Vetor

Neste exercício, definiremos uma classe chamada `Vector` para representar vetores n-dimensionais.

O método `__init__` da sua classe deve tomar como argumento uma lista contendo os números do vetor.

Sua classe deve fornecer os seguintes métodos:

Observe que não esperamos que você saiba essas operações de cabeça (ou mesmo, necessariamente, que você já as tenha aprendido antes). Se você não tiver certeza do que algo significa, tente fazer uma pesquisa na internet ou pedir ajuda no Piazza!

- um método `as_list()`, que retorna uma lista contendo os números do vetor.
- um método `size()`, que retorna um inteiro contendo o número de elementos no vetor.
- um método `magnitude()`, que deve retornar a magnitude do vetor.
- um método `euclidean_distance(other)`, que deve retornar a distância euclidiana entre `self` e `other`, onde `other` é uma instância de `Vector` com o mesmo tamanho.
- um método `normalized()`, que deve retornar um vetor unitário (vetor de comprimento 1) apontando na mesma direção do vetor original.
- um método `cosine_similarity(other)`, que deve retornar o cosseno do ângulo (https://en.wikipedia.org/wiki/Cosine_similarity) entre `self` e `other`, onde `self` e `other` são vetores do mesmo tamanho.
- um método `__add__(other)`, que retorna uma *nova instância* de `Vector` representando a soma da instância original e `other`:
  - se `other` for uma instância de `Vector` com tamanho apropriado, seu código deve realizar uma adição de vetor.
  - se `other` for uma instância de `Vector` cujas dimensões são inadequadas para adição de vetor, seu código deve retornar `None`
  - caso contrário, retorna `None` (indicando um erro)
- um método `__sub__(other)`, que deve se comportar de forma análoga a `__add__`, mas deve realizar a subtração.
- um método `__mul__(other)`, que retorna um objeto que representa o resultado da multiplicação da instância original e `other`:
  - se `other` for uma instância de `Vector` com dimensões apropriadas, seu código deve calcular e retornar o produto escalar (https://pt.wikipedia.org/wiki/Produto_escalar) de `self` e `other`. O resultado deve ser um `int` ou um `float`.
  - se `other` for uma instância de `Vector` cujas dimensões não permitem calcular o produto escalar, seu código deve retornar `None`.
  - se `other` for um `int` ou `float`, o resultado deve ser uma nova instância de `Vector`, obtida multiplicando cada entrada por `other`. Não deve alterar a instância original.
  - caso contrário, retorna `None` (indicando um erro)
- um método `__truediv__(other)`, que retorna uma *nova instância* de `Vector` representando o resultado da divisão da instância original por `other`:
  - se other for um `int` ou `float`, divida cada entrada por `other`
  - caso contrário, retorna `None` (indicando um erro)

Submissão

Quando estiver pronto (depois de ter simulado manualmente e testado em sua própria máquina e estiver convencido de que seu programa fará a coisa certa), faça upload do seu arquivo Python no **Problema 4.4** no Gradescope. Lembre de nomear seu arquivo `p4_4.py` .