# di1. PyDoc

Al instalar la distribución de Python, se nos instala pydoc, esta utilidad, nos permite recuperar y buscar rápidamente documentación desde el Shell así como crear páginas html de documentación.

## Ver opciones de comandos

Si abro un Shell y ejecuto:

|  |
| --- |
| python –m pydoc |

Nos aparecen todas las opciones de comandos que podemos utilizar para obtener información de las estructuras que podemos crear dentro de nuestro código, y crear nuestra propia documentación.

Veamos cada caso que nos aparece como opción al ejecutar el comando anterior:

### pydoc <name> ...

### Permite mostrar documentación de texto sobre algo. <nombre> puede ser el nombre de una palabra clave, tema, función, módulo o paquete de Python, o una referencia punteada a una clase o función dentro de un módulo o módulo en el paquete. Si <nombre> contiene un '\', se usa como la ruta a un archivo fuente de Python para documentar. Si el nombre es 'palabras clave', 'temas' o 'módulos', se muestra una lista de estas cosas.

Como ejemplo podemos consultar información sobre el módulo “random” el cual nos permite generar números aleatorios, de la siguiente forma:

|  |
| --- |
| python –m pydoc random |

Help on module random:

NAME

random - Random variable generators.

DESCRIPTION

integers

--------

uniform within range

sequences

---------

pick random element

pick random sample

pick weighted random sample

generate random permutation

distributions on the real line:

------------------------------

uniform

triangular

normal (Gaussian)

lognormal

negative exponential

gamma

beta

pareto

Weibull

-- Más --

Como podemos ver al final de la información que se despliega en el Shell nos aparece la opción de ver más, con lo cual si presionamos “**Enter**” nos seguirá apareciendo más información la cual se irá desplegando de a una línea por vez, es decir que cada “**Enter**” equivale a una nueva línea de descripción.

### pydoc -k <keyword>

### Permite buscar una palabra clave en las líneas de sinopsis de todos los módulos disponibles.

### pydoc -n <hostname>

Inicia un servidor HTTP con el número de puerto dado, (en este caso 54200)

