✈️ Homework 4: Book a Trip! ✈️

For this assignment, you will be writing methods so that **travelers can successfully book and pay for trips** (taxi rides, bus journeys, flights, etc.) from different **travel providers**. You will also be writing and fixing test cases to ensure that when a traveler books a specific trip, it is processed correctly and both the traveler and travel provider are satisfied!

Review the starter code thoroughly before beginning this assignment, as understanding how the classes interact with each other is important. Take notes or draw a diagram if necessary.

Overview

Traveler Class

The ***Traveler*** class represents a traveler who will request a specific trip. Most of the class has been provided for you (DO NOT CHANGE any of this code). You will write the ***book_trip*** and ***view_trip_history*** methods (see **Tasks to Complete** section).

Each *Traveler* object has 4 instance variables:

- **name -** a string representing a traveler's name
- **provider_id** - an integer representing the id of the travel provider the traveler works for. The default value is *None*, if the traveler does not work for the travel provider.
- **credits** - a float showing how many travel credits are in the traveler's account. The default value is 15 credits.
- **trip_history** - a list of dictionaries which stores a traveler's history of trip requests successfully booked using the *book_trip* method

The *Traveler* class also includes several methods:

- ***__init__*** - which initializes the traveler's attributes
- ***__str__*** - which returns the traveler's *name* and the amount of *credits* in their account as as a string
- ***add_credits(self, credits)*** - which adds the passed number of *credits* to the traveler's account
- ***book_trip(self, travel_provider_obj, trip_request)*** - which takes a *TravelProvider* object and a *trip_request* dictionary where:
    - The keys are *Trip* objects

    - The values are another dictionary with keys of *num_seats* and *first_class*. It returns a boolean value (True or False) that indicates if the request was successfully placed or not.
- ***view_trip_history(self)*** - Prints an output showing all of the trips a traveler has successfully booked

Trip Class

The ***Trip*** class represents a type of trip that can be booked (taxi, bus ride, flight, etc). The code for this class has been provided for you.

A *Trip* object has 2 instance variables:

- **type** - a string representing the type of trip
- **price** - a float representing the *price per seat* of a trip type

The *Trip* class includes 2 methods:

- ***__init__*** - which initializes the *Trip* object's type and price
- ***__str__*** - which returns the type and price as a string

TravelProvider Class

The ***TravelProvider*** class represents various travel providers (airline, rideshare, bus company, etc.).

Each *TravelProvider* object has 4 instance variables:

- **name** - a string which is the name of the travel provider
- **provider_id** - an integer representing the id of the travel provider (used for *travelers* who are also employees to track who they work for)
- **income** - a float representing the income the travel provider has collected
- **capacity** - a dictionary which holds the *Trip* objects as the keys and the available number of available seats for that trip from this travel provider as the value

The *TravelProvider* class also includes several provided methods (DO NOT CHANGE any of this code). You will complete additional methods in the **Tasks to Complete** section.

- ***__init__*** - which initializes the *TravelProvider* attributes
- ***__str__*** - which returns a string with the travel provider's name and income
- ***accept_payment(self, amount)*** - which takes an amount and adds it to the travel provider's total income

Some of the *TravelProvider* class and *Traveler class* functions make use of a *trip_request* dictionary and a *trip_history* list. Examples of these variables are provided below:

```
self.taxi = Trip('Taxi', 25.00)
self.rideshare = Trip('Rideshare', 40.00)
self.bus = Trip('Bus', 50.00)
self.airplane = Trip('Airplane', 100.00)

request =  {
    self.bus: {
        "num_seats": 2,
        "first_class": False
    },
    self.rideshare: {
        "num_seats": 3,
        "first_class": True
    }
}
```

```
self.bob = Traveler(name='Bob', provider_id=None)
self.alice = Traveler(name='Alice', provider_id=17, credits=125)
self.charlie = Traveler(name='Charlie', provider_id=1, credits=200)

self.bob.trip_history = [
    {
        "provider_name": "Global Getaways",
        "total_cost": 280,
        "travel": {
            self.bus: {
                "num_seats": 2,
                "first_class": False
            },
            self.rideshare: {
                "num_seats": 3,
                "first_class": True
            }
        }
    },
    {
        "provider_name": "Grand Travels",
        "total_cost": 200,
        "travel": {
            self.airplane: {
                "num_seats": 2,
                "first_class": False
            }
        }
    }
]
```

