Lista de Exercícios — Modularização em Python

Prof. Dr. Alysson Naves

Objetivo

Esta lista de exercícios tem como foco a prática de modularização de código em Python. Todos os problemas devem ser resolvidos utilizando **funções bem definidas e reutilizáveis**.

1. **Média de temperaturas semanais**
   Crie uma função que receba uma lista com 7 temperaturas e retorne a média. Depois, escreva funções para retornar o maior e menor valor da lista. Crie uma função final para exibir os resultados formatados.

2. **Conversor de medidas**
   Crie funções para converter metros em centímetros, milímetros e polegadas. O usuário informa o valor e a unidade de destino.

3. **Gerador de senha aleatória**
   Crie uma função que gere uma senha aleatória com N caracteres, podendo conter letras, números e símbolos. Use funções auxiliares para gerar os caracteres e para construir a senha.

4. **Tabuada modular**
   Crie uma função que recebe um número e imprime a tabuada dele de 1 a 10. Permita que o usuário digite vários números em sequência e utilize a função criada para cada um.

5. **Validador de CPF (Simplificado)**
   Crie uma função que verifique se um CPF está no formato `xxx.xxx.xxx-yy`. Use outra função para ler vários CPFs e verificar todos.

6. **Relatório de notas de aluno**
   Crie funções para:

   - Ler as notas de um aluno
   - Calcular a média
   - Verificar situação (Aprovado, Recuperação, Reprovado)
   - Exibir um relatório com os resultados

7. **Contador de vogais e consoantes**
   Crie uma função que recebe uma string e retorna a quantidade de vogais e consoantes. Separe as funções de leitura, contagem e exibição.

8. **Simulador de caixa eletrônico**
   Crie uma função que simula um saque e retorna quantas notas de 100, 50, 20, 10 e 5 são necessárias. Use funções auxiliares para modularizar.

9. **Jogo de adivinhação**
   Crie um jogo onde o computador sorteia um número de 1 a 100 e o jogador tenta adivinhar. Use funções para: Sortear número, ler palpite, verificar palpite e controlar tentativas.

10. **Validador de CPF**
    Crie uma função que verifique se um CPF é válido. A função deve seguir as seguintes etapas:

    - Remover automaticamente pontos e hífen da entrada, se existirem;
    - Verificar se o CPF possui exatamente 11 dígitos numéricos;
    - Rejeitar CPFs compostos por todos os dígitos iguais (ex: `111.111.111-11`);
    - Verificar os dois dígitos verificadores finais usando a regra oficial:
      - Para o primeiro dígito: multiplica-se cada um dos 9 primeiros dígitos por pesos de 10 a 2, somando os resultados. O dígito é 0 se o resto da divisão por 11 for menor que 2; caso contrário, é `11 - (resto)`.
      - Para o segundo dígito: repete-se o processo com os 9 dígitos originais mais o primeiro dígito verificador, usando pesos de 11 a 2.
    - A função deve retornar `True` se o CPF for válido e `False` caso contrário.

    O programa principal deve solicitar a entrada do usuário e exibir uma mensagem indicando se o CPF digitado é válido.

    Exemplo: É utilizado como exemplo o número: 123.456.789-07

    - Calcule a soma dos produtos dos nove digitos utilizando peso dois para unidade, peso 3 para dezena, peso 4 para centena e assim sucessivamente. Exemplo: $10 \times 1 + 9 \times 2 + 8 \times 3 + 7 \times 4 + 6 \times 5 + 5 \times 6 + 4 \times 7 + 3 \times 8 + 2 \times 9 = 210$
      - o resto da divisão de 210 por 11 é 1 então a dezena do número verificador é **0**.
    - Calcule a soma dos produtos dos dez digitos, onde o digito menos significativo passa a ser a dezena dos digitos verificadores, utilizando os seguintes pesos: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11; Exemplo: $11 \times 1 + 10 \times 2 + 9 \times 3 + 8 \times 4 + 7 \times 5 + 6 \times 6 + 5 \times 7 + 4 \times 8 + 3 \times 9 + 2 \times 0 = 255$
      - resto da divisão de 255 por 11 é 2 então a unidade do número verificador é 11-2=**9**.

    O 123.456.789-07 não é válido, pois deveria ser: 123.456.789-09

Instruções finais: Comente o código, nomeie bem as funções e tente dividir o problema em blocos menores sempre que possível.