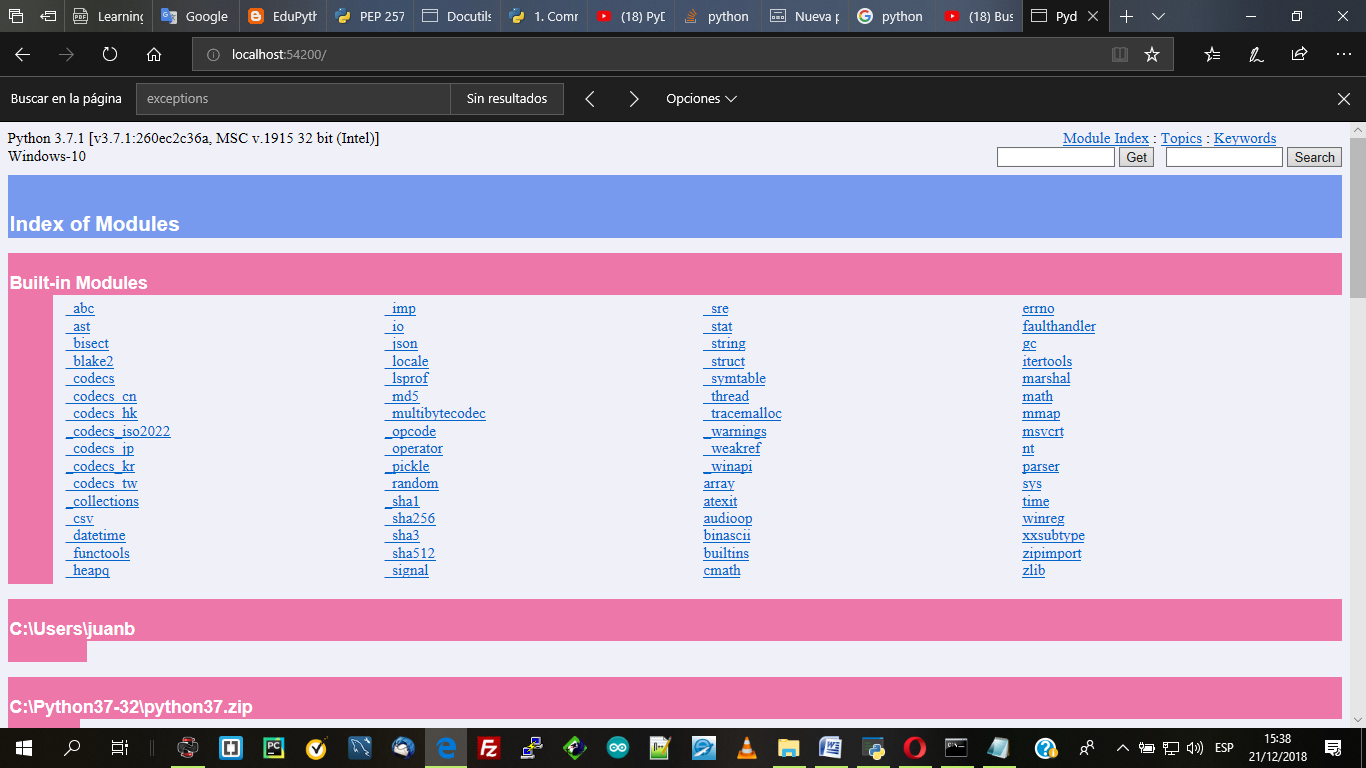
C:\Users\juanb>python -m pydoc -p 54200

Server ready at http://localhost:54200/

Server commands: [b]rowser, [q]uit

server>

Por lo que si ingresamos en la dirección dada en un explorador podemos acceder a la página desde la cual encontramos el listado de todos los módulos de nuestra distribución.



