Profa. Nádia Félix
Projeto TJ:
March 5, 2025

Tarefa 4: POO

Questão 1. Neste exercício, encontraremos nosso velho amigo, o vetor, mais uma vez, desta vez usando objetos.

1) Defina uma classe Vector. Cada vetor deve ter uma dimensão (um inteiro positivo) e uma lista ou tupla de suas entradas. O inicializador para sua classe deve tomar a dimensão como seu primeiro argumento e uma lista ou tupla de números (ints ou floats), representando as entradas do vetor, como seu segundo argumento. Se o usuário fornecer apenas uma dimensão e nenhuma entrada, o comportamento padrão deve ser criar um vetor com todos os elementos zeros da dimensão fornecida. O inicializador deve gerar um erro no caso em que a dimensão seja inválida (ou seja, tipo errado ou um número negativo) e também deve gerar um erro no caso de a dimensão e o número de entradas fornecidas discordarem.
2) Implemente um método Vector.get_dim() que retorna a dimensão do vetor e um método Vector.get_entries() que retorna as entradas do vetor como uma tupla.
3) Implemente o(s) operador(es) necessário(s) para suportar a comparação (igualdade, menor que, menor ou igual a, maior que, etc.) de objetos Vector. Diremos que dois objetos Vector são equivalentes se tiverem as mesmas coordenadas (normalmente nos preocuparíamos com a comparação de float aqui, mas vamos ignorar isso). Caso contrário, a comparação deve ser análoga às tuplas em Python, de modo que a comparação seja feita na primeira coordenada primeiro, depois na segunda coordenada, depois na terceira e assim por diante. Então, por exemplo, o vetor bidimensional (2, 4) é ordenado antes de (menor que) (2, 5). Tentar comparar dois vetores de dimensões diferentes deve resultar em um erro. Tentar comparar um objeto Vector a um objeto que não é do tipo Vector deve resultar em um erro.
4) Implemente os operadores de adição e subtração para objetos Vector, para que possamos escrever v1 + v2 e v1 - v2 para objetos Vector v1 e v2 da mesma dimensão. Adicionar dois objetos Vector deve produzir outro objeto Vector. Adicionar dois objetos Vector de dimensões diferentes deve resultar em um erro, assim como adicionar um Vector e um não-Vector.
5) Implemente um método Vector.dot para calcular o produto interno do chamador com outro objeto Vector. Ou seja, se v1 e v2 forem objetos Vector, v1.dot(v2) deve calcular seu produto interno. Seu método deve gerar um erro apropriado no caso em que o argumento não seja do tipo correto ou no caso em que as dimensões dos dois vetores não coincidam.
6) Gostaríamos também que nossa classe Vector suportasse multiplicação escalar. Multiplicação à esquerda ou à direita por um escalar, por exemplo, 2*v ou v*2, onde v é um objeto Vector, deve resultar em um novo objeto Vector com suas entradas todas multiplicadas pelo escalar fornecido. Isso deve suportar multiplicação por ints e floats. Implemente os operadores apropriados para dar suporte a esta operação de multiplicação. Muitas linguagens têm uma convenção para lidar com multiplicação de vetores que diferem em suas dimensões, mas vamos adiar este assunto. Seu método deve simplesmente levantar um erro apropriado no caso de v e w discordarem em suas dimensões. Seu método também deve levantar um erro ao tentar multiplicar um objeto Vector por qualquer coisa que não seja um Vector, int ou float.