eles.

20. **Filtragem de Strings Longas**: Crie uma função chamada filtra_strings que recebe uma lista de strings como parâmetro e retorna uma lista contendo apenas as strings com mais de 5 caracteres.

Nível Intermediário

1. **Soma de Lista**: Crie uma função chamada soma_lista que recebe uma lista de números como parâmetro e retorna a soma de todos os elementos.

2. **Filtragem de Números Ímpares**: Escreva uma função que recebe uma lista de números e retorna uma lista contendo apenas os números ímpares.

3. **Maior e Menor Valor de uma Lista**: Implemente um programa que pede ao usuário para inserir uma lista de números e imprime o maior e o menor valor da lista.

4. **Contagem de Letras**: Crie uma função chamada conta_letras que recebe uma string como parâmetro e retorna um dicionário com a contagem de cada letra.

5. **Distância entre Dois Pontos**: Escreva um programa que utiliza uma tupla para armazenar as coordenadas (x, y) de dois pontos e calcula a distância entre eles.

6. **Palavra Mais Longa**: Implemente uma função que recebe uma lista de palavras e retorna a palavra mais longa.

7. **Frase Invertida**: Crie um programa que solicita ao usuário uma frase e imprime a frase invertida.

8. **Filtragem de Números Divisíveis por 3 e 5**: Escreva uma função que recebe uma lista de números e retorna uma lista contendo apenas os números que são divisíveis por 3 e 5.

9. **Preço Total de uma Lista de Compras**: Implemente um programa que utiliza um dicionário para armazenar o preço de alguns produtos e calcula o preço total de uma lista de compras fornecida pelo usuário.

10. **Número de Caracteres de Cada String**: Crie uma função que recebe uma lista de strings e retorna uma lista com o número de caracteres de cada string.

11. **Números Únicos de uma Lista**: Faça um programa que lê uma lista de números e imprime apenas os números únicos (sem repetição).

12. **Filtragem de Palavras por Caractere Inicial**: Escreva uma função que recebe uma lista de palavras e um caractere, e retorna uma lista contendo apenas as palavras que começam com esse caractere.

13. **Média das Notas dos Alunos**: Implemente um programa que utiliza uma lista para armazenar os nomes de alunos e suas respectivas notas, e imprime a média das notas.

14. **Soma dos Quadrados dos Números Pares**: Crie uma função que recebe uma lista de números e retorna a soma dos quadrados dos números pares.

15. **Média de Idade**: Faça um programa que utiliza um dicionário para armazenar as idades de diferentes pessoas e imprime a média de idade.

16. **Ordenação de Strings por Comprimento**: Escreva uma função que recebe uma lista de strings e retorna uma lista com as strings ordenadas por comprimento, da menor para a maior.

17. **União e Interseção de Conjuntos**: Implemente um programa que solicita ao usuário dois conjuntos de números e imprime a união e a interseção desses conjuntos.

18. **Verificação de Disponibilidade de Produto**: Crie uma função que recebe um dicionário representando um estoque de produtos e verifica se um produto específico está disponível.

19. **Cidade com a Maior População**: Faça um programa que utiliza uma lista para armazenar nomes de cidades e suas respectivas populações, e imprime a cidade com a maior população.

20. **Conjunto de Palavras sem Repetição**: Escreva uma função que recebe uma lista de palavras e retorna um conjunto contendo as palavras sem repetição.

21. **Ordenação de Números sem Usar a Função Sort**: Implemente um programa que solicita ao usuário uma lista de números e imprime os números em ordem crescente, sem usar a função sort.

22. **Aluno com a Maior Média**: Crie uma função que recebe um dicionário representando notas de alunos e retorna o nome do aluno com a maior média.

23. **Nomes de Pessoas com Idade Acima da Média**: Escreva um programa que utiliza uma lista para armazenar nomes de pessoas e suas idades, e imprime os nomes das pessoas com idade acima da média.

24. **Verificação de Ordenação de Lista**: Implemente uma função que recebe uma lista de números e retorna True se a lista estiver ordenada de forma crescente, e False caso contrário.

25. **Busca de Livro por Título**: Crie um programa que utiliza um dicionário para armazenar informações sobre livros (título, autor, ano) e permite ao usuário buscar um livro pelo título.

26. **Lista Invertida**: Faça uma função que recebe uma lista de strings e retorna a lista invertida.

27. **Verificação de Igualdade de Listas**: Escreva um programa que solicita ao usuário duas listas e verifica se são iguais (têm os mesmos elementos na mesma ordem).

28. **Adição de Novo Produto ao Estoque**: Implemente uma função que recebe um dicionário representando o estoque de produtos e adiciona um novo produto com sua quantidade.

29. **Verificação de Número Repetido em Lista**: Crie uma função que recebe uma lista de números e retorna True se houver algum número repetido, e False caso contrário.