### pydoc -b

Inicia un servidor HTTP en un puerto arbitrario no utilizado y abre un navegador web para navegar interactivamente por la documentación. Esta opción se puede utilizar en combinación con -n y / o -p.

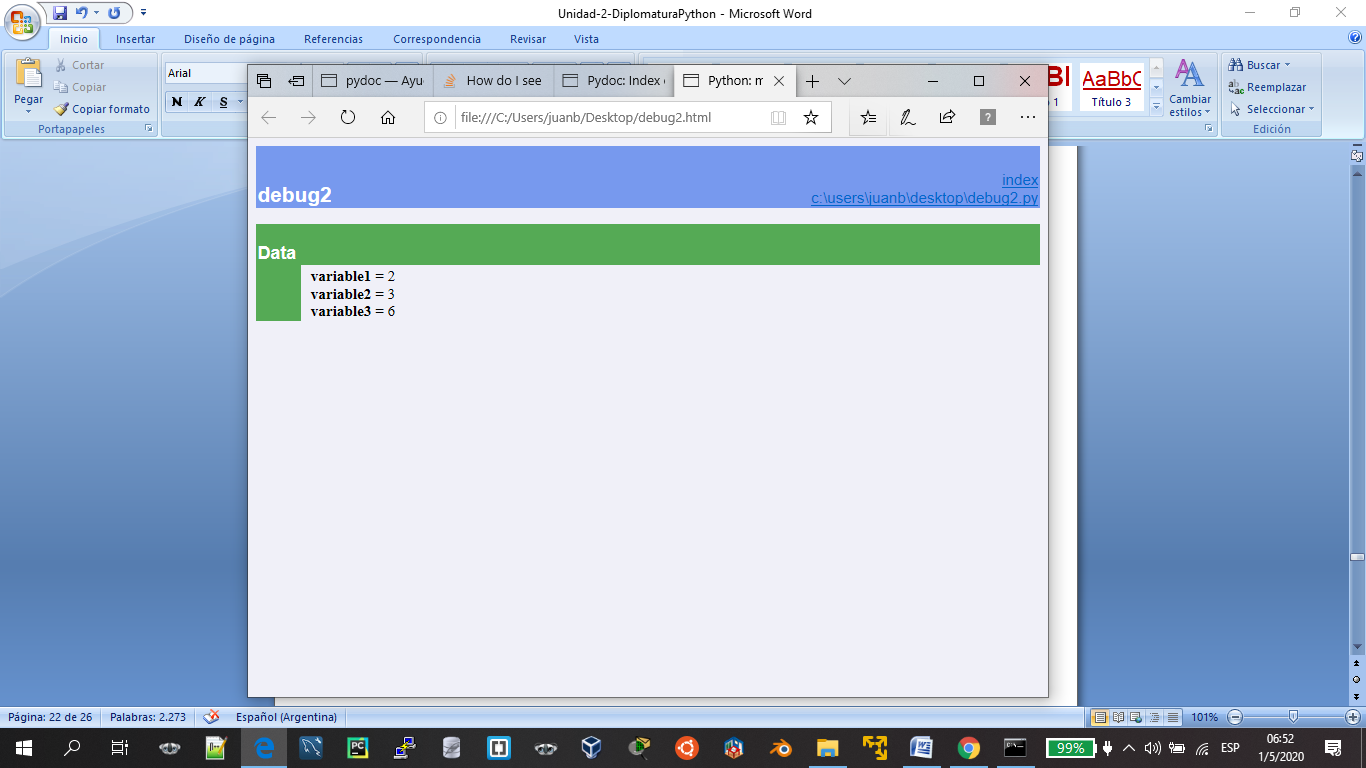
### pydoc -w <name> ...

