```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "1. What is a RESTful API?\n",
        "-  A RESTful API (Representational State Transfer API) is a type of web ser
vice   that follows the principles and constraints of REST architecture. It allows
 different software systems to communicate over the internet using HTTP requ
ests. RESTful APIs are widely used for building web services that are scalable,
stateless, and easy to integrate."
      ],
      "metadata": {
        "id": "-7tM6N1ju-u8"
      }
    },
    {
      "cell_type": "markdown",
      "source": [],
   "metadata": {
    "id": "cJcVLNhT4z0V"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "2. Explain the concept of API specification?\n",
    "-  An API specification is a detailed, structured description of how an API
behaves and how it should be used. It serves as a contract between the API pro
vider and the API consumer (developer), defining the endpoints, data formats, m
ethods, request/response structure, and authentication requirement\n",
    "- Endpoints\n",
    "- HTTP Methods\n",
    "- Request Parameter\n",
    "- Request and Response Formats\n",
    "- Response Codes\n",
    "- Authentication/Authorization\n",
    "- Data Models/Schemas"
   ],
   "metadata": {
    "id": "aPWj33RDvVvQ"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "3. What is Flask, and why is it popular for building APIs?\n",
    "-  Flask is a lightweight, Python-based web framework that's widely used f
or    building web applications and RESTful APIs. It is known for its simplicity, fl
exibility, and minimalism.\n",
    "- Why Flask Is Popular for Building APIs:-\n",
    "- Minimalist and Lightweight\n",
      "- Quick and Easy to Learn\n",
      "- Flexible and Unopinionated\n",
      "- Built-in Development Server\n",
      "- Large Ecosystem\n",
      "- Good Integration with Other Tools"
    ],
    "metadata": {
      "id": "Uv5xUXCiv7ms"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "4. What is routing in Flask?\n",
      "-  Routing in Flask is the process of associating URLs (routes) with    func
tions   (view functions) that handle requests to those URLs."
    ],
    "metadata": {
      "id": "itQ6xCzMwmaU"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "5. How do you create a simple Flask application?\n",
      "- To create a simple Flask application step by step:-\n",
      "- Install Flask\n",
      "- Create your flask App File\n",
      "- Run the flask app"
    ],
    "metadata": {
      "id": "gX90SAS0w5Yy"
    }
  },
  {
   "cell_type": "markdown",
   "source": [
    "6. What are HTTP methods used in RESTful APIs?\n",
    "-  In RESTful APIs, HTTP methods define the type of operation the client w
ants to perform on a resource (such as users, products, posts, etc.). Each meth
od has a specific purpose.\n"
   ],
   "metadata": {
    "id": "Oyh_SbMwxXpN"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "7. What is the purpose of the @app.route() decorator in Flask?\n",
    "-  Purpose of @app.route():-\n",
    "-  It maps a specific URL (route) to a function that handles the request.\n",
    "-  It allows you to define endpoints in your web application or API.\n"
   ],
   "metadata": {
    "id": "P3MF6DT3xkbu"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "8.  What is the difference between GET and POST HTTP methods?\n",
    "-   The main difference between GET and POST HTTP methods lies in how
 data is sent to the server and what they are used for."
   ],
   "metadata": {
      "id": "_PFLIKsVx4QQ"
     }
    },
    {
     "cell_type": "markdown",
     "source": [
      "9.  How do you handle errors in Flask APIs?\n",
      "-   Handling errors in Flask APIs ensures that your application returns clea
r, consistent, and meaningful responses when something goes wrong. Flask pro
vides multiple tools for catching and responding to errors gracefully.\n",
      "-   Using abort() to Trigger HTTP Errors\n",
      "-   Custom Error Handlers (@app.errorhandler)\n",
      "-   Try/Except Blocks for Manual Error Handling\n",
      "-   Raise HTTP Exceptions with werkzeug"
     ],
     "metadata": {
      "id": "uHH8-gQCyHxa"
     }
    },
    {
     "cell_type": "markdown",
     "source": [
      "10. How do you connect Flask to a SQL database?\n",
      "-   Connecting Flask to a SQL database is commonly done using an ORM (
Object-Relational Mapper) like SQLAlchemy, which makes database interaction
s easier and more Pythonic.\n",
      "-   Install Required Packages\n",
      "-   Set Up Flask and SQLAlchemy\n",
      "-   Define Database Models\n",
      "-   Create the Database and Tables\n",
      "-   Use Models to Perform CRUD Operation."