30. **Média das Temperaturas Diárias**: Faça um programa que utiliza uma lista para armazenar as temperaturas diárias de uma semana e imprime a média das temperaturas.

Nivel Avançado

1. **Ponto no Espaço Tridimensional**: Crie uma classe chamada Ponto que representa um ponto no espaço tridimensional (x, y, z). Adicione métodos para calcular a distância entre dois pontos e para verificar se um ponto está no plano xy.

2. **Avaliação de Expressão Matemática**: Escreva uma função que recebe uma expressão matemática como uma string e retorna o resultado da avaliação da expressão. Considere a presença de operadores aritméticos e parênteses.

3. **Frequência de Palavras em um Arquivo**: Implemente um programa que lê um arquivo de texto e conta a frequência de cada palavra no arquivo.

4. **Operações com Matrizes**: Crie uma classe Matriz que suporta operações como adição, multiplicação e transposição de matrizes.

5. **Cálculo Eficiente do Número de Fibonacci**: Escreva uma função que usa programação dinâmica para calcular o n-ésimo número de Fibonacci de forma eficiente.

6. **Simulação de Blackjack**: Implemente um programa que simula um jogo de blackjack, permitindo que um jogador jogue contra a inteligência artificial do dealer.

7. **Cálculo do Número de Fibonacci com Memoização**: Crie uma função que utiliza a técnica de memoização para calcular o n-ésimo número de Fibonacci.

8. **Caminho Mais Curto em um Grafo**: Escreva uma classe Grafo que represente um grafo direcionado ou não direcionado e inclua métodos para encontrar o caminho mais curto entre dois vértices.

9. **Ordenação Personalizada de Objetos**: Implemente um algoritmo de ordenação personalizado para ordenar uma lista de objetos com base em um atributo específico.

10. **Avaliação de Expressão Lógica**: Crie uma função que recebe uma expressão lógica como uma string e retorna True se a expressão for verdadeira e False caso contrário.

11. **Classificação de Dados com Aprendizado de Máquina**: Escreva um programa que utiliza um algoritmo de aprendizado de máquina simples para classificar dados de entrada em duas categorias.

12. **Gerenciamento de Tarefas**: Implemente um sistema de gerenciamento de tarefas que permite adicionar, remover e marcar tarefas como concluídas.

13. **Busca Binária em uma Lista Ordenada**: Crie uma função que implemente o algoritmo de busca binária em uma lista ordenada.

14. **Percorrer uma Árvore Binária**: Escreva uma classe Árvore que representa uma árvore binária e inclua métodos para percorrer a árvore em diferentes ordens (in-order, pre-order, post-order).

15. **Interação com uma Rede Social Fictícia**: Implemente um programa que utilize a API de uma rede social fictícia para obter dados de perfil de usuários, fazer postagens e interagir com outros usuários.

16. **Resolver o Problema da Mochila com Programação Dinâmica**: Crie uma função que utiliza a técnica de programação dinâmica para resolver o problema da mochila, otimizando a seleção de itens com base no valor e no peso.

17. **Simulação de um Sistema de Arquivos**: Escreva um programa que simula um sistema de arquivos, permitindo a criação, remoção e navegação por diretórios.

18. **Treinamento e Previsão com uma Rede Neural Simples**: Implemente uma classe RedeNeural que representa uma rede neural simples, com métodos para treinar e fazer previsões.

19. **Caminho Mais Curto em um Grafo Ponderado com Dijkstra**: Crie uma função que implementa o algoritmo de Dijkstra para encontrar o caminho mais curto em um grafo ponderado.

20. **Resolver o Problema do Caixeiro-Viajante com Algoritmo Genético**: Escreva um programa que utiliza um algoritmo genético para encontrar uma solução aproximada para o problema do caixeiro-viajante.

21. **Sistema de Recomendação Baseado nas Preferências do Usuário**: Implemente um sistema de recomendação que sugere itens com base nas preferências do usuário.

22. **Encontrar o Mínimo Local de uma Função com Minimização de Gradiente**: Crie uma função que utiliza o algoritmo de minimização de gradiente para encontrar o mínimo local de uma função.

23. **Simulação da Propagação de uma Epidemia**: Escreva um programa que simula a propagação de uma epidemia em uma população, levando em consideração diversos parâmetros.

24. **Fila de Prioridade com Inserção e Remoção Eficientes**: Implemente uma classe FilaPrioridade que suporta a inserção e remoção eficientes de elementos com base em uma prioridade associada a cada elemento.

25. **Encontrar Todos os Caminhos Mais Curtos em um Grafo Ponderado com Floyd-Warshall**: Crie uma função que utiliza o algoritmo de Floyd-Warshall para encontrar todos os caminhos mais curtos entre todos os pares de vértices em um grafo ponderado.

26. **Análise e Classificação de Sentimentos em Textos com Processamento de Linguagem Natural**: Escreva um programa que utiliza técnicas de processamento de linguagem natural para analisar e classificar sentimentos em um conjunto de textos.

