Universidade Federal Fluminense
Programação de Computadores
Prof. Luís Felipe Ignácio Cunha

Maratona 3 – Tuplas e Dicionários

Esta lista aborda tuplas e dicionários em Python, integrando conteúdos já vistos na disciplina, como listas, strings, subprogramação, recursividade, busca e ordenação.

1. **Agenda simples com dicionário**

   (a) **Objetivo:** Implementar um programa que constrói uma agenda telefônica utilizando um dicionário, em que cada nome está associado a um telefone. Além disso, se um nome já estiver na agenda, deve ser pedido outro nome ao usuário.

   (b) **Dados de entrada:** Cinco pares de valores, cada par composto por um nome (string) e um telefone (string ou inteiro), lidos do teclado.

   (c) **Saída esperada:** Ao final da leitura, o programa deve imprimir todos os contatos armazenados, um por linha, no formato:

   ```
   nome: telefone
   ```

2. **Contador de letras em uma string**

   (a) **Objetivo:** Implementar uma função que conte quantas vezes cada letra aparece em uma string, utilizando um dicionário.

   (b) **Dados de entrada:** Uma linha de texto (string), podendo conter letras, espaços e outros caracteres.

   (c) **Saída esperada:** Impressão de um dicionário em que cada chave é uma letra (considerando tudo em minúsculas e ignorando espaços) e o valor é a quantidade de ocorrências daquela letra na string.

3. **Tupla de dados pessoais**

   (a) **Objetivo:** Representar dados pessoais em uma tupla e exibir suas informações após desempacotamento.

   (b) **Dados de entrada:** Três valores lidos do teclado: nome (string), idade (inteiro) e altura (número real).

   (c) **Saída esperada:** Impressão das informações no formato:

   ```
   Nome: ...Idade: ...Altura: ...
   ```

   utilizando o desempacotamento da tupla para acessar os valores.

4. **Dicionário de notas e média da turma**

   (a) **Objetivo:** Armazenar em um dicionário as notas de alunos e determinar quais alunos ficaram acima da média da turma.

   (b) **Dados de entrada:**

   - Um número inteiro n representando a quantidade de alunos.
   - Para cada aluno, um nome (string) e uma nota (número real).

   (c) **Saída esperada:**

   - A média aritmética das notas da turma.
   - A lista dos alunos com nota estritamente maior do que a média, cada um com seu nome e nota.

5. **Consulta de nota**

   (a) **Objetivo:** Consultar a nota de um aluno em um dicionário, tratando o caso de aluno inexistente.

   (b) **Dados de entrada:**

   - Um dicionário já preenchido, no formato `nome` $\rightarrow$ `nota`.
   - Um nome (string) a ser consultado, lido do teclado.

   (c) **Saída esperada:**

   - Se o nome estiver no dicionário, imprimir a nota do aluno.
   - Caso contrário, imprimir uma mensagem informando que o aluno não foi encontrado.

6. **Conversão de dicionário para lista de tuplas**

   (a) **Objetivo:** Transformar um dicionário em uma lista de tuplas, facilitando outras operações sobre os dados.

   (b) **Dados de entrada:** Um dicionário `notas` no formato `nome` $\rightarrow$ `nota`, já preenchido no programa.

   (c) **Saída esperada:** Uma lista de tuplas no formato:

   $$[(\texttt{nome\_1}, \texttt{nota\_1}), (\texttt{nome\_2}, \texttt{nota\_2}), \ldots]$$

   impressa na tela.

7. **Soma de matrizes esparsas**

   (a) **Objetivo:** Somar duas matrizes esparsas representadas por dicionários que usam tuplas como chaves.

   (b) **Dados de entrada:** Dois dicionários A e B representando matrizes esparsas, com as seguintes chaves:

   - `"Número de Linhas"` e `"Número de Colunas"`, indicando as dimensões.
   - Tuplas (i, j) representando posições da matriz com valores inteiros diferentes de zero.

(c) **Saída esperada:** Um novo dicionário C representando a matriz soma, com mesmas chaves de dimensão e apenas as posições (i, j) cujo valor resultante seja diferente de zero, impressos de forma adequada.

8. **Dicionário de temperaturas com tuplas como chave**

   (a) **Objetivo:** Armazenar temperaturas de cidades em diferentes dias, utilizando tuplas como chaves em um dicionário.

