Introdução à Programação em Python
I Lista de Exercícios

Prof. Evandro C. R. Rosa
UNIVALI

Nome Completo: _____ Código de Aluno: _____

1. Crie um programa que verifique se uma palavra ou frase é um palíndromo, ou seja, se é lida da mesma maneira de trás para frente.

2. Desenvolva um programa que permita ao usuário converter uma lista de temperaturas de Celsius para Fahrenheit e vice-versa. O programa deve conter as seguintes funções:

   - `celsius_para_fahrenheit(celsius)`: converte uma lista de temperaturas em Celsius para Fahrenheit.
   - `fahrenheit_para_celsius(fahrenheit)`: converte uma lista de temperaturas em Fahrenheit para Celsius.
   - `exibir_temperaturas(lista)`: exibe a lista de temperaturas convertidas.

   Dicas:

   - Utilize as fórmulas de conversão:
     - Fahrenheit = (Celsius $\times$ 9/5) + 32
     - Celsius = (Fahrenheit - 32) $\times$ 5/9
   - Peça ao usuário para fornecer a lista de temperaturas e a direção da conversão.

3. Crie um programa que calcule a média ponderada das notas de um aluno. As notas e os pesos devem ser fornecidos pelo usuário e armazenados em listas. O programa deve ter as seguintes funções:

   - `calcular_media_ponderada(notas, pesos)`: recebe uma lista de notas e uma lista de pesos e calcula a média ponderada.
   - `solicitar_dados()`: solicita ao usuário as notas e os pesos correspondentes.

   Dicas:

   - Multiplique cada nota pelo seu peso, some os resultados e divida pela soma dos pesos para obter a média ponderada.
   - As listas de notas e pesos devem ter o mesmo tamanho.

4. Crie um programa que permita cadastrar alunos e suas notas, armazenando as informações em um dicionário. O programa deve ter as seguintes funções:

   - `cadastrar_aluno(nome, nota)`: adiciona um aluno e sua nota ao dicionário.
   - `mostrar_alunos()`: exibe a lista de alunos e suas respectivas notas.
   - `media_turma()`: calcula e retorna a média das notas da turma.

   Dicas:

   - Utilize um dicionário para armazenar os alunos (chaves) e suas notas (valores).
   - A função `mostrar_alunos` pode percorrer o dicionário e exibir o nome e a nota de cada aluno.

5. Escreva um programa que conte a frequência de cada palavra em uma frase fornecida pelo usuário. O programa deve incluir as seguintes funções:

   - `contar_palavras(frase)`: conta a frequência de cada palavra em uma frase e armazena em um dicionário.
   - `mostrar_frequencia(frequencia)`: exibe a palavra e o número de ocorrências de cada uma.

   Dicas:

   - Use o método `split()` para dividir a frase em palavras.
   - Utilize um dicionário para armazenar as palavras como chaves e suas frequências como valores.

6. Implemente um programa para gerenciar uma lista de compras. O programa deve conter as seguintes funções:

   - `adicionar_item(lista, item, quantidade)`: adiciona um item à lista de compras com sua quantidade.
   - `remover_item(lista, item)`: remove um item da lista.
   - `mostrar_lista(lista)`: exibe todos os itens da lista e suas respectivas quantidades.

   Dicas:

   - A lista de compras pode ser um dicionário onde as chaves são os itens e os valores são as quantidades.
   - Para remover um item, verifique se ele existe no dicionário antes de tentar removê-lo.

7. Crie um programa que peça ao usuário uma lista de números e forneça as seguintes estatísticas sobre a lista, utilizando funções:

   - `menor_numero(lista)`: retorna o menor número da lista.
   - `maior_numero(lista)`: retorna o maior número da lista.
   - `media(lista)`: retorna a média dos números da lista.
   - `ocorrencias(lista)`: retorna um dicionário com a frequência de cada número na lista.

   Dicas:

   - Use as funções embutidas `min()`, `max()` e `sum()` para calcular o menor, o maior e a média dos números.
   - Para contar a frequência de cada número, utilize um dicionário onde a chave é o número e o valor é o número de ocorrências.

8. Crie um sistema para registrar e analisar as notas de uma turma. O programa deve conter as seguintes funções:

   - `adicionar_nota(dicionario, aluno, nota)`: adiciona a nota de um aluno ao dicionário.
   - `mostrar_aprovados(dicionario)`: exibe os alunos aprovados (nota ≥ 7.0).
   - `media_turma(dicionario)`: retorna a média das notas da turma.

   Dicas:

   - Utilize um dicionário onde as chaves são os nomes dos alunos e os valores são as notas.
   - Para calcular a média da turma, some todas as notas e divida pelo número de alunos.

