```python
from pybricks import ev3devices
from pybricks.parameters import Direction, Stop
import pybricks.tools


# Wrapper class for motor to allow for run to stop without quiting program
class Motor():
    def __init__(self, port, config,
                 positive_direction=Direction.CLOCKWISE,
                 gears=None):
        self.config = config
        self.m = ev3devices.Motor(port, positive_direction=positive_direction,
                                  gears=gears)
        self.control = self.m.control

    def angle(self):
        return self.m.angle()

    def speed(self):
        return self.m.speed()

    def reset_angle(self, angle):
        self.m.reset_angle(angle)

    def hold(self):
        self.m.hold()

    def run(self, speed):
        self.m.run(speed)

    def run_time(self, speed, time, then=Stop.HOLD, wait=True):
        self.m.run_time(speed, time, then, False)
        while wait and self.control.done() == False and self.config.state.getState() != 3
        :
            pybricks.tools.wait(10)
        if then == Stop.HOLD:
            self.m.hold()
        elif then == Stop.BRAKE:
            self.m.brake()

    def run_angle(self, speed, rotation_angle, then=Stop.HOLD, wait=True):
        self.m.run_angle(speed, rotation_angle, then, False)
        while wait and self.control.done() == False and self.config.state.getState() != 3
        :
            pybricks.tools.wait(10)
        if then == Stop.HOLD:
            self.m.hold()
        elif then == Stop.BRAKE:
            self.m.brake()

    def run_target(self, speed, target_angle, then=Stop.HOLD, wait=True):
        self.m.run_target(speed, target_angle, then, False)
        while wait and self.control.done() == False and self.config.state.getState() != 3
        :
            pybricks.tools.wait(10)
        if then == Stop.HOLD:
            self.m.hold()
        elif then == Stop.BRAKE:
            self.m.brake()

    def run_until_stalled(self, speed, then=Stop.COAST, duty_limit=None):
        self.m.run_until_stalled(speed, then, duty_limit)
        if then == Stop.HOLD:
            self.m.hold()
        elif then == Stop.BRAKE:
            self.m.brake()

```

```
65      def track_target(self, target_angle):
66          self.m.track_target(target_angle)
67
68      def dc(self, duty):
69          self.m.dc(duty)
70
71      def stop(self):
72          self.m.stop()
73
```