Computação I

Lista de exercícios 4 – A estrutura de repetição while e o tipo lista.

Atenção! Leia as instruções antes de fazer a lista! A padronização do nome do arquivo e dos nomes das funções é muito importante e está explicada no arquivo de instruções!

Data de entrega: 13/10/2022

A menos que esteja explicitamente pedido na questão, não utilize nenhum método ou função já existente do Python exceto pelas funções print e len, e pelo método append. De forma simplificada, função é tudo o que precisa ser chamado com parênteses, e método é tudo o que precisa ser chamado com parênteses, mas que está associado a uma variável, isto é, variavel.metodo(args). Funções de transformação de tipo (int, float, str) também não são permitidas. Não importe nenhum módulo. Não é necessário testar se os dados passados por argumento são válidos. Nessa lista, utilize somente a estrutura de repetição while e **não utilize o for e não utilize recursão**. Não crie funções dentro de funções.

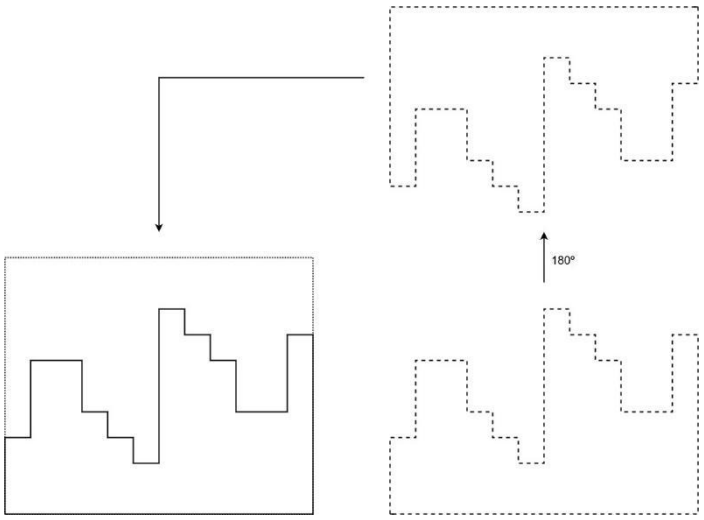
1. Escreva uma função em Python que recebe por argumento, e nessa ordem, uma lista de números inteiros não-nulos e um número inteiro positivo n, e retorna uma nova lista contendo apenas os elementos da lista de entrada que são divisíveis pelo número n passado como argumento.

2. Escreva uma função em Python que recebe duas listas de mesmo tamanho e que retorne uma lista contendo os elementos da primeira lista intercalados com os elementos da segunda lista, porém na ordem inversa em que tais elementos ocorrem na segunda lista. Isto significa que o primeiro elemento da lista de saída deve corresponder ao primeiro elemento da primeira lista, já o segundo elemento da lista de saída deve corresponder ao último elemento da segunda lista. Seguindo esta regra, o terceiro elemento da lista de saída deve ser o segundo elemento da primeira lista, enquanto que o quarto elemento da lista de saída deve ser o penúltimo elemento da segunda lista, e assim por diante. Por exemplo: para a entrada ([1,2,3,4], [5,6,7,8]), a saída deve ser [1,8,2,7,3,6,4,5].

3. Escreva uma função em Python que recebe por argumento três listas de números, onde as três listas possuem mesmo tamanho, e retorna uma nova lista com o resultado da soma, elemento a elemento, das duas primeiras listas e multiplicação da terceira. Isto é, elemento da posição 0 da lista 1 deve ser somado ao o elemento da posição 0 da lista 2 e o resultado da soma deve ser multiplicado pelo elemento da posição 0 da lista 3; o elemento da posição 1 da lista 1 deve ser somado ao o elemento da posição 1 da lista 2 e o resultado da soma deve ser multiplicado pelo elemento da posição 1 da lista 3; o elemento da posição 2 da lista 1 deve ser somado ao o elemento da posição 2 da lista 2 e o resultado da soma deve ser multiplicado pelo elemento da posição 2 da lista 3...

