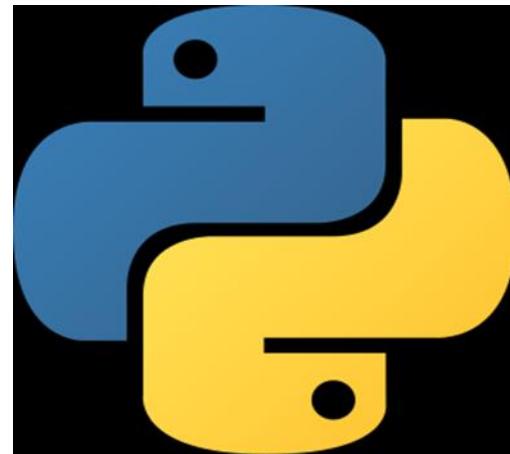


PYTHON

nivel III



Facilitador: Ing. Henry Duque.

Python nivel 3

Contenido

En Python nivel III, se estudiará Flask (Matraz en español) el cual es un framework minimalista escrito en Python, que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. En este nivel, se efectuarán ejercicios para el desarrollo de Aplicaciones en entorno Web a través del framework Python Flask, donde se cubrirán los siguientes aspectos:

- Introducción.
- Templates y Stactic.
- Formularios.
- Sesiones.
- Cookies.
- Integración de Bases de Datos a través de Flask.
- Servicios Web REST con Flask.

Sitio oficial: <https://palletsprojects.com/p/flask/>

Introducción

Desarrollado por Armin Ronacher, quien es un programador de software libre, de origen austriaco y creador del Flask (framework web desarrollado en Python).

Es un orador frecuente en conferencias de desarrolladores y tiene un blog popular sobre desarrollo de software y código abierto. Para mayor información consultar:

https://en.wikipedia.org/wiki/Armin_Ronacher

Preparación de ambiente

```
#  
# CREACIÓN DEL AMBIENTE VIRTUAL  
#
```

(1)

INSTALAR PYTHON 3.7 CON ACCESO A TRAVÉS DEL PATH (NO USAR LA INSTRUCCIÓN DE ABAJO)

```
C:\>set path=%path%;C:\Program Files\Python37;C:\Program Files\Python37\scripts
```

(2) #INSTALAR VIRTUALENV

```
C:\>pip install virtualenv --user
```

(3) #Añadir al path la siguiente ruta así:

```
C:\Users\hduqu\AppData\Roaming\Python\Python37\Scripts  
set path=%path%;C:\Users\hduqu\AppData\Roaming\Python\Python37\Scripts
```

(4) #CREAR EL AMBIENTE VIRTUAL

```
C:\>virtualenv TEST
```

(5) #SE ACTIVA EL ENTORNO VIRTUAL

```
>\test\scripts\activate
```

Preparación de ambiente (continuación)

(5) #SE ACTIVA EL ENTORNO VIRTUAL

>\test\scripts\activate

(6) #SE DESACTIVA EL ENTORNO VIRTUAL

>deactivate

(7) #INSTALACIÓN DE FLASK

(TEST) C:\>pip install flask

Preparación de ambiente (PRUEBA)

(8) GUARDAR EL SIGUIENTE PROGRAMA EN C:\TEST\SCRIPTS

#EJEMPLO01

```
from flask import *
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return 'Hola, alumnos UNEWEB. Flask se ha activado!'
```

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

Preparación de ambiente (PRUEBA)

(9) #EJECUCIÓN:

(TEST) C:\>ejemplo01.py

- * Serving Flask app "EJEMPLO01" (lazy loading)
- * Environment: production

WARNING: This is a development server. Do not use it in a production deployment.

Use a production WSGI server instead.

- * Debug mode: off
- * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Preparación de ambiente (CONSIDERACIONES)

(9) #EJECUCIÓN:

<http://127.0.0.1:5000>

Resultado

Entonces, ¿qué hizo ese código?

