



**¡Les damos la
bienvenida!**

¿Comenzamos?

Esta clase va a ser

- grabada
a

Semana 9. PYTHON

Git, GitHub y Django

Objetivos de la clase

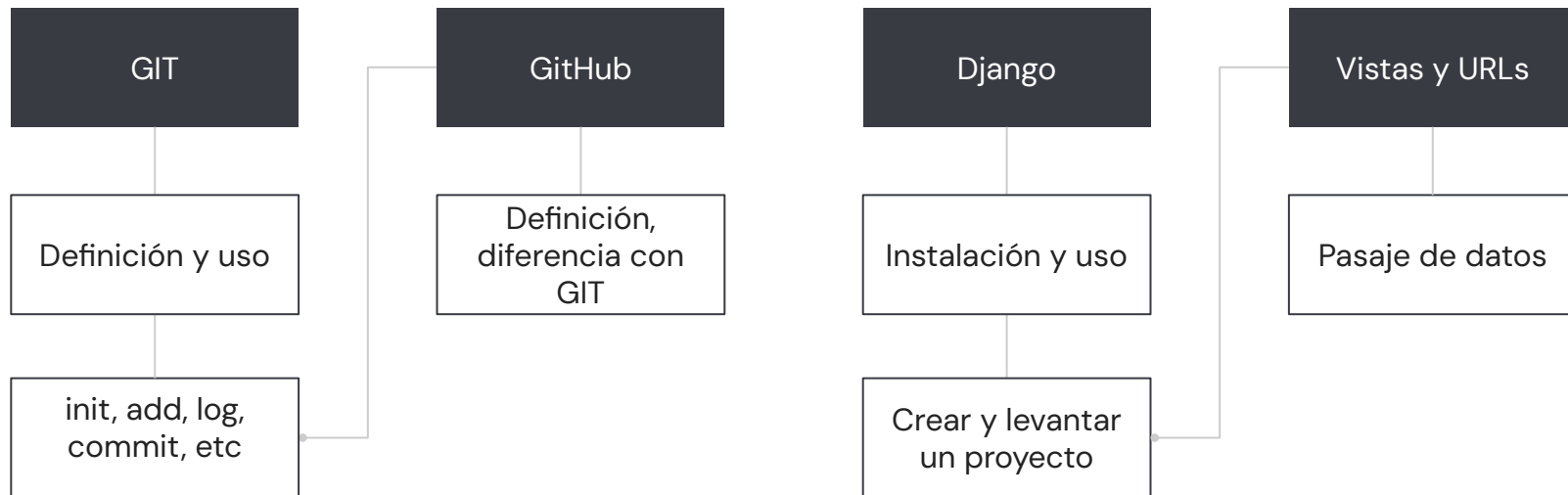
- **Crear** un proyecto y versiones con Git.
- **Utilizar** el repositorio GitHub.
- **Utilizar** MVC creando las primeras vistas.
- **Aplicar** el uso de templates



¡Recuerda esto!

Antes de iniciar esta clase,
debes abrir VSC y la página de
GitHub.

MAPA DE CONCEPTOS



Repasemos...



REPASO

Semana 9

En los videos de esta semana aprendiste sobre:

- ✓ GIT
- ✓ init, add, commit, log, status, push, etc
- ✓ GitHub, como crear un repositorio en la nube y conectarlo al local
- ✓ Django, creación de proyecto, primera prueba
- ✓ Creación de vistas y pasaje de datos

VIDEO N°9.1 – GIT

Recuperamos el tema visto

GIT es un **sistema de control de versiones** que nos permite ir administrando los cambios que vamos realizando en nuestros proyectos.

Se maneja en parte pasando por 3 estados: el área de trabajo (apenas guardamos los cambios de un archivo), el staging area (área intermedia de empaquetamiento o acumulación de cambios correctos), y el área de repositorio, que sería el paso final para generar una versión nueva.

Para utilizar Git tenemos que utilizar comandos como: init, add, push, commit, log, remote, etc.





VIDEO N° 9.2 – GITHUB

Recuperamos el tema visto

GitHub es una red social en la cual se manifiestan casi en su totalidad desarrolladores, ya que nos permite tener guardados nuestros proyectos en la nube.

Con simples pasos se puede tener una cuenta y subir un proyecto utilizando como intermediario GIT.

