```
import dates_utility as utility
from filters_base import FiltersBase
from models_base import ModelsBase

class MyFilter(FiltersBase):
    _filter_name = "MyFilter"
    _def_local_filter_configs = dict(model=None, filter_name=_filter_name)
    _local_def_output_configs = dict(scr_output=True, file_output=False,
                                     filter_statistics_dir='Filter_Statistics',
                                     model_states_dir='Model_States_Repository',
                                     observations_dir='Observations_Rpository')

    def __init__(self, filter_configs=None, output_configs=None):
        """ Constructor; MyFilter class implementation """
        err_msg = "A model object reference MUST be passed in 'filter_configs' as value to the key 'model'..."
        assert isinstance(filter_configs['model'], ModelsBase), err_msg

        # aggregate configurations, and attach filter_configs, output_configs to the filter object.
        filter_configs = utility.aggregate_configurations(filter_configs, MyFilter._def_local_filter_configs)
        output_configs = utility.aggregate_configurations(output_configs, MyFilter._local_def_output_configs)
        FiltersBase.__init__(filter_configs=filter_configs, output_configs=output_configs)
        self.model = self.filter_configs['model']

    def filtering_cycle(self):
        """ Carry out a single filtering cycle """
        FiltersBase.filtering_cycle()
        # Add further functionality if you wish...

    def forecast(self):
        """ Forecast step of the filter """
        #

    def analysis(self, *args, **kwargs):
        """ Analysis step of the filter """
        #
```