Permite crear una página html en base a la información agregada a los objetos de un determinado archivo. Podemos probarlo con nuestro ejercicio debug2.py, si accedemos al directorio en donde este se encentra desde el cmd y ejecutamos:

|  |
| --- |
| python -m pydoc -w debug2 |

Esto nos generará una página web con el mismo nombre conteniendo la información de nuestra programa, la cual se agregará en el mismo directorio en el cual tenemos el archivo.

Podemos hacer doble click sobre el archivo debug2.html que se acaba de crear para que la página web se abra.



# 2. sphinx

[**https://www.sphinx-doc.org/en/master/tutorial/index.html**](https://www.sphinx-doc.org/en/master/tutorial/index.html)

## Ventajas

* Presenta una estructura jerárquica
* Permite realizar referencias cruzadas
* Posee varios formatos de salida, como html y pdf
* Se pueden agregar extensiones: **https://sphinx-extensions.readthedocs.io/en/latest/**

**PASO 1 – Creamos un entorno virtual --🡪 documentación**

Instalo virtualenv

|  |
| --- |
| pip install virtualenv |

|  |
| --- |
| virtualenv para\_sphinx -p C:\Users\PC\Documents\Python\Python39\python.exe |

**PASO 2 – Instalamos dependencias**

|  |
| --- |
| python -m pip install sphinx |

**PASO 3 – Comprobamos instalación**

|  |
| --- |
| sphinx-build --version |

**PASO 4 – Creamos un directorio (mi\_proyecto), ingresamos en él y ejecutamos**

|  |
| --- |
| sphinx-quickstart docs |

Surgen unas series de preguntas que podemos responder de la siguiente forma:

* > Separate source and build directories (y/n) [n]: **y**
* > Project name: Write **utn**
* > Author name(s): Write **juan**
* > Project release []: **0.1**
* > Project language [en]: **es**