Tasks to Complete

➔ Complete the *TravelProvider Class*

- Complete ***calculate_trip_cost(self, trip_obj, num_seats, first_class, traveler_obj)***, a method that takes the *trip_obj* (*Trip* object)*, num_seats (number of seats), first_class* (Boolean variable that specifies if a first class trip request was made), and *traveler_obj* (*Traveler* object)
    - It returns the cost based on the trip *price* and *num_seats*.
    - **Note**: A first class request for any trip adds a **50% surcharge.**
- Complete ***add_seats(self, trip_obj, num_seats)****,* a method that takes the *trip_obj* (*Trip* object) and *num_seats* (number of seats).
    - If the *trip_obj* is not a key in the *capacity* dictionary, create a new entry in the capacity dictionary with the *trip_obj* as the key and *num_seats* as the value.
    - If the *trip_obj* is a key in the *capacity* dictionary, add the *num_seats* (number of seats) to the existing value.
- Complete ***process_trip_request(self, trip_request)***, a method which takes a dictionary that represents the traveler's trip request. The outer keys are *Trip* objects and the values are another dictionary with inner keys as *num_seats* and *first_class*. It checks if this travel provider has enough available seats for the requested trip using its *capacity* dictionary.
    - If not, return **False**.
    - Otherwise, it subtracts the requested seats for that *Trip* object from the travel provider's *capacity* dictionary, and returns **True**
        - **Note:** A trip request should only be fulfilled if the travel provider has enough seats for that trip type (i.e. *capacity* should not be modified if we cannot successfully process the trip request).
        - **Note:** You can assume that a trip type will only appear once in a given trip_request

➔ Complete the *Traveler Class*

- Complete the ***book_trip(self, travel_provider, trip_request)*** method
    - Call the ***calculate_trip_cost*** method on the *trip_obj* (*Trip* object) to calculate the cost of the trip.
    - Check if the traveler has the number of credits required or more in their *account*.
        - If they don't have enough credits, return **False.**

        - Otherwise, call the ***process_trip_request*** method on the *trip_obj* (*Trip* object).
            - If ***process_trip_request*** returns **True**, remove the total number of credits from the traveler's *account*. Then, call the ***accept_payment*** method to add it to the travel provider's income. Additionally, add the trip request to the ***trip_history*** list as a dictionary with keys for *provider_name*, *total_cost*, and the *trip_request* dictionary as well (there is an example of this structure in the picture above this section). Finally, return **True**.
            - Otherwise, return **False**.
- Complete the ***view_trip_history(self)*** method
    - Prints an output showing all of the trips a traveler has successfully booked using the ***trip_history*** variable
    - If the traveler has not booked any trips yet, it returns a message indicating that "{Traveler name} has no trips booked yet."
    - Example Output:

      Trip history for Bob:

      Trip 1
      Provider: Global Getaways
      Total Cost: 280
      Bus: (Number of Seats: 2, First Class: No)
      Rideshare: (Number of Seats: 3, First Class: Yes)

      Trip 2
      ...

➔ Write Test Cases

Note: Many test cases have already been written for you. **Please do not edit any test cases other than the ones that have comments above them explicitly asking you to do so.** These test case-related tasks are also explained below:

- Write test cases for the following scenarios for the ***book_trip*** method in ***test_traveler_book_trip***:
    - The traveler doesn't have enough credits in their account to book the trip.
    - The travel provider does not have enough seats to fulfill the trip request (the available seats for that trip type are less than the requested seats).
    - The travel provider does not have the requested trip type in the *capacity* dictionary.

    - **Your test case input should be written such that we can guarantee that book_trip returned False because of the scenario we are trying to test (i.e. a single test case input should not return False because the traveler doesn't have enough credits in their account to book the trip AND because the travel provider does not have enough seats to fulfill the trip request)**

As you are working on one test case, feel free to comment out the test cases that you are not working on, but **be sure to uncomment all test cases before you turn in your homework.**

Grading Rubric (60 points)

Note that if you use hardcoding (specifying expected values directly) in any of the methods by way of editing to get them to pass the test cases, or you edit any test cases other than the ones you have been directed to, you will NOT receive credit for those related portions.

- 10 points for correctly completing the ***Traveler*** class function ***book_trip***
- 5 points for correctly completing the ***Traveler*** class function ***view_trip_history***
- 30 points for correctly completing the ***TravelProvider*** class (10 points per method)
- 15 points for completing ***test_traveler_book_trip*** (5 points per scenario)

Extra Credit (6 points)

To gain extra credit on this assignment, please edit ***calculate_trip_cost()*** to follow this additional requirement.

Every traveler that works for the travel provider receives a **20% discount** (this occurs when the traveler's ***provider_id*** matches the travel provider's ***provider_id***). If the trip booked is labeled as ***first_class***, then the employee traveler still has to pay for the 50% upcharge **before** the discount is applied.