Q1. What is Flask Framework? What are the advantages of Flask Framework?

A1. Flask is a micro web framework for Python that is lightweight and easy to use. It is designed to be simple and minimalistic, allowing developers to build web applications quickly and efficiently. Some advantages of Flask include:

Simplicity: Flask has a simple and easy-to-understand API, making it a good choice for beginners and experienced developers.

Extensibility: Flask is highly extensible, allowing developers to add features as needed through its ecosystem of extensions.

Minimal Boilerplate: Flask does not require a lot of boilerplate code, so you can get started with writing your application logic quickly.

Built-in Development Server: Flask comes with a built-in development server for testing and development, making it convenient for local development.

Widely Supported: Flask is well-documented and has a strong community, which means you can find a lot of resources and support.

Q2. To create a simple Flask application to display "Hello World!!" in a Jupyter Notebook, you would typically use a Python script. Unfortunately, I cannot directly display screenshots, but I can provide you with the code to create the Flask application:

In [ ]:
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!!'

if __name__ == '__main__':
    app.run()
```

Q3. What is App routing in Flask? Why do we use app routes?

A3. App routing in Flask refers to the process of mapping URLs to view functions in your web application. It allows you to define the behavior of your application based on the URL requested by the client. App routes are defined using the @app.route() decorator in Flask.

We use app routes in Flask for the following reasons:

URL Handling: App routes allow you to specify which URL paths should trigger which view functions. This is fundamental for creating a navigable web application.

Modularity: Routes make your code modular and organized, as each route is associated with a specific view function, making it easier to manage and maintain the application.

Dynamic Content: Routes are essential for handling dynamic content, where parameters or variables can be part of the URL and are extracted by view functions.

Request Handling: Routes enable you to define how different HTTP methods (GET, POST, etc.) should be handled for specific URLs.

Clean and Readable Code: Using app routes results in clean and readable code, as it separates the URL handling logic from the application's core functionality.

Q4. To create the "/welcome" and "/" routes to display the specified information in Flask, you can modify the Flask application as follows:

In [ ]:
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return """Company Name: ABC Corporation
Location: India
Contact Detail: 999-999-9999"""

@app.route('/welcome')
def welcome():
    return "Welcome to ABC Corporation"

if __name__ == '__main__':
    app.run()
```

Q5. What function is used in Flask for URL Building? Write a Python code to demonstrate the working of the url_for() function.

A5. In Flask, the url_for() function is used for URL building. It generates a URL for a given view function and any arguments it needs. Here's an example:

In [ ]:
```
from flask import Flask, url_for

app = Flask(__name__)

@app.route('/')
def index():
    return "Welcome to my website."

@app.route('/user/<username>')
def user_profile(username):
    return f"Hello, {username}!"

if __name__ == '__main__':
    with app.test_request_context():
        print(url_for('index'))  # Output: '/'
        print(url_for('user_profile', username='john'))  # Output: '/user/john'
```