|  |
| --- |
| Tabla de lenguajes soportados |
| * ar – Arabic * bg – Bulgarian * bn – Bengali * ca – Catalan * cak – Kaqchikel * cs – Czech * cy – Welsh * da – Danish * de – German * el – Greek * en – English * eo – Esperanto * es – Spanish * et – Estonian * eu – Basque * fa – Iranian * fi – Finnish * fr – French * he – Hebrew * hi – Hindi * hi\_IN – Hindi (India) * hr – Croatian * hu – Hungarian * id – Indonesian * it – Italian * ja – Japanese * ko – Korean * lt – Lithuanian * lv – Latvian * mk – Macedonian * nb\_NO – Norwegian Bokmal * ne – Nepali * nl – Dutch * pl – Polish * pt – Portuguese * pt\_BR – Brazilian Portuguese * pt\_PT – European Portuguese * ro – Romanian * ru – Russian * si – Sinhala * sk – Slovak * sl – Slovenian * sq – Albanian * sr – Serbian * sr@latin – Serbian (Latin) * sr\_RS – Serbian (Cyrillic) * sv – Swedish * ta – Tamil * te – Telugu * tr – Turkish * uk\_UA – Ukrainian * ur – Urdu * vi – Vietnamese * zh\_CN – Simplified Chinese * zh\_TW – Traditional Chinese |

Luego de las preguntas nos queda la siguiente estructura:

|  |
| --- |
| docs  ├── build  ├── make.bat  ├── Makefile  └── source  ├── conf.py  ├── index.rst  ├── \_static  └── \_templates |

|  |  |
| --- | --- |
| build | Un directorio vacío (por ahora) que contendrá la documentación procesada. |
| make.bat | Scripts de conveniencia para simplificar algunas operaciones comunes de Sphinx, como representar el contenido. |
| Makefile | Un script de Python que contiene la configuración del proyecto Sphinx. Contiene el nombre del proyecto y la versión que especificó para sphinx-quickstart, así como algunas claves de configuración adicionales. |
| source | El documento raíz del proyecto, que sirve como página de bienvenida y contiene la raíz del “árbol de tabla de contenido” (o toctree). |

**PASO 5 – Ejecutamos el siguiente código y ya tenenemos el .html**

|  |
| --- |
| sphinx-build -b html docs/source/ docs/build/html |

El .html se encuentra en:

|  |
| --- |
| docs/build/html/index.html |

**PASO 6 – index.rst**

El archivo index.rst que creó sphinx-quickstart ya tiene algo de contenido y se representa como la página principal de su documentación HTML. Está escrito en **reStructuredText**, un poderoso lenguaje de marcas.

|  |
| --- |
| docs/source/index.rst |
| .. Lumache documentation master file, created by  sphinx-quickstart on Fri May 6 08:11:39 2022.  You can adapt this file completely to your liking, but it should at least  contain the root `toctree` directive.  Welcome to Lumache's documentation!  ===================================  .. toctree::  :maxdepth: 2  :caption: Contents:  Indices and tables  ==================  \* :ref:`genindex`  \* :ref:`modindex`  \* :ref:`search` |

Vamos a modificarlo de la siguiente forma

|  |
| --- |
| docs/source/index.rst |
| Welcome to Lumache's documentation!  **===================================**  **\*\***Lumache**\*\*** (/lu'make/) is a Python library for cooks and food lovers that  creates recipes mixing random ingredients. It pulls data from the `Open Food  Facts **database <https://world.openfoodfacts.org/>**`\_ and offers a **\*simple\*** and  **\*intuitive\*** API.  **.. note::**  This project is under active development. |

Esto muestra varias características de la sintaxis de reStructuredText, que incluyen:

* un encabezado de sección usando **===** para el subrayado,
* dos ejemplos de marcado en línea: **\*\*énfasis fuerte\*\*** (normalmente en negrita) y **\*énfasis\*** (normalmente en cursiva),
* un enlace externo en línea,
* y una nota de advertencia (una de las directivas disponibles)

**PASO 7 – Modificamos el contenido del html**

Ingresamos al directorio “docs” y reconstruimos el archivo .html

|  |
| --- |
| cd docs  make html |

**PASO 8 – Otros formatos**

Sphinx admite una variedad de formatos además de HTML, incluidos TEXT, EPUB y más. Podemos ver los diferentes formatos mediante el comando:

|  |
| --- |
| make help |

Una vez seleccionado el formato, por ejemplo “epub” se ejecuta:

|  |
| --- |
| make epub |

Esto crea un directorio nuevo con ese formato. Este formato puede ser leído mediante “Calibre”: <https://calibre-ebook.com/>

