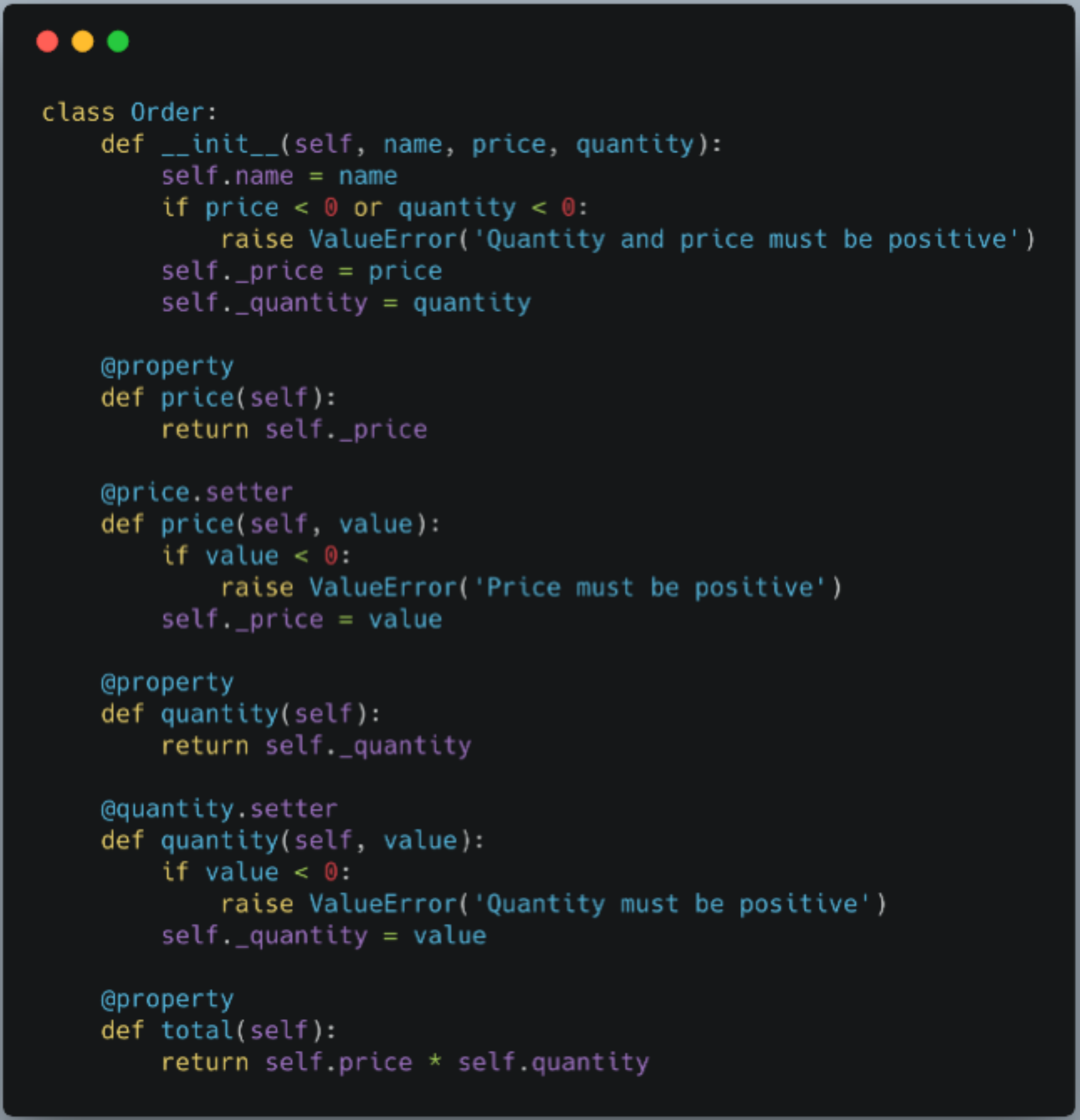
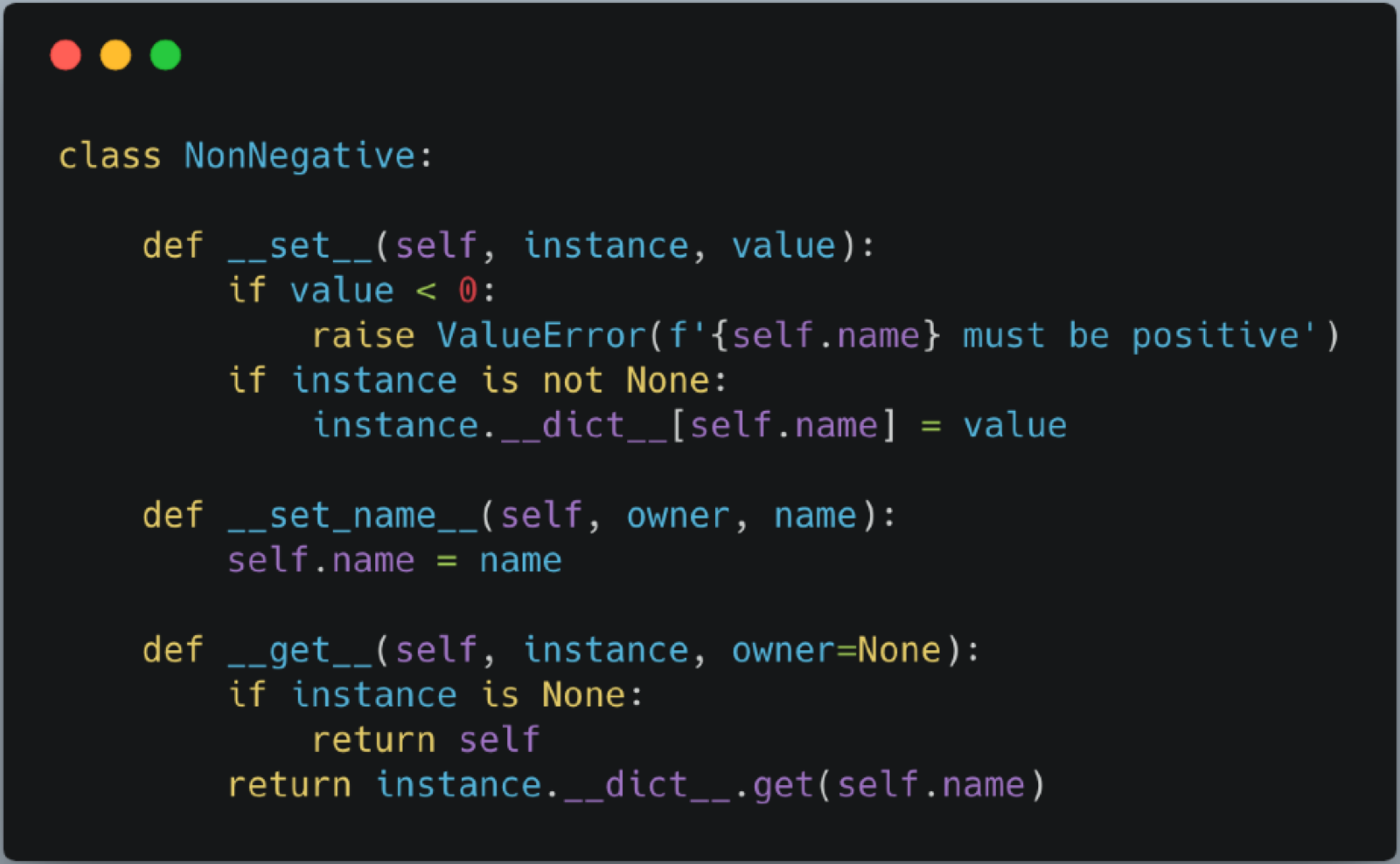
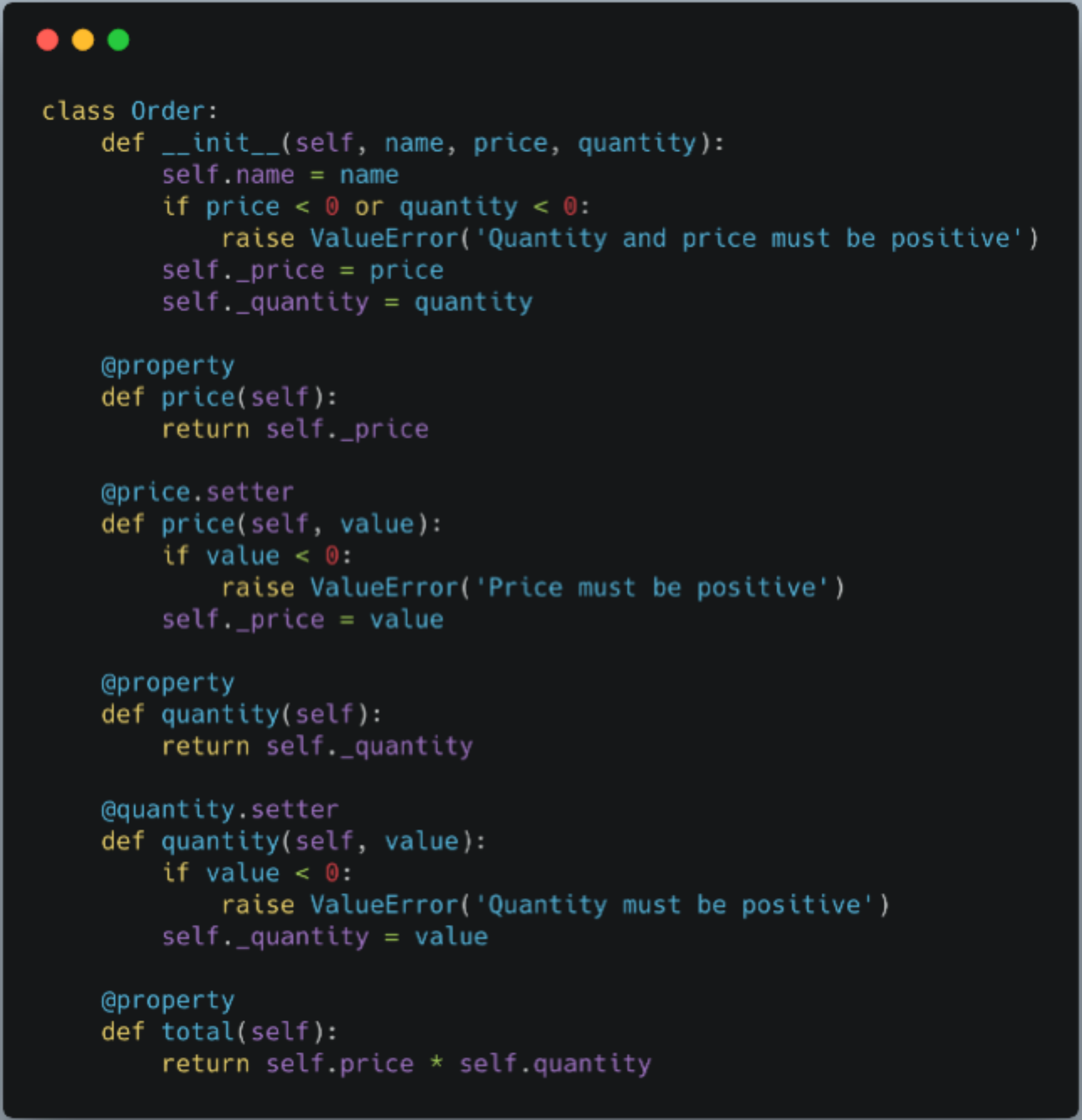


```python
class Order:
    def __init__(self, name, price, quantity):
        self.name = name
        if price < 0 or quantity < 0:
            raise ValueError('Quantity and price must be positive')
        self._price = price
        self._quantity = quantity

    @property
    def price(self):
        return self._price

    @price.setter
    def price(self, value):
        if value < 0:
            raise ValueError('Price must be positive')
        self._price = value

    @property
    def quantity(self):
        return self._quantity

    @quantity.setter
    def quantity(self, value):
        if value < 0:
            raise ValueError('Quantity must be positive')
        self._quantity = value

    @property
    def total(self):
        return self.price * self.quantity
```



```python
class NonNegative:

    def __set__(self, instance, value):
        if value < 0:
            raise ValueError(f'{self.name} must be positive')
        if instance is not None:
            instance.__dict__[self.name] = value

    def __set_name__(self, owner, name):
        self.name = name

    def __get__(self, instance, owner=None):
        if instance is None:
            return self
        return instance.__dict__.get(self.name)
```



```python
class Order:
    def __init__(self, name, price, quantity):
        self.name = name
        if price < 0 or quantity < 0:
            raise ValueError('Quantity and price must be positive')
        self._price = price
        self._quantity = quantity

    @property
    def price(self):
        return self._price

    @price.setter
    def price(self, value):
        if value < 0:
            raise ValueError('Price must be positive')
        self._price = value

    @property
    def quantity(self):
        return self._quantity

    @quantity.setter
    def quantity(self, value):
        if value < 0:
            raise ValueError('Quantity must be positive')
        self._quantity = value

    @property
    def total(self):
        return self.price * self.quantity
```