

Building dashboards with Dash

Practical examples with Python

First Edition

Building dashboards with Dash

Practical examples with Python

First Edition

Julio César Martínez
City, Mexico

booksvg.pdf

Secretaría Ejecutiva del Sistema Nacional Anticorrupción (SESNA)
Ciudad de México, México

This book was typeset using L^AT_EX softwareace*.

Preface

Table of Contents

1	Introducción	1
2	Introducción a Python	3
3	Estructura de datos	5
4	Programación con Python	7
5	Introducción a programación orientada a objetos con Python	9
6	Manejo de datos con pandas	11
7	Visualización con Matplotlib	13
8	Visualización con Plotly	15
9	Introducción a Dash	17
10	Mapas interactivos con dash_leaflet	19
11	Creación de ambientes virtuales	21
11.1	Ambiente virtual	21
12	Control de versiones con Git, y GitHub	25
12.1	Comandos útiles	29
13	Automatización con Docker	31
14	Deploy Dash app en servidor Linux	33

Chapter 1

Introducción

Chapter 2

Introducción a Python

Chapter 3

Estructura de datos

Chapter 4

Programación con Python

Chapter 5

Introducción a programación orientada a objetos con Python

Chapter 6

Manejo de datos con pandas

Chapter 7

Visualización con Matplotlib

Chapter 8

Visualización con Plotly

Chapter 9

Introducción a Dash

Chapter 10

Mapas interactivos con dash_leaflet

Chapter 11

Creación de ambientes virtuales

11.1 Ambiente virtual

Las aplicaciones en Python usualmente hacen uso de paquetes y módulos que no forman parte de la librería estándar. Las aplicaciones a veces necesitan una versión específica de una librería, sin embargo, a veces sucede que se requieren de varias versiones lo cual significa que no es posible una instalación de Python que permita cumplir los requerimientos de todas las aplicaciones. Si la aplicación A necesita la versión 2.0 de un módulo particular y la aplicación B necesita la versión 3.6, entonces los requerimientos entran en conflicto e instalar la versión 2.0 o 3.6 dejará una de las aplicaciones sin funcionar. Ello es un problema muy común, el cual puede solucionarse mediante la creación de un ambiente virtual.

Un ambiente virtual es un directorio que contiene una instalación de Python de una versión específica, además de paquetes adicionales. Por ejemplo, para solucionar el problema anterior, se requeriría de un ambiente virtual para la aplicación A con la versión 2.0 de Python, mientras que para la aplicación B se requerirá la versión 3.6, y, lo interesante es que el funcionamiento de uno no afectará el otro.

En las siguientes líneas vamos a describir la forma de crear un ambiente virtual en python:

- Instalar la versión requerida de [Python](#)
- Crear una carpeta
- Abrir el cmd
- Navegar hasta el directorio donde se creará el ambiente virtual, con `cd`

```
Microsoft Windows [Versión 10.0.19045.4046]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jcmartinez>cd Desktop
C:\Users\jcmartinez\Desktop>cd PSociales
C:\Users\jcmartinez\Desktop\PSociales>
```

Figure 11.1: Accediendo a la carpeta

- Escribimos en la terminal `C:\Python312\python.exe -m venv python312`. la primera sentencia hace referencia a la ruta del ejecutable de la versión de Python, es decir, la versión 3.12, las siguientes expresiones hacen referencia al módulo ambiente virtual `-m venv`, y finalmente la última expresión hace referencia al nombre del ambiente virtual `python312`. Para conocer la ruta de la versión de python requerida, basta con escribir en la terminal `python --version`, y para saber la versión actual escribimos `python --version`.

```
Microsoft Windows [Versión 10.0.19045.4046]
(c) Microsoft Corporation. Todos los derechos reservados.

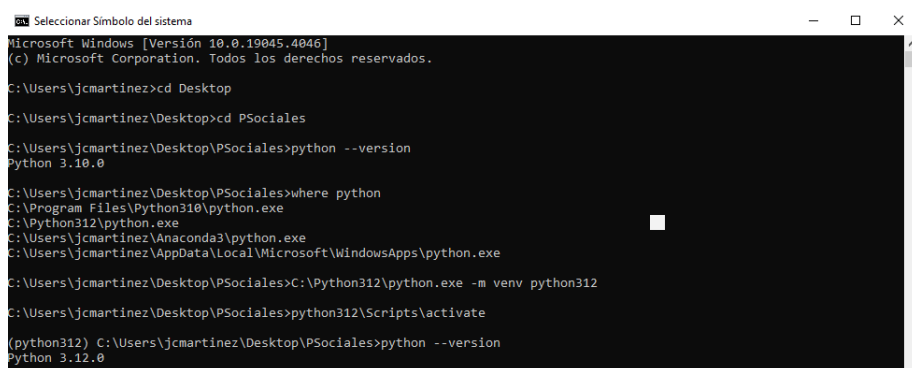
C:\Users\jcmartinez>cd Desktop
C:\Users\jcmartinez\Desktop>cd PSociales
C:\Users\jcmartinez\Desktop\PSociales>python --version
Python 3.10.0

C:\Users\jcmartinez\Desktop\PSociales>where python
C:\Program Files\Python310\python.exe
C:\Python312\python.exe
C:\Users\jcmartinez\Anaconda3\python.exe
C:\Users\jcmartinez\AppData\Local\Microsoft\WindowsApps\python.exe

C:\Users\jcmartinez\Desktop\PSociales>C:\Python312\python.exe -m venv python312
```

