# Hola mundo

from flask import Flask  
  
app= Flask(\_\_name\_\_)  
  
@app.route('/')  
def hello():  
 return "Hola mundo!"

mas correcto

from flask import Flask  
  
app= Flask(\_\_name\_\_)  
  
@app.route('/')  
def hello():  
 return "Hola mundo!"  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run()

o

if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

# comandos

//ejecutar servidor (NOMBRE DEL ARCHIVO py que contiene app sin poner la extencion [main.py])

> flask --app main run

> flask --app NOMBRE\_CON\_APP\_ARCHVIO run

// ejecucion modo debug (muestra los errores y se actualiza solo)

> flask --app main --debug run

# Renderizar html puro

@app.route(‘/html’)  
def hello\_html():  
 return “””  
 <html>  
 <body>  
 <h1>Hello World!</h1>  
 </body>  
 </html>  
 “””

# Múltiples direcciones

@app.route('/uno')  
@app.route('/dos')  
def hello():  
 return "Hola mundo!"

# Parámetros en ruta

@app.route('/parametro/<elparametro>')  
def con\_parametro(elparametro):  
 return f"Hola mundo! {elparametro}"

## Tipados

string , int , float , path , uuid  
@app.route('/parametro/<string:elparametro>')  
def con\_parametro(elparametro):  
 return f"Hola mundo! {elparametro}"

### path

tener cuidado con este tipo de dato porque ejecuta código

[GET] ruta…/parámetro/<script>alert(“hola”)</script>

@app.route('/parametro/<path:elparametro>')  
def con\_parametro(elparametro):  
 return f"Hola mundo! {elparametro}"

Para evitarlo se formatea el parámetro

from markupsafe import escape

@app.route('/parametro/<path:elparametro>')  
def con\_parametro(elparametro):  
 return f"Hola mundo! { escape(elparametro)}"

## multiples

@app.route('/parametro/<string:elparametro>/<int:elparametro2>')  
def con\_parametro(elparametro):  
 return f"Hola mundo! {elparametro} {elparametro2}"

## Opcionales

@app.route('/parametro/<string:elparametro>')

@app.route('/parametro/<string:elparametro>/<int:elparametro2>')  
def con\_parametro(elparametro,elparametro2=None**,**):  
 if elparametro2==None:  
 elparametro2="nada"

return f"Hola mundo! {elparametro} {elparametro2}"

# Renderizar plantillas

## /templates/

Ubicar aquí las plantillas, por defecto es la ruta en la que busca (al mismo nivel del main)

## render\_template

from flask import Flask**,** render\_template

@app.route('/')  
def con\_plantilla():  
 return render\_template("pagina1.html")

### Jinja2

En esencia es exactamente lo mismo que django, por lo tanto no lo voy a por todo

* La herencia de plantillas es igual
* Las condicionales y bucles son iguales
* Los filtros son iguales

#### Variables

La plantilla en Jinja2

<p>{{variable1}}</p>  
<p>{{variable2}}</p>

@app.route('/')  
def con\_plantilla():  
 return render\_template("pagina1.html"**,** variable1="valor1"**,** variable2="valor2")

#### Interpretar expresiones sencillas

<p>{{variable3\*2+4}}</p>

#### Variables en un dict

<p>{{variable1}}</p>  
<p>{{variable2}}</p>

<p>{{variable3\*2+4}}</p>

@app.route('/dict')  
def con\_plantilla\_dict():  
 data={  
 "variable1":"valor1dic"**,** "variable2" : "valor2dic"**,** "variable3" : **5** }  
 return render\_template("pagina1.html"**,** \*\*data)

#### Lista

<p>{{lista[0]}}</p>  
<p>{{lista[1]}}</p>  
<p>{{lista[2]}}</p>

@app.route('/lista')  
def con\_plantilla\_lista():  
 return render\_template("lista.html"**,** lista=["valor1"**,** "valor2"**,** "valor3"])

