```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


## 1.Create a module (.py file) for car
rental and import the built-in module DateTime to handle the rental time and bill.


#
In[3]:


import datetime

class CarRental:
    def __init__(self, inventory):

self.inventory = inventory

    def rent_car(self, car_type, rental_time):
        if car_type
in self.inventory and self.inventory[car_type] > 0:
            self.inventory[car_type] -=
1
            return self.generate_bill(car_type, rental_time)
        else:
            return
"Car not available for rental."

    def return_car(self, car_type):
        if
car_type in self.inventory:
            self.inventory[car_type] += 1
            return
"Car returned successfully."
        else:
            return "Invalid car
type."

    def generate_bill(self, car_type, rental_time):
        current_time =
datetime.datetime.now()
        return f"Bill generated for {car_type} rental for
{rental_time} hours. Total amount: ${rental_time * self.get_rental_rate(car_type)}"


def get_rental_rate(self, car_type):
        if car_type == "hourly":

return 10
        elif car_type == "daily":
            return 50
        elif
car_type == "weekly":
            return 200
        else:
            return 0


#
In[4]:


## 2.Create a class for renting the cars and define a constructor in it.


#
In[10]:


class CarRental:
    def __init__(self):
```

```
        pass


# In[11]:


## 3.Define a
method for displaying the available cars.
## Also, define methods for renting cars on an
hourly, daily and weekly basis, respectively.


# In[33]:


def display_available_cars():
#
Code to display available cars

 def rent_hourly(self, customer_id, car_id, rental_time):

return self._rent_car(customer_id, car_id, "hourly", rental_time)

def
rent_daily(self, customer_id, car_id, rental_time):
    return self._rent_car(customer_id,
car_id, "daily", rental_time)

def rent_weekly(self, customer_id, car_id,
rental_time):
    return self._rent_car(customer_id, car_id, "weekly",
rental_time)


# In[16]:


## 4.Inside these methods, make sure that the number of requested
cars is positive and lesser than the total available cars.


# In[17]:


def
check_requested_cars(requested_cars, total_available_cars):
    if requested_cars > 0 and
requested_cars < total_available_cars:
        return True
    else:
        return False



    import datetime


# In[20]:


## 5.Store the time of renting a car in a variable, which
can later be used in the bill while returning the car.


# In[22]:


# Store the current time
when renting the car
rental_time = datetime.datetime.now()

# Use the rental_time variable in
the bill when returning the car
```

```
# In[23]:


## 6.Define a method to return the cars using
rental time, rental mode (hourly, daily, or weekly), and the number of cars rented.


#
In[24]:


def return_cars(rental_time, rental_mode, num_cars):
    pass


# In[25]:


##
7.Inside the return method; update the inventory stock, calculate the rental period, and
generate the final bill.


# In[26]:


def generate_final_bill(inventory_stock,
rental_period):
    # Update inventory stock
    inventory_stock -= 1

    # Calculate
rental period
    rental_days = rental_period.days

    # Generate final bill

final_bill = rental_days * 10

    return final_bill


# In[27]:


## 8.Create a class for
customers and define a constructor in it.


# In[28]:


class Customer:
    def __init__(self,
name, age, email):
        self.name = name
        self.age = age
        self.email =
email


# In[29]:


## 9.Define methods for requesting the cars and returning them.


#
In[30]:


def request_car(car_id):
    # Code to request a car with the given car_id

pass
```

```
def return_car(car_id):
    # Code to return a car with the given car_id
    pass


# In[
]:
```