UNIVERSIDADE FEDERAL DO CEARÁ
Departamento de Computação
Prof. Lincoln Souza Rocha
CK0442- Técnicas de Programação para Ciência de Dados - T02

TP07
2024.2

1. **Implementando Funções/Expressões Lambda**

   Descrição: Sobre funções/expressões lambda em Python, implemente o que se pede:

   1. **Soma Simples**
      - Crie uma função lambda que receba dois números e retorne sua soma. Teste a função com diferentes pares de números.
   2. **Verificação de Paridade**
      - Escreva uma função lambda que receba um número e retorne `True` se ele for par e `False` caso contrário.
   3. **Elevar ao Quadrado**
      - Use uma função lambda dentro da função `map()` para elevar ao quadrado todos os números de uma lista.
   4. **Composição de Funções**
      - Crie duas funções lambda: (i) uma que converte um valor em Celsius para Fahrenheit; e (ii) outra que arredonda o valor para o inteiro mais próximo. Em seguida, componha essas funções para criar uma única função que converta Celsius para Fahrenheit e retorne o valor arredondado. Teste com uma lista de temperaturas em Celsius.

2. **Implementando com a Função Map**

   Descrição: Sobre a função `map()` do Python, implemente o que se pede:

   1. **Converter para Maiúsculas**
      - Dada a lista ["python", "lambda", "map"], use a função `map()` para converter todos os elementos para maiúsculas.
   2. **Raiz Quadrada**
      - Escreva uma função que, com o auxílio de `map()` e uma função lambda, calcule a raiz quadrada de uma lista de números.
   3. **Análise de Strings**
      - Dada a lista de frases ["Python é incrível", "Lambda são úteis", "Map é funcional"], use `map()` para criar uma lista de dicionários, onde cada dicionário contenha: (i) o número de palavras na frase; e (ii) o comprimento total da frase (em caracteres). Exemplo de saída para a frase dada: [{"palavras": 3, "caracteres": 17}, {"palavras": 3, "caracteres": 16}, {"palavras": 3, "caracteres": 15}]

3. **Implementando com a Função Filter**

Descrição: Sobre a função `filter()` do Python, implemente o que se pede:

1. **Filtrar Números Positivos**
   - Use a função `filter()` para obter somente os números positivos de uma lista como [-10, 15, -20, 25, 0, 30].
2. **Filtrar Palavras Curtas**
   - Dada uma lista de palavras, use `filter()` para selecionar apenas as palavras com mais de 5 caracteres.
3. **Filtragem baseada em Múltiplas Condições**
   - Dada uma lista de números inteiros, use `filter()` para selecionar apenas aqueles que: (i) são múltiplos de 3 ou de 5; e (ii) são positivos. Retorne o resultado como uma lista.

4. **Implementando com a Função Reduce**

Descrição: Sobre a função `reduce()` do Python, implemente o que se pede:

1. **Produto de uma Lista**
   - Utilize a função `reduce()` para calcular o produto dos números em uma lista.
2. **Maior Número**
   - Use a função `reduce()` para encontrar o maior número em uma lista.
3. **Construção de um Dicionário**
   - Use `reduce()` para transformar uma lista de tuplas como [("a", 1), ("b", 2), ("a", 3)] em um dicionário, onde as chaves são únicas e os valores correspondem à soma dos valores associados àquela chave. Exemplo de saída esperada: {"a": 4, "b": 2}.

5. **Implementando Geradores e Expressões Geradoras**

Descrição: Sobre geradores e expressões geradoras do Python, implemente o que se pede:

1. **Gerador de Pares**
   - Crie um gerador que produza todos os números pares entre 1 e 50. Use um loop for para exibir os números gerados.
2. **Soma de Quadrados com Expressão Geradora**
   - Use uma expressão geradora para calcular a soma dos quadrados dos números de 1 a 10.
3. **Gerador de Sequência Infinita**
   - Crie um gerador infinito que produza números da sequência de Fibonacci. Exiba os primeiros 10 números gerados.
4. **Gerador Personalizado**
   - Crie um gerador que receba dois números, start e end, e produza todos os números primos dentro desse intervalo. Teste o gerador imprimindo os números gerados para o intervalo de 10 a 50.
5. **Gerador Infinito com Condição de Parada**
   - Crie um gerador infinito que produza números da sequência de Fibonacci. Ele deve parar automaticamente quando o número gerado exceder um valor limite fornecido como argumento (por exemplo, 10.000).