CSE304-PYTHON PROGRAMMING WITH WEBFRAMEWORKS

Django Projects to Practice

1. ABC Travels company at Trichy is desirous of designing a web page to enable their customers to book tickets online. All their buses are leaving only from Trichy Central Bus stand. All passengers should board the bus only at starting point, but they may get down at any of the allowed destinations. Define a Model called BusDetails with fields Bus_No, Departure_Time, Destinations, Seats_Available, TicketCosts. The Destinations should include all cities along the route separated by comma and the corresponding ticket amount should be in TicketCosts as integer separated by comma. Using Admin Interface insert at least 10 records into BusDetails. Design a Form called Ticket with fields Bus_No, Destination, and No_of_Persons. Define a function in views.py that displays the Ticket Form for booking tickets. After getting input from the user, check the availability of seats for that bus. If required number of seats are available, then deduct the booked seats from Seats_Available, calculate and display the total ticket amount. Otherwise display the message “Sorry, Ticket not booked”.

2. Define a model called Grocery having the following fields: Name, Type (ChoiceField-Oil, Grains, Cosmetics), Quantity, RatePerUnit, and Amount. Design a form by inheriting ModelForm to get input for all fields except Amount. Define two functions in views: one is to read all records from the database and display them; another function to display the form to the user getting new Grocery item detail, calculate the Amount and insert it as a new record. Insert 10 Grocery records.

3. Define a model called FoodItem having the following fields: Name, Type (ChoiceField-Vegetable, Fruit, Nuts), and VitaminPresent. Design a customized admin interface by inheriting from ModelAdmin to display the records in Ascending / Descending order of VitaminPresent field and within that alphabetical order of Name field, to filter records based on Type field, Search on Name field. Insert 10 records with all possible combinations.

4. In number theory, a number n is called perfect if the sum of its proper divisors is equal to n, deficient if the sum of its proper divisors is less than n, and abundant if the sum of its proper divisors exceeds n. Write a Web-based Django application to perform the following: [Note: No necessity to create Model or Form]
   i. Write a function in views.py that returns the list of proper divisors for a given number [Note: Proper divisor means all the divisors of that number except that number itself. For eg. Proper divisors of 10 are: 1, 2, and 5]
   ii. Write a function in views.py that generates and returns the list of perfect, deficient and abundant numbers within a given range using list comprehension
   iii. Create a webpage using template systems to render and print the result

5. In number theory, Amicable numbers are two different numbers so related that the sum of the proper divisors of each is equal to the other number. The smallest pair of amicable numbers is (220, 284). They are amicable because the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110, of which the sum is 284; and the proper divisors of 284 are 1, 2, 4, 71 and 142, of

which the sum is 220. Write a Python program to compute the sum of all the amicable numbers under a given number. Write a Web-based Django application to perform the following: [Note: No necessity to create Model or Form]

i. Write a function that returns the list of proper divisors for a given number [Note: Proper divisor means all the divisors of that number except that number itself. For eg. Proper divisors of 10 are: 1, 2, and 5]
ii. Write a function that generates and returns the list of perfect, deficient and abundant numbers within a given range using list comprehension
iii. Write a function that takes two numbers as parameters and checks and returns whether they are amicable or not.
iv. Create a webpage using template systems to render and print whether the following list of pairs given as tuples are amicable or not using the function.
[(150, 290), (220, 284), (1184, 1210), (1350, 1851)]

6. A weighted directed graph G=(V,E) may be represented as a dictionary of edges where for each edge (start vertex, end vertex) tuple is stored as key with its weight as value.
   i. Create a module named graph.py to define a python class named Graph that has the following function attributes: __init__, transformEdges, and sortEdges
   ii. In __init__ function, read input from a text file named graph.txt that contains the graph details in the following format and initializes the edge_list attribute as a dictionary. First Line contains no. of edges n, and subsequent n lines contains for each edge start vertex, end vertex and its weight.
   iii. In transformEdges function, create a list of tuples (start vertex, end vertex, weight) from edge_list using list comprehension, and return the list
   iv. In sortEdges function, convert the edge_list attribute as list of tuples using the transformEdges function, sort them and return the list of tuples in increasing order of their edge weight.
   v. Create a webpage using template systems that takes the graph represented as dictionary render and print the edges in increasing order of their weight using the function sortEdges.

7. Create a csv file that contains name of person, status (Professional / Politician / Employee), income per year, list of fixed deposit values, dictionary of assets and their values in the following format:

```
Name, Status, Income, FD_List, Asset_Value_Dict
AAA, Professional, 800000, [1200000, 45000,300000, 200000], {House:4500000, Car:600000, Land:4000000, Jewels:1000000}
BBB, Politician, 10000000, [2000000, 1000000, 25000000], {House:30000000, Car:2000000, Land:100000000, Jewels:25000000}
CCC, Employee, 700000, [500000, 20000000, 150000], {House:1000000, Jewels:250000}
…….
```

   i. Create module called person to define a python class named Person to hold attributes name (string), status (string), income (integer), total deposits value (sum of all FD values - integer) and total assets value (sum of all assets values - integer). Provide functions to initialize its members, to return string representation of object by concatenating all details as a string separated by comma. Also define another function to check the validity of candidature using the following rules:

ii. If the person is a Professional and his total deposits exceeds 10 times of his annual income or the total assets value exceeds 25 times of his annual income, then raise an ValueError Exception “IT Raid Alert” message

iii. If the person is a Politician and his total deposits and total assets value exceeds 10 times of his annual income, then raise ValueError Exception “Disproportionate Assets Alert” message

iv. If the person is an Employee and his total deposits or total assets value exceeds 20 times of his annual income, then raise ValueError Exception with “Scam Alert” message

v. Create another module called candidate and define createCand_List() function to create Person objects from the csv file and store those objects as a module level list object called candidate_list.

vi. Create a webpage using template systems that takes candidate_list, checks the validity of candidature of each person to render and display the alert message if exception is raised (or else “Good” message if no exception) along with the details of each person.