SOFTWARE 1 PRACTICAL

OPERATOR OVERLOADING

Week 9 – Practical 8

Before attempting this practical, you should complete part A. The aim of this practical is to improve the functionality of the class Vector you implemented earlier this week. You will do so by refactoring your code and overloading existing operators. All tests from previous practical should still work. You should try to ensure backward compatibility.

Exercise 1:

We are now able to compare two vectors, however it would be nice if we could use the operator == instead of `.equals(…)`. Fortunately Python allows to overload operators such as +, *, == and !=. To overload the operator ==, we must ***override*** the method `__eq__`. (for the operator `!=`, override the method `__ne__`).

Implement the two methods:

```
def __eq__(self, other_vector):
    <body>

def __ne__(self, other_vector):
    <body>
```

Exercise 2:

Instead of using the method `add(…)` and `scalar_product(…)` we would like to overload the operators + and *. The vector addition operator is commutative, i.e. `v1+v2 == v2+v1` and we can override the method `__add__` to overload the + operator. When considering the multiplication, it is a little bit more complicated, `3 * v1` is allowed, but `v1 * 3` is not. Investigate the methods `__mul__` and `__rmul__`. One other programming shortcut we could find useful is `v1 += v2` for `v1 = v1+v2`. It can be implemented by overloading `__iadd__`. Similarly, the shortcut `v1 *= 3` for `v1 = 3 * v1` can be implemented by overloading `__imul__`. Implement all these operators overloading.

Exercise 3:

There is a way to add the following functionality to our class Vector.

```
>>> vector1 = Vector(1, 2, 3, 5, 6, 1)
>>> vector1[2] += 5
>>> print(vector1)
 <1.0, 2.0, 8.0, 5.0, 6.0, 1.0>
```

Search the web to find out how it can be done (I will NOT provide the answer). For method with undefined number of parameters search `*args`. **Note**, the tests provided in previous practical should still be working, including the test for the __init__ method.