Computação I

Lista de exercícios 9 – Valores padrão para argumentos e tratamento de exceção
Atenção! Leia as instruções antes de fazer a lista! - Data de entrega: 01/12/2022
Não importe nenhum módulo. Não é necessário testar se os dados passados por argumento são válidos a menos que pedido na questão. Veja no enunciado da questão se há alguma restrição para a utilização de métodos e funções do Python. Se nada for dito no enunciado, eles estão liberados. Utilize tratamento de exceção nas questões 2, 3 e 4.

1. Faça uma função em Python que receba como entradas, nesta ordem: uma lista de números ordenada de forma crescente, e uma string indicando uma determinada informação estatística que se deseja extrair da lista de entrada. As informações que podem ser especificadas pelo segundo argumento são: 'media', 'mediana' e 'variancia'. A média que deve ser calculada pela função é a média aritmética dos números da lista de entrada. A mediana de um conjunto de dados ordenado corresponde ao valor numérico central que divide o conjunto de entrada em duas metades iguais (isto é, metade dos elementos do conjunto devem ser menores ou iguais do que a mediana, e metade dos elementos do conjunto devem ser maiores ou iguais mediana). Se um conjunto numérico ordenado de forma crescente ou decrescente tiver um número ímpar de elementos, então a mediana será o elemento central do conjunto. Por exemplo: para o conjunto (1,2,3,4,5), a mediana é 3. No entanto, se o conjunto tiver um número par de elementos, então a mediana será a média aritmética entre os dois elementos centrais do conjunto. Por exemplo: para o conjunto (1,2,3,4), os dois elementos centrais são '2' e '3' e, portanto, a mediana do conjunto será (2+3)/2 = 2,5. Já a variância de um conjunto de N números $(x_1, x_2, \cdots, x_n)$ pode ser calculada através da fórmula abaixo:

$$variancia = \frac{1}{N} \cdot \sum_{i=1}^{N} (x_i - media)^2$$

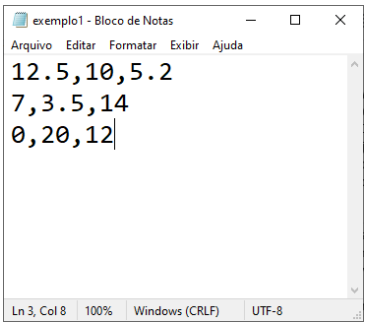
   Na fórmula acima, $media$ corresponde à média aritmética do conjunto numérico em questão. A função deve retornar a informação estatística especificada pela string, e que é calculada considerando os números presentes na lista de entrada. Se a string for diferente de todas as opções especificadas, retorne -1. **O segundo argumento da sua função deve ter valor padrão igual a 'media'.** Assuma que os dados de entrada sempre serão válidos (isto é, os números sempre serão inteiros ou floats) e que os números da lista de entrada estarão sempre ordenados em forma crescente. Não utilize a função "sum()" do Python.

2. Faça uma função em Python que recebe, nesta ordem: uma lista de números representando numeradores de uma divisão, e uma lista de números representando denominadores de uma divisão. A função deve retornar uma lista, cujos elementos correspondem à divisão dos elementos da primeira lista pelos elementos da segunda lista, localizados em posições correspondentes em cada lista (isto é, elemento na posição 0 da lista 1 dividido pelo elemento na posição 0 da lista 2, elemento na posição 1 da lista 1 dividido pelo elemento na posição 1 da lista 2, e assim por diante). Sua função deve percorrer as listas elemento a elemento e **usar tratamento de exceção para tentar converter os elementos de cada lista para float e para tentar calcular a razão entre eles. Utilize uma única estrutura de tratamento de exceções (apenas um 'try' e múltiplos 'except')**. Se nenhuma exceção for gerada, o quociente da divisão entre os números de cada lista de entrada deve ser incluído na lista que será retornada. Se o elemento de alguma lista não puder ser convertido para float, a exceção ValueError será gerada e você deve encerrar a função retornando o inteiro -1. Se algum elemento da segunda lista for igual a zero, a exceção ZeroDivisionError será gerada ao tentar fazer a divisão usando esse elemento. Caso a exceção ZeroDivisionError seja gerada, você deve incluir a string 'inf' na lista que será retornada e continuar realizando a divisão entre elementos subsequentes, e acrescentando os resultados de tais divisões à lista. Caso as listas tenham tamanhos diferentes, a exceção IndexError será gerada ao tentar acessar uma posição inexistente na lista de menor tamanho. Se a exceção IndexError for gerada, então a função deve retornar a lista que foi montada até aquele ponto. **Não utilize**