**PASO 9 – Extensiones**

Las extensiones pueden habilitarse desde el archivo de configuración que se encuentra en: docs/source/conf.py

Por ejemplo podemos cambiar de tema de vista a alguno de estos formatos:

Propios de sphinx

<https://www.sphinx-doc.org/en/master/usage/theming.html#builtin-themes>

De terceros

<https://sphinx-themes.org/#themes>

Por ejemplo podemos instalar este tema:

https://sphinx-themes.org/sample-sites/sphinx-rtd-theme/

Primero lo cargamos:

|  |
| --- |
| pip install sphinx-rtd-theme |

Luego modificamos el archivo de configuración:

|  |
| --- |
| html\_theme = 'sphinx\_rtd\_theme' |

Donde 'sphinx\_rtd\_theme' es el nombre del tema.

**PASO 10 – Estructuración de su documentación en varias páginas**

El archivo **index.rst** creado por sphinx-quickstart es el documento raíz, cuya función principal es servir como página de bienvenida y contener la raíz del “árbol de la tabla de contenido” (o toctree). Sphinx le permite ensamblar un proyecto a partir de diferentes archivos, lo cual es útil cuando el proyecto crece.

Como ejemplo, cree un nuevo archivo docs/source/usage.rst (junto a index.rst) con estos contenidos:

|  |
| --- |
| usage.rst |
| Usage  =====  Installation  ------------  To use Lumache, first install it using pip:  .. code-block:: console  (.venv) $ pip install lumache |

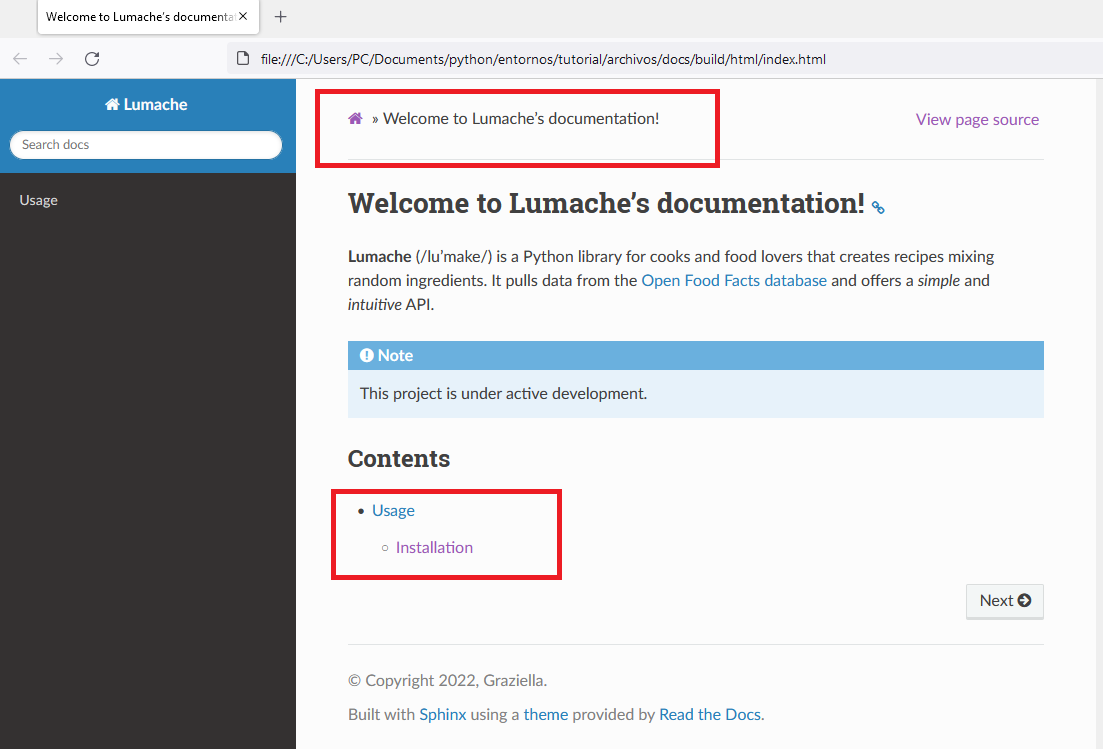
Este nuevo archivo contiene dos encabezados de sección, texto de párrafo normal y una directiva de bloque de código que representa un bloque de contenido como código fuente, con resaltado de sintaxis apropiado (en este caso, texto de consola genérico).

