

<https://github.com/EryonVF/U4-Practica-2.git>

Paso 1:

Crear un archivo de Python (practica1.py)

Importar del módulo flask la clase Flask (from flask import Flask)

Crear una aplicación flask a partir de una instancia de la clase Flask ( app = Flask(\_\_name\_\_ ), donde el argumento \_\_name\_\_ se refiere al nombre del módulo actual de python

Crear la ruta principal y asociarla a una función si parámetro que imprima el mensaje “bienvenido”

@app.route('/')

Crea la ruta

def welcome():

Crea la función que representa a una vista

return 'Bienvenido'

Cuerpo de la función, debe estar con 1 tabulador a la derecha, envía el mensaje “Bienvenido”

\*\*Recuerde que los nombres de rutas y las funciones de vista deben ser únicos. Sin embargo, varias rutas pueden compartir a la misma vista

El código final debe verse así

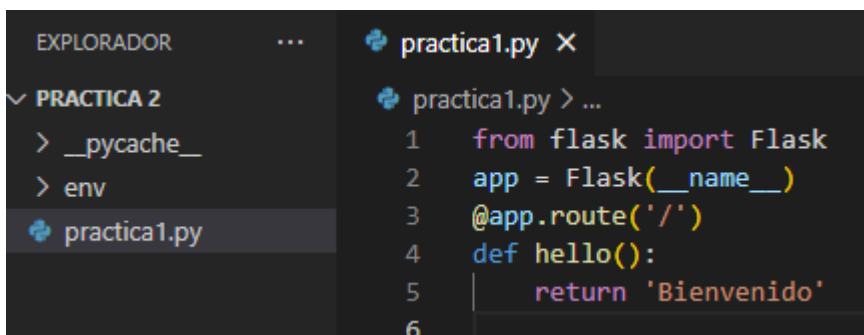
*from flask import Flask*

*app = Flask(\_\_name\_\_)*

*@app.route('/')*

*def hello():*

*return 'Bienvenido'*

A screenshot of a code editor interface. On the left, a file explorer shows a folder named 'PRACTICA 2' containing files like '\_\_pycache\_\_', 'env', and 'practica1.py'. The 'practica1.py' file is selected. On the right, the code editor shows the following Python code:

```
1  from flask import Flask
2  app = Flask(__name__)
3  @app.route('/')
4  def hello():
5      return 'Bienvenido'
6
```

Para ejecutar puede indicar cual es la aplicación y después ejecutar flask

```
set FLASK_APP=practical1
```

```
flask run
```

o bien, hacer todo en una sola línea

```
flask --app=practical1 run
```

Para que se ejecute en modo de prueba (debug) puede escribir

```
flask --app=practical1 --debug run
```