     ],
     "metadata": {
      "id": "qcKFlYx_yxUz"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "11. What is the role of Flask-SQLAlchemy?\n",
      "-   Flask-SQLAlchemy is an extension for Flask that simplifies the integrati
on of SQL databases with Flask applications by providing a convenient and pow
erful Object-Relational Mapping (ORM) layer."
    ],
    "metadata": {
      "id": "Rav-nYmizPPq"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "12. What are Flask blueprints, and how are they useful?\n",
      "-   Flask Blueprints are a way to organize your Flask application into small
er, reusable components or modules. They let you group routes, templates, stat
ic files, and other code related to a specific feature or section of your app.\n"
    ],
    "metadata": {
      "id": "Qwx56k9VzalU"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "13. What is the purpose of Flask's request object?\n",
      "-   Flask’s request object represents the incoming HTTP request that a cli
ent (like a browser or API consumer) sends to your Flask application.\n",
      "-   Purpose of Flask’s request Object:-\n",
      "-  It gives you access to all data sent by the client as part of the request.\
n",
      "-  You can retrieve form data, query parameters, headers, cookies, uploade
d     files, and more.\n",
      "-  It helps your app understand what the client wants and respond accordin
gly."
    ],
    "metadata": {
      "id": "KDJKAODTzlIL"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "14. How do you create a RESTful API endpoint using Flask?\n",
      "-   Creating a RESTful API endpoint in Flask involves defining a route that r
esponds to HTTP methods (like GET, POST) and returns data—typically in JSON
 format.\n",
      "-   Steps to Create a RESTful API Endpoint in Flask:-\n",
      "-   Import Flask and JSON utilitiez\n",
      "-   Initialize your Flask app\n",
      "-   Define the API endpoint with appropriate HTTP methods\n",
      "-   Run the Flask app"
    ],
    "metadata": {
      "id": "21vVgNsq0Ad8"
    }
  },
  {
    "cell_type": "markdown",
    "source": [
      "15. What is the purpose of Flask's jsonify() function?\n",
      "-   The purpose of Flask's jsonify() function is to convert Python data stru
ctures (like dictionaries or lists) into a JSON-formatted HTTP response that ca
n be sent back to the client."
     ],
     "metadata": {
      "id": "8xakmea30lwe"
     }
    },
    {
     "cell_type": "markdown",
     "source": [
      "16. Explain Flask’s url_for() function?\n",
      "-   Flask’s url_for() function is used to generate URLs for your Flask routes
 dynamically based on the name of the view function (endpoint) instead of hard
coding URLs."
     ],
     "metadata": {
      "id": "ogF8YZma0t3R"
     }
    },
    {
     "cell_type": "markdown",
     "source": [
      "17. How does Flask handle static files (CSS, JavaScript, etc.)?\n",
      "-   By default, Flask expects static files to be in a folder named static at th
e root of your project.\n",
      "-   These files are accessible via URLs starting with /static/.\n",
      "-   For example, a file static/style.css is served at /static/style.css."
     ],
     "metadata": {
      "id": "-MzrYRgu08QZ"
     }
    },
  {
   "cell_type": "markdown",
   "source": [
    "18. What is an API specification, and how does it help in building a Flask A
PI?\n",
    "-   It describes the contract between the API provider and consumers.\n",
    "-   Common formats include OpenAPI (Swagger), RAML, and API Blueprint.
