

# Desarrollando una pagina web en pocas horas



Juan Vladimir  
@juanvladimir13

# Herramientas



Flask



# Lenguaje python



Python es poderoso ... y rápido  
juega bien con los demás  
corre por todas partes  
es amigable y fácil de aprender  
es opensource

- ✓ Estas son algunas de las razones por las que las personas que usan Python prefieren no usar nada más.
- ✓ Python puede ser fácil de aprender, ya sea que sea un programador por primera vez o tenga experiencia con otros lenguajes.

# Estructura de datos en python3



Una estructura de datos es una forma particular de **organizar datos** para que puedan ser **utilizados de manera eficiente**.

## List

```
colores = ['azul', 'verde', 'blanco']
```

## Tuple

```
coodenada = ( 12356 , 52525 )
```

## Dict

```
usuario = { 'username': '@juanvladimir13', 'anio': 1987, 'peso': 51 }
```

# Funciones en python3



## Funcion

Una función es un bloque de código con un nombre asociado, que recibe cero o más argumentos como entrada, devuelve un valor y/o realiza una tarea.

```
def fib(n):    # write Fibonacci series up to n
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()

# Now call the function we just defined:
fib(2000)
```

# Funciones decoradas



## Decorator

Es una función que retorna otra función, usualmente aplicada como una función de transformación empleando la sintaxis **@envoltorio**

Donde **a** recibe como parámetro **b** para dar como salida a **c**. Esta es una pequeña "formula".

**a(b) -> c**

# Funciones decoradas



```
def funcion_a(funcion_b):  
    def funcion_c(*args, **kwargs):  
        print('Antes de la ejecución de la función a decorar')  
        result = funcion_b(*args, **kwargs)  
        print('Después de la ejecución de la función a decorar')  
  
        return result  
  
    return funcion_c  
  
@funcion_a  
def suma(a, b):  
    return a + b
```

# Flask



Flask es un **micro-framework de aplicación web** WSGI ligero. Está diseñado para que la puesta en marcha sea rápida y sencilla, con la capacidad de escalar a aplicaciones complejas.

```
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

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



# HTML5



HTML5 es la última versión de HTML. El término representa dos conceptos diferentes:

- ✓ Se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos.
- ✓ Contiene un conjunto más amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser más diversas y de gran alcance. A este conjunto se le llama HTML5 y amigos, a menudo reducido a HTML5 .

<https://tools.ietf.org/html/rfc7992>

<https://developer.mozilla.org/es/docs/HTML/HTML5>

---

# Protocolo HTTP



El Protocolo de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol, abreviado HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

HTTP/1.1 (1999/2000)

## REQUEST

GET/POST/PUT/DELETE http://mysitio.com →

X-HEADER: X-value



← **Status Code 200 OK**

X-HEADER: x-value

respuesta en texto plano

← **RESPONSE**

<https://tools.ietf.org/html/rfc2616#section-9.5>

<https://tools.ietf.org/html/rfc2616>

# Metodos de peticion de HTTP



## GET

El método GET solicita una **representación del recurso especificado**. Las solicitudes que usan GET **solo deben recuperar datos** y no deben tener ningún otro efecto.

<https://blogspot.com/?username=juanvladimir13&year=2020>

<https://wordpress.com/juanvladimir13/13/11/2020>

## POST

Envía datos para que sean procesados por el recurso identificado en la URI de la línea petición. Los datos se incluirán en el cuerpo de la petición.

---

## Protocolo sin estado



Un protocolo sin estado (en inglés stateless protocol) es un protocolo de comunicaciones que **trata cada petición como una transacción independiente que no tiene relación con cualquier solicitud anterior**, de modo que la comunicación se compone de pares independientes de solicitud y respuesta.

Ejemplos de protocolos sin estado son:

- ✓ Internet Protocol (IP)
- ✓ Hypertext Transfer Protocol (HTTP)



# Crear un entorno virtual



Un entorno virtual es un entorno Python en el que el intérprete Python, las bibliotecas y los scripts instalados en él están aislados de los instalados en otros entornos virtuales.

## Crear un entorno virtual

```
python3 -m venv /path/to/new/virtual/environment
```

## Activar en entorno virtual

### Linux

```
source /path/to/new/virtual/environment/bin/activate
```

### Windows

```
path\to\new\virtual\environment\Scripts\activate.bat
```

<https://docs.python.org/es/3/library/venv.html>

---

# Importando y exportando un proyecto de python



## Exportando proyecto

- ✓ Ingresar al entorno virtual
- ✓ Guardar modulos de proyecto

```
python3 -m pip freeze > requirements.txt
```

## Importando proyecto

- ✓ Crear un entorno virtual
- ✓ Ingresar al entorno virtual
- ✓ Instalar librerias de proyecto

```
python3 -m pip install -r requirements.txt
```

---

# Contactos



<https://www.facebook.com/juanvladimir13>



[juanvladimir13@gmail.com](mailto:juanvladimir13@gmail.com)



<https://twitter.com/juanvladimir13>



[@juanvladimir13](https://t.me/juanvladimir13)



<https://www.linkedin.com/in/juanvladimir13>



<http://juanvladimir13.wordpress.com>



<https://www.instagram.com/juanvladimir13>



<http://juanvladimir13.blogspot.com/>



[@juanvladimir13](https://wa.me/juanvladimir13)



<https://github.com/juanvladimir13/>



<https://www.youtube.com/c/juanvladimir13>



<https://bitbucket.org/juanvladimir13>

