```
# Name: Trien Bang Huynh
# Assignment 2: numpy, matplotlib, tkinter
# tuition.py: Tuition class

import numpy as np
import os
import matplotlib.pyplot as plt


class FileNotFound(Exception):
    '''
    A custom class to handle "file not found"
    '''
    pass

class Tuition:
    '''
    A Tuition class which has data from the 3 input files and methods to analyze
and plot the data.
    '''
    def print_return_value(func):
        '''
        A decorator that prints the return value of the function that it decorates.
        '''
        def wrapper(*args, **kwargs):
            result = func(*args, **kwargs)
            print(f"Return value of {func.__name__}: {result}")
            return result
        return wrapper


    def __init__(self) -> None:
        try:
            costsArr =
np.loadtxt(os.path.join(os.path.dirname(os.path.abspath(__file__)), "costs.csv"),
dtype=int, delimiter=",")
            self._costsArrView = costsArr[ costsArr > 0]
            self._costsArrView.shape = (49,19)

        except IOError:
            raise FileNotFound("costs.csv not found")

        try:
            statesArr =
np.loadtxt(os.path.join(os.path.dirname(os.path.abspath(__file__)), "states.csv"),
dtype=str, delimiter=",")
            self._statesArrView = statesArr[statesArr != "Alaska"]

        except IOError:
            raise FileNotFound("states.csv not found")

        try:
            yearsArr =
np.loadtxt(os.path.join(os.path.dirname(os.path.abspath(__file__)), "years.csv"),
dtype=str, delimiter=",")

            self._years = np.zeros(len(yearsArr), dtype=int)
            for i, year_string in enumerate(yearsArr):
                beforeDash, afterDash = year_string.split('-')
```

```
                self._years[i] = int(beforeDash)

        except IOError:
            raise FileNotFound("years.csv not found")

    @print_return_value
    def plot_tuition_distribution(self):
        '''
        A method that plots tutition distribution and return number of states being
plotted to help the user have an overview of current tuition rate across all the
state.
        '''
        tuition_2022 = self._costsArrView[:, -1] # tution of states except Alaska
in the most recent year (2022-23)


        plt.hist(tuition_2022, edgecolor='black')
        plt.xlabel('Tuition')
        plt.ylabel('Number of College')
        plt.title('Tuition Distribution for 2022-23')

        return len(tuition_2022)

    @print_return_value
    def plot_lowest_tuition_states(self, num_states:int):
        '''
        A method that plots the tuition of the correct number of states (inputted
from user) that have the lowest tuition rate of the recent year (2022) and return
the state has lowest tuition.
        '''

        tuition_2022 = self._costsArrView[:, -1]

        sorted_indices = np.argsort(tuition_2022) # sort indices of tuition costs
        lowest_indices = sorted_indices[:num_states]  # get indices of the first N
lowest tution
        lowest_tuition = tuition_2022[lowest_indices]  #  get the first N lowest
tution

        lowest_states = self._statesArrView[lowest_indices] # get the first N
states with lowest tution

        plt.barh(lowest_states, lowest_tuition)
        plt.xlabel('Tuition')
        plt.ylabel('State')
        plt.title(f'Tuition of {num_states} States with the Lowest Tuition')
        plt.tight_layout()

        lowest_state_index = np.argmin(lowest_tuition) # get index of the lowest
tuition
        lowest_state = lowest_states[lowest_state_index]
        return lowest_state

    @print_return_value
    def plot_tuition_trend(self):
        '''
        A method that plots the tuition trend of the one state with the smallest
increase in tuition and return the state with largest increase.
        '''
```

```
        tuition_2022_last = self._costsArrView[:, -1]
        tuition_2022_first = self._costsArrView[:, 0]


        tuition_changes = tuition_2022_last - tuition_2022_first

        sorted_indices = np.argsort(tuition_changes)  # Sort indices based on
tuition changes
        largest_increase_indices = sorted_indices[-5:]  # Get the indices of the 5
states with the largest increase
        smallest_increase_index = sorted_indices[0]  # Get the index of the state
with the smallest increase


        states_with_largest_increase =
self._statesArrView[largest_increase_indices]
        state_with_smallest_increase =self._statesArrView[smallest_increase_index]

        # Plot the tuition trends for the states

        costs2022 = self._costsArrView[:,:]

        for i, state in enumerate(states_with_largest_increase):
            tuition = costs2022[largest_increase_indices[i], :]
            plt.plot(self._years, tuition, marker='o', label=state)

        tuition_smallest = costs2022[smallest_increase_index, :]

        plt.plot(self._years, tuition_smallest, marker='*',
label=state_with_smallest_increase)
        plt.xticks(self._years.astype(int),rotation=45, ha='right')
        plt.xlabel('Year')
        plt.ylabel('Tuition')
        plt.title('Tuition Trend for 5 States with Largest Increase and 1 State
with Smallest Increase')
        plt.legend(loc = 'best')

        # plt.show()

        return states_with_largest_increase[-1]

    @print_return_value
    def get_tuition_statistics(self):
        '''
        A method to show the minimum, maximum, mean, and median of the most current
tuition from all the states
        '''

        tuition_2022 = self._costsArrView[:,-1]

        minimum = np.round(np.min(tuition_2022))
        maximum = np.round(np.max(tuition_2022))
        mean = int(np.round(np.mean(tuition_2022)))
        median = int(np.round(np.median(tuition_2022)))

        return [minimum, maximum, mean, median]
```

```
def main():
    tuition = Tuition()
    tuition.plot_tuition_distribution()
    plt.show()
    tuition.plot_lowest_tuition_states(4)
    plt.show()
    tuition.plot_tuition_trend()
    plt.show()

if __name__ == "__main__":
    main()
```