



CURSO: PYTHON SIN FRONTERAS: HTML, CSS, FLASK Y MYSQL

SECCIÓN: EL MICRO FRAMEWORK FLASK

Flask es uno de los frameworks más populares que podemos usar con Python para hacer aplicaciones Web, podremos hacer nuestras propias API's así como despachar archivos que podrán ser renderizados con el navegador, pero para poder lograr correr nuestras aplicaciones hay un par de cosas que tenemos que hacer si queremos tener éxito:

AMBIENTE VIRTUAL

Lo primero que deberemos tener es un ambiente virtual, ¿esto es necesario?, la verdad es que podrías correr tu aplicación en tu local haciendo las instalaciones, pero si queremos compartir este código y que todos puedan trabajar con este sin problemas, probablemente sea lo más recomendado trabajar con un ambiente virtual.

Esto encapsulará nuestro código e instalaciones, permitiendo que todo funcione desde este ambiente, así sin tener problemas de compatibilidad, posiblemente tengas que encender el ambiente con cada reinicio de tu equipo o de tu editor de texto, pero antes de hacer cualquier instalación o correr el código, asegúrate que tu ambiente virtual esté encendido.

Ambiente virtual en MacOS y Linux

Si sigues los pasos del curso no tendrás problema en utilizar estos comandos para terminales Linux, el ambiente virtual se activa usando el siguiente comando:

```
. venv/bin/activate o source ./venv/bin/activate
```

Ambiente virtual en Windows

Para un ambiente Windows cambia un detalle, usando powershell se activaría con el siguiente comando:

```
source .\venv\Scripts\activate o .\venv\Scripts\activate
```

Mientras que en CMD puedes comenzar a activarlo con el siguiente comando:

```
.\venv\Scripts\activate
```



El cómo identificar tu terminal, lo veremos más adelante en estas notas.

VARIABLES DE ENTORNO

Para que el comando `flask run` funcione, así como los proyectos que veremos más adelante tendremos que definir variables de entorno, cuando guardamos estas variables lo que hacemos es guardar valores en nuestro sistema operativo, al reiniciar nuestro equipo estas variables se borran de la memoria y tendremos que definir las de nuevo.

Pero primero vamos a un tema importante, identifica qué terminal de comandos estás usando, si estás en Linux o en MacOS no deberías tener mayor problema, ya que los comandos son los mismos y no debemos preocuparnos por eso, sin embargo, en Windows esto sí depende y debemos identificar nuestra terminal para saber qué comandos usar

En la lectura de la sección 2 y 3, vimos las diferentes terminales de comandos que pudiste elegir si utilizas Windows, las cuales eran CMD, powershell, git Bash y WSL 2.

Si estás utilizando Git Bash y WSL 2, no tendrás ningún problema, ya que los comandos son los que se utilizan en Linux, mientras que CMD y powershell utilizando los Windows, pero con unas cuantas diferentes entre ambas.

¿Cómo identifiqué mi terminal de comandos con VsCode?

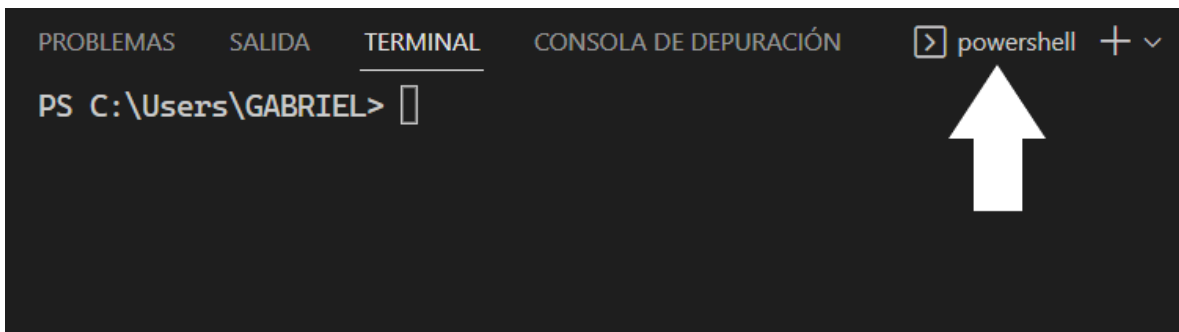
Si estás utilizando VsCode al momento de abrir la terminal integrada, con