Existen distintas formas de conectar nuestro repositorio de GIT local con nuestro repositorio en la nube, pero GitHub lo hace más simple y nos guía para realizarlo una vez que creamos un repositorio nuevo en la plataforma.





VIDEO N°9.3 - DJANGO

Recuperamos el tema visto

Django es un **framework de desarrollo web**. Utiliza un patrón MTV similar al MVC.

Para crear un proyecto debemos instalar Django y luego ejecutar `django-admin startproject`.

Lo que genera este comando de primeras es un archivo `manage`, que es el encargado de casi todo lo que queramos solicitarle al proyecto, y una carpeta con los archivos iniciales necesarios para iniciar con configuraciones básicas.

A partir de ahora, nuestros pasos más básicos serán crear una vista, un path que referencie a esta y un template que lo veremos en la clase de hoy.



CODERHOUSE

Contenido pregrabado

Si en algún momento quieres buscar un fragmento de contenido pregrabado y no recuerdas dónde hacerlo, puedes consultar [este resumen](#). Luego, podrás ir directo a buscar el video o podcast que necesites.

¿Preguntas?

Te invitamos a dejar tu
pregunta a través de/del
[chat](#)



Puesta en común microdesafío

¡Vamos a recuperar lo trabajado durante la semana!

Duración: **10 minutos.**



PUESTA EN COMÚN – MICRODESAFÍO

GitHub

Crea el primer repositorio en GitHub con el material generado en los ejercicios “Tablas”, “Otra opción” y “Agenda de contactos” de la semana pasada.

¡Buen trabajo! 🧐

Paso a paso: Repaso

- 1: git init → inicializamos el repo (única vez)
- 2: git add → pasamos a stage los cambios
- 3: git commit → confirmamos los cambios
- 4: git log → revisamos el historial de commits
- 5: git push → actualizamos el repo de GitHub con los commits nuevos

¿Preguntas?

Te invitamos a dejar tu
pregunta a través de/del
[chat](#)



Para pensar

Con lo visto en el contenido pregrabado
¿Cuál es la diferencia principal entre Git y GitHub?

Contesta mediante el chat de Zoom

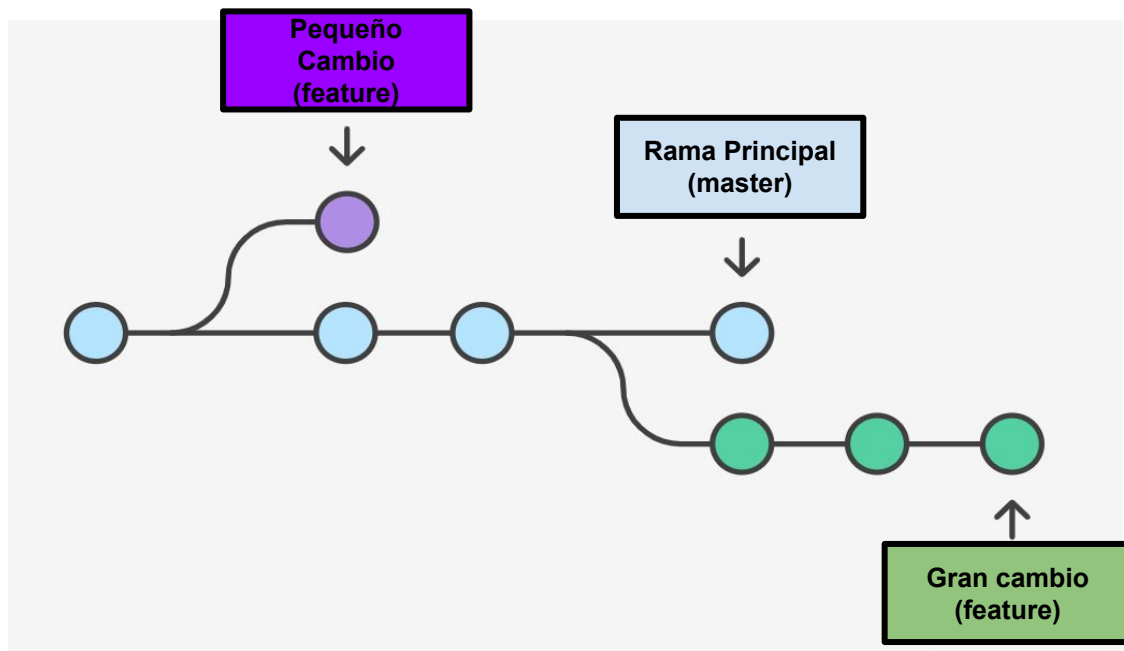
Ramas

Ramas

Para añadir una nueva función o solucionar un error (sin importar su tamaño), generas una nueva rama para alojar estos cambios. Esto te da la oportunidad de organizarte mejor con los cambios o correcciones experimentales.

