```
# def multiplesOfThreeAndFive():
    sum = 0
    value = range(1,1000)
    for x in value:
        if (x % 3 == 0 or x % 5 == 0)
            sum = sum + x;
    print(sum)

multiplesOfThreeAndFive()
```

```
def fibonacci():
    a = 1
    b = 2
    sum = 0
    c = 0
    while (c <= 4000000)
        c = a + b
        a = b
        b = c
        if (c % 2 == 0)
            sum = sum + c;
print(sum)

fibonacci()
```

```
for x in range(1,11):
```

Python virtual environments.

The venv module supports creating lightweight "virtual environments" each with their own independent set of python packages installed in the site directories. A virtual environment is created on the top of an existing Python installation, known as virtual environment's "base" python, and may optionally be isolated from the packages in the base environment.

- Virtual environment is the isolated environment specific for the installation of dependencies independently with the system-wide Python installation.

The importance of virtual environment are:

1) Dependency isolation:
→ Virtual environments is isolated environment that contains specific Python interpreter and software libraries needed for a project. They are isolated from other virtual environment and the system-wide Python installation, ensuring that dependencies one of one project do not interfere with those of another.

Disposable nature.

→ Virtual environment are disposable in nature i.e they can be deleted recreated from scratch. They donot contain project code, instead, they provide dependencies and environment to run sepesately.

Sandboxing.

They provide a sandboxed environment for your project, which means you can expesiment with different packages and configurations without worying about affecting other projects or System Python environment.

Version Control

Virtual environments enable you to specify the exact version of packages required for your project, ensuring consistency across different environments and making it easier to reproduce your development environment.

⇒ Basic commands in python virtual environment

- mkvirtualenv
- rmvirtualenv
- workon
- deactivate.

At first we need to install virtualenvwrapper.

pip install virtualenvwrapper.

mkvirtualenv.

This command is used to create a new virtual environment. 'mkvirtualenv' not only creates the virtual environment but also automatically activates.

mkvirtualenv myenv.

rmvirtualenv.

→ This command is used to remove an existing virtual environment. It deletes the virtual environment directory & all its content.

rmvirtualenv myenv.

