```python
                                       "Expecting %d got %d" % (len(self.step_offsets), k))

    def step_offset(self, step):
        if self.step_info == None:
            return 0
        else:
            if step >= len(self.step_offsets):
                return len(self.data)
            else:
                return self.step_offsets[step]

    def get_wave(self, step=0):
        return self.data[self.step_offset(step):self.step_offset(step + 1)]


class Trace(DataSet):
    """Class used for storing generic traces that report to a given Axis."""

    def __init__(self, name, datatype, datalen, axis):
        super().__init__(name, datatype, datalen)
        self.axis = axis

    def get_point(self, n, step=0):
        if self.axis is None:
            return super().get_point(n)
        else:
            return self.data[self.axis.step_offset(step) + n]

    def get_wave(self, step=0):
        if self.axis is None:
            return super().get_wave()
        else:
            return self.data[self.axis.step_offset(step):self.axis.step_offset(step + 1)]


class DummyTrace(object):
    """Dummy Trace for bypassing traces while reading"""

    def __init__(self, name, datatype):
        """Base Class for both Axis and Trace Classes.
```