👉 Podemos crear una rama escribiendo
`"git branch mi-rama"`

Ramas



Así funciona

Git Branch: creando ramas

Veamos cómo crear una rama.

```
/* Paso 1: Verifico en cuál rama estoy */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch  
*master  
/* Paso 2. Creo la rama que voy a usar para el cambio */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch mi_rama  
/* Paso 3: Verifico que se creó la rama */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch -l  
*master  
mi_rama  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

Git Branch: movernos entre ramas

¿Será muy complicado hacerlo?

```
/* Para moverme a la rama que cree uso el comando de git checkout */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git checkout mi_rama  
Switched to branch 'mi_rama'  
/* Verifico nuevamente que me movi de rama */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch -l  
master  
*mi_rama  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```


Git Branch D: borrando ramas

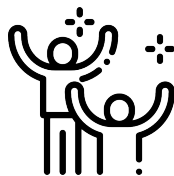
Penúltimo paso 😊

```
/* Paso 1: Me muevo a la rama principal "master" */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git checkout master  
/* Paso 2: Verificar que se está en la rama de master */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch  
*master  
mi_rama  
/* Paso 3: Procedo a borrar la rama que ya no voy a usar */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch -D mi_rama  
Deleted branch mi_rama (was 6d6c28c)  
/* Paso 4: Verificar que se borró la rama */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch  
*master
```

Git checkouts: listar commits

Así como nos movemos entre ramas, nos podemos mover entre commits. Recuerden que al hacer cambios, adherirlos y comitearlos, se crea un historial de dichos cambios, los logs.

La posibilidad de volver a un commit en específico es una ventaja de los controladores de versiones, que permiten volver a un estado anterior si se presenta un problema, error o cambio inesperado.



Git checkouts listar commits

Comenzamos listando.

```
/* Para ver los commits realizados, los listamos con el comando git log --oneline para verlos
en una sola línea*/
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git log --oneline
/* Se listan todos los cambios que se han realizado sobre el index.html */
fc59b88 (HEAD -> nueva_rama) Ahora agregamos un título
6bcff19 Agregar un texto al index.html
41e6121 (master) Primer archivo del repositorio
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

Git checkout: mover a un commit

/* Supongamos que me equivoqué en agregar el título, quiero volver al punto anterior del texto, busco el número de commit y muevo hacia ese punto */

```
john@MyShopSolutions: ~/Documents/Proyectos_Coder/mi_repositorio$ git checkout 6bcff19
```

Note: checking out 6bcff19.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

```
git checkout -b <new-branch-name>
```

HEAD is now at 6bcff19... Agregar un texto al index.html

Git checkout: movernos a un commit

```
/* Si verifico donde estoy parado co git branch se puede observar que se está en el  
commit y el index.html ha cambiado*/
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch
```

```
* (HEAD detached at 6bcff19)
```

```
master
```

```
nueva_rama
```

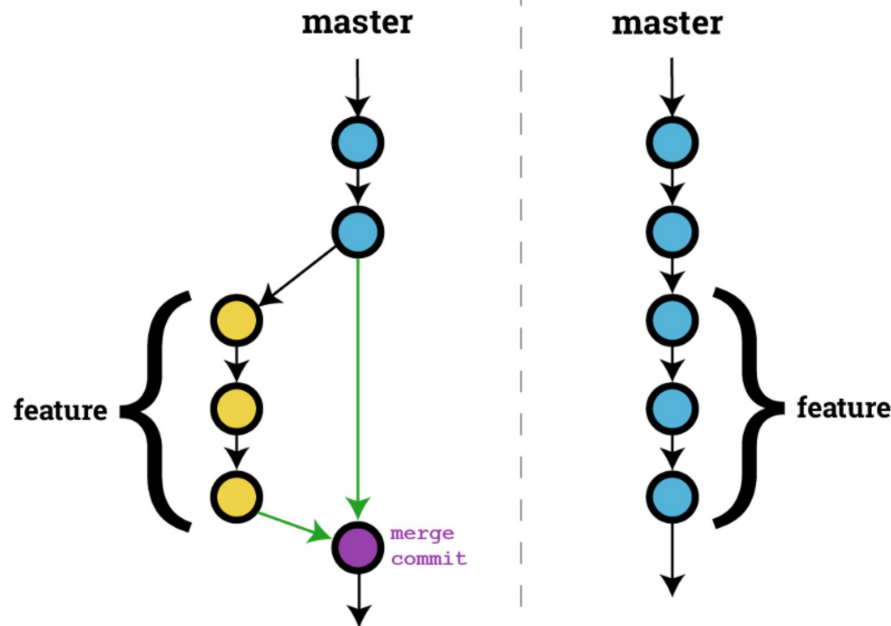
```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