Figure 11.2: Creación de ambiente virtual

- Después de ello, no debe pasar nada, ningún resultado en la terminal, lo cual significa que el ambiente virtual se ha creado correctamente.
- Para activar el ambiente virtual escribimos en la terminal `python312\Scripts\activate`, cuando parezca el nombre del ambiente virtual entre corchetes (`python312`) ya se habrá activado.



```
Seleccionar Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.4046]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jcmartinez>cd Desktop

C:\Users\jcmartinez\Desktop>cd PSociales

C:\Users\jcmartinez\Desktop\PSociales>python --version
Python 3.10.0

C:\Users\jcmartinez\Desktop\PSociales>where python
C:\Program Files\Python310\python.exe
C:\Python312\python.exe
C:\Users\jcmartinez\Anaconda3\python.exe
C:\Users\jcmartinez\AppData\Local\Microsoft\WindowsApps\python.exe

C:\Users\jcmartinez\Desktop\PSociales>C:\Python312\python.exe -m venv python312

C:\Users\jcmartinez\Desktop\PSociales>python312\Scripts\activate

(python312) C:\Users\jcmartinez\Desktop\PSociales>python --version
Python 3.12.0
```

Figure 11.3: Activación de ambiente virtual

Chapter 12

Control de versiones con Git, y GitHub

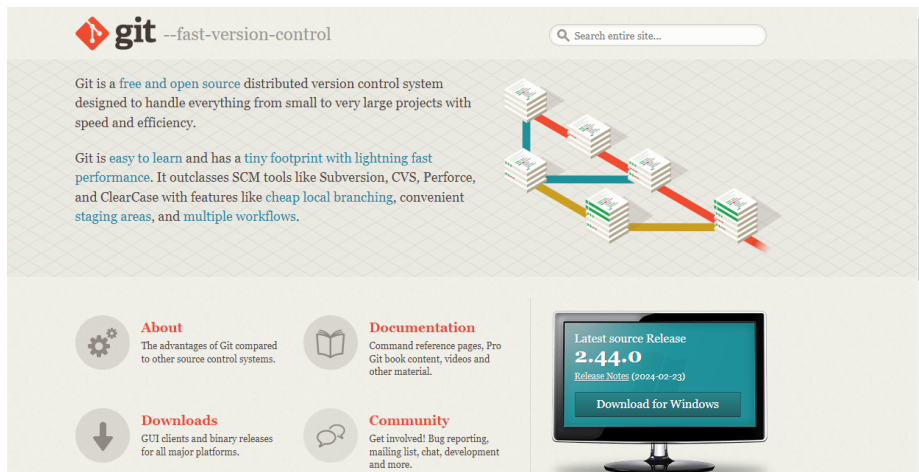


Figure 12.1: Git

Conexión con repositorio remoto *GitHub*:

- Descargar e instalar [Git](#)
- Crear una cuenta en [GitHub](#)
- Crear una nueva carpeta que contendrá archivos del primer repositorio, p.e., *ProgramasSociales*, acceder a la carpeta, dar click con el botón derecho del mouse, y seleccionar *Git Bash Here*