\n",
    "-   Includes details like:\n",
    "-   Available endpoints (URLs)\n",
    "-   Supported HTTP methods (GET, POST, etc.)\n",
    "-   Request parameters and body structure\n",
    "-   Response formats and status codes\n",
    "-   Authentication and security requirements\n",
    "-   Error messages and codes\n",
    "\n",
    "-   How Does an API Specification Help When Building a Flask API?\n",
    "-   Clear Design Blueprint\n",
    "-   Improves Communication\n",
    "-   Speeds Up Development\n",
    "-   Facilitates Testing and Validation\n",
    "-   Enhances Documentation\n"
   ],
   "metadata": {
    "id": "QIL8pmrB1Uf-"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "19. What are HTTP status codes, and why are they important in a Flask AP
I?\n",
    "-   They are part of the HTTP response.\n",
    "-   Categorized into groups based on their first digit:\n",
    "-   1xx: Informational (rarely used in APIs)\n",
    "-   2xx: Success (e.g., 200 OK, 201 Created)\n",
    "-   3xx: Redirection (e.g., 301 Moved Permanently)\n",
    "-   4xx: Client errors (e.g., 400 Bad Request, 404 Not Found)\n",
    "-   5xx: Server errors (e.g., 500 Internal Server Error)\n",
    "\n",
    "-   Why Are HTTP Status Codes Important in a Flask API?\n",
    "-   Communicate Outcome\n",
    "-   Guide Client Behavior\n",
    "-   Debugging and Error Handling\n",
    "-   RESTful Best Practice"
   ],
   "metadata": {
    "id": "YabAc0Eu1_vP"
   }
  },
  {
   "cell_type": "markdown",
   "source": [
    "20. How do you handle POST requests in Flask?\n",
    "-   Step-by-step: Handling POST Requests in Flask:-\n",
    "-   Import necessary modules\n",
    "-   Create your Flask app\n",
    "-   Define a route that accepts POST requests\n",
    "-   Run your app"
   ],
   "metadata": {
    "id": "-UV3Nr-H2vGP"
   }
  },
  {
   "cell_type": "markdown",
  "source": [
    "21.  How would you secure a Flask API?\n",
    "-    Use HTTPS\n",
    "-    Authentication\n",
    "-    Authorization\n",
    "-    Input Validation & Sanitization\n",
    "-    Rate Limiting\n",
    "-    Error Handling\n",
    "-    CORS (Cross-Origin Resource Sharing)\n",
    "-    Keep Dependencies Updated\n",
    "-    Use Secure Headers"
  ],
  "metadata": {
    "id": "Hi-S4jcT3QOT"
  }
},
{
  "cell_type": "markdown",
  "source": [
    "22.  What is the significance of the Flask-RESTful extension?\n",
    "-    Significance of Flask-RESTful:-\n",
    "-    Simplifies Resource-Based API Design\n",
    "-    Automatic Request Parsing and Validation\n",
    "-    Built-in Response Formatting\n",
    "-    Improved Routing\n",
    "-    Error Handling and Abortion\n",
    "-    Extensible and Compatible"
  ],
  "metadata": {
    "id": "NVfPxcy_3z1d"
  }
},
{
```

"cell_type": "markdown",
"source": [
"23. What is the role of Flask’s session object?\n",
"-   Role of Flask’s session Object\n",
"-   Stores data on a per-user basis between requests (like cookies, but more secure).\n",
"-   Keeps user state without requiring server-side storage (data is stored client-side but signed to prevent tampering).\n",
"-   Useful for things like:\n",
"-   User authentication sessions (e.g., remembering logged-in users).\n",
"-   Flash messages.\n",
"-   Storing user preferences temporarily."
