Instruções:

- A Lista 2 - Python deverá ser entregue via Google Classroom, conforme informações na tarefa
- A nota dessa Lista irá compor a média final do semestre
- O Arquivo deve ser nomeado da seguinte forma: 1-L1-CAIO.c onde
  - 1: o número do exercício
  - LI: o número da lista
  - CAIO: o seu nome em letra maiúscula e sem acento. Colocar o sobrenome caso outro aluno tenha o seu nome (CAIOSILVA)
  - .c: é a extensão do arquivo que pode ser .c ou .ipynb
  - Não colocar espaço entre as palavras, seguir exatamente o exemplo
- Coloque no cabeçalho de cada exercício o seu nome e o enunciado da questão.
- Para o arquivo em Python, entregue um arquivo para cada exercício.
- Farei comparação entre os exercícios, não copie do colega.
- Não utilize ferramentas de inteligência artificial.
- A explicação é o critério que mais vale pontos.
- Sugestão de aplicações on-line: Google Colab

1. Escreva um programa em Python que solicite ao usuário seu nome e exiba uma mensagem de boas-vindas personalizada, seguindo os critérios abaixo:

   (a) O nome deve ser validado para garantir que não contenha números ou caracteres especiais (apenas letras e espaços são permitidos). Caso contrário, solicite a entrada novamente.

   (b) O programa deve exibir a mensagem de boas-vindas com o nome corretamente formatado:
   - Apenas a primeira letra de cada nome deve estar em maiúscula (ex.: ”mArIa da sIlvA”deve ser corrigido para ”Maria Da Silva”).
   - O nome deve ser exibido com um espaçamento uniforme, removendo espaços extras.

   (c) Além da mensagem padrão `"Bem-vindo, <Nome>!"`, o programa deve incluir uma saudação que varia conforme o horário do dia:
   - Manhã (05h - 12h): `"Bom dia, <Nome>!"`
   - Tarde (12h - 18h): `"Boa tarde, <Nome>!"`
   - Noite (18h - 05h): `"Boa noite, <Nome>!"`

   (d) A mensagem final deve ser exibida com um design estilizado utilizando caracteres especiais (por exemplo, linhas ou caixas ASCII) para torná-la visualmente agradável.

2. Crie um programa em Python que peça ao usuário para digitar três números inteiros. O programa deve comparar os números e exibir o maior deles.

3. Escreva um programa em Python que solicite ao usuário uma sequência de números separados por espaço (por exemplo, ”2 4 6 8”). O programa deve usar um loop ‘for‘ para calcular a média desses números e exibir o resultado.

4. Escreva um programa em Python que peça ao usuário para digitar um número inteiro positivo. O programa deve gerar a sequência de Fibonacci até o número especificado usando um loop ‘while‘.

5. Crie um programa em Python que solicite ao usuário uma palavra e, em seguida, verifique se a palavra é um palíndromo (ou seja, se ela pode ser lida da mesma forma de trás para frente). O programa deve exibir ”A palavra é um palíndromo”ou ”A palavra não é um palíndromo”de acordo com o resultado.

6. Faça um programa na Linguagem Python que leia 2 números e em seguida pergunte ao usuário qual operação ele deseja realizar (soma, subtração, divisão ou multiplicação). O resultado da operação deve ser acompanhado de uma frase que diga se o resultado da operação é:

   - par ou ímpar;
   - positivo ou negativo;
   - inteiro ou decimal

7. Crie um programa em Python que peça ao usuário a senha “123senai”. Se a senha for incorreta, continuar pedindo até que a senha esteja correta. Em seguida, o usuário deve ter acesso a um gerador de tabuada, capaz de gerar a tabuada de qualquer número inteiro entre 1 a 10. O usuário deve informar de qual número ele deseja ver a tabuada. Se o número inserido for par, mostrar a tabuada dos números pares. Se não, imprimir os números ímpares.

8. Os números primos possuem várias aplicações dentro da Computação, por exemplo na Criptografia. Um número primo é aquele que é divisível apenas por um e por ele mesmo. Faça um programa em Python que peça um número inteiro e determine se ele é ou não um número primo.

9. Faça um programa em Python que calcule o fatorial de um número inteiro fornecido pelo usuário. Ex.: 5! =120.

10. Crie um programa em Python que simule um sistema de login com senha. O programa deve permitir ao usuário tentar até 3 vezes inserir a senha correta. Se a senha inserida estiver correta, o programa deve exibir uma mensagem de boas-vindas. Se a senha estiver errada após 3 tentativas, o programa deve exibir uma mensagem de erro.

    (a) O programa deve armazenar uma senha fixa (por exemplo, ”senha123”).

    (b) O programa deve permitir ao usuário tentar inserir a senha até 3 vezes. Para cada tentativa, o programa deve verificar se a senha está correta utilizando a estrutura `if`.

    (c) Caso a senha esteja correta, o programa deve exibir uma mensagem de boas-vindas e encerrar o loop de tentativas.

    (d) Caso a senha esteja errada após 3 tentativas, o programa deve exibir uma mensagem de erro e finalizar a execução.

    (e) Utilize o `while` para controlar o número de tentativas e o `for` para iterar sobre as tentativas de senha.

11. Crie um programa que simule uma conta bancária utilizando apenas listas, estruturas de controle `if-else` e `for`. O programa deve realizar as seguintes operações:

    (a) Crie uma lista chamada `conta`, onde o primeiro item será o `titular` (uma string) e o segundo item será o `saldo` (um número decimal, inicialmente 0).

    (b) O programa deve permitir ao usuário realizar os seguintes tipos de operação:

    - Depósito: o valor deve ser adicionado ao saldo da conta.
    - Saque: o valor deve ser subtraído do saldo, mas somente se houver saldo suficiente. Caso contrário, uma mensagem de erro deve ser exibida.
    - Exibir Saldo: o programa deve mostrar o saldo atual da conta.

    (c) O programa deve realizar um loop, onde o usuário pode escolher entre realizar uma operação de depósito, saque ou exibir o saldo. O loop deve continuar até que o usuário escolha a opção para sair.

12. Escreva um programa em Python que ajude a calcular a data de vencimento de um prazo processual, seguindo as regras abaixo:

    (a) O usuário deve informar:

    - A data inicial do prazo (no formato `DD/MM/AAAA`).
    - O número de dias corridos concedidos para o prazo.

    (b) O programa deve calcular a data final do prazo somando os dias corridos informados à data inicial, considerando que todos os dias (incluindo fins de semana e feriados) são contados.

    (c) O programa deve exibir a data final do prazo.