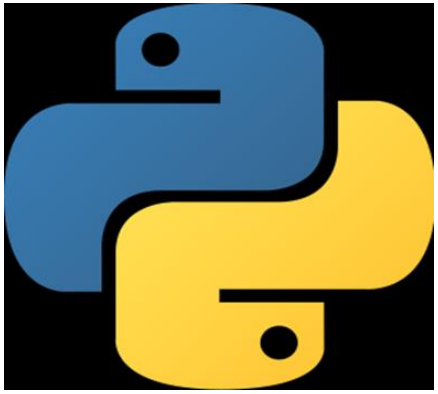
# PYTHON
# nivel III

Flask



# 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://flask.palletsprojects.com/en/2.3.x/

# Preparación de ambiente

#

# CREACIÓN DEL AMBIENTE VIRTUAL

#

(1) #INSTALAR VIRTUALENV

C:\>pip install virtualenv --user

(2) #CREAR EL AMBIENTE VIRTUAL

C:\>virtualenv TEST

(3) #SE ACTIVA EL ENTORNO VIRTUAL

>\test\scripts\activate

# Preparación de ambiente (continuación)

(4) #SE ACTIVA EL ENTORNO VIRTUAL

>\test\scripts\activate

(5) #SE DESACTIVA EL ENTORNO VIRTUAL

>deactivate

(6) #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+DEVUELTO

#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>UNEWEB</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] = 'x1= 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

```
>>> import os
>>> print(os.urandom(16))
b'\x84\x11g\xfa\xd86^\xd1\xc3K\x94m"\xd0\x02V'
```

# Bases de Datos

- Ejecutar desde su entorno virtual:

pip install mysql-connector-python

**Nota Importante:**

- Transcribir y probar los ejemplos del 1 al 16, según el procedimiento, descrito en los vídeos del curso. Se espera que el estudiante envíe estos ejercicios para su revisión, así como soporte de las pruebas efectuadas.

# Referencias documentación

- Python:

https://www.python.org/

- Flask:

https://flask.palletsprojects.com/en/2.3.x/

- Jinja

https://jinja.palletsprojects.com/en/3.1.x/

- SQLAlchemy

https://www.sqlalchemy.org/

- Otros

https://www.w3schools.com/bootstrap/default.asp