Exercícios de Revisão – prof. Joísa – 2022/2

Não serão disponibilizadas as soluções. Devem ser buscadas as monitorias de reforço. O conteúdo trabalhado nos exercícios pode ser cobrado em avaliações.

Exercício 1:
Nas funções dos itens 1A e 1B abaixo dê preferência ao comando de repetição while

1.A) Escreva uma função, denominada gera_determinada, que receba um número inteiro x e gere x vezes randomicamente um inteiro entre 1 e 4. Cada vez que um número seja gerado (1,2,3 ou 4) deve ser exibido o número por extenso ('um', 'dois','três' ou 'quatro'). Imediatamente antes do término da função deve ser exibida a mensagem fim .
IMPORTANTE: teste sua função chamando-a no bloco principal

1.B) Escreva uma função, denominada gera_indeterminada, que gere repetidas vezes randomicamente um inteiro entre 0 e 4. Caso seja gerado o valor 0, deve ser exibida a mensagem fim e a função termina. Caso seja gerado 1,2,3 ou 4 deve ser exibido o número por extenso.
Observe que nesse caso é necessário gerar um número antes da entrar na repetição (antes do while)
IMPORTANTE: teste sua função chamando-a no bloco principal

1.C) Escreva uma funcao (não recursiva) que receba uma string e um caractere e retorne o numero de ocorrencias do caractere. Faça uso de uma estrutura de repeticao dentro da funcao. Atenção: caso o caractere corresponda a uma letra devem ser levadas em conta as ocorrências tanto de maiúsculas quanto de minúscula correpondentes a esse caractere
Obs: não use o método count

Escreva no bloco principal código para ler uma string e um caractere e testar sua função

Exercício 2:

Q2A)
Escreva uma função, denominada exibe_iguais_na_posicao, que receba duas strings e, para as posições (índices) em que em ambas as strings tenham um mesmo caractere, exiba posição e caractere.

Q2B)
Escreva uma função, denominada decodifica_texto, que:

- receba 3 strings:
  - a primeira com um texto codificado a ser decodificado e exibido
  - a segunda e a terceira strings definindo toda a codificação usada, sendo respectivamente a msg original e a mesma msg codificada, escrita de acordo com a codificação adotada
- construa e retorne uma string correspondente ao texto decodificado

Obs1: apenas alguns caracteres foram mudados na codificação

Obs2: o texto a ser decodificado e a mensagem original não possuem acentuação, caracteres especiais, dígitos(algarismos) ou maiúsculas

Dica: use .find para encontrar um caractere na msg codificada

Exemplo:
A chamada da função com o texto codificado '*m!=m?s4*@!=d&=c+@&s=&=s&54?m&54+s' e a msg 'arquipelago tranquilo' e sua respectiva msg codificada '!@1*?2&3!-+=4@!51*?3+', retornaria a seguinte string resultante da decodificação:
'uma mistura de cores e sentimentos'

No BP teste sua função duas vezes: a primeira com o texto codificado do exemplo acima e a segunda com o texto codificado '!@3&1*?5=&s4@!-+*=!=f&s4!'

Exercício 3:

Escreva uma função, denominada para_um_encontro, que:
- receba uma lista com as 3 coisas favoritas de uma pessoa em busca de romance e uma lista de possíveis parceiros e suas respectivas 3 coisas favoritas
- retorne 1 lista com os possíveis parceiros que tenham ao menos 2 coisas favoritas em comum.

É conveniente fazer uma função auxiliar que retorne quantas coisas comuns há em 2 listas de coisas favoritas.

Escreva um programa completo para testar sua função.