- Primero importamos la clase [Flask](#). Una instancia de esta clase será nuestra aplicación WSGI (Web Server Gateway Interface).
- A continuación, creamos una instancia de esta clase. El primer argumento es el nombre del módulo o paquete de la aplicación. Si está usando un solo módulo (como en este ejemplo), debe usarlo `__name__` porque dependiendo de si se inicia como aplicación o si se importa como módulo, el nombre será diferente (en '`__main__`' comparación con el nombre de importación real). Esto es necesario para que Flask sepa dónde buscar plantillas, archivos estáticos, etc.
- Luego usamos el decorador [route\(\)](#) para decirle a Flask qué URL debería activar nuestra función.
- La función recibe un nombre que también se utiliza para generar URL para esa función en particular, y devuelve el mensaje que queremos mostrar en el navegador del usuario.

Acceso a datos de solicitud

#EJEMPLO02

```
from flask import Flask, request #módulo para requerimientos
app = Flask(import_name=__name__)
@app.route("/echo")
def echo():
    to_dato1 = request.args.get("dato1")
    to_dato2 = request.args.get("dato2")
    response = "<u>" + to_dato1 + "</u><br>" + "<i>" + to_dato2 + "</i>"
    return response
```

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

#PARA HACER LA PRUEBA, COLOCAR EN LA URL:

#localhost:5000/echo?dato1=ESTE+MENSAJE+SERÁ+DEVUELTO&dato2=Y+ESTE+TAMBIÉN

Acceso a rutas (1/2)

#EJEMPLO04

```
from flask import *
from flask import Flask
mensaje = '<h1 align="center">FLASK ES UN FRAMEWORK
MINIMALISTA</h1><h2 align="center">PYTHON NIVEL III</h2>'
```

```
app = Flask(__name__)
@app.route('/')
def hola():
    return mensaje
```

Acceso a rutas (2/2)

```
if __name__ == '__main__':
    #app.run()
    app.run(host="127.0.0.1",port=5001)
```

OTRO HOST U OTRO PUERTO
app.run()

Para ejecutar la aplicación en otro servidor o puerto, se debe efectuar la siguiente modificación, en el programa anterior:

```
# app.run(host="127.0.0.1",port=5001)
```

FORMA DE EJECUCIÓN
http://127.0.0.1:5001/

Acceso a rutas (Otro ejemplo 1/2)

#EJEMPLO03

```
from flask import Flask, request # se incorpora módulo  
de solicitud  
  
app = Flask(import_name=__name__)  
  
@app.route("/echo")  
  
def echo():  
    to_echo = request.args.get("echo", "")  
    response = "{}".format(to_echo)  
    return response
```

Acceso a rutas (Otro ejemplo 2/2)

```
#EJEMPLO03 (continuación)

@app.route('/projects/')
def projects():
    return '<font color="red">EL SITIO DE PROYECTOS</font>'

@app.route('/about')
def about():
    return '<font color="blue">EL SITIO ACERCA DE
NOSOTROS</font>'

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

Acceso a rutas (Otro ejemplo Prueba)

#PARA HACER LA PRUEBA, COLOCAR EN LA URL:

#localhost:5000/echo?echo=ECO+ESTE+MENSAJE+SERÁ+DEVUEL
TO

#localhost:5000/projects/

#localhost:5000/about

Acceso a ruta y pase de valores (1/3)

#EJEMPLO05

```
from flask import Flask, request #módulo
import math #módulo
app = Flask(import_name=__name__)
@app.route("/ec2gdo")
def ec2gdo():
    a = eval(request.args.get("a"))
    b = eval(request.args.get("b"))
    c = eval(request.args.get("c"))
    resultado = ""
    subR = 0
    x1 = 0
    x2 = 0
```

Acceso a ruta y pase de valores (2/3)

