```
# Name: Trien Bang Huynh
# Assignment 1: States class

import csv
import os

class States:
    '''
    A class which stores data from the input file and has methods to look up the
data.
    '''

    '''
    A decorator that prints the name of the function that it decorates to a log
file.
    '''
    def log_function(func):
        def wrapper(*args, **kwargs):
            with open('log_file.txt', 'a') as f:
                f.write(f"{func.__name__}\n")
            return func(*args, **kwargs)
        return wrapper

    filename = "statesPop.csv"

    def __init__(self, fileName = filename) -> None:
        self.data = []
        with open(os.path.join(os.path.dirname(os.path.abspath(__file__)),
fileName)) as f:
            reader = csv.reader(f)
            for row in reader:
                state_name, pop_1990, pop_2000, pop_2010, pop_2020, pop_2021,
growth_rate = row
                pop_1990, pop_2000, pop_2010, pop_2020 = map(int, (pop_1990,
pop_2000, pop_2010, pop_2020))
                growth_rate = float(growth_rate)
                self.data.append((state_name, pop_1990, pop_2000, pop_2010,
pop_2020, growth_rate))

    '''
    A method to list the states with their population for a certain year, sorted in
descending order by population
    '''
    @log_function
    def byPop(self, year):
        index = (year - 1990) // 10 + 1
        return ((state[0], state[index]) for state in sorted(self.data, key=lambda
x: x[index], reverse=True))

    '''
    A generator method that yields a state and its growth rate, based on the user
choice of positive or negative growth.
    '''
    @log_function
    def growth(self, is_positive):
        if is_positive:
            index = 0
            while index < self.max_num_states :
                if self.data[index][5] > 0:
                    yield (self.data[index][0],self.data[index][5])
                index += 1
        else:
            index = 0
            while index < self.max_num_states :
                if self.data[index][5] < 0:
                    yield (self.data[index][0],self.data[index][5])
                index += 1

    '''
    A method to check whether at least one state has dropped in population between
2 given years.
    '''
    @log_function
    def drop(self, start_year, end_year):
        start_index = (start_year - 1990) // 10 + 1
        end_index = (end_year - 1990) // 10 + 1
        return any(state[start_index] > state[end_index] for state in self.data)

    '''
    A property method that returns the maximum number of states.
    '''
    @property
    def max_num_states(self):
        return len(self.data)
```