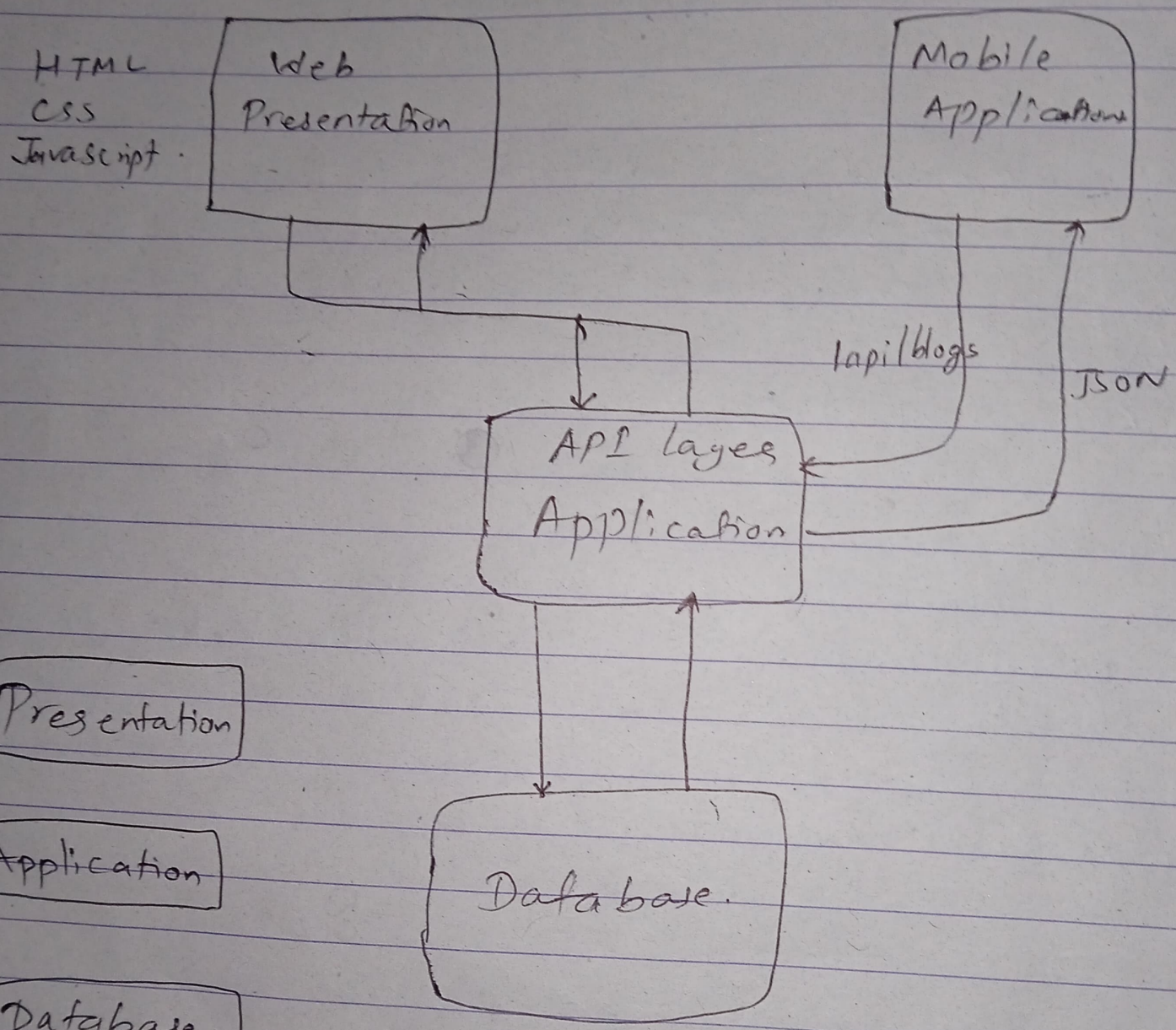
workon

→ This command is used to activate a virtual environment. It lists all available virtual environment and allows you to activate the one you want to work with.

workon myenv.

deactivate. used to deactivate the currently deactivate. active virtual environment

Modern Web Architecture.



Presentation

Application

Database.

For the projeet like this (mentioned in question), I would prefer to use modern web architecture. Modern Web Architecture is the architecture which consists of three tiers i.e presentation, application & database. ~~In presentation~~ Modern web architecture helps to perform the communication between three tiers which is the key to make a secure & well web applications. Here,

In presentation layer, all the task related to viewing of the projects is considered. i.e the client or users can see the interface. ~~and~~ It is the topmost layer of the application responsible for interacting with users and presenting information to them in a human-readable format. It helps to represent the graphical elements.

- The presentation layer communicates with application layer to request and receive data.
- Examples of technologies, HTML, CSS, JS.

) Application Layer.

→ The application layer also known as business logic layer, contains the core logic and functionality of the application.

It processes user requests, performs business operations and arranges interactions between

presentation layer & data-layer:
- The technologies involved are Java, C#, Python, Node etc.
- The application layer implements the business rules, workflows, validations, and calculations required to fulfill user requests & produce meaningful outcomes.

Database Layer.
- The database layer is responsible for managing the data like storing, retrieval of data by the application.
- It provides mechanisms for storing, updating, deleting data, as well as enforcing data integrity, security & access controls.

→ The DBMS used are MySQL, MongoDB etc.

Mobile Applications lies ~~in the application~~
→ It receives the data by requesting in application layer. It gets api and use api for manipulating data.

This architecture would be the best selection because:

- Seperation of Concerns.
  - Each tier focuses on speeific aspect of the application's functionality, promoting modularity, maintainability & reusability of code.

- Scalability.
  The modular architecture allows different components of the app to be scaled independently, making it easier to accomodate changes in ~~future~~ uses demand or feature requirements

- Platform Flexibility :
  The seperation of ~~de~~ layers enables developers to adapt the application logic & data access ~~to~~ mechanisms of different platforms. (eg:- Android, iOS) while maintaining a consistent user experience.

- Testing & debugging.
  With clear seperation of concerns, it becomes easier to write unit tests, conduct code reviews, and debug issues within speeific layers of the application.

Security:

By enforcing data access controls and encryption mechanisms at the data layer, the developer can ensure that sensitive information stored on the device or transmitted over the network is protected from unauthorized access or tampering.

3 b) What are HTTP status Codes? Explain its advantages along with its categorization.

=> HTTP stands for HyperText Transfer Protocol. The status codes that HTTP used for indicating several things which has certain meaning over web in HTTP are HTTP status codes.

An HTTP status code is a server response to browser's request. When we visit a website, your browser sends a request to the site's server, and the server then responds to the browser's request with a three-digit code: the HTTP status code.

Some of the common HTTP status code are:

1xxs - Information responses:
The server is thinking through the request.

2xxs - Success!
The request was successfully completed and the server gaves the browser the expected response.

3xxs - Redirection: You got redirected somewhere else. The request was received, but there is a redirect of some kind.

4xxs - Client errors - Page not found. The site or page couldn't be reached. (The request was made but the isn't valid - this is an error on the website's side of the conversation (often appears when a page doesn't exist).

5xxs - Server errors.
Failure. A valid request was made by the client but server failed to complete the request.

- Standardization:
HTTP status codes provide a standardized way for servers to communicate the outcome of requests to clients, ensuring interoperability & consistency across different systems & platforms.

- Error Handling.
→ Status code helps clients & developers identify & troubleshoot errors more efficiently by providing specific information about the nature of error & suggesting corrective actions

- Automated Handling:
Certain status code such as redirection codes, allow clients to automatically handle certain situations without user intervention, improving the uses experience & reducing the need for manual intervention.

