UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

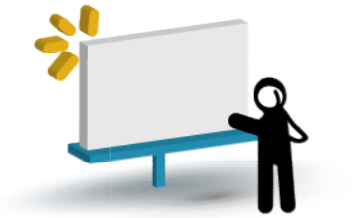
UNIDAD DIDÁCTICA VIII

DIPLOMATURA EN PYTHON

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo IV – Nivel Intermedio

Unidad VIII – Documentación.



Presentación:

Un tema no menos que trataremos en esta unidad es sobre cómo obtener información sobre python y cómo documentar de forma correcta nuestro trabajo, estos temas suelen ser un tanto periféricos al lenguaje en sí, pero se convierten en un conocimiento esencial al tener que profundizar en el lenguaje y al realizar trabajos en equipo con otros desarrolladores.

Objetivos:

Que los participantes:

Comiencen a documentar sus trabajos.

Aprendan a trabajar con git.

Bloques temáticos:

1.- Fuentes de documentación.

2.- PyDoc.

3.- Git.

Consignas para el aprendizaje colaborativo

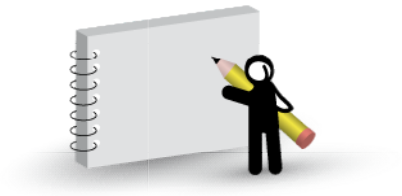
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1.- Fuentes de documentación.

Podemos encontrar documentación sobre un programa que estemos analizando en diferentes fuentes tanto dentro como fuera del documento.

1.- Comentarios.

Un primer lugar donde podemos encontrar información como ya hemos venido viendo en las unidades anteriores es en la implementación de comentarios, tanto de una línea con la utilización de (#):

```
# Este es un comentario de una línea.
```

Como multilínea con la utilización de las triples comillas:

```
“““Este es
            un comentario
                        multilinea.
”””
```

2.- dir()

Si recordamos hemos comentado que en python todos los elementos son objetos, y todos los objetos tienen asociados métodos, si le pasamos como parámetro al método dir() un objeto, este nos retorna todos los métodos que el objeto puede implementar, tanto los propios de la clase (si es que es una clase nuestra) como aquellos métodos que vienen incluidos en el núcleo de la distribución que estamos utilizando. Podemos ver que si por ejemplo tenemos una lista frutas, dado que la lista es un objeto en python, al aplicarle el método dir(), nos encontramos que obtenemos todos los métodos por defecto asociados a los objetos listas:

info1.py

```
1   lista =  ['Manzana', 'Pera', 'Limón']
2   print(dir(lista))
```

Retorna:

```
'__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
'__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__',
'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

Lo mismo pasa con un objeto definido a partir de una clase creada por nosotros, salvo que en este caso tendremos tanto los métodos creados en la clase como los que vienen asociados a las clases en general por defecto. En cuanto a los métodos por defecto podemos obtener información sobre ellos si utilizamos el método “help()” como se muestra a continuación.

info2.py

```
1   class MiString():
2       def __init__(self, mi_string):
3           self.mi_string = mi_string
4
5       def imprimir(self):
6           pass
7
8   objeto = MiString('Manzana')
9   print(dir(objeto))
10  print(objeto.__class__)
11  print('--------------------------')
12  print(help(objeto.__init__))
```

Retorna:

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__',
'__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'imprimir',
'mi_string']
<class '__main__.MiString'>
--------------------------
Help on method __init__ in module __main__:

__init__(mi_string) method of __main__.MiString instance
    Initialize self.  See help(type(self)) for accurate signature.
```

Este punto no es muy complicado de comprender, sin embargo la pregunta podría ser **¿cómo agregamos información sobre el método imprimir que hemos declarado en nuestra clase?.** Este es el tema de nuestro próximo punto.

3.- PEP 257 - Convenciones de documentación.

Los PEP'S son Propuestas de mejora de Python, en particular el PEP 257 documenta la semántica y las convenciones asociadas con las cadenas de documentación de Python. El objetivo de este PEP es estandarizar la estructura de alto nivel de las cadenas de documentación: que deben contener y cómo decirlo (sin tocar ninguna sintaxis de marcado dentro de las cadenas de documentación). El PEP contiene convenciones, no leyes o sintaxis.