9. Crie um programa que conte a frequência de cada caractere em uma string fornecida pelo usuário. O programa deve conter as seguintes funções:

   - `contar_caracteres(string)`: conta a frequência de cada caractere e armazena os resultados em um dicionário.
   - `mostrar_frequencia(frequencia)`: exibe o caractere e o número de vezes que ele aparece na string.

   Dicas:

   - Utilize um dicionário para armazenar os caracteres como chaves e suas frequências como valores.
   - Ignore espaços em branco ou outros caracteres que não sejam letras ou números.

10. Crie um programa para gerenciar uma lista de tarefas. O programa deve permitir adicionar, remover e marcar tarefas como concluídas. Ele deve conter as seguintes funções:

    - `adicionar_tarefa(lista, tarefa)`: adiciona uma nova tarefa à lista de tarefas.
    - `remover_tarefa(lista, tarefa)`: remove uma tarefa da lista.
    - `concluir_tarefa(lista, tarefa)`: marca uma tarefa como concluída.
    - `mostrar_tarefas(lista)`: exibe a lista de tarefas, indicando quais foram concluídas.

    Dicas:

    - Utilize uma lista de dicionários, onde cada tarefa é um dicionário com as chaves `tarefa` e `concluida` (booleano que indica se a tarefa foi concluída).

11. Desenvolva um programa que permita ao usuário filtrar uma lista de produtos com base em critérios como preço e categoria. O programa deve conter as seguintes funções:

    - `adicionar_produto(produtos, nome, preco, categoria)`: adiciona um produto à lista de produtos com seu nome, preço e categoria.
    - `filtrar_por_preco(produtos, preco_minimo, preco_maximo)`: exibe todos os produtos cujo preço está dentro do intervalo fornecido.
    - `filtrar_por_categoria(produtos, categoria)`: exibe todos os produtos de uma categoria específica.

    Dicas:

    - Utilize uma lista de dicionários, onde cada produto tem as chaves `nome`, `preco` e `categoria`.
    - Filtre os produtos iterando sobre a lista e verificando os critérios.

12. Crie um jogo onde o programa escolhe uma palavra aleatória de uma lista, embaralha as letras e pede ao usuário para adivinhar a palavra correta. O programa deve conter as seguintes funções:

    - `embaralhar_palavra(palavra)`: embaralha as letras de uma palavra.
    - `escolher_palavra(lista_palavras)`: escolhe uma palavra aleatória de uma lista de palavras.
    - `jogar(lista_palavras)`: inicia o jogo e interage com o usuário.

    Dicas:

    - Utilize o módulo `random` para escolher a palavra e embaralhá-la.
    - Crie uma lista de palavras que o programa poderá utilizar.

13. Escreva um programa para calcular a média das notas de alunos em diferentes disciplinas. O programa deve permitir que o usuário insira as notas por disciplina e retorne a média de cada uma. Ele deve conter as seguintes funções:

    - `adicionar_nota(disciplinas, disciplina, nota)`: adiciona a nota de uma disciplina à lista de notas daquela disciplina.
    - `calcular_media_disciplinas(disciplinas)`: calcula e retorna a média de notas de cada disciplina.
    - `mostrar_disciplinas(disciplinas)`: exibe todas as disciplinas com as médias de notas calculadas.

    Dicas:

    - Utilize um dicionário onde as chaves são as disciplinas e os valores são listas de notas.
    - Para calcular a média de uma disciplina, some as notas da lista e divida pela quantidade de notas.

14. Escreva um programa que permita ao usuário adicionar produtos com preços e listar os produtos em ordem crescente ou decrescente de preço. Ele deve conter as seguintes funções:

    - `adicionar_produto(produtos, nome, preco)`: adiciona um produto à lista de produtos.
    - `ordenar_produtos(produtos, ordem)`: ordena os produtos por preço em ordem crescente ou decrescente, conforme a escolha do usuário.
    - `mostrar_produtos(produtos)`: exibe todos os produtos com seus preços.

    Dicas:

    - Utilize uma lista de dicionários, onde cada dicionário contém o nome e o preço de um produto.
    - A função `sorted()` pode ser usada para ordenar a lista com base no preço.

15. Desenvolva um programa para ajudar o usuário a controlar suas despesas pessoais. O programa deve permitir registrar despesas em diferentes categorias e calcular o total gasto em cada categoria. Ele deve conter as seguintes funções:

    - `adicionar_despesa(despesas, categoria, valor)`: registra uma nova despesa em uma categoria específica.
    - `calcular_total_categoria(despesas, categoria)`: calcula o total de despesas em uma categoria específica.
    - `mostrar_resumo_despesas(despesas)`: exibe o total de despesas em todas as categorias.

    Dicas:

    - Utilize um dicionário onde as chaves são as categorias e os valores são listas de despesas.
    - Para calcular o total, some todos os valores dentro da lista de uma categoria.