- SEO & accessibility.
Proper use of status codes, especially redirection codes, can have positive effects on SEO and accessibility by ensuring that URLs are correctly indexed and that users are directed to the correct resource.

Q4)

⇒ Create database trekapp.

```
Create tables treks (
    id int Auto_increment PRIMARY KEY,
    place_name varchar(50) Not Null,
    milestones varchar(50) Not Null,
    total_days int Not Null,
    price int Not Null,
    user_name varchar
)
```

```
Create table users (
    user_id int AUTO_INCREMENT PRIMARY KEY,
    place_name varchar(50) NotNull,
    user_name varchar(50) NotNull,
    age int notNull )
```

```
INSERT INTO treks (place_name, milestones,
    total_days, price) VALUES ('Test',
    'Test', 5, 10000);
```

```
SELECT * from trek;
```

```
UPDATE treks SET placename = 'Pokhara',
    milestones = 'Damauli' where id = '2';
```

```
DELETE from trek where id = 2;
```

→ Via Forms, title = request.form['title'], category = request.form['category'], description = request.form['description'], member_count = request.form['member_count'].

- Via JSON payload.

req = request.json title = req.get("title"), category = req.get("category") description = req.get("description") member_count = req.get("member_count").

5b)

⇒ GET / PUT / DELETE

GET → The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

PUT.

The PUT method replaces all current representations of the target resource with the request payload.

DELETE

DELETE method deletes the specified resource.

HTTP vs HTTPS.

HTTP	HTTPS
→ Hypertext transfer Protocol	- Hypertext transfer Protocol Secure.
Default port for HTTP is 80	- Default port for HTTPS is 443.
→ It is used for older text-based websites	→ It is used all modern websites
→ It is less secure that no additional seeeurity features	→ It is secured with the use of SSL certificates for public-key encryption
→ It made communicati- on over the internet possible.	improves web authority, trust, search engine rankings.
→ It is comparatively fast then HTTPS.	→ It is slower than HTTP.

GET vs POST.

→ GET	POST
1) Data is visible to everyone in the URL	- Data is not visible to everyone in the URL.
2) GET is less secured than POST because data sent is part of the URL.	- POST is more secured than GET because the parameters are not stored in browser history or in web server logs.
3) Parameters remains in browsers history	- Parameters are not saved in browsers history.
4) The length of the URL is limited (2048 characters)	- There is no such restrictions.
5) GET request can be cached, bookmarked	- POST request cannot be cached and bookmarked.
6) Only ASCII characters allowed.	- No restrictions. Binary data is also allowed.

PUT	POST.
→ It is called when you have to modify a single resource.	→ It is called when you have to add a child resource.
→ PUT method responses cannot be cached.	- POST method responses can be cached.
→ Can use UPDATE query in PUT.	- Create query is used in POST.
→ Client decides which URI resource should have	- server decides which URI resource should have.
→ PUT works as specific	- POST work as abstract.
→ PUT method is idempotent	- POST method is not idempotent.
→ If you send same PUT request multiple times, the result will remain same.	- If we send POST request multiple times, you will receive different result.

Render template in flask

→ First we need to create a directory named 'template' in project directory. Flask will look for templates in this directory by default. Then, we can add html files in template directory.

- To render a template in flask, you'll need to use the 'render_template' ~~from~~ function from flask module.

```
from flask import Flask, render_template.

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html', name='wor

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

→ for run python app.py

→ We defined the route '/' using the '@app.route()'
→ index function returns the result of render_template
which takes the name of the template file ('index.ht
as the first argument and any additional data
to pass to the template as keyword argument.
Here we're passing the name='world'.

When you run the Flask application & visit the root
URL ('http://localhost:5000/' by default), Flask
will render 'index.html'.

#56)

```
from flask import Flask, ~~render_template~~
                        request, jsonify.

app = Flask(__name__)

# Sample data.
books = {
    1: {"title": "Book1", "author": "Author 1"},
    2: {"title": "Book2", "author": "Author 2"}
}

@app.route('/books', methods=['GET'])
def get_books():
    return jsonify(books)

@app.route('/books/<int:book_id>', methods=['GET'])
def get_book(book_id):
    if book_id in books:
        return jsonify(books[book_id])
    else:
        return jsonify({"error": Book not found'})

@app.route('/books/<int:book_id>', methods=['PUT']
def update_book(book_id):
    if book_id in books:
        data = request.get_json()
        books[book_id] = data.
```

```
    return jsonify ({"Message": "Book updated
                    successfully"}).
else:
    return jsonify ({"error": "Book Not Found"}), 404.

@app.route ('/books/<int:book_id>', methods=['DELETE'])
def .delete_book (book_id)
    if book_id in books:
        del books[book_id]
    . return jsonify ({"Message": Book
                    deleted successfully"})
    else:
        return jsonify ({"error": "Book not found"})
                                    404

if __name__ = "__main__":
    app.run (debug = True).
```