Una cadena de documentación (Docstring) es un literal de cadena que se produce como la primera declaración en un módulo, función, clase o definición de método. Dicha cadena de documentos se convierte en el atributo especial __doc__ de ese objeto.

Normalmente, todos los módulos deben tener cadenas de documentación, y todas las funciones y clases exportadas por un módulo también deben tener cadenas de documentación. Los métodos públicos (incluido el constructor __init__) también deben tener cadenas de documentación. Se puede documentar un paquete en la cadena de documentación del módulo del archivo __init__.py en el directorio del paquete.

Los literales de cadena que aparecen en otras partes del código Python también pueden actuar como documentación. No son reconocidos por el compilador de bytecode de Python y no son accesibles como atributos de objeto de tiempo de ejecución (es decir, no asignados a __doc__), pero las herramientas de software pueden extraer dos tipos de cadenas de documentación adicionales:

Los literales de cadena que se producen inmediatamente después de una asignación simple en el nivel superior de un módulo, clase o método __init__ se denominan "**cadenas de documentación de atributos**".

Los literales de cadena que se producen inmediatamente después de otra cadena de documentación se denominan "**cadenas de documentación adicionales**".

Para mantener la coherencia:

- Utilice siempre "" "comillas dobles triples" "" alrededor de cadenas de documentación.
- Utilice r "" comillas dobles triples sin formato "" "si usa alguna barra diagonal inversa en sus cadenas de documentación.
- Para las cadenas de documentación de Unicode, use u "" "Cadenas de comillas triples de Unicode" "".

Hay dos formas de cadenas de documentación: de una sola línea y cadenas de documentación de varias líneas.

Modifiquemos nuestro ejemplo anterior agregándole comentarios para ejemplificar lo dicho hasta aquí, desde nuestro programa la información se retorna con __doc__:

info3.py

```
1   class MiString():
2       """
3       Esta es una clase
4       destinada a ...
5       """
6       def __init__(self, mi_string):
7           self.mi_string = mi_string
8
9       def imprimir(self):
10          """
11          Este es un método que en principio no hace nada
12          pero que luego será utilizado con
13          un fín específico.
14          """
15          pass
16
17  objeto = MiString('Manzana')
18  print("------------Información sobre la clase-------------")
19  print(objeto.__doc__)
20  print("--------Información sobre el método imprimir()-------")
21  print(objeto.imprimir.__doc__)
```

El programa nos retorna:

```
------------Información sobre la clase-------------

    Esta es una clase
    destinada a ...
```

```
--------Información sobre el método imprimir()-------

    Este es un método que en principio no hace nada
    pero que luego será utilizado con
    un fín específico.
```

Nota: No se ponen líneas ni antes ni después de un doctstring.

Nota: Podemos encontrar en el PEP 258 las especificaciones de diseño de las Docutils para obtener una descripción detallada de los atributos y las cadenas de documentación adicionales. Si bien esto puede servir como un documento de diseño interesante, ya no está programado para ser incluido en la biblioteca estándar.

2.- PyDoc

pydoc es una utilidad que se instala al instalar Python. 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 clases, funciones, módulos, e incluso de creación de nuestra propia documentación, nos retorna. 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” 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>

