Uber Cab Booking Assignment -- Selvakumar P --program submitted on *8/4/2020*

In [ ]:

```
import cmd

class Vehicle:
    def __init__(self, base, travel, efficency):
        self.base = base
        self.travel = travel
        self.trips = []
        self.efficency = efficency

    def cost(self, length):
        if not (3 <= length <= 50):
            raise ValueError('Invalid trip length')
        return (self.base + self.travel * length) * 1.06

    def distance(self):
        return sum(trip[0] for trip in self.trips)

    def record_trip(self, distance, people):
        if len(self.trips) + 1 > 24:
            raise ValueError("Can't store more than 24 trips")
        if self.distance() + distance > 350:
            raise ValueError("Can't store more than 350KM")
        self.trips.append((distance, people))

    def average_cost(self):
        return self.distance() * self.efficency

    def total_people(self):
        return sum(trip[1] for trip in self.trips)

    def gross_income(self):
        return sum(self.cost(distance) for distance, _ in self.trips)


def args(values, metas):
    values = values.split()
    if len(values) > len(metas):
        keys = ', '.join(m[0] for m in metas)
        raise ValueError(f'Too many keys should have {keys}')

    if len(values) < len(metas):
        keys = ', '.join(m[0] for m in metas[len(values):])
        raise ValueError(f'Missing {keys}')

    output = []
    for value, meta in zip(values, metas):
        try:
            value = meta[1](value)
        except ValueError as e:
            raise ValueError(f'Invalid value for {meta[0]}')
        output.append(value)
    return output


class VehicleRenter(cmd.Cmd):
    intro = 'Welcome to the taxi recorder.   Type help or ? to list commands.\n'
    prompt = '> '
    vehicles = {
        'UberMini':         Vehicle(2.50, 1.00, 7.4/100),
        'UberPrime': Vehicle(4.00, 1.25, 8.6/100),
        'Ubervan':      Vehicle(5.00, 1.50, 9.2/100),
    }

    def get_vehicle(self, vehicle):
        v = self.vehicles.get(vehicle, None)
        if v is None:
            keys = ', '.join(self.vehicles.keys())
            raise ValueError(f'Invalid vehicle {vehicle!r}, options are {keys}')
        return v

    def do_vehicles(self, _):
        """List all the vechicles available."""
        print(', '.join(self.vehicles.keys()))

    def do_cost(self, arg):
        """
        Calulate the cost of a trip.

        cost {vechicle name} {distance}
        cost UberMini 3
        """
        try:
            vehicle, distance = args(arg, (('vehicle', str), ('distance', float)))
            print(self.get_vehicle(vehicle).cost(distance))
        except ValueError as e:
            print(e)

    def do_trip(self, arg):
        """
        Record a trip.

        trip {vechicle name} {distance} {people}
        trip UberMini 3 1
        """
        try:
            vehicle, distance, people = args(arg, (('vehicle', str), ('distance', float), ('people'
, int)))
            self.get_vehicle(vehicle).record_trip(distance, people)
        except ValueError as e:
            print(e)


    def do_stats(self, _):
        """See the stats of the vehicles"""
        for name, v in self.vehicles.items():
            print(
                f'{name}:\n'
                f'  trips: {len(v.trips)}\n'
                f'  distance: {v.distance()}\n'
                f'  people: {v.total_people()}\n'
                f'  gross income: {v.gross_income():.2f}\n'
                f'  fuel: {v.average_cost():.2f}\n'
                f'  net profit: {v.gross_income() - v.average_cost():.2f}\n'
            )

if __name__ == '__main__':
    VehicleRenter().cmdloop()
```

```
Welcome to the taxi recorder.   Type help or ? to list commands.

> help

Documented commands (type help <topic>):
========================================
cost  help  stats  trip  vehicles

> stats
UberMini:
  trips: 0
  distance: 0
  people: 0
  gross income: 0.00
  fuel: 0.00
  net profit: 0.00

UberPrime:
  trips: 0
  distance: 0
  people: 0
  gross income: 0.00
  fuel: 0.00
  net profit: 0.00
```

```
Ubervan:
  trips: 0
  distance: 0
  people: 0
  gross income: 0.00
  fuel: 0.00
  net profit: 0.00

> vehicles
UberMini, UberPrime, Ubervan
> cost UberMini 3
5.83
> trip UberMini 3 1
> stats
UberMini:
  trips: 1
  distance: 3.0
  people: 1
  gross income: 5.83
  fuel: 0.22
  net profit: 5.61

UberPrime:
  trips: 0
  distance: 0
  people: 0
  gross income: 0.00
  fuel: 0.00
  net profit: 0.00

Ubervan:
  trips: 0
  distance: 0
  people: 0
  gross income: 0.00
  fuel: 0.00
  net profit: 0.00
```