],
"metadata": {
"id": "kQbFSGIg4QnY"
}
},
{
"cell_type": "markdown",
"source": [
"# Practical"
],
"metadata": {
"id": "65Ty-X6F42X6"
}
},
{
"cell_type": "code",
"source": [
"#1.How do you create a basic Flask application?\n",
"\n",
"from flask import Flask\n",
"\n",

```
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/')\n",
    "def hello_world():\n",
    "    return 'Hello World!!'\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug = True)\n",
    "\n",
    "\n",
    "\n"
   ],
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "ioNON8PG47LX",
    "outputId": "3dde39e4-223f-4b4f-a9e4-5a2b0075bf28"
   },
   "execution_count": null,
   "outputs": [
    {
     "metadata": {
      "tags": null
     },
     "name": "stdout",
     "output_type": "stream",
     "text": [
      " * Serving Flask app '__main__'\n",
      " * Debug mode: on\n"
     ]
    },
    {
        "metadata": {
          "tags": null
        },
        "name": "stderr",
        "output_type": "stream",
        "text": [
          "INFO:werkzeug:\u001b[31m\u001b[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.\u001b[0m\n",
          " * Running on http://127.0.0.1:5000\n",
          "INFO:werkzeug:\u001b[33mPress CTRL+C to quit\u001b[0m\n",
          "INFO:werkzeug: * Restarting with stat\n"
        ]
      }
    ]
  },
  {
    "cell_type": "code",
    "source": [
      "#2. How do you serve static files like images or CSS in Flask?\n",
      "\n",
      "import os\n",
      "\n",
      "os.makedirs('static', exist_ok=True)\n",
      "\n",
      "with open('static/style.css', 'w') as f:\n",
      "    f.write('body {background-color: lightblue;}')\n",
      "\n",
      "from flask import Flask, url_for\n",
      "\n",
      "app = Flask(__name__)\n",
      "\n",
      "@app.route('/')\n",
        "def home():\n",
        "    return '''\n",
        "    <html>\n",
        "    <head>\n",
        "        <link rel=\"stylesheet\" href=\"{}\">\n",
        "    </head>\n",
        "    <body>\n",
        "        <h1>Hello, styled Flask!</h1>\n",
        "    </body>\n",
        "    </html>\n",
        "    '''.format(url_for('static', filename='style.css'))\n",
        "\n",
        "if __name__ == '__main__':\n",
        "    app.run()\n",
        "\n"
      ],
      "metadata": {
        "id": "-PrqP3Wk5Ui-"
      },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "#3. How do you define different routes with different HTTP methods in Fla
sk?\n",
        "from flask import Flask, request\n",
        "\n",
        "app = Flask(__name__)\n",
        "\n",
        "@app.route('/')\n",
        "def home():\n",
    "    return \"This is a GET request\"\n",
    "\n",
    "@app.route('/submit', methods=['POST'])\n",
    "def submit():\n",
    "    data = request.form.get('data')\n",
    "    return f\"Received POST data: {data}\"\n",
    "\n",
    "\n",
    "@app.route('/both', methods=['GET', 'POST'])\n",
    "def both_methods():\n",
    "    if request.method == 'POST':\n",
    "        return \"POST request received\"\n",
    "    else:\n",
    "        return \"GET request received\"\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n",
    "\n"
   ],
   "metadata": {
    "id": "vQK1jBwr6-SV"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#4.  How do you render HTML templates in Flask?\n",
    "from flask import Flask, render_template\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/')\n",
    "def home():\n",
    "    return render_template('index.html')\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n",
    "\n",
    "\n",
    "#templates/index.html\n",
    "\n",
    "\n",
    "<!DOCTYPE html>\n",
    "<html>\n",
    "<head>\n",
    "    <title>Flask Template</title>\n",
    "</head>\n",
    "<body>\n",
    "    <h1>Hello from Flask template!</h1>\n",
    "</body>\n",
    "</html>"
   ],
   "metadata": {
    "id": "Fa_yP4vt7ZIT"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#5. How can you generate URLs for routes in Flask using url_for?\n",
