Building a 2D Vector Class in Python

This tutorial is designed to guide you through the process of building a comprehensive 2D Vector class in Python, focusing on the implementation of various dunder (double underscore) methods and properties. This exercise will help reinforce your understanding of Python's object-oriented programming features.

In class we have built the following class Vector:

```python
class Vector:
    '''This is the class Vector to manipulate vectors in 2d'''

    @staticmethod
    def from_polar(rho,theta):
        return Vector(rho * math.cos(theta), rho * math.sin(theta))

    def __init__(self, x, y):

        if not (isinstance(x,(int,float)) and isinstance(y,(int,float))):
            raise TypeError('No way, I want numbers')

        # set the coordinates

        self.x = x
        self.y = y

    def __repr__(self):
        return f'Vector({self.x},{self.y})'

    def __str__(self):
        return f'This is Vector({self.x},{self.y})'

    def __add__(self, other_vector):
        if not isinstance(other_vector, Vector):
            raise TypeError('Please add a vector to a vector')
        return Vector(self.x + other_vector.x, self.y + other_vector.y)

    def __eq__(self, other_vector):
        if not isinstance(other_vector, Vector):
            raise TypeError('Please add a vector to a vector')
        return (self.x == other_vector.x) and (self.y == other_vector.y)

    @property
    def norm(self):
        return math.sqrt(self.x**2 + self.y**2)

    def normalized(self, inplace = False):
        '''This method normalizes the vector. By convention, (0,0) is nomalized to (0,0)'''
        norm_of_the_vector = self.norm

        if inplace:
            if norm_of_the_vector == 0.:
                self.x = 0.
                self.y = 0.
            else:
                self.x /= norm_of_the_vector
                self.y /= norm_of_the_vector
            return None
        else:
            return Vector(self.x/norm_of_the_vector, self.y/norm_of_the_vector) if norm_of_the_vector != 0. else Vector(0.,0.)
```

1. Recall what is a static method? a property? a dunder method?

2. Implement two properties `rho` and `theta` for the polar coordinates.

3. Implement the `__sub__` method for vector subtraction.

4. Implement the `__neg__` method for vector negation.

5. Implement the `__mul__` method for scalar multiplication. Also, implement `__rmul__` to handle multiplication when the Vector instance is on the right side of the * operator.

6. Implement the `__bool__` method to return False if the vector is the zero vector.

7. Write a method to compute the inner product with another vector.

8. Write a method to rotate the vector by an angle θ. It should have a keyword parameter `inplace`.

9. Write a method to plot the vector using `matplotlib`.

10. Improve the previous method so that it takes an `Axes` object. The goal is to be able to add a vector on a given graph or to plot several vectors on the same graph.

Congratulations on completing the tutorial! You have now built a fully functional 2D Vector class that supports a variety of operations, deepening your understanding of Python's special methods and soem of the object-oriented programming principles.