```
#EJEMPLO05
```

```
if a == 0:  
    resultado = "<b>ERROR:VALOR DE a DEBER DIFERENTE DE CERO</b>"  
else:  
    subR = b*b-4*a*c  
    if subR<0:  
        resultado = "<u>ERROR:EXPRESIÓN SUB RADICAL NO PUEDE SER NEGATIVA</u>"  
    else:  
        x1 = (-b-math.sqrt(subR))/(2*a)  
        x2 = (-b+math.sqrt(subR))/(2*a)  
        resultado = '<h1 align="center">x1='+str(x1)+',<br>x2='+str(x2)+'</h1>'  
  
return resultado  
  
if __name__ == "__main__":  
    app.run()
```

Acceso a ruta y pase de valores (3/3)

#EJEMPLO05

#PARA HACER LA PRUEBA, COLOCAR EN LA URL:

#localhost:5000/ec2gdo?a=0&b=1&c=1 <--- CASO 1

#localhost:5000/ec2gdo?a=1&b=1&c=1 <--- CASO 2

#localhost:5000/ec2gdo?a=1&b=4&c=1 <--- CASO 3

Templates (Plantillas)

Generar HTML desde Python requiere del lenguaje jinja2, a través del cual Flask configura un motor de plantillas automáticamente.

Para ello se requiere de un proceso denominado renderizar, donde una plantilla pasa por el método render_template() de Flask.

Para mayor información:

<https://jinja.palletsprojects.com/en/2.10.x/templates/>

Templates (Ejemplo)

- 1.) Dentro del directorio /Scripts, crear el directorio /templates. La ruta debería quedar así:
..../Scripts/templates

- 2.) Desarrollar el ejemplo06.py. Este programa contendrá el siguiente código:

```
#EJEMPLO06
```

```
from flask import Flask, render_template
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

Templates (Ejemplo)

#EJEMPLO06 (Continuación)

```
def inicio():
```

```
    enlaces = [['uneweb', 'https://uneweb.edu.ve/'],
               ['tutoriales uneweb', 'https://uneweb.edu.ve/tutoriales/'],
               ['flask', 'https://palletsprojects.com/p/flask/']]
```

```
    return render_template('plantilla.html', lista=enlaces)
```

```
app.run()
```

Templates (Ejemplo)

3.) Guardar en el directorio/Scripts/templates, el archivo plantilla.html, el cual tiene el siguiente contenido.

```
<!DOCTYPE html>
<html>
<head>
    <title>FLASK - USO DE PLANTILLAS (TEMPLATES)</title>
</head>
<body>
    <h1>Listado de referencias</h1>
    <ul>
        {% for nombre, enlace in lista %}
        <li><a href='{{ enlace }}'> {{ nombre.capitalize() }}</a></li>
        {% endfor %}
        <ul>
</body>
```

Static (Estático)

Las aplicaciones web dinámicas también necesitan archivos estáticos. Por lo general, de ahí provienen los archivos CSS y JavaScript. Flask puede hacerlo. Simplemente cree una carpeta llamada static en su paquete o al lado de su módulo y estará disponible en /static.

Para generar URL para archivos estáticos, se debe usar el 'static' a través de la función:

```
url_for('static', filename='style.css')
```

El archivo debe almacenarse en el sistema de archivos como static/style.css.

Static (Ejemplo)

1.) Dentro del directorio /Scripts, crear el directorio /static. La ruta debería quedar así:

..../Scripts/static

2.) Crear el archivo style.css, dentro de este nuevo directorio, el cual contendrá el siguiente código:

Static (Ejemplo)

```
h1 {  
font: bold 20px verdana, sans-serif;  
}
```

```
h2 {  
font: bold 14px verdana, sans-serif;  
}
```

```
h3 {  
font: bold 12px verdana, sans-serif;  
}
```

Modificar el archivo plantilla.html, para incorporar el siguiente cambio:

Static (Ejemplo)

```
<!DOCTYPE html>
<html>
<head>
    <title>FLASK - USO DE PLANTILLAS (TEMPLATES)</title>
    <link rel="stylesheet" href="{{ url_for("static", filename="style.css") }}">
</head>
<body>
    <h1>Listado de referencias</h1>
    <h2>Python nivel III</h2>
    <h3>UNEWB</h3>
    <ul>
        {% for nombre, enlace in lista %}
        <li><a href='{{ enlaces }}'> {{ nombre.capitalize() }}</a></li>
        {% endfor %}
        <ul>
</body>
```

Formularios (Ejemplo 1)