4. Escreva uma função em Python que receba uma lista contendo números inteiros, ordenados de forma crescente, e que retorne a soma de todos os produtos dois-a-dois distintos entre os elementos desta lista. Por exemplo: para a entrada [3,4,5], a saída deve ser 97 (3*3 + 3*4 + 3*5 + 4*4 + 4*5 + 5*5). Observe que apenas os produtos dois-a-dois distintos devem ser considerados. No exemplo acima, consideramos apenas 3*4, pois 4*3 é o mesmo produto (isto é, o primeiro elemento vezes o segundo elemento é o mesmo que o segundo elemento vezes o primeiro elemento). Da mesma forma, consideramos apenas 4*5, pois 5*4 é o mesmo produto.

5. (Inspirada na OBI 2020 - Nível Júnior - Fase 1 - Modalidade Programação - Turno A) M. C. Escher foi um artista gráfico holandês que fazia incríveis ilustrações onde preenchia a tela com objetos auto-similares, cujos contornos encaixam neles próprios, criando simetrias geométricas muito impressionantes. Veja um exemplo dessa ideia na figura da próxima página, que mostra um objeto que é um perfil ortogonal definido por uma sequência de números naturais representando a sequência de alturas. Podemos pegar uma cópia do objeto, rotacionar 180 graus e encaixar perfeitamente no objeto original, formando um retângulo.



Em termos mais gerais, se uma sequência de N números naturais representando a sequência de alturas for A_0, A_1, A_2, . . . , A_{N-3}, A_{N-2}, A_{N-1} , o perfil definido será chamado de perfil Escher se tivermos A_0 + A_{N-1} igual a A_1 + A_{N-2} igual a A_2 + A_{N-3}, e assim por diante. Faça uma função em Python que recebe por argumento uma lista de números inteiros ou floats, com no mínimo dois elementos, que representa a sequência de alturas que definem o perfil. A função deve decidir se o perfil é Escher, ou não, e deve retornar o **booleano** True, se o perfil for Escher; ou False, senão.

6. O Triângulo de Pascal é uma ferramenta matemática com propriedades bastante interessantes e úteis. Ele é construído visualmente empilhando linhas representando diferentes sequências, conforme ilustrado na figura abaixo.

$$
\begin{array}{lll}
n = 0 & \longrightarrow & \mathbf{1} \\
n = 1 & \longrightarrow & \mathbf{1}\ \ \mathbf{1} \\
n = 2 & \longrightarrow & \mathbf{1}\ \ \mathbf{2}\ \ \mathbf{1} \\
n = 3 & \longrightarrow & \mathbf{1}\ \ \mathbf{3}\ \ \mathbf{3}\ \ \mathbf{1} \\
n = 4 & \longrightarrow & \mathbf{1}\ \ \mathbf{4}\ \ \mathbf{6}\ \ \mathbf{4}\ \ \mathbf{1} \\
 & & \vdots
\end{array}
$$

Dentre uma de suas propriedades, está o fato de que cada linha do triângulo contém os coeficientes numéricos que multiplicam os termos do binômio de Newton de uma determinada ordem. Assumindo que a contagem das linhas começa a partir do número 0, isto quer dizer que a linha n deste triângulo contém os coeficientes numéricos dos termos do binômio de Newton de ordem n. Por exemplo: a linha 0 ($n = 0$) contém os coeficientes numéricos dos termos do binômio de Newton de ordem 0 ($(x + y)^0 = \mathbf{1}$). A linha 2 ($n = 2$) contém os coeficientes numéricos dos termos do binômio de Newton de ordem 2 ($(x + y)^2 = \mathbf{1}x^2 + \mathbf{2}xy + \mathbf{1}y^2$).

Há três regras que permitem definir as sequências em cada linha deste triângulo: i) a primeira linha ($n = 0$) sempre contém apenas um único termo de valor 1; ii) as demais linhas sempre começam e terminam com o número 1; e iii) em cada linha, os termos intermediários são obtidos somando o número imediatamente acima do termo atual, na linha anterior, com o número localizado acima e à esquerda ao termo atual, também na linha anterior. Por exemplo: na linha $n = 4$, o segundo termo,

de valor 4, é obtido somando o valor do termo imediatamente acima (3) com o valor do termo acima e à esquerda (1), ambos na linha anterior. Da mesma forma, o terceiro termo da linha $n = 4$, de valor 6, é obtido somando o valor do termo imediatamente acima (3) com o valor do termo acima e à esquerda (3), ambos na linha $n = 3$.

Faça uma função em Python que receba um número n maior ou igual a 0, o qual identifica uma linha do Triângulo de Pascal, e que retorne uma lista contendo os valores numéricos da sequência definida em tal linha do Triângulo de Pascal.