La estructura del documento está determinada por la sucesión de estilos de encabezado, lo que significa que, al usar --- para la sección "Instalación" después de === para la sección "Uso", ha declarado que "Instalación" es una subsección de “Uso”.

Para completar el proceso, agregue una directiva toctree al final de index.rst que incluya el documento que acaba de crear, de la siguiente manera:

|  |
| --- |
| index.rst |
| Welcome to Lumache's documentation!  ===================================  \*\*Lumache\*\* (/lu'make/) is a Python library for cooks and food lovers that  creates recipes mixing random ingredients. It pulls data from the `Open Food  Facts database <https://world.openfoodfacts.org/>`\_ and offers a \*simple\* and  \*intuitive\* API.  .. note::  This project is under active development.  **Contents**  **--------**  **.. toctree::**  **usage** |

Este paso inserta ese documento en la raíz del toctree, por lo que ahora pertenece a la estructura de su proyecto, que hasta ahora se ve así:

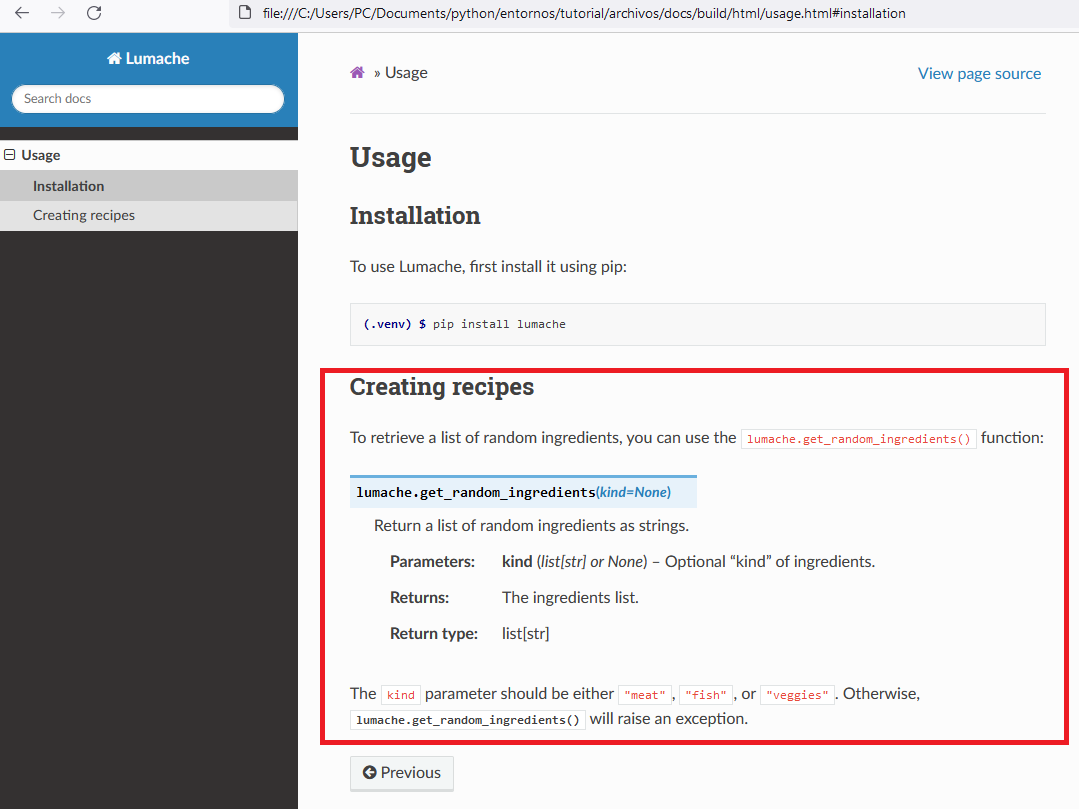


PASO 11 – Descripción de código

Sphinx ofrece varios roles y directivas para documentar objetos de Python, todos agrupados en el dominio de Python. Por ejemplo, puede usar la directiva py:function para documentar una función de Python, de la siguiente manera:

Modificamos el archivo usage.rst así:

|  |
| --- |
| usage.rst |
| Usage  =====  Installation  ------------  To use Lumache, first install it using pip:  .. code-block:: console  (.venv) $ pip install lumache  **Creating recipes**  **----------------**  **To retrieve a list of random ingredients,**  **you can use the ``lumache.get\_random\_ingredients()`` function:**  **.. py:function:: lumache.get\_random\_ingredients(kind=None)**  **Return a list of random ingredients as strings.**  **:param kind: Optional "kind" of ingredients.**  **:type kind: list[str] or None**  **:return: The ingredients list.**  **:rtype: list[str]**    **The ``kind`` parameter should be either ``"meat"``, ``"fish"``,**  **or ``"veggies"``. Otherwise, :py:func:`lumache.get\_random\_ingredients`**  **will raise an exception.** |



# 4 Documentar proyecto con Sphinx

PASO 1 - Ingresamos al directorio de nuestro proyecto

PASO 2 – Ejecutamos:

|  |
| --- |
| sphinx-apidoc -o docs . -f -F -H MiProyecto -A "Juan" -V 0.0.1 -R 0.0.1 |

PASO 3 – Modifico el archivo docs/config.py así:

|  |
| --- |
|  |
| import os  import sys  sys.path.insert(0, os.path.abspath('../'))  ------------------------  language = 'es'  --------------------------  html\_theme = 'sphinx\_rtd\_theme'  ------------------------ |

PASO 4 – Ingresamos a “docs” y ejecutamos

|  |
| --- |
| make html |

# 3 GIT

# Instalación

## Configuramos el nombre

Abrimos el gitbash e iniciamos cargando nuestros datos:

|  |
| --- |
| git config --global user.name "juan" |

Si quiero ver la configuración pongo el mismo comando pero sin el nombre

|  |
| --- |
| git config --global user.name |

Si quiero cambiarlo puedo eutarlo de nuevo con la modificación:

|  |
| --- |
| git config --global user.name "marcelo" |

## Configuramos el email

Hacemos lo mismo para configurar el correo

|  |
| --- |
| git config --global user.email juanbarretor@gmail.com |

## Configuramos los colores

|  |
| --- |
| git config --global color.ui true |

## Ver estado de configuraciones

|  |
| --- |
| git config --global --list |

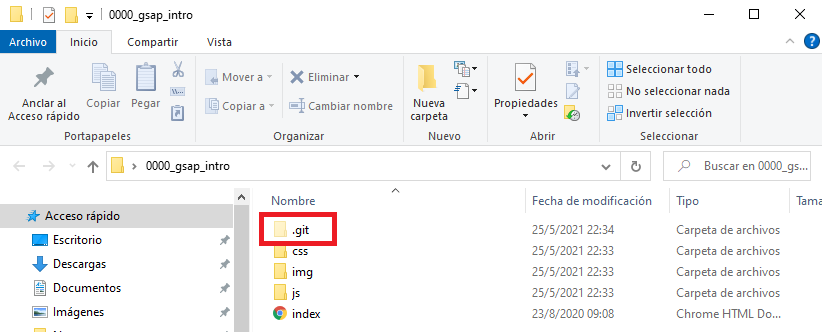
# Comenzamos con un proyecto

## INICIAMOS EL PROYECTO

Ingresamos al directorio del proyecto desde el gitbash y ejecutamos:

