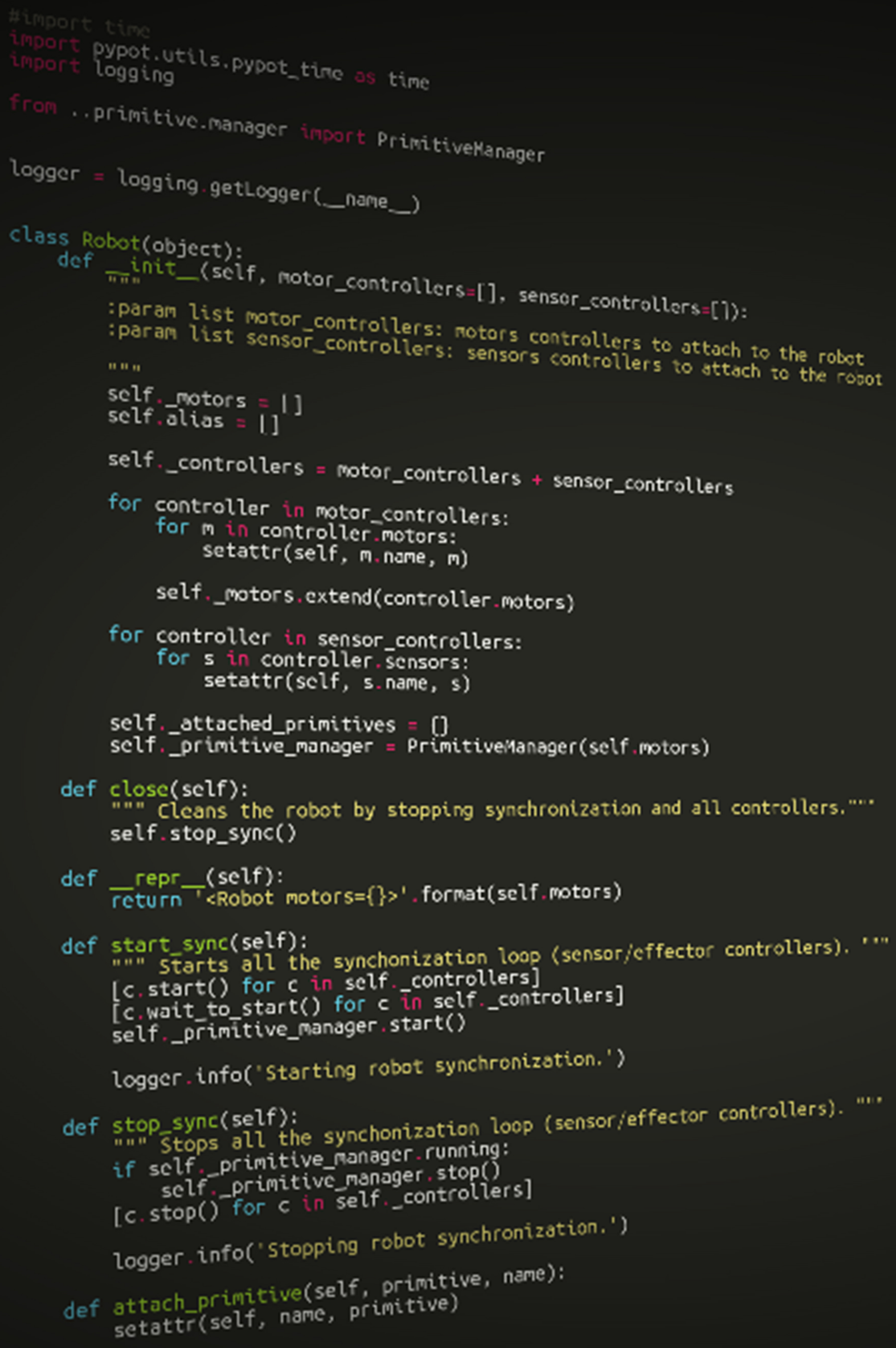

```python
#import time
import pypot.utils.pypot_time as time
import logging

from ..primitive.manager import PrimitiveManager

logger = logging.getLogger(__name__)

class Robot(object):
    def __init__(self, motor_controllers=[], sensor_controllers=[]):
        """
        :param list motor_controllers: motors controllers to attach to the robot
        :param list sensor_controllers: sensors controllers to attach to the robot

        """
        self._motors = []
        self.alias = []

        self._controllers = motor_controllers + sensor_controllers

        for controller in motor_controllers:
            for m in controller.motors:
                setattr(self, m.name, m)

            self._motors.extend(controller.motors)

        for controller in sensor_controllers:
            for s in controller.sensors:
                setattr(self, s.name, s)

        self._attached_primitives = {}
        self._primitive_manager = PrimitiveManager(self.motors)

    def close(self):
        """ Cleans the robot by stopping synchronization and all controllers."""
        self.stop_sync()

    def __repr__(self):
        return '<Robot motors={}>'.format(self.motors)

    def start_sync(self):
        """ Starts all the synchonization loop (sensor/effector controllers). """
        [c.start() for c in self._controllers]
        [c.wait_to_start() for c in self._controllers]
        self._primitive_manager.start()

        logger.info('Starting robot synchronization.')

    def stop_sync(self):
        """ Stops all the synchonization loop (sensor/effector controllers). """
        if self._primitive_manager.running:
            self._primitive_manager.stop()
        [c.stop() for c in self._controllers]

        logger.info('Stopping robot synchronization.')

    def attach_primitive(self, primitive, name):
        setattr(self, name, primitive)
```