Computação I

Lista de exercícios 8 – Dicionários e módulos
Atenção! Leia as instruções antes de fazer a lista! - Data de entrega: 16/11/2022
A menos que esteja explicitamente pedido na questão, não utilize nenhuma função ou método já existente no Python exceto pelas funções print, len, range, e pelo método append. **Atenção! Os métodos keys e values não são permitidos**! Não utilize funções de transformação de tipo. Importe somente o módulo pedido nas questões 3 e 4. Não é necessário testar se os dados passados por argumento são válidos a menos que pedido na questão.

1. Faça uma função que recebe, por argumento, um número inteiro de 1 a 10 e retorna a string do número romano referente àquele número. Defina um dicionário dentro da função que permita o mapeamento (nesse caso as chaves devem ser números). Se o número passado não for válido, retorne uma string vazia. Para verificar se o argumento é válido ou não, defina o dicionário, utilize o operador in e veja se o argumento é uma chave pertencente ao dicionário (lembre-se que ao utilizar o operador in com o dicionário, o in verifica dentre as chaves).

2. A moda é uma medida estatística que corresponde ao elemento que aparece mais vezes dentro de um conjunto de dados. Um conjunto de dados pode conter múltiplas modas, caso dois ou mais elementos apareçam um mesmo número de vezes, e sejam os mais frequentes do conjunto. Por exemplo: a lista [0,0,4,3,4,2,1] apresenta duas modas: 0 e 4 (observe que ambos os elementos aparecem duas vezes, e são os mais frequentes da lista, já que os demais aparecem apenas uma vez). Faça uma função em Python que receba como argumento de entrada uma lista de valores e retorne uma lista contendo os elementos que representam as modas da lista de entrada.

3. Um professor que costuma preparar provas de múltipla escolha percebeu uma tendência ao preparar os gabaritos de suas provas, de forma que uma das letras sempre possuía muito mais acertos que as outras. Para evitar essa tendência, ele deseja um programa que defina aleatoriamente qual deve ser a letra certa para cada uma das questões. Escreva uma função que recebe por argumento, e nessa ordem, um inteiro que representa a quantidade de questões e uma lista de strings com as possíveis letras da múltipla escolha. Sua função deve retornar um dicionário onde as chaves são os números das questões (do tipo inteiro, começando de 1 e indo de 1 em 1 até o número de questões) e os valores são strings que representam os gabaritos. Para cada questão, o gabarito deve ser uma das letras da lista passada por argumento, escolhida de forma aleatória. Para fazer a escolha aleatória, importe o módulo random e utilize obrigatoriamente a função choice (ou importe diretamente a função choice do módulo random). Por exemplo, se a entrada for (5,['a','b','c']), isso quer dizer que há 5 questões com 3 múltiplas escolhas cada (a, b ou c). Como há 5 questões, então o dicionário precisa ter 5 pares chave-valor e as chaves serão de 1, 2, 3, 4 e 5. Utilizando a função random.seed(0) antes de fazer a chamada da função questao3 com entrada (5,['a','b','c']), a saída é {1: 'b', 2: 'b', 3: 'a', 4: 'b', 5: 'c'}.

4. Escreva uma função em Python que receba duas tuplas de entrada, contendo informações sobre dois personagens de um jogo de RPG. Todos os elementos são inteiros positivos, e representam, nesta ordem: os pontos de vida do personagem, o dano mínimo causado pela arma do personagem, o dano máximo causado pela arma do personagem, os pontos de armadura do personagem, e a chance de causar um dano crítico (em porcentagem). A função deve simular uma batalha entre ambos os personagens. O vencedor será o primeiro a reduzir os pontos de vida de seu oponente a 0. A batalha é dividida em turnos, onde em cada turno os personagens começam com uma determinada quantidade de pontos de vida e se atacam uma vez. O primeiro ataque é sempre feito pelo primeiro personagem. O dano que um personagem causa no outro é obtido da seguinte forma: primeiro, sorteia-se aleatoriamente um número inteiro entre o dano mínimo e máximo que aquele personagem pode causar. Depois, sorteia-se aleatoriamente um outro número entre 0 e 100, que determinará se o dano a ser causado será crítico ou não, baseado na chance de dano crítico