Guardar el siguiente archivo en la ruta: /template/plantilla1.html

```
<!DOCTYPE html>
<html>
<head>
    <title>FORMULARIO</title>
</head>
<body>
    <form method="POST">
        <input name="text">
        <input type="submit">
    </form>
</body>
</html>
```

Formularios (Ej1 continuación)

Guardar en .../scripts el archivo #EJEMPLO07.py

```
from flask import Flask, request, render_template
app = Flask(__name__)
@app.route('/')
def formulario():
    return render_template('plantilla1.html')
@app.route('/', methods=['POST'])
def formulario_post():
    text = request.form['text']
    processed_text = text.upper()
    return processed_text
if __name__ == "__main__":
    app.run()
```

Formularios (Ejemplo 2)

Guardar el siguiente archivo en la ruta: /template/plantilla1.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Formulario - Ecuación Resolvente</title>
    <link rel="stylesheet" href="{{ url_for("static", filename="style2.css") }}">
</head>
<body>
    <div id="Encabezado" align="center">
        <h1>Ecuación de 2do. Grado (Resolvente)</h1>
    </div>
    <div id="Captura">
        <form method="POST" action="">
```

Formularios (Ej2 continuación)

```
<table align="center" border="1">  
    <tr>  
        <td>Ingrese a:</td>  
        <td><input type="text" name="va"></td>  
    </tr>  
    <tr>  
        <td>Ingrese b:</td>  
        <td><input type="text" name="vb"></td>  
    </tr>  
    <tr>  
        <td>Ingrese c:</td>  
        <td><input type="text" name="vc"></td>  
    </tr>
```

Formularios (Ej2 continuación)

```
<tr>
    <td colspan="2" align="center">
        <input type="submit"
value="CALCULAR">
        <input type="reset"
value="LIMPIAR">
    </td>
</tr>
</table>
</form>
</div>
```

Formularios (Ej2 continuación)

```
<div id="Resultado">  
    <ul>  
        { % for i in lista %}  
            <li>{{ i }}</li>  
        { % endfor %}  
    </ul>  
</div>  
</body>  
</html>
```

Formularios (Ej2 continuación)

Guardar el archivo style2, en la ruta:../scripts/static

```
*{  
    font: bold 20px verdana, sans-serif;  
}  
  
table{  
    margin-top: 35px;  
    box-shadow: 5px 5px 5px blue;  
    border-radius: 5px;  
}
```

Formularios (Ej2 continuación)

```
#Encabezado{  
    margin-top:    40px;  
    background-color: white;  
    border: 1px solid gray;  
    color: black;  
    height: 50%;  
    padding: 20px;  
    width: 50%;  
    margin-left: 25%;  
    box-shadow: 5px 5px 5px blue;  
    border-radius: 5px;  
}
```

Formularios (Ej2 continuación)

```
#Resultado{  
    margin-top:    40px;  
    background-color: white;  
    border: 1px solid gray;  
    color: black;  
    height: 50%;  
    padding: 20px;  
    width: 50%;  
    margin-left: 25%;  
    box-shadow: 5px 5px 5px red;  
    border-radius: 5px;  
}
```

Formularios (Ej2 continuación)

Guardar en .../scripts el archivo #EJEMPLO08.py

```
from flask import Flask, request, render_template  
import math #módulo  
  
app = Flask(__name__)  
  
@app.route('/')  
def formulario():  
    return render_template('plantilla2.html')
```

Formularios (Ej2 continuación)

```
@app.route('/', methods=['GET','POST'])
def formulario_post():
    if request.method == "POST":
        a = eval(request.form['va'])
        b = eval(request.form['vb'])
        c = eval(request.form['vc'])
        resultado = ["""",0,0]
        subR = 0
        x1 = 0
        x2 = 0
```

Formularios (Ej2 continuación)

```
if a == 0:  
    resultado[0] = "ERROR:VALOR DE a DEBER DIFERENTE  
DE CERO"  
    resultado[1] = 'x1= ERROR'  
    resultado[2] = 'x2= ERROR'  
else:  
    subR = b*b-4*a*c  
    if subR<0:  
        resultado[0] = "ERROR:EXPRESIÓN SUB RADICAL NO  
PUEDE SER NEGATIVA"  
        resultado[1] = 'x1= ERROR'  
        resultado[2] = 'x2= ERROR'
```

Formularios (Ej2 continuación)