En los tres casos de habilita el URL base `http://127.0.0.1:5000` (servidor local para desarrollo), pero en modo depurador no se requiere detener y volver a ejecutar la aplicación cada vez que se haga un cambio

```
C:\Users\TheOne\Desktop\Codigos\Progra web\Python\Practica 2>py -m venv env
C:\Users\TheOne\Desktop\Codigos\Progra web\Python\Practica 2>env\Scripts\activate
(env) C:\Users\TheOne\Desktop\Codigos\Progra web\Python\Practica 2>pip install Flask
Collecting Flask
  Obtaining dependency information for Flask from https://files.pythonhosted.org/packages/36/42/015c23096649b908c809c69388a805a571a3b
ea44362fe87e33fc3afa01f/flask-3.0.0-py3-none-any.whl.metadata
  Downloading flask-3.0.0-py3-none-any.whl.metadata (3.6 kB)
Collecting Werkzeug>=3.0.0 (from Flask)
  Obtaining dependency information for Werkzeug>=3.0.0 from https://files.pythonhosted.org/packages/c3/fc/254c3e9b5feb89ff5b9076a2321
8dafbc99c96ac5941e900b71206e6313b/werkzeug-3.0.1-py3-none-any.whl.metadata
  Downloading werkzeug-3.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.1.2 (from Flask)
  Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
----- 133.1/133.1 kB 1.3 MB/s eta 0:00:00
Collecting itsdangerous>=2.1.2 (from Flask)
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.1.3 (from Flask)
  Obtaining dependency information for click>=8.1.3 from https://files.pythonhosted.org/packages/00/2e/d53fa4befbf2cfa713304affc7ca78
0ce4fc1fd8710527771b58311a3229/click-8.1.7-py3-none-any.whl.metadata
  Downloading click-8.1.7-py3-none-any.whl.metadata (3.0 kB)
Collecting blinker>=1.6.2 (from Flask)
  Obtaining dependency information for blinker>=1.6.2 from https://files.pythonhosted.org/packages/bf/2b/11bcd7dee4923253a4a21bae3b
e854bcc4f06295bd827756352016d97c/blinker-1.6.3-py3-none-any.whl.metadata
  Downloading blinker-1.6.3-py3-none-any.whl.metadata (1.9 kB)
Collecting colorama (from click>=8.1.3->Flask)
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->Flask)
  Obtaining dependency information for MarkupSafe>=2.0 from https://files.pythonhosted.org/packages/44/44/dbaf65876e258facd65f586dde1
58387ab89963e7f2235551afc9c2e24c2/MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl.metadata
  Downloading MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl.metadata (3.0 kB)
Downloaded flask-3.0.0-py3-none-any.whl (99 kB)
----- 99.7/99.7 kB 6.0 MB/s eta 0:00:00
Downloaded blinker-1.6.3-py3-none-any.whl (13 kB)
Downloaded click-8.1.7-py3-none-any.whl (97 kB)
----- 97.9/97.9 kB 5.5 MB/s eta 0:00:00
Downloaded werkzeug-3.0.1-py3-none-any.whl (226 kB)
----- 226.7/226.7 kB 13.5 MB/s eta 0:00:00
Downloaded MarkupSafe-2.1.3-cp312-cp312-win_amd64.whl (16 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, blinker, Werkzeug, Jinja2, click, Flask
Successfully installed Flask-3.0.0 Jinja2-3.1.2 MarkupSafe-2.1.3 Werkzeug-3.0.1 blinker-1.6.3 click-8.1.7 colorama-0.4.6 itsdangerous
-2.1.2

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(env) C:\Users\TheOne\Desktop\Codigos\Progra web\Python\Practica 2>set FLASK_APP=practical1.py
(env) C:\Users\TheOne\Desktop\Codigos\Progra web\Python\Practica 2>flask run
* Serving Flask app 'practical1.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [27/Oct/2023 17:54:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Oct/2023 17:54:30] "GET /favicon.ico HTTP/1.1" 404 -
```



Bienvenido

## Paso 2: Creando rutas

Modifique la ruta por `@app.route('/welcome')`, ejecute y pruebe la liga `http://127.0.0.1:5000` y después la liga `http://127.0.0.1:5000/welcome`

```
from flask import Flask
app = Flask(__name__)
@app.route('/welcome')
def hello():
    return 'Bienvenido'
```



Bienvenido

## Paso 3: Enviando 1 parámetro

Modifique la ruta por `@app.route('/welcome/<name>')` se está agregando el parámetro name, pero también hay que enviarlo a la función, así que modifica la función hello agregándole el parámetro, debe quedar así: `def hello(name)` adicionalmente hay que manipular el parámetro return `'Bienvenido '+name`, ahora prueba la liga pruebe la liga `http://127.0.0.1:5000/welcome` y después la liga `http://127.0.0.1:5000/welcome/Juan Perez`



## Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.



