Flask

1. Launch a Web App using Flask

```
from flask import Flask
'''
 It creates an instance of the Flask class,
 which will be your WSGI (Web Server Gateway Interface) application.
'''
###WSGI Application
app=Flask(__name__)

@app.route("/")
def welcome():
    return "Welcome to this best Flask course.This should be an amazing course"

@app.route("/index")
def index():
    return "Welcome to the index page"

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

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a producti
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

2. Render Template

```
from flask import Flask,render_template
'''
 It creates an instance of the Flask class,
 which will be your WSGI (Web Server Gateway Interface) application.
'''
###WSGI Application
app=Flask(__name__)

@app.route("/")
def welcome():
    return "<html><H1>Welcome to the flask course</H1></html>"

@app.route("/index")
def index():
    return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

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

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a producti
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Flask App</title>
</head>
<body>
    <h1>Welcome to My Flask App!</h1>
    <p>This is a simple web application built with Flask.</p>
</body>
</html>
```

about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>About</title>
</head>
<body>
    <h1>About</h1>
    <p>This is the about page of my Flask app.</p>
</body>
</html>
```

3. GET and POST Requests

```
from flask import Flask,render_template,request
'''
 It creates an instance of the Flask class,
 which will be your WSGI (Web Server Gateway Interface) application.
'''
###WSGI Application
app=Flask(__name__)

@app.route("/")
def welcome():
    return "<html><H1>Welcome to the flask course</H1></html>"
```

```
@app.route("/index",methods=['GET'])
def index():
    return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/form',methods=['GET','POST'])
def form():
    if request.method=='POST':
        name=request.form['name']
        return f'Hello {name}!'
    return render_template('form.html')

@app.route('/submit',methods=['GET','POST'])
def submit():
    if request.method=='POST':
        name=request.form['name']
        return f'Hello {name}!'
    return render_template('form.html')


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

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a producti
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Form</title>
    <link rel="stylesheet" href="{{ url_for('static',filename='css/style.css') }}">
</head>
<body>
    <h1>Submit a Form</h1>
    <form action="/submit" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name">
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

4. Jinja 2 Template Engine

```
### Building Url Dynamically
```

```
## Variable Rule
### Jinja 2 Template Engine

### Jinja2 Template Engine
'''
{{  }} expressions to print output in html
{%...%} conditions, for loops
{#...#} this is for comments
'''

from flask import Flask,render_template,request,redirect,url_for
'''
 It creates an instance of the Flask class,
 which will be your WSGI (Web Server Gateway Interface) application.
'''
###WSGI Application
app=Flask(__name__)

@app.route("/")
def welcome():
    return "<html><H1>Welcome to the flask course</H1></html>"

@app.route("/index",methods=['GET'])
def index():
    return render_template('index.html')

@app.route('/about')
def about():
    return render_template('about.html')

## Variable Rule
@app.route('/success/<int:score>')
def success(score):
    res=""
    if score>=50:
        res="PASSED"
    else:
        res="FAILED"

    return render_template('result.html',results=res)

## Variable Rule
@app.route('/successres/<int:score>')
def successres(score):
    res=""
    if score>=50:
        res="PASSED"
    else:
        res="FAILED"

    exp={'score':score,"res":res}

    return render_template('result1.html',results=exp)

## if confition
@app.route('/sucessif/<int:score>')
def successif(score):

    return render_template('result.html',results=score)

@app.route('/fail/<int:score>')
def fail(score):
    return render_template('result.html',results=score)
```

```
@app.route('/submit',methods=['POST','GET'])
def submit():
    total_score=0
    if request.method=='POST':
        science=float(request.form['science'])
        maths=float(request.form['maths'])
        c=float(request.form['c'])
        data_science=float(request.form['datascience'])

        total_score=(science+maths+c+data_science)/4
    else:
        return render_template('getresult.html')
    return redirect(url_for('successres',score=total_score))

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

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a producti
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

result.html

```
<h1>

  Based on the marks You have  {{ results }}

  {% if results>=50 %}
  <h1>You have passed with marks {{results}}</h1>
  {% else %}
  <h2>You have failed with marks {{results}} </h2>
  {% endif %}

</h1>
```

result1.html

```
<html>
<h2>
    Final Results
</h2>
<body>

     {% for key,value in results.items()   %}

      {# This is the comment section #}
      <h1>{{ key }}<h1>
      <h2>{{ value }}</h2>

     {% endfor %}
```

```
</body>
</html>
```

getresult.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="{{ url_for('static',filename='css/style.css') }}">
<script type="text/javascript" src="{{ url_for('static',filename='script/script.js') }}">

</script>

</head>
<body>

<h2>HTML Forms</h2>

<form action="/submit" method='post'>
  <label for="Science">Science:</label><br>
  <input type="text" id="science" name="science" value="0"><br>
  <label for="Maths">Maths:</label><br>
  <input type="text" id="maths" name="maths" value="0"><br><br>
  <label for="C ">C:</label><br>
  <input type="text" id="c" name="c" value="0"><br><br>
  <label for="datascience">Data Science:</label><br>
  <input type="text" id="datascience" name="datascience" value="0"><br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called "/submit".</p>

</body>
</html>
```

5. PUT and DELETE Requests

```
### Put and Delete-HTTP Verbs
### Working With API's--Json

from flask import Flask, jsonify, request

app = Flask(__name__)

##Initial Data in my to do list
items = [
    {"id": 1, "name": "Item 1", "description": "This is item 1"},
    {"id": 2, "name": "Item 2", "description": "This is item 2"}
]

@app.route('/')
def home():
```

```
    return "Welcome To The Sample To DO List App"

## Get: Retrieve all the items

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

## get: Retireve a specific item by Id
@app.route('/items/<int:item_id>',methods=['GET'])
def get_item(item_id):
    item=next((item for item in items if item["id"]==item_id),None)
    if item is None:
        return jsonify({"error":"item not found"})
    return jsonify(item)

## Post :create a new task- API
@app.route('/items',methods=['POST'])
def create_item():
    if not request.json or not 'name' in request.json:
        return jsonify({"error":"item not found"})
    new_item={
        "id": items[-1]["id"] + 1 if items else 1,
        "name":request.json['name'],
        "description":request.json["description"]
    }
    items.append(new_item)
    return jsonify(new_item)

# Put: Update an existing item
@app.route('/items/<int:item_id>',methods=['PUT'])
def update_item(item_id):
    item = next((item for item in items if item["id"] == item_id), None)
    if item is None:
        return jsonify({"error": "Item not found"})
    item['name'] = request.json.get('name', item['name'])
    item['description'] = request.json.get('description', item['description'])
    return jsonify(item)

# DELETE: Delete an item
@app.route('/items/<int:item_id>', methods=['DELETE'])
def delete_item(item_id):
    global items
    items = [item for item in items if item["id"] != item_id]
    return jsonify({"result": "Item deleted"})

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

```
 * Serving Flask app '__main__'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a producti
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
```

sample.json

```
{"name": "New Item", "description": "This is a new item"}

{"name": "Updated Item", "description": "This item has been updated"}
```