#### Pasar una variable dict

<p>{{variable\_dict["key"]}}</p>

@app.route('/dict2')  
def con\_plantilla\_dict2():  
 return render\_template("dict.html"**,** variable\_dict={"key":"valor"}  
 )

#### Pasar un objeto

class Persona:  
 def \_\_init\_\_(self**,** nombre**,**edad):  
 self.nombre=nombre  
 self.edad=edad

<p>{{variable\_clase.nombre}}</p>  
<p>{{variable\_clase.edad}}</p>

@app.route('/clase')  
def con\_plantilla\_clase():  
 return render\_template("persona.html"**,** variable\_clase=Persona(  
 nombre="Julio"**,**edad=**2** )  
 )

#### If elif else

{% if variable == 1 %}

<p> era 1 </p>

{% elif variable == 2 %}  
 <p> era 2 </p>  
 {% else %}  
 <p> no lo era</p>  
{% endif %}

#### For

{% for elemento in lista %}  
 <p>{{ elemento }}</p>  
{% endfor %}

##### For dict

{% for key, valor in variable\_dict.items() %}  
 <p>{{ key }}</p>  
 <p>{{ valor }}</p>  
{% endfor %}

#### Filtros personalizados app.add\_template\_filter

<p>{{variable1|filtronombre}}</p>

@app.add\_template\_filter  
def filtronombre(valor):  
 return f"tu nombre es {valor}"

Otra manera de registrar un filtro

<p>{{variable1|filtronombre}}</p>

def filtronombre2(valor):  
 return f"tu nombre es {valor}"  
app.add\_template\_filter(filtronombre2**,**'filtronombreenplantilla')

#### pasar y usar funciones

<p>{{lafuncion("valorarepetir",5)}}</p>

def repeat(s**,**n):  
 return s\*n  
  
@app.route('/confuncion')  
def con\_plantilla\_confuncion():  
 return render\_template("confuncion.html"**,** lafuncion=repeat  
 )

###### funciones globales app.add\_template\_global

no es necesario pasarla a las plantillas porque ya están presentes

<p>{{funcionglobal("valorarepetir",5)}}</p>

@app.add\_template\_global

def funcionglobal(s**,**n):  
 return s\*n  
  
@app.route('/confuncion')  
def con\_plantilla\_confuncion():  
 return render\_template("confuncion.html")

## Archivos estáticos

### /static/

Poner en esta carpeta los archivos

### Plantilla

/static/css/micss.css

… href=”{{ url\_for(‘static’,filename=’ css/micss.css’) }}”

# url\_for

obtener una ruta a la vista de una función, no incluye la base de la url

ruta=url\_for("funcion\_vista")

## tiene parámetros la ruta

ruta=url\_for("funcion\_vista",nombre\_parametro1=”valor”,nombre\_parametro2=”valor2”)

## usar en plantillas

es una función global asi que se puede usar en las plantillas

<a href=”{{ url\_for("funcion\_vista") }} />

# POST

La idea es usar el objeto request global para acceder a los datos de un formulario

<p>{{contenido}}</p>  
<form class="form" method="POST">  
 <label for="entry" class="form-label" >Nombre:</label>  
 <input id="entry" name="contenido" class="form-submit">  
 <button type="submit">Enviar</button>  
</form>

from flask import Flask**,** render\_template**,** request

@app.route('/formulario'**,**methods=["GET"**,**"POST"])  
def con\_plantilla\_formularios():  
 contenido="vacio"  
 if request.method=="POST":  
 contenido=request.form.get("contenido")  
 return render\_template("formularios.html"**,** contenido=contenido  
 )

# mongoDB

instalar la librería

* pip install pymongo[srv]

## conectarse

* 1ro crear la conexión

from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017")

* 2do agregar la conexión a la app  
  app.db=client.prueba
* 3ro realizar consultas  
  usuarios= [usuario for usuario in app.db.usuarios.find({})]
* 4to insertar un valor

usuario={"nombre":contenido}  
app.db.usuarios.insert\_one(usuario)