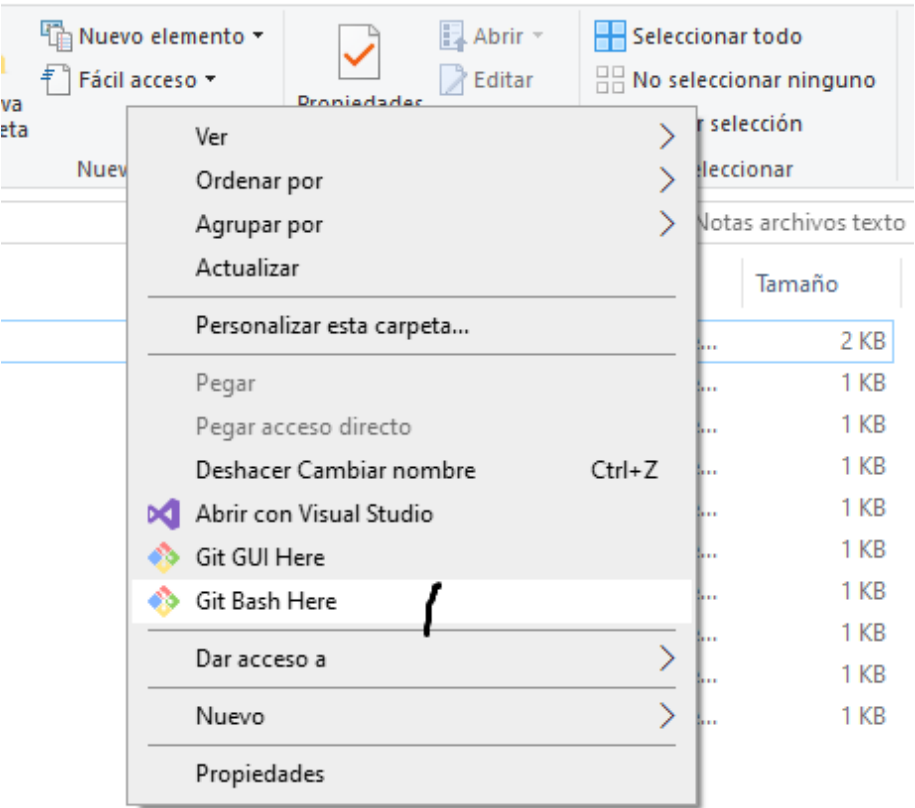


Figure 12.2: Consola Git Bash

- Una vez que se abrió la consola de *Git Bash*, introducimos los siguientes comandos: `git config global user.name` seguido del nombre del repositorio remoto (*GitHub*) creado "SESNA-Inteligencia", posteriormente escribiremos `,git config global user.email` seguido del correo con el cual se creó el repositorio remoto (*GitHub*):

```
jcmartinez@SESNA-URPP29 MINGW64 ~/Desktop/Notas archivos texto
$ git config --global user.name SESNA-Inteligencia

jcmartinez@SESNA-URPP29 MINGW64 ~/Desktop/Notas archivos texto
$ git config --global user.email jcmartinez@sesna.gob.mx

jcmartinez@SESNA-URPP29 MINGW64 ~/Desktop/Notas archivos texto
$
```

Figure 12.3: Consola Git Bash

- En el repositorio remoto *GitHub* nos dirigimos a la sección de repositorios, damos click en *New* para crear uno nuevo, y en el apartado *section name** agregamos el nombre del nuevo repositorio, y damos click en *Create repository* al final de la página.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository](#)

Required fields are marked with an asterisk (*).

Owner *

Repository name *

SESNA-Inteligencia / primerrepositorio

primerrepositorio is available.

Great repository names are short and memorable. Need inspiration? How about [legendary-octo-happiness](#)?

Description (optional)

☒ Public

Anyone on the Internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#)

Add gitignore

gitignore template: None

Choose which files not to track from a list of templates. [Learn more about gitignore files](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses](#)

☐ You are creating a public repository in your personal account.