27. **Operações de Inserção e Remoção em uma Árvore AVL**: Implemente uma classe ÁrvoreAVL que representa uma árvore AVL e inclua métodos para realizar operações de inserção e remoção.

28. **Convolução de Duas Funções com Transformada de Fourier**: Crie uma função que utiliza a transformada de Fourier para realizar a convolução de duas funções.

29. **Simulação do Comportamento de um Sistema Operacional**: Escreva um programa que simula o comportamento de um sistema operacional, gerenciando processos concorrentes e recursos do sistema.

30. **Encontrar a Árvore de Abrangência Mínima em um Grafo Ponderado com Kruskal**: Implemente uma classe GrafoPonderado que representa um grafo ponderado e inclua métodos para encontrar a árvore de abrangência mínima usando o algoritmo de Kruskal.

Especialista

1. **Lambda e Map**: Crie uma função lambda que eleva um número ao quadrado. Use a função map para aplicar essa função lambda a uma lista de números.

2. **List Comprehension**: Use a list comprehension para criar uma lista dos quadrados dos números de 1 a 10.

3. **Dict Comprehension**: Use a dict comprehension para criar um dicionário que mapeia números de 1 a 10 para seus quadrados.

4. **Tuple Comprehension**: Use a tuple comprehension (gerador) para criar uma tupla dos quadrados dos números de 1 a 10.

5. **Função Recursiva**: Escreva uma função recursiva para calcular o fatorial de um número.

6. **Lambda e Filter**: Crie uma função lambda que verifica se um número é par. Use a função filter para filtrar os números pares de uma lista.

7. **Estruturas de Controle**: Escreva um programa que usa estruturas de controle (if, for, while) para imprimir os números de 1 a 10, mas pula os números divisíveis por 3.

8. **Função de Ordem Superior**: Escreva uma função de ordem superior que recebe uma função e uma lista como parâmetros, e aplica a função a todos os elementos da lista.

9. **Lambda e Reduce**: Use a função reduce do módulo functools juntamente com uma função lambda para calcular o produto de uma lista de números.

10. **Geradores**: Escreva um gerador que produz os números na sequência de Fibonacci.

11. **Decoradores**: Escreva um decorador que registra a execução de uma função, imprimindo uma mensagem antes e depois da chamada da função.

12. **Manipulação de Arquivos**: Escreva um programa que lê um arquivo de texto e conta a frequência de cada palavra no arquivo.

13. **Expressões Regulares**: Escreva um programa que usa expressões regulares para verificar se uma string é um endereço de e-mail válido.

14. **Manipulação de Exceções**: Escreva um programa que solicita ao usuário um número e imprime o inverso do número. Use a manipulação de exceções para lidar com entradas inválidas.

15. **Programação Orientada a Objetos**: Escreva uma classe Círculo que representa um círculo. A classe deve ter um método para calcular a área do círculo.

16. **Testes Unitários**: Escreva testes unitários para a classe Círculo usando o módulo unittest.

17. **Manipulação de Datas**: Escreva um programa que solicita ao usuário sua data de nascimento e imprime sua idade.

18. **Web Scraping**: Escreva um programa que faz web scraping de um site e extrai informações específicas. (Nota: sempre respeite os termos de serviço do site)

19. **Acesso a Banco de Dados**: Escreva um programa que acessa um banco de dados SQLite, cria uma tabela, insere dados na tabela, consulta os dados e os imprime.

20. **Multithreading**: Escreva um programa que faz download de várias páginas da web em paralelo usando multithreading.

21. **Multiprocessing**: Modifique o programa anterior para usar multiprocessing em vez de multithreading, e compare a diferença de desempenho.

22. **Async IO**: Reescreva o programa de download de páginas da web usando Async IO em vez de multithreading ou multiprocessing.

23. **Pandas e Numpy**: Escreva um programa que usa pandas e numpy para analisar um conjunto de dados.

24. **Matplotlib**: Modifique o programa anterior para visualizar os resultados da análise usando matplotlib.

25. **Scikit-learn**: Escreva um programa que usa scikit-learn para fazer uma tarefa simples de aprendizado de máquina, como classificação ou regressão.

26. **TensorFlow ou PyTorch**: Escreva um programa que usa TensorFlow ou PyTorch para treinar uma rede neural simples em um conjunto de dados.

27. **Flask ou Django**: Escreva uma aplicação web simples usando Flask ou Django.

28. **Requests ou Scrapy**: Escreva um programa que usa requests ou scrapy para fazer web scraping de um site.

29. **Tkinter ou PyQt**: Escreva uma aplicação de desktop simples usando Tkinter ou PyQt.

30. **Keras**: Escreva um programa que usa Keras para treinar uma rede neural convolucional em um conjunto de dados de imagens.