`ctrl + ñ` o `ctrl + ``, con este atajo abrirás la terminal de comandos y puedes ver qué tipo de terminal estás usando

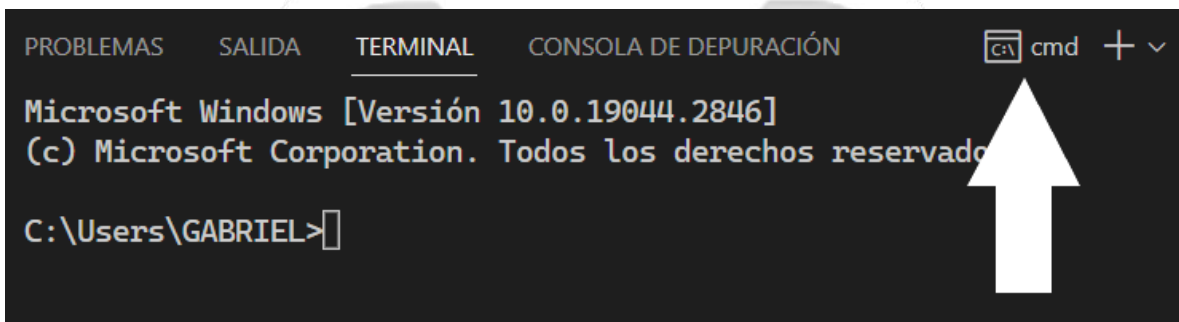


En este caso, se trata de una terminal zsh en un equipo Windows usando WSL 2, esta terminal es una que utiliza comandos Windows, vamos a ver otros ejemplos:

Esta es una terminal Windows usando el intérprete de comandos powershell:



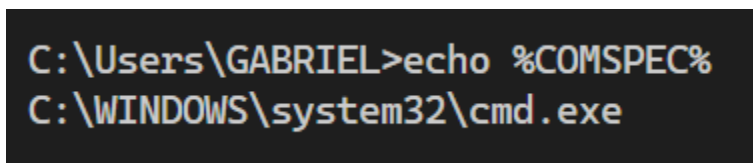
Y esta es una terminal Windows usando el intérprete de comandos CMD



¿Cómo identifico mi terminal de comandos si no uso la terminal de VsCode?

Si estás usando una terminal aparte de la que tiene por defecto el editor VsCode, normalmente habrás abierto la terminal buscando la aplicación CDM, powershell o la terminal por defecto que estás usando por lo que deberías saber con esto y el nombre del software cuál es la terminal, pero si no das con ello te dejaré un par de comandos que podrás usar para identificar tu terminal:

- Sí, tu terminal es CMD podrías usar el comando `echo %COMSPEC%`, con esto la terminal te dará la ruta de la terminal y su archivo ejecutable `cmd.exe`



Sí, tu terminal es powershell con el comando:

```
echo $PSVersionTable.PSEdition
```

con este es que vas a poder confirmar, aquí te saldrán dos opciones como resultado “Desktop” o “Core”, esto te dirá que utilizas powershell.

```
PS C:\Users\GABRIEL> echo $PSVersionTable.PSEdition
Desktop
```

Para terminales que usen comandos Linux es un tanto más fácil, con el comando: `echo $0` te saldrá la ruta de donde se está utilizando el recurso, podría salirte también la opción bash adicional a la zsh.

```
> echo $0
/bin/zsh
```

Como definimos las variables de entorno:

Bien, ya que hemos identificado la terminal de comandos estamos utilizando es momento de ver cómo guardamos las variables de entorno:

- **Terminales que usan comandos Linux**

Para esta terminal tendremos que usar la siguiente sintaxis con el comando:

```
export <nombre de la variable>=<valor>
```

En el caso de la variable flask tendremos el siguiente:

```
export FLASK_APP=todo
```

Si queremos ver que variables tenemos guardadas, vamos a usar el comando `env`, este comando nos dará la lista de variables de entorno que tiene nuestro sistema operativo guardadas, hasta el final de la lista normalmente veremos las que hayamos guardado nosotros con los comandos, pero si quieres ver una variable de entorno específica puedes usar

```
echo ${<nombre de la variable>}
```

Por ejemplo:

```
echo $FLASK_APP
```

```
> echo $FLASK_APP  
todo
```

Si la variable existe veremos su valor, si esta no está guardada simplemente no veremos nada en la terminal después de este comando.

