**Definition:**

* **WSGI :**  WSGI stands for Web Server Gateway Interface. Simple calling convention for web servers to forward requests to web applications or frameworks written in the Python programming language.

* **ASGI** : ASGI stands for Asynchronous Server Gateway Interface. Intended to provide a standard interface between async-capable Python web servers, frameworks, and applications

**Difference between ASGI and WSGI:**

1. WSGI is a Web Server Gateway Interface while ASGI is a Asynchronous Server Gateway Interface.

2. WSGI works synchronously while ASGI performs the asynchronous functionality.

3. Asynchronous function will await the execution of a promise and it will always return a promise. Synchronous function is a function that does not return until the work is completed or has failed

4. WSGI applications take a single request and return a response at a time. ASGI allows multiple incoming events and outgoing events for each application.

5. WSGI supported frameworks : Django, Flask, Falcon, Pyramid etc.

ASGI supported framework : Django, Falcon, FastAPI etc.

**Example of the WSGI and ASGI:**

**WSGI:**

**-**  WSGI works synchronously, which means requests and responses work

simultaneously.

- As mentioned below, function is written in a simple term because of synchronization.

When the function gets the request it will pass the response as a “Hello World!” and

status of the response 200.

- To run wsgi app we can use gunicorn gunicorn app:app .

**def application(environ, start\_response):**

**body = b'Hello world!\n'**

**status = '200 OK'**

**headers = [('Content-type', 'text/plain')]**

**start\_response(status, headers)**

**return [body]**

**ASGI:**

**-** ASGI is structured as a single, asynchronous callable. It takes a scope, which is a dict

containing details about the specific connection and send an asynchronous callable

that lets the application send event messages to the client, and receive an

asynchronous callable which lets the application receive event messages from the

client.

- Await function is used to wait for the promise.It could be used within the async block

only. It makes the code wait until the promise returns a result. It only makes the async

block wait.

**async def app(scope, receive, send):**

**if scope['type'] == 'lifespan':**

**while True:**

**message = await receive()**

**if message['type'] == 'lifespan.startup':**

**... # Do some startup here!**

**await send({'type': 'lifespan.startup.complete'})**

**elif message['type'] == 'lifespan.shutdown':**

**... # Do some shutdown here!**

**await send({'type': 'lifespan.shutdown.complete'})**

**Return**

**else:**

**pass # Handle other types**