Python - Classes, Objects, Constructors, Recursion, Modules and Packages

1. Create a python module calculator.py with class Calculator. The calculator class should have functions add(x,y), sub(x,y), mul(x,y), div(x,y). Create another file python runner.py and import calculator module here and call add, sub, mul and div functions.

2. Create a simple python class OperatingSystem with following methods **Hint: Use OS module**
    1. show_directories("path") -> This function should print list of directories in a given path. Example: show_directories("C://myfolder") should list files inside myfolder of C drive.
    2. get_current_working_directory() -> This function should print the current working directory.
    3. copyFile(source, destination) -> This function should copy a file from a given source path to given destination path. For example: source : C:/talentpy/test.txt and destination is D:/talent/myfolder. Then this function should copy test.txt to D:/talent/myfolder.

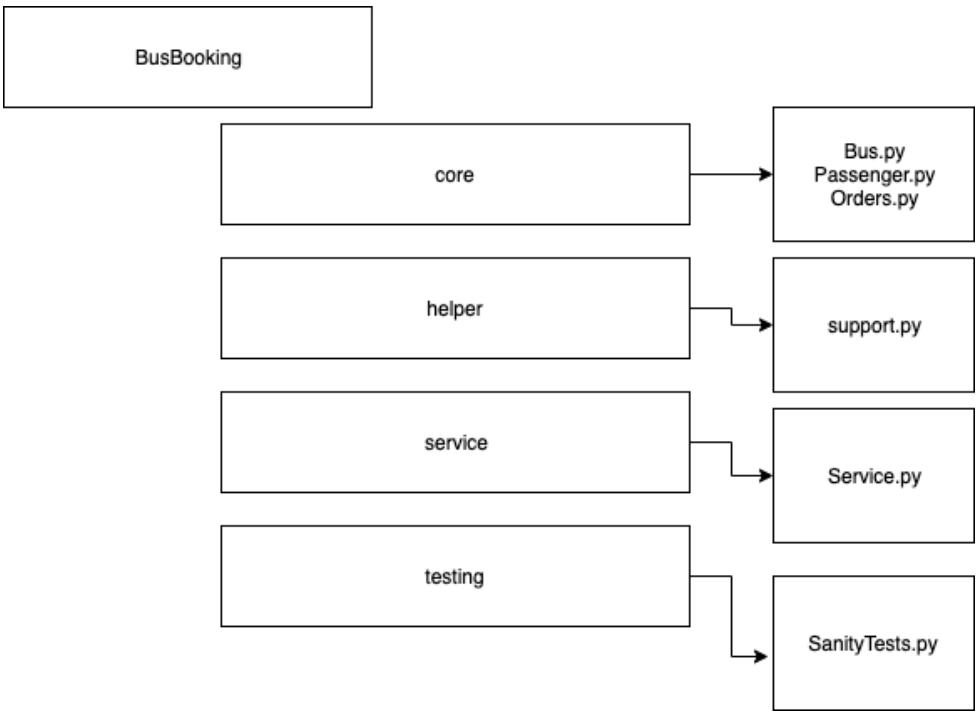
    Create another file *runner.py* and import OperatingSystem class and call show_directories, get_current_working_directory and copyFile methods.

3. Create a python class DateOperations with following methods **Hint: Use Datetime module**
    1. get_current_date() -> This function should return today's date in format DD/MM/YYYY. Example: 18/03/1994.

    Create a python class MathOperations with following methods **Hint: Use math module**
    1. get_square_root(no) -> This function should return square root of given number
    2. get_power_val(x,y) -> The function should return X power Y as output. (Example: X=3, Y=2 , then 3 power 2 => 3 * 3 = 9)

    Create another file *runner.py* and import DateOperations and MathOperations classes and call the methods get_current_date, get_square_root and get_power_val.