Git: fusionar (merge)

Una vez que tenemos una rama (o más), podemos experimentar características nuevas.

Para luego **FUSIONARLAS** con la rama **MASTER**.

A continuación veamos cómo hacerlo...



Git Merge

/* Paso 1: Ubicarse en la rama master, que es a donde quiero fusionar los cambios usando el comando de git checkout master. */

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git checkout master
```

/* Paso 2: Verificar que estoy en master con git branch. Se puede observar en el archivo de index.html que no tiene ni título ni texto. */

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch  
*master
```

Nueva_rama

/* Paso 3: Realizar la fusión. Hacer el merge con el comando **git merge nueva_rama***/

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git merge nueva_rama
```

Updating 41e6121..fc59b88

Fast-forward

index.html | 2 ++

1 file changed, 2 insertions(+)



Ejemplo en vivo

Vamos a crear una rama para listar commits en GIT.

Listar commits

1

Crear una rama con git
branch **nueva_rama**

2

Cambiar de rama con git
checkout **nueva_rama**

3

Verificar que cambie de
rama con git branch -l

Paso a paso

4

Agregar al index.html un
texto nuevo

5

Verificar que hubo un
cambio en el index.html
con git status

6

Adherir el cambio con
git add

Listar commits

7

Comitear el cambio con
`git commit -m`
"Agregando texto al HTML"

8

Agregar un título al
`index.html` y repetir los
pasos para poder
comitear el cambio.



Repaso

- ✓ **Git Init:** indicarle que en ese directorio, donde ejecutamos este comando, será usado con GIT.
- ✓ **Git Add:** Agregar todos los archivos creados, modificados, eliminados al estado 2 (stage).
- ✓ **Git Commit - m "mensaje":** mensaje obligatorio para mostrar que hemos cambiado, por ejemplo al estado 3.
- ✓ **Git log -- online:** para conocer los códigos de los commits realizados.



Repaso

- ✓ **Git checkout rama:** para cambiar de rama e ir a un commit específico (debemos conocer su código anteriormente).
- ✓ **Git merge rama:** Debemos estar en un MASTER para funcionar.
- ✓ **Git branch rama:** creación de una rama (si queremos eliminar una rama ponemos `git branch -D nombre-rama`).



Break

¡En 10 minutos volvemos!

Plantillas Django

¿Qué son?

Las plantillas son archivos que nos permiten separar la vista de la estética, es decir guardar en un archivo separado de todo lo que guardábamos en “documento”, para así enviar por la HttpResponse.

Entonces, ¿podríamos decir que así se crea un template? 😊

¡Exacto! 🙌

Recordemos que Django se basaba en Modelo, Vista, Template.



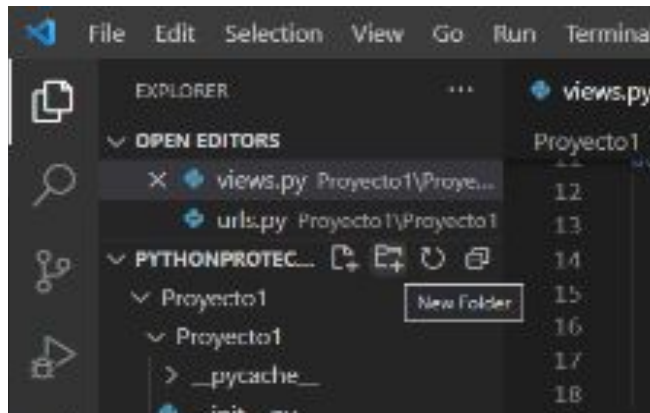
Ejemplo en vivo: Creando nuestro primer template

Vamos a crear nuestro primer template. Esto necesitará de un “template”, propiamente dicho, es decir, lo que se muestra. Además, necesitaremos un “contexto”, esto es, para manejar contenido que cambia, por ejemplo nuestro nombre en el ejercicio anterior. Y por último crear un “render” es decir, una transformación a página web.

Duración: **15 minutos**

Paso a paso

1 Dentro del proyecto creamos una carpeta que se puede llamar plantillas.



2 Dentro de esa nueva carpeta creamos un archivo `template1.html`.

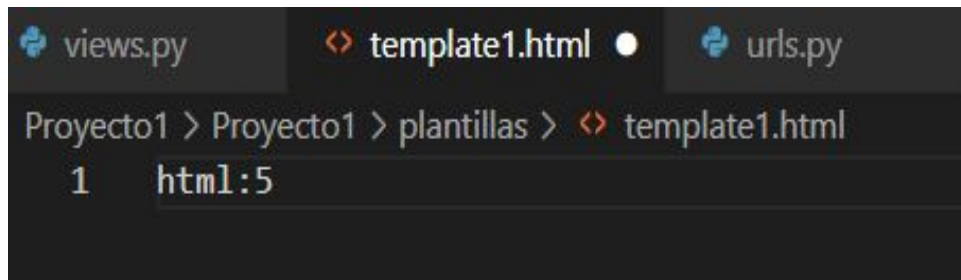


Paso a paso

3

Dentro de `template1`, escribimos `html:5` y damos clic en "enter".

Se crea automáticamente un esqueleto.



The screenshot shows a code editor with three tabs at the top: `views.py`, `template1.html` (which is the active tab), and `urls.py`. The breadcrumb navigation at the top of the editor area reads: `Proyecto1 > Proyecto1 > plantillas > template1.html`. The main editing area shows a single line of code: `1 html:5`, where the line number '1' is on the left and the code 'html:5' is on the right.

Paso a paso

4

Dentro de `<body>` `</body>`, escribimos lo que queremos que se vea en nuestra página web.