```
else:  
    x1 = (-b-math.sqrt(subR))/(2*a)  
    x2 = (-b+math.sqrt(subR))/(2*a)  
    resultado[0] = "RAICES OBTENIDAS:"  
    resultado[1] = 'x1='+str(x1)  
    resultado[2] = 'x2='+str(x2)  
    return render_template('plantilla2.html',  
lista=resultado)
```

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

Sesiones

- Session le permite almacenar información específica para un usuario de una solicitud a la siguiente. Esto se implementa sobre las cookies para usted y firma las cookies criptográficamente. Lo que esto significa es que el usuario podría mirar el contenido de su cookie pero no modificarlo, a menos que conozca la clave secreta utilizada para firmar.
- Para usar sesiones, debe establecer una clave secreta. Así es como funcionan las sesiones:

Sesiones (Ejemplo)

```
from flask import Flask, session, redirect, url_for,  
escape, request
```

```
app = Flask(__name__)
```

```
# Establezca la clave secreta en algunos bytes  
aleatorios.
```

```
# ¡Mantén esto realmente en secreto!
```

```
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'
```

```
#app.secret_key =  
b'\x84\x11g\xfa\xd86^\\xd1\xc3K\x94m"\xd0\x02V'
```

Sesiones (Ejemplo continuación)

```
@app.route('/')
def index():
    if 'username' in session:
        return 'Efectuó Login como: %s' %
escape(session['username'])
    return 'Ud. no ha efectuado Login.'
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        session['username'] = request.form['username']
    return redirect(url_for('index'))
```

Sesiones (Ejemplo continuación)

```
return ""  
    <div align="center">Ingrese su código de  
usuario </div>  
  
    <form method="post">  
        <p><input type=text name=username>  
        <p><input type=submit value>Login>  
    </form>
```

""

Sesiones (Ejemplo continuación)

```
@app.route('/logout')
def logout():
    # Borra el username de la session, si se
    # encuentra activo
    session.pop('username', None)
    return redirect(url_for('index'))

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

Cómo generar buenas claves secretas

Desde Python efectuar los siguientes comandos:

```
>>> import os
```

```
>>> print(os.urandom(16))
```

```
b'\x84\x11g\xfa\xd86^\xd1\xc3K\x94m"\xd0\x02V'
```

Cookies

Las **cookies** son un fragmento de información almacenado por un navegador web. Más específicamente, en el disco duro de la persona que visita la página. Esta información almacenada, puede ser recuperada nuevamente por el **servidor web** en posteriores visitas a la misma página.

Las principales **funciones de las cookies** son:

Recordar acceso, conocer si un usuario ya ha visitado la pagina anteriormente y actuar en consecuencia. Como por ejemplo, ocultar cierto contenido.

Cookies

Veamos como podemos *almacenar cookies con Flask*. Solo debemos utilizar el método `set_cookie`. Veamos un ejemplo:

Cookies (Ejemplo 1)

```
from flask import Flask, make_response  
  
# Creamos la app Flask  
app = Flask(__name__)  
  
@app.route('/')  
  
def index():  
    # Creamos la respuesta  
    resp = make_response("")  
  
    # Almacenamos la cookie  
    resp.set_cookie('username', 'username_1')  
  
    # retornamos la respuesta  
    return resp  
  
if __name__ == '__main__':  
    # iniciamos la aplicación  
    app.run()
```

Cookies (Ejemplo 1)

Ahora veamos el resultado. Nos dirigimos a <http://localhost:5000>. Para poder ver la **cookie almacenada**. Tendremos que ir a **inspeccionador de elementos ** y dirigirnos a la pestaña **Application**. Y entramos en cookies:

Cookies (Ejemplo2)

```
from flask import Flask, request
```

```
# Creamos la app Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