4. Create a python project with following structure -



Bus Booking Application:

Package core:

1. Bus.py -> Create a class Bus with parameterised constructor that can take bus_no, name, start_city, end_city, days_of_availability (list), total_seats_availaibility (Dictionary having days and seats available for given day. This class should have functions as follows -
    1. is_bus_available_only_given_day(day) -> True if the bus is available on given day else False.
    2. is_seat_available(day) -> True if seat available else False (use total_seats_availaibility to check)
    3. is_bus_of_route(start, end) -> True if the bus running between given start and destination city.
    4. book_bus(start, end, day) -> call the following
        1. is_bus_of_route -> If False return “This bus is not available”. Else step (2)
        2. is_bus_available_only_given_day() -> If False, return “Bus not running on given day”, else step (3)
        3. is_seat_available(day) -> If False, return “No seats found!” Else, update the seat availability for the bus and return the message “Booked bus : Bus No”.

Example:

- B = Bus(“B101”, “ABC Travels”, “Chennai”, “Madurai”, [“Monday”, “Tuesday”], {“Monday” : 1, “Tuesday” : 2”))
- B.is_bus_available_only_given_day(“Friday”) -> **False**
- B.is_seat_available(“Tuesday”) -> **True**
- B.is_bus_of_route(“Chennai”, “Madurai”) -> **True**
- B.is_bus_of_route(“Chennai”, “Trichy”) -> **False**
- B.book_bus(“Chennai”,”Madurai”, “Monday”) -> **Book bused: B101**
- **For next time, Monday -> 0, then** B.book_bus(“Chennai”,”Madurai”, “Monday”) -> **No seats found!**

2. Passenger.py -> Create a class Passenger with parameterised constructor that can take passenger_name, start_city, end_city, day_of_request.
    1. request_bus_ticket(passenger_name, start_city, end_city, day_of_request) -> Call book_ticket function in Orders.py file.

Example:

3. Orders.py -> Create a class Orders with constructor which initialises 3 buses as follows. **(This is just an example…)**

```
class Orders:
    def __init__(self):
        self.buses = []
        self.b1 =  Bus(“B101”, “ABC Travels”, “Chennai”, “Madurai”, [“Monday”, “Tuesday”], {“Monday” : 1, “Tuesday” : 4”))
        self.b2 =  Bus(“B102”, “Python Travels”, “Chennai”, “Madurai”, [“Monday”, “Tuesday”], {“Monday” : 1, “Sunday” : 1”))
        self.b3 =  Bus(“B103”, “talentpY Travels”, “Chennai”, “Madurai”, [“Monday”, “Tuesday”], {“Friday” : 1, “Saturday” : 2”))
        self.buses.append(b1)
```

```
        self.buses.append(b2)
        self.buses.append(b3)
```

This order class should have a function **book_ticket(passenger_name, start, end, day_of_request)** which iterates through self.buses and tries to book a ticket by calling book_bus function of bus class.

Example:

1. If Passenger.py calls book_ticket("John", "Chennai", "Madurai", "Friday"). Then the book_ticket function should iterate through self.buses
    1. b1 -> Ticket booking will be failed since it is not going to operate on Friday
    2. b2 -> Fails for same reason
    3. b3 -> Books ticket and it should return "Booked bus: B103".

Package Helper:
Create a file support.py in this package and the support.py should have class MyHelper. This MyHelper should have following methods -

1. print_time() -> prints current time

Package Service:
Create a file Service.py and inside it the class MyService with following constructor

```
class MyService:
    def __init__(self):
        # HERE CALL print_time() from package helper
        # HERE Create Passenger object
        # HERE Create Order object

    def _service_operation(self):
        # USE passenger object  and call request_bus_ticket and try to book bus and return relevant message at all steps. If it is impossible to book for a given request then print "No bus available for your request".
        # Finally again call print_time() from package helper
```

Package Testing:
Create a file SanityTests.py and class TestService.
This TestService constructor should call **MyService class -> _service_operation function**

5**.** Create a python recursive function to sum up all digits of the given number N until it becomes a single digit number.

1. Example : 208 => 2+0+8 => 10 => 1+0 = 1
2. Example: 21 => 2+1 = 3

****