```
views.py x template1.html urls.py
Proyecto1 > Proyecto1 > plantillas > template1.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10
11 </body>
12 </html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

  Excelente!!!!!!.... Es muy fácil usar plantillas.

</body>
</html>
```

Paso a paso

5

5a. Llamamos a `template1` desde una nueva vista (`probandoTemplate`). Aquí crearemos el contexto y el render.

```
def probandoTemplate(self):  
  
    miHtml = open("C:/Users/nico_/Desktop/PythonProtecto1/Proyecto1/Proyecto1/plantillas/template1.html")  
  
    plantilla = Template(miHtml.read()) #Se carga en memoria nuestro documento, template1  
    ##OJO importar template y contex, con: from django.template import Template, Context  
  
    miHtml.close() #Cerramos el archivo  
  
    miContexto = Context() #EN este caso no hay nada ya que no hay parametros, IGUAL hay que crearlo  
  
    documento = plantilla.render(miContexto) #Aca renderizamos la plantilla en documento  
  
    return HttpResponse(documento)
```

Paso a paso

6. Agregamos la ruta a la nueva vista.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('saludo/', saludo),      #ojo la url no hace
    path('segundavista/', segunda_vista),
    path('diaDeHoy/', diaDeHoy),
    path('miNombreEs/<nombre>', miNombreEs),
    path('probandoTemplate/', probandoTemplate),
]
```



Excelente !!!!!!! Es muy fácil usar plantillas.

¿Qué hemos logrado?



Puesta en común

Logramos separar lo que construye a la página web, es decir el **HTML**, del procesamiento de los datos dado por la vista.

De esta manera podemos tener un diseñador trabajando en el HTML y nosotros desarrollando en **Python/Django** sin saber nada de HTML.

Entornos virtuales y paquetes en Python

Entornos virtuales

Los entornos virtuales se pueden describir como directorios de instalación aislados. Este aislamiento te permite localizar la instalación de las dependencias de tu proyecto, sin obligarte a instalarlas en todo el sistema.

