**Python Questions :**

**I have a couple of problems that I would like you to work progressively on paper using python code without any import modules.**

**1. If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000. (Answer: 233168) (solution code attached: problem\_1\_solution\_code.py)**

**Solution code attached: problem\_1\_solution\_code.py**

**Link:**

**2. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ... By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms. (Answer: 4613732) (solution code attached: problem\_2\_solution\_code.py)**

**Solution code attached: problem\_2\_solution\_code.py**

**Link:**

**Once you have them written on paper, can you put that code on the system and make changes as required and share the changed code as well.**

**Django Questions :**

**Setting up a new environment:**

python3 -m venv env

source env/bin/activate

**# Now, we're inside a virtual environment, we can install our package requirements.**

pip install django

pip install djangorestframework

**1. What command line interface instructions are required to create a django project called**

**‘company’ and application called ‘staff’?**

**Getting started:**

# Okay, we're ready to get coding. To get started, let's create a new project 'company' to work with.

**django-admin startproject company**

**cd company**

# Once that's done we can create an app 'staff' that we'll use to create a sample Web API.

**python manage.py startapp staff**

**2. models.py - Please use the django ORM framework to create the tables below.**

**Employee table**

**Employee name**

**Department**

**Email**

**Date of birth**

**Picture**

**Department table**

**Department name**

**Manager (employee)**

**Comments:**

* ‘department’ field in the Employee table should be ForeignKey which refers to the Department Table, but it generates circular dependency. To avoid circular dependency, it is handled by the EmployeeForm class of forms.py.
* **Other Solution to handle circular dependency:**

**class EmployeeDepartment(models.Model):**

department = models.ForeignKey(Department, to\_field='id', related\_name='dept', on\_delete=models.CASCADE,)

employee = models.ForeignKey(Employee, to\_field='id', related\_name='manager', on\_delete=models.CASCADE,)

**3. admin.py - Please implement the code to allow the admin interface to be used to**

**manipulate the database for both the Employee and Department objects. When viewing**

**the Employee objects you should be able to filter by department and date of birth. When**

**creating/editing a Department object there should be inlines for each of the Employee**

**objects in the Department.**

**4. What command line interface instructions are required to be able to test the admin interface after you have completed #2 and #3 above?**

**# Make sure you’re inside the project directory where manage.py is exists.**

company$ **python3 manage.py shell**

Python 3.8.10 (default, Jun 2 2021, 10:49:15)

[GCC 9.4.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

(InteractiveConsole)

>>> **from staff.models import Department, Employee**

>>> **employee = Employee(name='Hari',department='BA')**

>>> **employee**

<Employee: Hari>

>>> **dept =Department(name="BA",manager=employee)**

>>> **dept**

<Department: BA>

>>> exit

**5. views.py (Department) - Using generic class-based views please write the following views for the Department objects: list and detail**

**6. forms.py - Create a ModelForm class to create an Employee object only entering the Employee name and Department fields.**

**7. views.py (Employee) - Using a generic class-based view use the form in #6 in a create view for Employee objects. Is your models.py class for Employee compatible with the form and view?**

**Yes, Done by filling department names through the values of department Object. (Please refer code comments are Added)**

**8. What template files would be required for the views in #5 and #7 to work? Where would you put these files?**

# 5 to find a template for list  **<app\_name/modelname\_list.html>**

**staff/department\_list.html**

# 5 to find a template for detail  **<app\_name/modelname\_detail.html>**

**staff/department\_detail.html**

# 7 to find a template for detail  **<app\_name/modelname\_form.html>**

**staff/employee\_form.html**