daquele personagem. Se o dano for crítico, então o dano total será o dano sorteado inicialmente multiplicado por 2. Caso contrário, o dano total será o próprio dano sorteado inicialmente. Por fim, se o dano total do personagem for maior do que os pontos de armadura de seu oponente, então a quantidade de pontos de vida que o oponente perderá após esse ataque será o dano total subtraído de seus respectivos pontos de armadura. Caso contrário, o oponente não perderá pontos de vida naquele turno. Um personagem será derrotado se a quantidade restante de pontos de vida, após o ataque de seu oponente, for menor ou igual a 0. Considere que um personagem só pode atacar o outro caso ainda tenha pontos de vida restantes (isto é, um valor superior a 0). Ou seja, o segundo personagem só poderá atacar o primeiro caso tenha pontos de vida restantes após o ataque do primeiro personagem, e vice-versa. A função deve retornar uma tupla, cujos elementos são, nesta ordem: o número 1 ou 2 (dependendo de qual jogador venceu a batalha), a quantidade de pontos de vida restantes do vencedor, e quantos turnos decorreram até a batalha se encerrar. Para o sorteio de números aleatórios, utilize obrigatoriamente a função randint do módulo random.

5. Escreva uma função que recebe por argumento dois dicionários, onde o primeiro contém os nomes das matérias que um aluno fez em um período (strings) como chaves e notas de cada uma dessas matérias (floats) como valores; e o segundo contém como chaves os nomes de diversas matérias do curso do aluno (strings) como chaves e a quantidade de créditos dessas matérias (ints) como valores. A sua função deve retornar o CR do aluno naquele período sabendo que o CR é calculado como uma média ponderada, onde a quantidade de créditos é usada como peso. Por exemplo, se as entradas forem ({'Calculo I': 8.0,'Computacao I': 9.5}, {'Quimica': 2,'Calculo I': 5,'Computacao I': 4, 'Fisica Experimental': 3}), a saída deve ser 8.666666666666666, pois (8*5 + 9.5*4)/(5 + 4) = 8.666666666666666. Considere que as strings com nomes das matérias serão escritas exatamente da mesma forma e que todas as chaves do primeiro dicionário serão também chaves do segundo dicionário.

6. Escreva uma função em Python que receba uma tupla de tuplas, representando os times que ocuparam as quatro primeiras colocações de um campeonato realizado em diferentes temporadas. Cada subtupla representa a classificação de um ano específico, e ela contém quatro strings, em que a ordem das strings nesta subtupla corresponde às colocações dos times naquela temporada do campeonato. Por exemplo: para a subtupla (‘Time B’, ‘Time D’, ‘Time A’, ‘Time C’) que representa a classificação em uma certa temporada, o “Time B” ficou em primeiro, o “Time D” em segundo, o “Time A’ em terceiro, e o “Time C” em quarto. A função deve retornar um dicionário contendo, como chaves, os times que já apareceram nas quatro primeiras posições em pelo menos uma temporada do campeonato e, como valores, uma lista contendo as colocações que o respectivo time assumiu em cada temporada do campeonato (cada posição desta lista representa uma temporada específica). Se um time, em alguma temporada, não aparecer entre as quatro primeiras posições, o elemento referente àquela temporada, na lista associada ao time em questão, deve aparecer com o valor ‘-’. Por exemplo: para a entrada ((‘Time A’, ‘Time D’, ‘Time C’, ‘Time B’), (‘Time D’, ‘Time B’, ‘Time E’, ‘Time A’), (‘Time B’, ‘Time C’, ‘Time D’, ‘Time A’)), a saída deve ser {‘Time A’: [1, 4, 4], ‘Time B’: [4, 2, 1], ‘Time C’: [3, ‘-’, 2], ‘Time D’: [2, 1, 3], ‘Time E’: [‘-’, 3, ‘-’]} (a ordem em que as chaves são exibidas pode mudar, mas o conteúdo de cada uma delas deve ser exatamente as respectivas listas acima). O ‘Time A’ foi primeiro colocado na primeira temporada do campeonato (pois está na primeira posição da primeira subtupla), e depois quarto colocado no segundo e terceiro anos de campeonato (pois está na quarta posição da segunda e da terceira subtuplas). Portanto, a lista associada ao ‘Time A’ é [1, 4, 4]. Já o ‘Time C’ foi terceiro colocado na primeira temporada, não figurou entre os quatro primeiros na segunda temporada (porque ele não aparece na segunda subtupla), e foi segundo colocado na terceira temporada. Logo, sua lista é [3, ‘-’, 2]. O ‘Time E’ não figurou entre os quatro primeiros colocados na primeira e na terceira temporada, enquanto que, na segunda temporada, ele ficou em terceiro lugar. Portanto, sua lista é [‘-’, 3, ‘-’].