   (b) **Dados de entrada:**

   - Um número inteiro n representando a quantidade de leituras.
   - Para cada leitura: uma cidade (string), um dia (string) e uma temperatura (inteiro).
   - Ao final, uma cidade C (string) cuja média de temperatura deve ser calculada.

   (c) **Saída esperada:**

   - A média das temperaturas cadastradas apenas para a cidade C.
   - Uma mensagem informando que não há dados, caso a cidade C não possua registros.

9. **Agrupamento de alunos por nota**

   (a) **Objetivo:** Construir um novo dicionário que agrupa alunos de acordo com a nota obtida.

   (b) **Dados de entrada:** Um dicionário `notas` no formato `nome` $\rightarrow$ `nota`, previamente preenchido.

   (c) **Saída esperada:** Um dicionário `agrupado` no formato:

   `nota` $\rightarrow$ `[lista de nomes com essa nota]`

   e a impressão desse dicionário na tela.

10. **Frequência de palavras em uma frase**

    (a) **Objetivo:** Contar quantas vezes cada palavra aparece em uma frase, utilizando um dicionário.

    (b) **Dados de entrada:** Uma linha de texto (string) contendo uma frase, lida do teclado.

    (c) **Saída esperada:**

    - Um dicionário em que a chave é a palavra e o valor é a quantidade de ocorrências.
    - A impressão das palavras em ordem alfabética, juntamente com suas contagens.

11. **Fibonacci com memoização usando dicionário**

    (a) **Objetivo:** Implementar uma função recursiva para calcular o n-ésimo número de Fibonacci, utilizando um dicionário para armazenar resultados já calculados (memoização).

    (b) **Dados de entrada:** Um número inteiro $n \geq 0$, lido do teclado.

    (c) **Saída esperada:** O valor de $F(n)$, correspondente ao n-ésimo número da sequência de Fibonacci, impresso na tela.

12. **Agrupando palavras por tamanho**

    (a) **Objetivo:** Agrupar palavras de uma frase de acordo com o seu tamanho, utilizando um dicionário em que as chaves são comprimentos de palavras.

(b) **Dados de entrada:** Uma linha de texto (string) contendo uma frase, lida do teclado.

(c) **Saída esperada:**

- Um dicionário em que cada chave é um inteiro representando o tamanho das palavras, e o valor é a lista de palavras com aquele tamanho.
- A impressão do dicionário, mostrando os comprimentos em ordem crescente.

13. **Boletim com tupla de três notas**

(a) **Objetivo:** Representar o boletim de uma turma em um dicionário, utilizando tuplas para armazenar as três notas de cada aluno.

(b) **Dados de entrada:**

- Um número inteiro n representando a quantidade de alunos.
- Para cada aluno: um nome (string) e três notas (números reais).

(c) **Saída esperada:**

- A média de cada aluno.
- Uma lista de pares `(nome, média)` ordenada em ordem decrescente de média.
- Para cada aluno, a indicação de situação: `APROVADO` (média ≥ 6), `RECUPERAÇÃO` ($4 \leq$ média < 6) ou `REPROVADO` (média < 4).

14. **Busca binária recursiva em lista de tuplas**

(a) **Objetivo:** Implementar a busca binária de forma recursiva em uma lista de tuplas, para localizar o preço associado a um código.

(b) **Dados de entrada:**

- Um número inteiro n representando a quantidade de produtos.
- Uma lista de n tuplas `(código, preço)`, com os códigos inteiros em ordem crescente.
- Um código (inteiro) a ser buscado na lista.

(c) **Saída esperada:**

- Se o código for encontrado, o preço correspondente deve ser impresso.
- Caso o código não esteja presente, deve ser impressa uma mensagem informando que o produto não foi encontrado.

15. **Sistema de cache com dicionário e tuplas como chaves**

(a) **Objetivo:** Implementar um sistema simples de cache para uma função "cara", utilizando um dicionário que mapeia tuplas de parâmetros para resultados.

(b) **Dados de entrada:**

- Diversos conjuntos de parâmetros numéricos (a, b, c), lidos do teclado, que serão usados como entrada para a função.

(c) **Saída esperada:**

- Para cada chamada, o resultado da função aplicada a (a, b, c).
- Mensagens que indiquem quando o resultado foi obtido a partir de um cálculo novo e quando foi recuperado do cache (dicionário).