Lista 4 - Funções e Módulos

*Resolva os exercícios com a linguagem Python, tente seguir o caminho mais **pythônico** possível.*

1. Escreva uma função chamada `formatar_data` que recebe três parâmetros: dia, mês e ano. A função deve retornar uma string com a data formatada no formato "dd/mm/aaaa". Permita que os parâmetros sejam passados com seus respectivos nomes e em qualquer ordem. Teste a função com diferentes combinações de argumentos nomeados.
2. Implemente uma função chamada `aplicar_operacao` que recebe até três parâmetros: até dois números e uma função de operação (soma, subtração, multiplicação, divisão, resto, potência, raiz, fatorial, logaritmo, cosseno, seno, tangente). A função `aplicar_operacao` deve retornar o resultado da operação aplicada aos números. Crie funções separadas para cada operação matemática e teste a função `aplicar_operacao` com elas.
3. Para cada problema a seguir defina uma função **recursiva**, faça a simulação por substituição e desenhe o fluxo de chamadas e retornos:
   a. Calcular o resto da divisão inteira usando subtração.
   b. Calcular o quociente da divisão inteira usando subtração.
   c. Calcular o produto de dois naturais usando adição.
   d. Calcular a soma de dois naturais usando as funções `suc(n)` e `pred(n)` que devolvem, respectivamente, o sucessor e o predecessor de um natural n.
4. Crie uma função recursiva para calcular o n-ésimo termo de uma sequência definida por:$T(n) = T(n-1) + 2 \times T(n-2)$ com condições iniciais $T(0) = 0$ e $T(1) = 1$. Utilize memoização para evitar cálculos repetidos. Em seguida, defina uma expressão lambda que encapsule essa função recursiva com memoização.
5. Crie um pacote chamado `texttools` que contenha dois módulos:
   - **preprocessing.py**: contendo funções para limpar e tokenizar um texto (remover pontuações, transformar para minúsculas e dividir em palavras).
   - **statistics.py**: contendo funções para calcular a frequência de palavras e retornar as N palavras mais comuns, utilizando expressões lambda, `map` e compreensões.

   Em seguida, crie um programa principal que:
   1. Leia um arquivo de texto (ou uma string grande).
   2. Use o pacote `texttools` para processar o texto.
   3. Exiba as 5 palavras mais frequentes e suas contagens.
6. Crie um módulo que leia dados de um arquivo CSV contendo informações de vendas (por exemplo: `data, produto, quantidade, valor`). Implemente funções que:
   a. Leiam o arquivo e retornem uma lista de dicionários (cada linha do CSV como um dicionário).
   b. Calcule o total de vendas para cada produto.
   c. Utilize funções como `map`, `filter` e compreensões de dicionários para processar os dados.
7. Crie um pacote chamado `estatistica` com dois módulos:
   - **leitor.py:** Contendo uma função `ler_csv(caminho)` que leia um arquivo CSV e retorne uma lista de dicionários, convertendo automaticamente valores numéricos.

- **analise.py:** Contendo uma função `estatisticas(dados, campo)` que receba os dados lidos (lista de dicionários) e o nome de um campo numérico, e retorne um dicionário com estatísticas básicas (média, mínimo, máximo e total).

8. Crie um módulo chamado `introspecao.py` que contenha uma função chamada `mostrar_info(obj)`. Essa função deverá:
   - Exibir o nome do objeto (se possível).
   - Exibir seu identificador e tipo.
   - Imprimir sua docstring (caso exista).
   - Listar os métodos e atributos do objeto (usando a função `dir()`).

   Requisitos:
   - Utilize a função `help()` e os atributos especiais como `__doc__` e `__dict__` quando aplicável.
   - Inclua uma docstring detalhada na função explicando seu funcionamento.
9. Implemente um interpretador que avalie expressões aritméticas simples compostas pelas operações +, -, *, / e parênteses. Utilize funções recursivas para realizar o parsing e a avaliação da expressão. Além disso, defina as operações básicas usando expressões lambda em um dicionário de operadores.

   Requisitos:
   - Crie um módulo chamado `interprete.py` contendo:
     - Um dicionário que mapeia os operadores para funções lambda (por exemplo, `{'+' : lambda a, b: a + b, ...}`).
     - Uma função recursiva `avaliar(expressao)` que recebe uma string com a expressão e retorna o resultado numérico.
   - O interpretador deve lidar com a precedência dos operadores e com parênteses.
10. Desenvolva um simulador de jogo de dados onde, ao lançar os dados, diferentes eventos são disparados com base no resultado. Organize o código em um pacote chamado `jogo` com os módulos:
    - **dados.py:** Função para simular o lançamento de um dado (usando `random.randint`).
    - **eventos.py:** Implementação de um sistema de eventos que permite registrar callbacks para determinados resultados (por exemplo, se o dado mostrar 6, disparar o evento "sorte").

    Requisitos:
    - No módulo `eventos.py`:
      - Implemente funções para registrar e disparar eventos (use um dicionário para mapear eventos a listas de callbacks).
      - Crie um decorador `@evento(nome)` que registre automaticamente uma função para um evento.
    - No módulo `dados.py`:
      - Implemente uma função `lançar_dado()` que retorne um número aleatório de 1 a 6.
    - No script principal (`main.py`), integre os módulos para:
      - Registrar callbacks usando o decorador.
      - Simular o lançamento do dado e disparar o evento correspondente.
      - Exibir mensagens personalizadas para cada evento disparado.