    "from flask import Flask, url_for\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/')\n",
    "def home():\n",
    "    return \"Home Page\"\n",
    "\n",
    "@app.route('/user/<username>')\n",
    "def profile(username):\n",
    "    return f\"User: {username}\"\n",
    "\n",
    "@app.route('/links')\n",
    "def links():\n",
    "    # Generate URLs dynamically\n",
    "    home_url = url_for('home')\n",
    "    profile_url = url_for('profile', username='alice')\n",
    "    return f'Home URL: {home_url} <br> Profile URL: {profile_url}'\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n",
    "\n"
   ],
   "metadata": {
    "id": "4e8VKR0Q8Mxw"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#6. How do you handle forms in Flask?\n",
    "from flask import Flask, render_template, request\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/form', methods=['GET', 'POST'])\n",
    "def form():\n",
    "    if request.method == 'POST':\n",
    "        name = request.form['name']\n",
    "        email = request.form['email']\n",
    "        return f\"Received: Name={name}, Email={email}\"\n",
    "    return render_template('form.html')\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n"
   ],
   "metadata": {
    "id": "lr4hpgmw8yBE"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#7.  How can you validate form data in Flask?\n",
    "\n",
    "from flask import Flask, request, render_template\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/form', methods=['GET', 'POST'])\n",
    "def form():\n",
    "    error = None\n",
    "    if request.method == 'POST':\n",
    "        name = request.form.get('name')\n",
    "        email = request.form.get('email')\n",
    "\n",
    "        if not name or not email:\n",
    "            error = \"Both name and email are required!\"\n",
    "        else:\n",
    "            return f\"Form submitted: Name = {name}, Email = {email}\"\n",
    "\n",
    "    return render_template('form.html', error=error)\n"
   ],
   "metadata": {
    "id": "RzstnS2w9Bff"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#8. How do you manage sessions in Flask?\n",
    "from flask import Flask, session, redirect, url_for, request\n",
    "\n",
    "app = Flask(__name__)\n",
    "app.secret_key = 'your_secret_key'  # Required to use sessions securely\n
",
    "\n",
    "@app.route('/')\n",
    "def index():\n",
    "    if 'username' in session:\n",
    "        return f'Logged in as {session[\"username\"]}'\n",
    "    return 'You are not logged in.'\n",
    "\n",
    "@app.route('/login', methods=['GET', 'POST'])\n",
    "def login():\n",
    "    if request.method == 'POST':\n",
    "        session['username'] = request.form['username']\n",
    "        return redirect(url_for('index'))\n",
    "    return '''\n",
    "        <form method=\"post\">\n",
    "            <input type=\"text\" name=\"username\">\n",
    "            <input type=\"submit\" value=\"Login\">\n",
    "        </form>\n",
    "    '''\n",
    "\n",
    "@app.route('/logout')\n",
    "def logout():\n",
    "    session.pop('username', None)\n",
    "    return redirect(url_for('index'))\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n"
  ],
  "metadata": {
    "id": "YQyLsL299X0D"
  },
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "#9. How do you redirect to a different route in Flask?\n",
    "from flask import Flask, redirect, url_for\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/')\n",
    "def home():\n",
    "    return 'Home Page'\n",
    "\n",
    "@app.route('/login')\n",
    "def login():\n",
    "    return 'Login Page'\n",
    "\n",
    "@app.route('/go-to-login')\n",
    "def go_to_login():\n",
    "    return redirect(url_for('login'))  # Redirects to /login\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n"
  ],
  "metadata": {
    "id": "XKFjaamA9mY7"
  },
  "execution_count": null,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