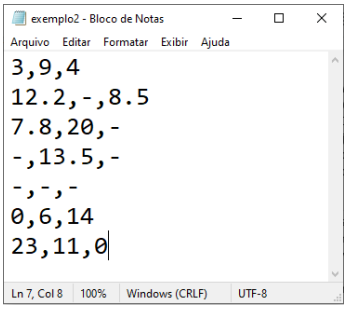
nenhuma estrutura condicional (if/elif/else) dentro da função, pois o objetivo aqui é praticar o uso do tratamento de exceção.

3. Faça uma função em Python que recebe por argumento um dicionário de produtos de uma loja de biscoitos e interage com um usuário que deseja adicionar ou alterar as informações nesse dicionário. O dicionário possui como chave o nome do produto e como valor uma lista, onde a primeira posição é o preço, a segunda é o peso do pacote, em gramas, e a terceira é o preço por grama. A função deve interagir com o usuário para pedir o nome do produto que deseja alterar ou adicionar. **Com o nome do produto, utilize tratamento de exceção para tentar acessar os valores do dicionário usando o nome digitado como chave. Se a chave não existir no dicionário, a exceção KeyError será gerada. Se a exceção KeyError for gerada, informe que um novo produto será inserido.** Se nenhuma exceção for gerada, informe que o produto será alterado e imprima os valores antigos. Peça para o usuário digitar o preço. **Ao pedir o preço, utilize tratamento de exceção para garantir que o valor digitado pelo usuário possa ser convertido para float e que o valor é maior que 0 e menor que 100 (utilize o raise para definir os limites e levante a exceção ValueError se os limites forem violados).** Caso o valor seja inválido, imprima uma mensagem de erro e repita a interação até que o valor digitado seja válido. **Se nenhuma exceção for gerada, peça para o usuário digitar o peso. Utilize tratamento de exceção mais uma vez para garantir que, ao pedir a quantidade, o programa identifica se o valor digitado pode ser convertido para float para garantir que o valor não é negativo (de novo, utilize o raise para definir os limites e levante a exceção ValueError se o valor for negativo).** Se a exceção ValueError for gerada, imprima uma mensagem de erro e repita a interação até que um valor não negativo e que possa ser convertido para float seja digitado. **Se nenhuma exceção for gerada, calcule o preço por grama fazendo preço/peso. Se o peso for igual a zero, a exceção ZeroDivisionError será gerada.** Se a exceção ZeroDivisionError for gerada, imprima uma mensagem de erro indicando que o peso não pode ser zero e volte a pedir o peso e a fazer as verificações necessárias (ValueError e ZeroDivisionError). A função deve retornar o dicionário atualizado.

4. Faça uma função em Python que recebe uma string, que é o nome de um arquivo, por argumento. A função deve ler o arquivo de texto que contém, em cada linha, informações sobre o tempo em que três máquinas ficaram ligadas em um dia dentro de um laboratório. Considere que o arquivo terá ao menos uma linha, e pode ter um número indefinido de linhas. Em cada linha, as informações do tempo de cada máquina são separadas por vírgula, conforme mostrado no exemplo abaixo.