• Terminales Powershell

Para la terminal powershell, podremos guardar las variables de entorno con la siguiente sintaxis `$env:<nombre de la variable> = <valor>` como la siguiente:

```
$env:MI_VARIABLE = "valor"
```

y vamos a ver si se guardó correctamente de esta manera:

```
echo $env:MI_VARIABLE
```

Si la variable fue guardada correctamente, veremos su valor en la consola

```
PS C:\Users\GABRIEL> $env:MI_VARIABLE = "valor"  
PS C:\Users\GABRIEL> echo $env:MI_VARIABLE  
valor
```

• Terminales CMD

Por último, vamos a ver las terminales CMD,

```
set MI_VARIABLE=mi_valor
```

Y se ven de esta manera:

```
echo %MI_VARIABLE%
```



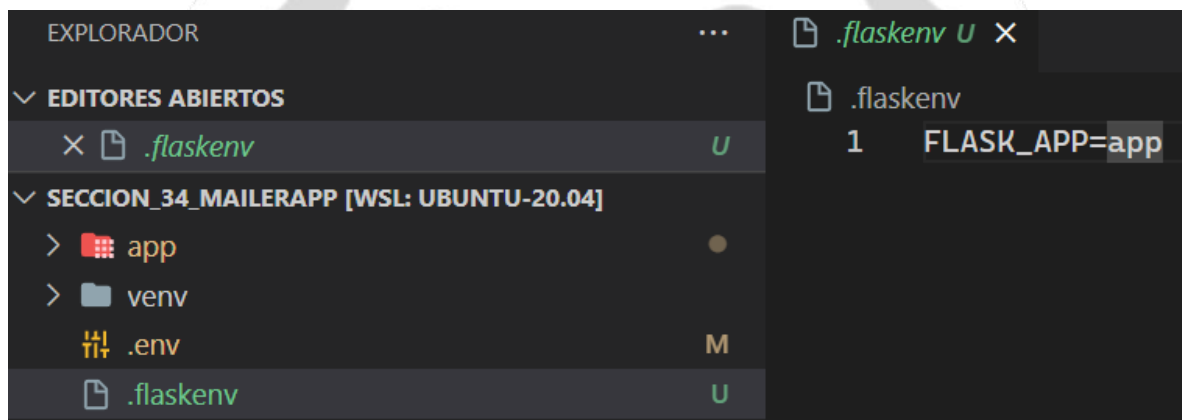
Si la variable fue guardada correctamente, veremos su valor en la consola:

```
C:\Users\GABRIEL>set MI_VARIABLE=mi_valor  
  
C:\Users\GABRIEL>echo %MI_VARIABLE%  
mi_valor
```

Con esto, si tienes las variables de entorno guardadas correctamente tu proyecto deberá funcionar, recuerda que el valor de `FLASK_APP` debe ser igual al nombre de la carpeta que se encuentra el proyecto.

Otra forma de definir las variables es usando un módulo externo, pero eso lo puedes aprender en la clase definiendo “**Variables de entorno**” de la sección “**Proyecto:Mailer**”, si quieres, puedes adelantarte y ver el procedimiento de este módulo, es a tu decisión.

Pero igualmente si no quieres definir `FLASK_APP` en cada reinicio, algo que ha funcionado es usar un archivo llamado `.flaskenv` en la raíz del proyecto, el cual nos permite guardar la variable de entorno, entonces el archivo tendría el siguiente contenido:



The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing the project structure under 'SECCION_34_MAILERAPP [WSL: UBUNTU-20.04]'. It includes folders 'app' and 'venv', and files '.env' and '.flaskenv'. The '.flaskenv' file is selected. On the right, the Editor shows the content of the '.flaskenv' file, which is a single line: '1 FLASK_APP=app'.

Mi carpeta se llama `app`, por lo que el valor de la variable de entorno será `app`.