* Flask is a micro framework that can create web applications. It is not opinionated like some other larger frameworks and does not bind the user to a specific set of tools.
* The main features of Flask include the following.
  + Flask has a web server that runs applications in development mode.
  + Flask also comes with a debugger to help debug applications.
  + The debugger shows interactive traceback and stack trace in the browser.
  + Flask uses standard Python logging for application logs, and you can use the same logger to log custom messages about your application.
  + Flask provides a way to test different parts of your application.
  + The testing feature enables developers to follow a test-driven approach.
  + You can use frameworks like Pi test and coverage to ensure your code works as desired.
  + Finally, developers can access the request and response objects to pull arguments and customize responses.
* Some of the popular community extensions that can be added to your application are the following :
  + Flask-SQLAlchemy add support for ORM called SQLAlchemy to Flask, giving developers a way to work with database objects in Python.
  + Flask-Mail provides the ability to set up an SMTP mail server.
  + Flask-Admin lets you add admin interfaces to Flask applications easily.
  + Flask-Uploads allow you to add customized file uploading to your application.
  + Flask-CORS allows your application to handle Cross-Origin Resource Sharing, making cross-origin JavaScript requests possible.
  + Flask-Migrate adds database migrations to SQLAlchemy ORM.
  + Flask-User adds user authentication,authorization, and other user management activities.
  + Marshmallow adds extensive object serialization and deserialization support to your code.
  + Finally, celery is a powerful task queue that can be used for simple background tasks and complex multi-storage programs and schedules.
* You can create a server by instantiating the Flask class.
* Use the @app decorator to create URL handlers.
* Return string messages or use the jsonify() method to return JSON objects.
* You can set application configuration from environment variables, Python files, app. config object directly.

Error Handling

Every HTTP response contains a three-digit code indicating different error and success status, and the client is responsible for consuming this error code. Valid response code ranges from 100 to 599.

* Informational : Error codes from 100 to 199 indicate the request has been received and are informational.   
  Successful : Error codes from 200 to 299 indicate the request has been received and the requested operation is successful.
* Redirection : Error codes in the 300s indicate the request is proper and there is a redirection on the server.
* Client error : Error codes in the range from 400 to 499 show an error in the request. The request is not proper and cannot be fulfilled
* Sever error : And finally, error codes from 500 to 599 indicate an error on the server side.