Cada linha corresponde à medição de um dia. O primeiro número representa a quantidade de horas que a máquina 1 ficou ligada, o segundo número representa a quantidade de horas que a máquina 2 ficou ligada, e o terceiro número representa a quantidade de horas que a máquina 3 ficou ligada. Por simplicidade, considere que esses números sempre representarão valores válidos de horas diárias (ou seja, sempre estarão entre 0 e 24, de modo que não é preciso verificar a validade deles). No entanto, podem haver dias em que a medição de tempo de uma ou mais máquinas não foi feita. Nesses casos, o caractere '-' substitui o número de horas da máquina em questão naquele dia, conforme o exemplo abaixo.



A função deve retornar um dicionário contendo três chaves: 'M1', 'M2' e 'M3'. O valor de cada chave corresponde a uma tupla com dois elementos. O primeiro elemento da tupla é a quantidade de dias em que houve uma medição válida de tempo para tal máquina. Uma medição para tal máquina é válida, se não existir um caractere '-' registrado para aquela máquina em uma dada linha do arquivo. O segundo elemento da tupla é a porcentagem de tempo que tal máquina ficou ligada, considerando o total de horas disponíveis **nos dias em que houve uma medição válida**. Por exemplo: se uma máquina teve três medições válidas, e ficou ligada um total de 36 horas ao longo desses três dias, então a porcentagem de tempo que ela ficou ligada é de 100*(36/72) = 100*(0,5) = 50% (o denominador 72 representa o número máximo de horas em três dias).

Utilize tratamento de exceção ao tentar abrir o arquivo para leitura. Se não for possível abrir o arquivo para leitura, a exceção FileNotFoundError será gerada. Se a exceção FileNotFoundError for gerada, imprima uma mensagem de erro, indicando que o arquivo não existe, e retorne -1 (lembre-se: imprima com o print, retorne com o return). **Utilize tratamento de exceção também ao tentar converter as informações de tempo de cada máquina para float. Se alguma conversão não for possível, a exceção ValueError será gerada.** Se a exceção ValueError for gerada, imprima uma mensagem indicando a linha do arquivo em que isso ocorreu, e quais máquinas não tiveram medições de tempo registradas naquela linha (veja o exemplo de execução no arquivo .pdf com os testes), mas continue convertendo os valores de horas das demais linhas e utilize-os para calcular a porcentagem final de tempo em que cada máquina ficou ligada. Considere que haverá ao menos uma medição válida para cada máquina. **Não use nenhuma estrutura condicional (if/elif/else) dentro da função, pois o objetivo aqui é praticar o uso do tratamento de exceção.**

Dica: O método split(), do tipo string, pode ajudar muito nessa questão. Ele funciona da seguinte forma: considerando uma variável do tipo string **s**, a sintaxe para chamar o método split é **s.split(separador)**, onde separador é uma outra string. O split retorna uma lista de strings com pedaços de **s** entre a string separadora. Isto é, se **s = Fernanda.Oliveira**, então **s.split('.')** retorna a lista **['Fernanda' , 'Oliveira']** (note que o separador em si não aparece no valor de retorno). Veja alguns exemplos desse método realizados no shell do Python:

```
>>> s = 'Fernanda.Oliveira'
>>> nomes = s.split('.')
>>> nomes
['Fernanda', 'Oliveira']
>>> frase = 'Ola! Meu nome eh Fernanda'
>>> palavras = frase.split(' ')
>>> palavras
['Ola!', 'Meu', 'nome', 'eh', 'Fernanda']
>>> # Veja que a variavel com a string original nao muda:
>>> frase
'Ola! Meu nome eh Fernanda'
>>> # Quando o separador estah na ultima posicao da string, aparece uma
string vazia na ultima posicao da lista resultante:
>>> teste1 = frase.split('a')
>>> teste1
['Ol', '! Meu nome eh Fern', 'nd', '']
>>> # Quando o separador nao existe na string, o resultado eh uma lista
com a string completa:
>>> teste2 = frase.split('A')
>>> teste2
['Ola! Meu nome eh Fernanda']
```