Bienvenido Eryon

```
from flask import Flask

app = Flask(__name__)

@app.route('/welcome/<name>')
def hello(name):
    return 'Bienvenido ' + name
```

#### Paso 4: Tipos de datos

Definiendo el tipo de dato del argumento. En flask se puede utilizar los siguientes tipos de datos como argumento en las rutas:

<b>int:</b> Captura un valor entero.	@app.route('/usuario/<int:id>')
<b>float:</b> Captura un valor en punto flotante.	@app.route('/precio/<float:valor>')
<b>path:</b> Captura una cadena que puede incluir barras diagonales (útil para rutas).	@app.route('/ruta/<path:segmento>')
<b>string:</b> Este es el tipo de datos predeterminado y captura una cadena de caracteres.	@app.route('/nombre/<string:nombre>')

Modifique la ruta para tener un parámetro tipo entero @app.route('/wellcome <int:ncontrol>'), realice las modificaciones necesarias a la función hello.

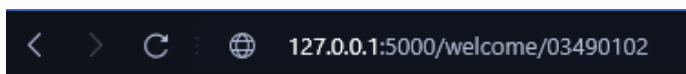
```
from flask import Flask

app = Flask(__name__)

@app.route('/wellcome <int:ncontrol>')
def hello(ncontrol):
    return 'Bienvenido ' + str(ncontrol)
```

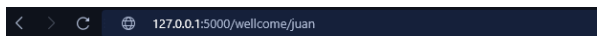
Pruebe las ligas:

<http://127.0.0.1:5000/wellcome/03490102>



Bienvenido 3490102

<http://127.0.0.1:5000/wellcome/juan>



#### Not Found

The requested URL was not found on the server. If you entered the URL manually please check :

### Paso 5: Enviando parámetros con tipo y formato F

Modifique la ruta para que reciba una cadena y un entero `@app.route('/wellcome <int:ncontrol>')`,

Considere que Python permite formatear cadena anteponiendo la letra F, Las cadenas formateadas son una forma conveniente de combinar texto estático con valores variables en una cadena de una manera legible y mantenible. Por ejemplo, si `ncontrol` es un argumento puede formatearse la cadena usando f'El número recibido es: {ncontrol}', note que la variable va entre llaves y solo un conjunto de comillas simples que agrupa todo:

### Paso 6: Recibiendo 2 parámetros

Modifica la ruta para que reciba dos parámetros `@app.route('/wellcome/<name>/<int:ncontrol>')` realiza las modificaciones en la función de vista necesarias para desplegar las dos variables. Ejecute y pruebe con la liga:

[http://127.0.0.1:5000/wellcome/juana maria/0749105](http://127.0.0.1:5000/wellcome/juana%20maria/0749105)

```
from flask import Flask

app = Flask(__name__)

@app.route('/wellcome/<name>/<ncontrol>')
def hello(name,ncontrol):
    return f'Bienvenido ' + name + ncontrol
```



Bienvenido juana maria0749105

### Paso 7: Asociando rutas

Crear varias rutas con el mismo nombre y responder según los parámetros enviados, uniendo los casos anteriores podemos formar las siguientes rutas de la siguiente forma

```
from flask import Flask

app = Flask(__name__)

@app.route('/wellcome/')
@app.route('/wellcome/<name>')
@app.route('/wellcome/<int:ncontrol>')
@app.route('/wellcome/<name>/<int:ncontrol>')
def bienvenido(name=None,ncontrol=None):
    if name== None and ncontrol==None:
        return 'Bienvenido '
    if name!= None and ncontrol == None:
        return f'Bienvenido {name}'
    if name == None and ncontrol != None:
        return f'El número recibido es: {ncontrol}'
    else:
        return f'Bienvenido {name} tu numero de control es: {ncontrol}'
```



Bienvenido juana maria tu numero de control es: 749105

Como se observa hay una sola función de vista llamada `bienvenido` la cual reacciona según los parámetros recibidos. Considere el valor `None` equivalente a `NULL` de java, y que Python permite modificar el tipo de dato de una variable de forma dinámica simplemente asignándole un valor, por ejemplo, verifique el siguiente código

```
a=8
print(a)
print(type(a))
a='mexicali'
print(a)
print(type(a))
a=4.5
print(a)
print(type(a))
a=None
print(a)
print(type(a))
```