**Lista de Exercícios**

*Cada exercício deve estar em um arquivo separado com a indicação do exercicio: ex1.py, ex2.py, etc. Os arquivos devem estar em um repositório privado no GitHub que deve ser compartilhado com o professor:* ***alexandre-prof-up****. No blackboard deve ser enviado* ***APENAS*** *o link para o repositório. Quaisquer comentários sobre os exercícios devem estar no Readme do GitHub.*

** Não utilize bibliotecas/módulos para uso de outras estruturas (numpy, collections, etc). Neste momento, utilizas apenas as estruturas estudadas nas aulas (listas, tuplas, sets, dicionários). Pode-se utilizar, contudo, bibliotecas auxiliares como 'math', 'random', e outras para decorações de saída, se for o caso.*

*-Fazer um cabeçalho para cada arquivo:*

*-Explicação da estratégia;*

*-Detalhamento das estruturas usadas*

*-Documentar cada função com docstring*

**Usando listas:**

1) Crie uma versão do jogo da velha 4x4. As regras são as mesmas da versão 3x3.

2) Crie um jogo da velha NxN em que o usuário deve definir as dimensões do tabuleiro (sempre quadrado).

3) Desenvolver o jogo https://term.ooo/ a partir do arquivo lista_palavras.txt. O jogo deve ser jogado por meio do terminal, mantendo a lógica do jogo original. Devem aparecer as letras que já foram descobertas, as letras já tentadas no teclado e assim por diante. Atente-se à formatação.

**Prática de dicionários:**

4) "Banco de dados" de dicionários.

Crie um menu com três opções:

**1**-cadastrar usuário

**2**-imprimir usuários

**0**-encerrar

Ao iniciar, o programa deve solicitar ao usuário os nomes dos campos que serão obrigatórios para os cadastros.

Na sequência, deve mostrar o menu e iniciar o fluxo normal de execução.

**Opção 1**. Crie uma função cadastrar_usuario para cadastrar um usuário de maneira flexível.

A função deve receber uma tupla com os campos obrigatórios para cadastro.

Estes campos devem ser definidos globalmente em tempo de execução pelo usuário. (opção anterior)

O usuário pode cadastrar quantos campos quiser além dos obrigatórios até digitar "sair".

Deve então retornar ao menu.

A função deve retornar o dicionário gerado e armazenar no dicionário global chamado banco_usuarios

**Opção 2**. Criar uma outra função imprimir_usuarios com 4 possibilidades de invocação:

- Caso a função não receba argumentos, deve imprimir todos os usuários com todas suas infos.

- Caso receba vários nomes, deve imprimir todos dados de todos os usuários com os nomes especificados.

exemplo: imprimir_usuarios("alberto", "joaquina", "enzo", "valentina")

- Caso receba uma série de elementos campo e valor deve imprimir apenas os dados completos

de todos os usuários que satisfazem as condições:

- Receber nomes e campos, valor: apenas nos usuários listados, imprimir os dados dos usuários que

satisfazem às condições dadas. (usar *args e **kwargs facilita a defin=ção da fun=ção)

exemplo de execução da opção 2:

1 - impirmir todos

2 - filtrar por nomes

3 - filtrar por campos

4 - filtrar por nomes e campos

**2**

*alberto, maria, enzo*

>> deve imprimir os dados de alberto, maria e enzo

**outro exemplo**

3

**digite o campo de busca:**

*endereço*

**digite o endereço**

*xv de novembro*

**mais algum campo:**

*idade*

**digite a idade**

*20*

**mais algum campo?**

*sair*

*>> Deve imprimir os dados dos usuários que possuem o endereço xv de novembro a e idade 20. Retorna ao menu inicial*

**outro exemplo:**

4

**digite os nomes**

*alberto, maria, jose*

**digite os campos**

*idade*

**digte a idade**

*10*

**outro campo**

*sair*

*>> deve imprimir os dados dos usuários, entre alberto, maria e josé, que possuem 10 anos. Retorna ao menu inicial.*