Create repository

Figure 12.4: Consola Git Bash

- Una vez creado, debe aparecer la siguiente información en el repositorio

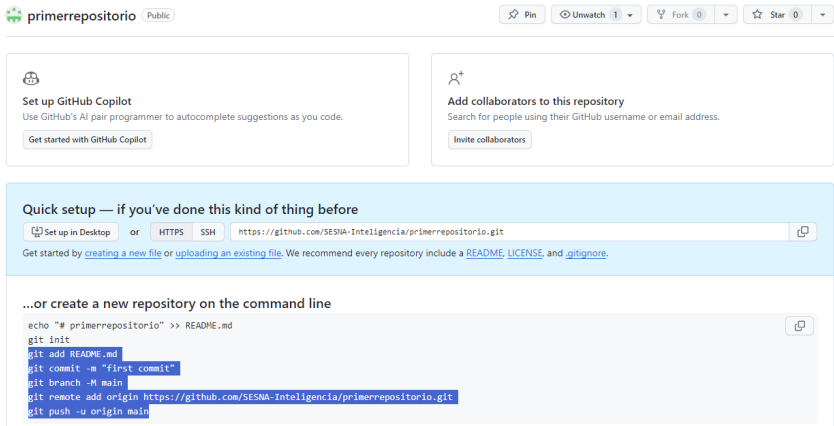


Figure 12.5: Creación de ambiente virtual

- para subir la información de la carpeta al repositorio remoto, vamos crear un archivo *.txt* llamado *prueba.txt*, y dentro del archivo agregamos la frase *Documento de prueba*, guardamos y cerramos.
- escribimos en la consola de Git *git init* para indicar que estamos iniciando con Git.
- escribimos el comando *git commit -m "first commit"* para nombrar e identificar rápidamente los cambios realizados en el commit.
- escribimos *git branch -M main* para indicar la creación de una llama llamada "main".
- escribimos *git remote add origin https://github.com/SESNA-Inteligencia/primerrepositorio.git* para establecer la conexión con el repositorio remoto.
- finalmente escribimos *git push -u origin main* para subir los archivos al repositorio remoto.

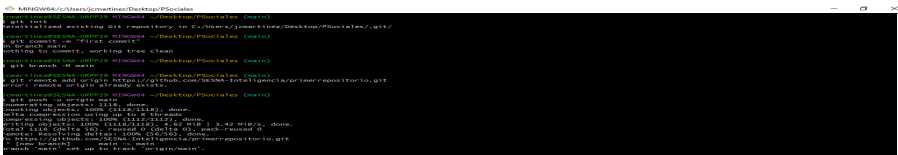


Figure 12.6: Creación de ambiente virtual

- Nos dirigimos al repositorio remoto y observaremos ya se han subido los archivos de la carpeta local.

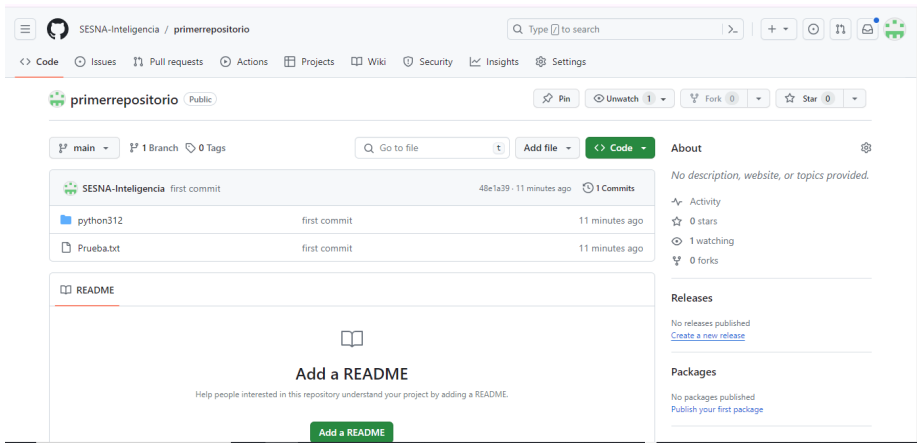


Figure 12.7: Creación de ambiente virtual

12.1 Comandos útiles

- *git checkout*: permite movernos de un commit a otro.
- *git clone* : permite descargar localmente un repositorio remoto.
- *git add* : Guarda el estado del archivo en preparación para realizar un commit
 - *git add "filename.txt"*: agrega "filename.txt" al área de preparación (staging area)
 - *git add .*: agrega todos los cambios al área de preparación (staging area)
 - *git switch*: Cambia a la rama especificada
- *git commit -m [mensaje]* : Registra los cambios del archivo permanentemente en el historial de versiones
- *git brach*: Enumera todas las ramas en el repositorio actual
- *git branch [name]*: Crea una nueva rama

Chapter 13

Atomatización con Docker

Chapter 14

Deploy Dash app en servidor Linux