```
def index():
```

```
# Obtenemos la cookie
```

```
    user = request.cookies.get('username')
```

```
    # retornamos la respuesta
```

```
    return "<h1>Bienvenido " + user + "</h1>"
```

```
if __name__ == '__main__':
```

```
# iniciamos la aplicación
```

```
app.run()
```

Cookies (Ejemplo 3)

```
from flask import Flask, request, make_response  
from random import randint
```

```
# Creamos la app Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

```
def index():
```

```
# Leemos la cookie "usuario"
```

```
name_user = request.cookies.get('usuario')
```

Cookies (Ejemplo 3 continuación)

Comprobamos si las cookies existen. De no ser así:

```
if name_user == None:  
    # Creamos la respuesta  
    res = make_response("<h1>Bienvenido </h1>")  
    # Guardamos la cookie "usuario"  
    res.set_cookie('usuario', "usuario" +  
    str(randint(1000, 10000)))  
    # Retornamos la respuesta  
return res
```

Cookies (Ejemplo 3 continuación)

Bases de Datos

- Ejecutar desde su entorno virtual, el siguiente comando a fin de disponer del modulo, para acceso a bases de datos mysql:

```
> pip install mysql-connector-python
```

- Crear desde mysql, una base de datos con el mismo nombre de su entorno virtual (TEST#_BD, donde #, corresponde al número asignado)

Bases de Datos

Crear la tabla USUARIO, siguiendo las siguientes instrucciones:

```
CREATE TABLE USUARIO  
(USERNRO INT AUTO_INCREMENT,  
DOC CHAR(1),  
ID VARCHAR(10),  
NOMBRE VARCHAR(40),  
APELLIDO VARCHAR(40),  
SEXO CHAR(1),  
FECHANAC DATE,  
DIRECCION TEXT,  
CORREO VARCHAR(80),  
TELF VARCHAR(40),  
PRIMARY KEY(USERNRO));
```

Bases de Datos

- Crear en la carpeta \templates el formulario, que permitirá el registro de la información:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>EJEMPLO13 - FLASK - REGISTRO DE
USUARIOS</title>
</head>
<body>
    <header>
        <h1 align="center">REGISTRO DE
USUARIOS</h1>
    </header>
```

Bases de Datos (Formulario)

```
<section>
<form method="post">
<table align="center">
    <tr>
        <td>IDENTIFICACIÓN:</td>
        <td>
            <select name="vDoc">
                <option value="E">E</option>
                <option value="P">P</option>
                <option value="V">V</option>
            </select>
            <input type="text"
                   name="vId"
                   required="">
        </td>
    </tr>
```

Bases de Datos (Formulario)

```
<tr>
    <td>NOMBRE:</td>
    <td>
        <input type="text"
            name="vNom"
            required="">
    </td>
</tr>
<tr>
    <td>APELIDO:</td>
    <td>
        <input type="text"
            name="vApe"
            required="">
    </td>
</tr>
```

Bases de Datos (Formulario)

```
<tr>
    <td>SEXO:</td>
    <td>
        <input type="radio"
            name="vSex"
            value="F"
            checked="">FEMENINO
        <input type="radio"
            name="vSex"
            value="M">MASCULINO
    </td>
</tr>
<tr>
    <td>FECHA DE NACIMIENTO:</td>
    <td>
        <input type="date"
            name="vFecha"
            required="">
    </td>
</tr>
```

Bases de Datos (Formulario)

```
<tr>
    <td>DIRECCIÓN:</td>
    <td>
        <textarea name="vDir"
                  cols="20"
                  rows="5"
                  required=""></textarea>
    </td>
</tr>
<tr>
    <td>CORREO ELECTRÓNICO:</td>
    <td>
        <input type="email"
                  name="vCorreo"
                  required="">
    </td>
</tr>
<tr>
```

Bases de Datos (Formulario)

```
<td>TELÉFONO:</td>
<td>
    <input type="telf"
        name="vTelf"
        required="">
</td>
</tr>
<tr>
    <td align="center" colspan="2">
        <input type="submit" value="GUARDAR">
        <input type="reset" value="LIMPIAR">
    </td>
</tr>
</table>
```

Bases de Datos (Formulario)

