Módulo 07
Funções Avançadas (Aula 10)

Questões de Aprendizagem

Questão 1:

Defina uma classe `Endereco` usando o decorator `@dataclass`. A classe deve ter os seguintes atributos: `rua`, `numero`, `cidade`, `estado` e `cep`. Crie uma instância da classe `Endereco` e imprima seus atributos. Em seguida, crie outra instância com os mesmos valores e verifique se as duas instâncias são consideradas iguais pelo operador `==`.

Questão 2:

Utilize a função `sorted()` com uma função lambda para ordenar uma lista de tuplas. Cada tupla contém o nome de um produto e seu preço. A lista deve ser ordenada em ordem decrescente de preço.

Questão 3:

Você tem uma lista de caminhos de arquivos. Utilize a função `filter()` com uma função lambda para obter apenas os caminhos que correspondem a arquivos com extensão ".txt".

Questão 4:

Use as funções `map()` e `filter()` com funções lambda para processar uma lista de números. Primeiro, use `map()` para elevar cada número ao quadrado. Em seguida, use `filter()` para manter apenas os números que são maiores que 10.

Questão 5:

Escreva uma função de alta ordem chamada `aplicar_operacao(numeros, operacao)` que recebe uma lista de números e uma função `operacao` como argumentos. A função `aplicar_operacao` deve aplicar a função `operacao` a cada número na lista e retornar uma nova lista com os resultados. Em seguida, crie duas funções, `dobrar(x)` que retorna o dobro de `x`, e `triplicar(x)` que retorna o triplo de `x`. Utilize a função `aplicar_operacao` para dobrar todos os números em uma lista e, em seguida, triplicar todos os números em outra lista.

Módulo 07
Funções Avançadas (Aula 10)

Questões de Aprendizagem

Questão 6:

Você tem uma lista de dicionários, onde cada dicionário representa um aluno com as chaves "nome" e "notas" (uma lista de notas). Escreva uma função `obter_melhor_aluno(alunos, criterio)` que recebe a lista de alunos e uma função `criterio` como argumentos. A função `criterio` deve receber um dicionário de aluno e retornar um valor numérico pelo qual os alunos serão comparados. Use a função `max()` com o parâmetro `key` para encontrar o aluno com o maior valor retornado pela função `criterio`. Teste sua função com duas funções de critério diferentes: uma que retorna a média das notas do aluno e outra que retorna a maior nota do aluno.

Questão 7:

Crie uma função geradora `gerar_primos(maximo)` que retorne todos os números primos até um valor máximo fornecido. Utilize a função `next()` para obter os primeiros 5 números primos gerados pela função.

Questão 8:

Considere uma função `buscar_dados_api(url)` que faz uma requisição a uma API externa para buscar dados. Essa operação pode ser demorada e a API pode retornar os mesmos dados para a mesma URL várias vezes. Utilize o decorator `@cache` para otimizar a função, evitando requisições repetidas para a mesma URL. Crie um exemplo onde você chama a função várias vezes com a mesma URL e, em seguida, com URLs diferentes, e observe o comportamento do `@cache` (você pode simular a chamada à API usando a função `time.sleep()` para introduzir um atraso e imprimindo uma mensagem para indicar quando a função está sendo executada).

Módulo 07
Funções Avançadas (Aula 10)

Questões de Aprendizagem

Questão 9:

Crie um pipeline de geradores para processar uma lista de strings. O pipeline deve consistir de três etapas:

- Uma função geradora `converter_maiusculas(strings)` que recebe uma sequência de strings e retorna cada string convertida para maiúsculas.
- Uma função geradora `filtrar_strings(strings, tamanho_minimo)` que recebe uma sequência de strings e um inteiro `tamanho_minimo`, e retorna apenas as strings que têm um comprimento maior ou igual a `tamanho_minimo`.
- Uma função geradora `limitar_quantidade(strings, quantidade)` que recebe uma sequência de strings e um inteiro `quantidade`, e retorna no máximo `quantidade` strings da sequência.

Aplique o pipeline a uma lista de strings de sua escolha, definindo `tamanho_minimo` como 5 e `quantidade` como 3.

Questão 10:

Considere um cenário onde você precisa processar um arquivo de texto muito grande, contendo milhões de linhas. Cada linha do arquivo representa uma transação financeira com o seguinte formato: `data,hora,valor,descrição`. Seu objetivo é calcular o valor total das transações para cada dia. Escreva uma função geradora chamada `processar_transacoes(nome_arquivo)` que leia o arquivo linha por linha, extraia a data e o valor de cada transação, e retorne um dicionário onde as chaves são as datas e os valores são a soma dos valores das transações para aquele dia.