Muchas veces se recomienda a la hora de programar, usar este tipo de herramientas. Es importante mencionar que los entornos virtuales, pueden crearse en diferentes lenguajes de programación, en este caso en particular, veremos como crearlos en Python.

Entornos virtuales

Para crear un entorno virtual, debemos decidir en qué carpeta lo queremos crear y ejecutar el módulo venv como script con la ruta a la carpeta. Por ejemplo, creamos una carpeta en el Escritorio y allí lo ejecutamos.

PROBLEMAS

7

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

JUPYTER

```
PS C:\Users\layla> cd Escritorio
PS C:\Users\layla\Escritorio> mkdir ejemplo_entorno_virtual_python
```

Entornos virtuales

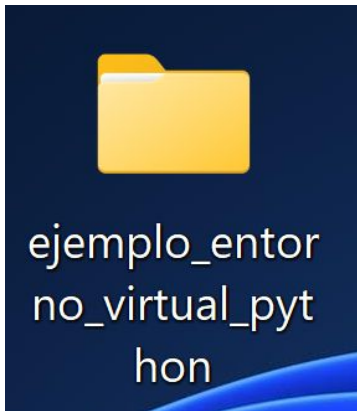
Mode	LastWriteTime	Length	Name
d----	7/8/2022 16:31		ejemplo_entorno_virtual_python

Ingresamos a la carpeta y ejecutamos el siguiente comando: `python -m venv tutorial-env`

- PS C:\Users\layla\Escritorio> `cd ejemplo_entorno_virtual_python`
- PS C:\Users\layla\Escritorio\ejemplo_entorno_virtual_python> `python -m venv tutorial-env`

Entornos virtuales

Chequeamos el Escritorio y la creación de las carpetas:



📁	Nombre	Fecha de modifica...	Tipo	Tamaño
📁	tutorial-env	7/8/2022 16:36	Carpeta de arc...	

El comando anterior creó el directorio tutorial-env si no existe, y también directorios dentro de él que contienen una copia del intérprete de Python y varios archivos de soporte.

Entornos virtuales

Ahora solo nos queda activarlo,. Para ello, estando dentro de la terminal, en el mismo lugar donde se encuentra tutorial-env, ejecutar el comando:

En windows >> source <nombre del directorio del entorno virtual>/Scripts/activate

En linux o mac>> source <nombre del directorio del entorno virtual>/bin/activate

Luego, para desactivarlo solo es necesario escribir "desactivate"

¿Preguntas?

Te invitamos a dejar tu
pregunta a través de/del
[chat](#)



MATERIAL AMPLIADO

- ✓ [Git Cheat Sheet](#) | **GitHub Education**
- ✓ [Tutoriales de uso GitHub](#) | **GitHub**
- ✓ [Cómo usar la integración Git en Visual Studio Code](#) | **Digital Ocean**
- ✓ [Tutorial Django básico \(Python\)](#) | **Developer**
- ✓ [Creación de entornos virtuales](#) | **Python Org**
- ✓ [Entornos virtuales y paquetes](#) | **Python Org**

¿Preguntas?

Resumen de la clase hoy

- ✓ Git: instalación, configuración, repositorio y ramas.
- ✓ GitHub: definición, creación de repositorio, suba de proyecto.
- ✓ Crear un primer proyecto en Django.
- ✓ Crear primeras vistas
- ✓ Relacionar un template con una vista.
- ✓ Entornos virtuales en Python

La próxima semana

Los próximos temas que vamos a ver



Contenido Pregrabado

- ✓ Video 10.1 –Mejoras en las plantillas
- ✓ Video 10.2 –Modelo
- ✓ Video 10.3 – Profundizando MVT
- ✓ Microdesafío – Crea tu modelo



Clase en vivo (2h)

- ✓ Actividad individual: Agregando cargadores
- ✓ Ejemplo en vivo: Carga en BD
- ✓ Actividad individual: Mi Django a Github
- ✓ Vistas y URLs(URLs avanzadas)
- ✓ Ejemplo en vivo: Generar URL.PY
- ✓ Ejemplo en vivo: Template

La **próxima** semana

Recuerda que, a partir de ahora, tienes disponible el contenido pregrabado en la plataforma y que **es necesario que lo veas en forma previa a la próxima clase.**

Opina y valora
esta clase

Muchas gracias.

#DemocratizandoLaEducación

**¡Gracias por estudiar
con nosotros! ✨**