```
</form>
```

```
<div align="center" id="msj">
```

```
    {{resp}}
```

```
</div>
```

```
</section>
```

```
</body>
```

```
</html>
```

Bases de Datos (Servidor web Flask)

Desarrollar en el directorio \Scripts, el siguiente programa para el ingreso de datos:

#EJEMPLO13

```
import mysql.connector  
from flask import Flask, request, render_template
```

```
app = Flask(__name__)
```

```
@app.route('/')  
def formulario():  
    return render_template('ejemplo13.html')
```

Bases de Datos (Servidor web Flask)

```
@app.route('/', methods=['GET','POST'])

def formulario_post():
    if request.method == "POST":
        bd = mysql.connector.connect(
            host="localhost",
            user="root",
            passwd="",
            database="TEST_BD")

    flaskCursor = bd.cursor() # prompt de comandos
```

Bases de Datos (Servidor web Flask)

```
sql = "INSERT INTO USUARIO (DOC, ID, NOMBRE, APELLIDO, SEXO, \  
    FECHANAC, DIRECCION, CORREO, TELF) \  
    VALUES \  
    (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
```

```
vDoc = request.form['vDoc']  
vId = request.form['vId']  
vNom = request.form['vNom']  
vApe = request.form['vApe']  
vSex = request.form['vSex']  
vFecha = request.form['vFecha']  
vDir = request.form['vDir']  
vCorreo = request.form['vCorreo']  
vTelf = request.form['vTelf']
```

Bases de Datos (Servidor web Flask)

```
valores = (vDoc,vId,vNom,vApe,vSex,vFecha,vDir,vCorreo,vTelf)
```

```
flaskCursor.execute(sql,valores)
```

```
bd.commit()
```

```
resultado = "USUARIO INGRESADO CON ÉXITO"
```

```
bd.close()
```

```
return render_template('ejemplo13.html', resp=resultado)
```

```
if __name__ == "__main__":
```

```
    app.run()
```

Servicios Web REST

Los 4 principios de REST

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

- Utiliza los métodos HTTP de manera explícita
- No mantiene estado
- Expone URIs con forma de directorios
- Transfiere XML, JavaScript Object Notation (JSON), o ambos

Servicios Web REST

Una vez preparado el entorno de desarrollo desde el idle de python, efectue las siguientes pruebas:

```
>>> import urllib.request  
>>> with  
urllib.request.urlopen("http://localhost/digitalcoin/acredita_servicio.p  
hp?TIPO=X") as url:  
    s = url.read()  
    print(s)
```

Obtendrá el siguiente resultado:

```
b'<ALUMNOS><ALUMNO><IDALUMNO>1</IDALUMNO><NOMBRE>HENRY  
ALBERTO</NOMBRE><APELLIDO>DUQUE  
ANDRADE</APELLIDO></ALUMNO><ALUMNO><IDALUMNO>2</IDALUMNO><NOMB  
RE>AARON DAVID</NOMBRE><APELLIDO>DUQUE  
ROMERO</APELLIDO></ALUMNO><ALUMNO><IDALUMNO>3</IDALUMNO><NOMBR  
E>SOFIA ELENI</NOMBRE><APELLIDO>DUQUE  
ROMERO</APELLIDO></ALUMNO></ALUMNOS>'
```

Servicios Web REST

```
with
```

```
urllib.request.urlopen("http://localhost/digitalcoin/acredita_servicio.php?TIPO=J") as url:
```

```
    s = url.read()
```

```
    print(s)
```

```
b'{"ALUMNOS": [{"IDALUMNO": "1", "NOMBRE": "HENRY  
ALBERTO", "APELLIDO": "DUQUE  
ANDRADE"}, {"IDALUMNO": "2", "NOMBRE": "AARON  
DAVID", "APELLIDO": "DUQUE  
ROMERO"}, {"IDALUMNO": "3", "NOMBRE": "SOFIA ELENI", "APELLIDO": "DUQUE  
ROMERO"}]]'
```

```
>>>
```

FIN