Python Programming Challenge - Quaternions

Introduction

The topic of this challenge is the implementation of a quaternions class in Python. In mathematics, the quaternion number system extends the complex numbers. A quaternion q is an expression of the form:

$$q = a + bi + cj + dk\,, \tag{1}$$

where a, b, c, d are real numbers and i, j, k are symbols that can be interpreted as unit-vectors pointing along the three spatial axes. The real part of a quaternion q is defined as:

$$\Re(q) = \Re\left(a + bi + cj + dk\right) = a\,. \tag{2}$$

Addition and subtraction of two quaternions are defined as follows:

$$\begin{aligned} q_1 + q_2 &= (a_1 + b_1 i + c_1 j + d_1 k) + (a_2 + b_2 i + c_2 j + d_2 k) \\ &= (a_1 + a_2) + (b_1 + b_2)\, i + (c_1 + c_2)\, j + (d_1 + d_2)\, k \qquad (3) \\ q_1 - q_2 &= (a_1 + b_1 i + c_1 j + d_1 k) - (a_2 + b_2 i + c_2 j + d_2 k) \\ &= (a_1 - a_2) + (b_1 - b_2)\, i + (c_1 - c_2)\, j + (d_1 - d_2)\, k\,. \qquad (4) \end{aligned}$$

Multiplication of a quaternion q with a scalar s is defined as:

$$s \cdot q = s \cdot (a + bi + cj + dk) = sa + sbi + scj + sdk\,. \tag{5}$$

The conjugation q^* of a quaternion q is defined as:

$$q^* = a - bi - cj - dk\,. \tag{6}$$

The norm $|q|$ of an quaternion q is defined as:

$$|q| = \sqrt{a^2 + b^2 + c^2 + d^2}\,. \tag{7}$$

Task

Use Python to implement a quaternion class called `Quaternion` with:

- suitable attributes which are set in the `init` function
- an addition according to equation (3) which is callable with +
- a function named `scalar_multiplication` according to equation (5) which takes `scalar` as an argument
- a function named `real_part` according to equation (2)
- a function named `conjugate` according to equation (6)
- a function named `norm` according to equation (7)
- a function named `create_real_quarternion` which takes an scalar argument called `scalar` and returns a quarternion with $a =$ `scalar`, $b = c = d = 0$.
- a function named `create_multiple_real_quarternions` which takes a list of scalars (`e.g [2, 4, 5]`) as argument (`scalar_list`) and returns a list of quarternions with $a =$ `scalar`, $b = c = d = 0$, where a for every quaternion corresponds to the corresponding element in the input list.
- When calling `print` on a quaternion $a + ib + jc + kd$ the output should be `a + bi + cj + dk`. If the quaternion is for example $1 + 2i + 3j + 4k$ the output should be `1 + 2i + 3j + 4k`.

Create a second class called `QuaternionHelper`. This class should only have a single function `perform_on_quaternion` with the two arguments `input_quaternion` (a quaternion object) and `action` (a string). Possible values for `action` are: 'real_part', 'conjugate', 'norm', 'itself', 'triple'. For each of these values a different action is to be triggered:

- **real_part:** print the real part of the input quaternion
- **conjugate:** print the conjugate of the input quaternion
- **norm:** print the norm of the input quaternion
- **itself:** print the input quaternion
- **triple:** print the result of multiplication of the input quaternion and 3
- **other input:** Raise an appropriate error.