    "#10.  How do you handle errors in Flask (e.g., 404)?\n",
    "from flask import Flask, render_template\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/')\n",
    "def home():\n",
    "    return 'Welcome to the homepage!'\n",
    "\n",
    "# Handle 404 errors (Page Not Found)\n",
    "@app.errorhandler(404)\n",
    "def page_not_found(e):\n",
    "    return render_template('404.html'), 404\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n",
    "<!DOCTYPE html>\n",
    "<html>\n",
    "<head><title>Page Not Found</title></head>\n",
    "<body>\n",
    "    <h1>404 - Page Not Found</h1>\n",
    "    <p>Oops! The page you're looking for doesn't exist.</p>\n",
    "    <a href=\"{{ url_for('home') }}\">Go to Home</a>\n",
    "</body>\n",
    "</html>\n"
   ],
   "metadata": {
    "id": "gSKArSb99ydp"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#11. How do you structure a Flask app using Blueprints?\n",
    "from flask import Blueprint, render_template\n",
    "\n",
    "auth_bp = Blueprint('auth', __name__, url_prefix='/auth')\n",
    "\n",
    "@auth_bp.route('/login')\n",
    "def login():\n",
    "    return render_template('login.html')\n",
    "from flask import Flask\n",
    "from auth.routes import auth_bp  # Import your blueprint\n",
    "\n",
    "app = Flask(__name__)\n",
    "app.secret_key = 'your_secret'\n",
    "\n",
    "# Register blueprint\n",
    "app.register_blueprint(auth_bp)\n",
    "\n",
    "@app.route('/')\n",
    "def home():\n",
    "    return \"Home Page\"\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n",
    "<!DOCTYPE html>\n",
    "<html>\n",
    "<head><title>Login</title></head>\n",
    "<body>\n",
    "    <h1>Login Page (from auth blueprint)</h1>\n",
    "</body>\n",
    "</html>\n"
   ],
   "metadata": {
    "id": "f4XXYDTA-Dg_"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#12. How do you define a custom Jinja filter in Flask?\n",
    "def reverse_string(s):\n",
    "    return s[::-1]\n",
    "from flask import Flask\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.template_filter('reverse')\n",
    "def reverse_string(s):\n",
    "    return s[::-1]\n",
    "app.jinja_env.filters['reverse'] = reverse_string"
  ],
  "metadata": {
    "id": "m_TAWE07-T7v"
  },
  "execution_count": null,
  "outputs": []
 },
 {
  "cell_type": "code",
  "source": [
    "#13. How can you redirect with query parameters in Flask?\n",
    "from flask import Flask, redirect, url_for, request\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/')\n",
    "def home():\n",
    "    return redirect(url_for('greet', name='Alice', lang='en'))\n",
    "\n",
    "@app.route('/greet')\n",
    "def greet():\n",
    "    name = request.args.get('name')\n",
    "    lang = request.args.get('lang')\n",
    "    return f\"Hello {name}, language: {lang}\"\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n"
   ],
   "metadata": {
    "id": "MotEew3x-i0a"
   },
   "execution_count": null,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "#14.  How do you return JSON responses in Flask?\n",
    "from flask import Flask, jsonify\n",
    "\n",
    "app = Flask(__name__)\n",
    "\n",
    "@app.route('/api/data')\n",
    "def get_data():\n",
    "    return jsonify({\n",
    "        'name': 'Alice',\n",
    "        'age': 30,\n",
    "        'status': 'active'\n",
    "    })\n",
    "\n",
    "if __name__ == '__main__':\n",
    "    app.run(debug=True)\n"
   ],
   "metadata": {
    "id": "eVPoFZHB-xO2"
   },
      "execution_count": null,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "#15. How do you capture URL parameters in Flask?\n",
        "from flask import Flask\n",
        "\n",
        "app = Flask(__name__)\n",
        "\n",
        "@app.route('/user/<username>')\n",
        "def show_user(username):\n",
        "    return f'Hello, {username}!'\n"
      ],
      "metadata": {
        "id": "kLjV3AxP-9V3"
      },
      "execution_count": null,
      "outputs": []
    }
  ]
}
```