Considere a lista abaixo de possíveis parceiros para uma pessoa

```
lparc= [['lala', ['restaurante','bar','praia']],
      ['nena', ['cinema','leitura','praia']],
      ['vivi', ['cinema','leitura','teatro']],
      ['lele', ['restaurante','bar','praia']],
      ['nina', ['cinema','leitura','praia']],
      ['tita', ['cinema','leitura','teatro']],
      ['leda', ['viagem','danca','esporte']],
      ['gege', ['praia','leitura','viagem']],
      ['tati', ['cinema','leitura','dormir']],
      ['babi', ['viagem','cachorro','praia']],
      ['tata', ['cachorro','leitura','praia']],
      ['zaza', ['cinema','gato','cachorro']]
      ]
```

Por exemplo, com a lista de coisas favoritas da pessoa contendo ['praia','leitura','tv'] e a lista de parceiros lparc acima seria retornada:

```
['nena', 'nina', 'gege', 'tata']
```

Exercício 4:

Considere as listas lHerois e lVotos em que lVotos[K] é a quantidade de pessoas que escolheram lHeroi[K] como seu heroi favorito.
Assuma que as quantidades de votos são distintas.

Escreva uma função, denominada cria_lista_ordenada_dec, que receba duas listas como as descritas e construa e retorne uma nova lista, ordenada decrescentemente por qtdVotos, em que cada elemento é uma listinha [heroi,qtdVotosDoHeroi] . As listas originais devem ficar vazias ao final da função.

A ideia de uma possível solução é determinar o máximo de votos da lista e o heroi correspondente, retirando ambos de suas listas originais e adicionando um novo elemento no final da nova lista.
Repetir esse procedimento até que as listas originais fiquem vazias (basta testar uma delas)

Essa solução deve usar adequadamente:
- a função len, que recebe uma lista e retorna seu tamanho
- a função max, que recebe uma lista e retorna o maior valor da lista
- o método index de lista, que retorna a posição de um elemento na lista
- o método pop de lista ou o método remove, para fazer a retirada de elemento da lista
- o método append

Teste sua função com as listas abaixo:

```
lHeroi=['batman','superman','wolverine','vampira','tempestade',
        'xavier','hulk','flash','spyderman','venom']
lVotos=[ 2990, 2100, 3350, 1122, 1213,
        2451, 2855, 1002, 2705, 1567]
```

Para as listas acima a lista retornada é:

```
[['wolverine', 3350], ['batman', 2990], ['hulk', 2855], ['spyderman', 2705],
 ['xavier', 2451], ['superman', 2100], ['venom', 1567], ['tempestade', 1213],
 ['vampira', 1122], ['flash', 1002]]
```

Exercício 5:

Em um arquivo de nome apostas16junho.txt há o nome, a quantidade de apostas e as apostas de 6 números de jogadores da megasena, organizados da seguinte forma:

```
NOME_DO_JOGADOR_1,QTD_APOSTAS_JOG1
os seis números da primeira aposta do jogador1 separados por ,
os seis números da segunda aposta do jogador1 separados por ,
...
NOME_DO_JOGADOR_2,QTD_APOSTAS_JOG2
os seis números da primeira aposta do jogador2 separados por ,
...
```

Escreva um programa que simule um sorteio da megasena.

Para isso devem ser sorteados 6 numeros entre 1 e 60 sem repetição (dica: use random.sample, importando o módulo random). Exiba os números sorteados.
Em seguida devem ser lidos os dados do arquivo e informado, para cada jogador, cada aposta e quantos números ele acertou na aposta.

Exercício 6:

Escreva um programa para processar o resultado de uma campanha pró-animais solicitando doações em dinheiro. Ao final devem ser exibidos o total arrecadado, o número de pessoas que fizeram doações, os valores mínimo, médio e máximo das doações.

Não se sabe antecipadamente quantas doações serão feitas. Deve-se optar por um indicador de final de entrada de dados. Nesse caso, qualquer valor menor ou igual a 0 indica o fim da arrecadação (leitura) das doações.

Obs: trata-se de uma repetição indeterminada, não se sabe quantas serão as doações. NESSE CASO É NECESSÁRIO FAZER UMA PRIMEIRA LEITURA ANTES DA REPETIÇÃO