Profa. Nádia Félix
Projeto TJ: Python - Análise de Dados
February 10, 2025

Tarefa 1: Iteração, Sequências e Dicionários

Instruções sobre como redigir e enviar seu trabalho estão disponíveis na página do curso. O não cumprimento dessas instruções resultará na perda de pontos. Direcione quaisquer perguntas ao professor.

Leia isto primeiro.

1) Comece cedo! Se você encontrar dificuldades ou tiver dúvidas, é melhor identificar esses problemas com antecedência, e não nas horas anteriores ao prazo de entrega!
2) Caso tenha dúvidas ou encontre problemas, não envie e-mails diretamente ao professor. Publique no fórum de discussão para que seus colegas também possam se beneficiar caso tenham a mesma dúvida.
3) Certifique-se de fazer backup do seu trabalho! Recomenda-se, no mínimo, salvar em uma pasta do Dropbox ou, melhor ainda, usar o Git, que vale o esforço de aprender.

Questão 1. Diversão com Strings

Nesta tarefa, você implementará funções simples para manipulação de strings. Não é necessário realizar verificações de erro. Pense cuidadosamente sobre o comportamento esperado em casos especiais (e.g., string vazia, strings com comprimento 1, etc.).

1) Palíndromos: Um palíndromo é uma palavra ou frase que se lê da mesma forma para frente e para trás. Exemplo: "level", "kayak", "pop". Frases como "rats live on evil star" e "Was it a car or a cat I saw?" também são palíndromos se ignorarmos espaços e pontuação.
   (a) Escreva uma função chamada is_palindrome, que recebe uma string como argumento e retorna um valor booleano (True se for palíndromo e False caso contrário).
   (b) Ignore espaços e capitalização ao avaliar se a string é um palíndromo. Por exemplo, "tacocat" e "T A C O cat" devem ser considerados palíndromos.
2) Palavras abecedárias: Uma palavra é "abecedária" se suas letras aparecem em ordem alfabética. Exemplo: "adder" e "beet" são abecedárias, enquanto "dog" e "cat" não são.
   (a) Escreva uma função chamada is_abecedarian que receba uma string e retorne True se a string for abecedária e False caso contrário. Ignore espaços e capitalização.
   (b) Remover vogais: Escreva uma função chamada remove_vowels que receba uma string e retorne a string sem vogais. As vogais incluem: a, e, i, o, u (maiúsculas e minúsculas).
      (i) Exemplo:
         (A) remove_vowels('cat?') -> ct?
         (B) remove_vowels('goAT!') -> gT!
         (C) Não use métodos embutidos como str.replace.

Questão 2. Diversão com Listas

Nesta tarefa, você implementará operações simples com listas.

1) Reverso de uma lista: Escreva a função list_reverse, que recebe uma lista e retorna uma nova lista com os elementos em ordem reversa.
   (a) Exemplo:
      (i) list_reverse([1, 2, 3]) -> [3, 2, 1]
      (ii) Se a entrada não for uma lista, apresente uma mensagem de erro apropriado.
2) Verificar ordenação: Escreva a função is_sorted, que recebe uma sequência e retorna True se a sequência estiver em ordem não decrescente e False caso contrário.
   (a) A solução deve usar apenas uma passagem pela lista para verificar a ordenação.
3) Busca binária: Escreva a função binary_search, que recebe uma lista ordenada e um elemento, e retorna True se o elemento estiver presente na lista e False caso contrário.
   - Implemente a busca binária sem usar funções embutidas como in.
   - Considere cuidadosamente os casos base: listas vazias, de comprimento 1, etc.

Questão 3. Mais diversão com Strings

1) Histograma de caracteres: Escreva a função char_hist, que recebe uma string e retorna um dicionário onde as chaves são caracteres e os valores são o número de vezes que cada caractere aparece. Ignore maiúsculas e minúsculas, e trate 'G' e 'g' como o mesmo caractere.
2) Histograma de bigramas: Um bigrama é um par de caracteres consecutivos. Escreva a função bigram_hist que recebe uma string e retorna um dicionário com bigramas como chaves e o número de ocorrências como valores.

Questão 4. Tuplas como Vetores

1) Multiplicação escalar: Escreva a função *vec_scalar_mult*, que realiza a multiplicação de uma tupla por um número escalar.
   (a) Verifique se os tipos dos argumentos são válidos, levantando um erro apropriado se não forem.
2) Produto interno: Escreva a função *vec_inner_product* que calcula o produto interno de dois vetores representados por tuplas.
3) Matriz válida: Escreva a função *check_valid_mx*, que verifica se um argumento é uma matriz válida representada por uma tupla de tuplas.
4) Multiplicação *matrizvetor*: Escreva a função *mx_vec_mult*, que realiza a multiplicação de uma matriz por um vetor.

Questão 5. Mais diversão com Vetores

1) Vetores esparsos: Escreva a função *is_valid_sparse_vector* que verifica se uma entrada é um vetor esparso válido (dicionário com chaves inteiras não negativas e valores numéricos).
2) Produto interno esparso: Escreva a função *sparse_inner_product* que calcula o produto interno de dois vetores esparsos.