|  |
| --- |
| git init |

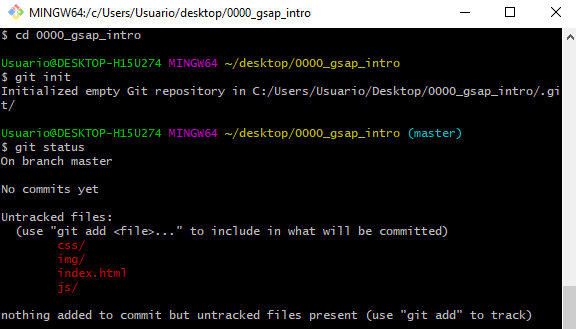
Para que git inicie el seguimiento de lo que pasa dentro de este directorio destinado al proyecto. En el directorio de crea un directorio llamado .git destinado a guardar los cambios.



## VEMOS EL ESTADO DEL PROYECTO

Ejecutamos:

|  |
| --- |
| git status |



## PARA AGREGAR ARCHIVOS

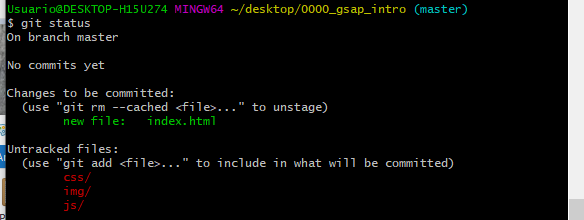
Para agregar todo un proyecto ejecutamos:

|  |
| --- |
| git add -A |

Para agregar un archivo, por ejemplo index.html

|  |
| --- |
| git add index.html |

Al hacer ahora un “git status” nos muestra lo que se ha agregado:



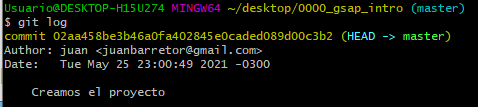
## PARA GUARDAR CAMBIOS

Ejecutamos:

|  |
| --- |
| git commit -m “Descripción del cambio”  Ejemplo  git commit -m "Creamos el proyecto" |

## VER DETALLE DE TODOS LOS COMMITS

|  |
| --- |
| git log |



## GUARDAR LOS COMMITS EN TXT

|  |
| --- |
| git log > commits.txt |

## VAMOS A OTRO PUNTO EN LA RAMA

Ejecutamos:

|  |
| --- |
| git checkout id |

En donde el id es el identificador del commit que aparece en amarillo en el gitbash.

Si quiero ir al último commit ejecuto:

|  |
| --- |
| git checkout master |

## ELIMINAR COMMINT

Usamos “reset”

El primero que tenemos es:

|  |
| --- |
| git reset --soft id |

Este no toca nuestra área de trabajo (no modifica el código) solo borra el commit pero no el código.

El segundo es:

|  |
| --- |
| git reset --mixed id |

Borra el “Staging Area” sin tocar el área de trabajo.

El tercero es:

|  |
| --- |
| git reset --hard id |

Borra todo el commit.

## Modificar descripción de commit

|  |
| --- |
| git commit --amend -m “texto modificado” |

## VER DOCUMENTACIÓN

|  |
| --- |
| git help |

De forma más específica por ejemplo con “status”

|  |
| --- |
| git help status |

## PARA VER DIFERENCIAS REALIZADAS A UN ARCHIVO

Por ejemplo podemos hacer un

|  |
| --- |
| git status |

Para saber que archivos cambiaron, y luego si por ejemplo el index.html cambio, podemos hacer un:

|  |
| --- |
| git diff index.html |

Esto antes del commit.

## PARA NO INCLUIR ARCHIVOS EN EL SEGUIMIENTO

Creo un archivo .gitignore