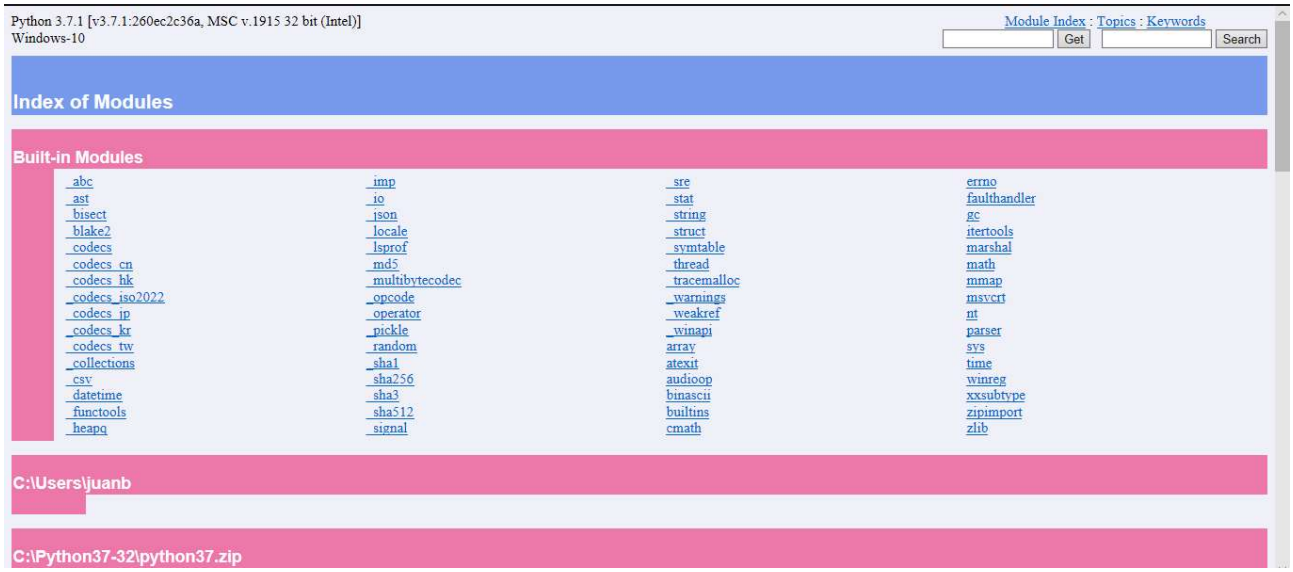
Permite buscar una palabra clave en las líneas de sinopsis de todos los módulos disponibles.

pydoc -n <hostname>

Inicia un servidor HTTP con el nombre del host dado (por defecto es localhost)

```
C:\Users\juanb>python -m pydoc -n localhost
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 -p <port>

Inicie un servidor HTTP en el puerto dado en la máquina local.

pydoc -b

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

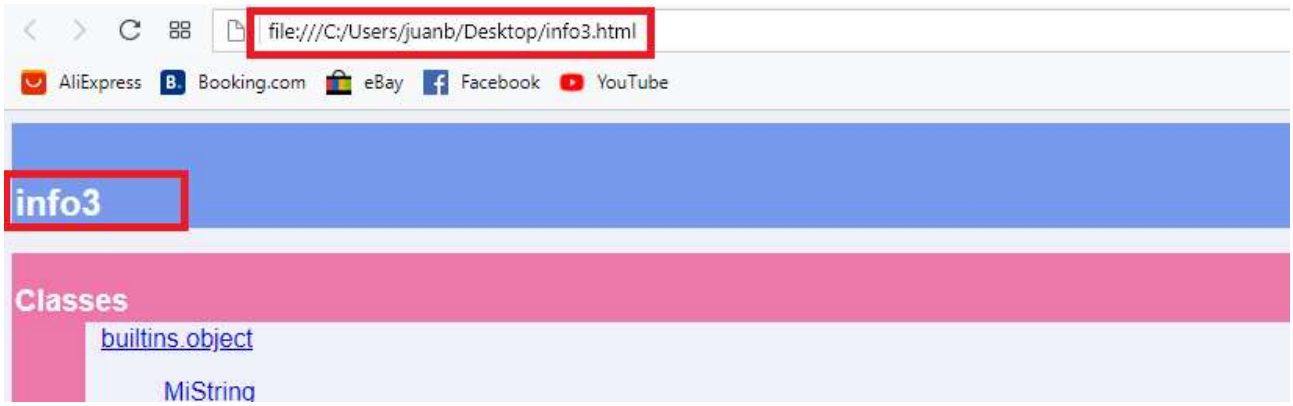
pydoc -w <name> ...

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

```
python -m pydoc -w info3
```

Y se nos generará una página web con el mismo nombre conteniendo la información de nuestra clase y sus componentes, la cual se agregará en el mismo directorio en el cual tenemos el archivo “info3.py”

Podemos hacer doble click sobre el archivo info3.html que se acaba de crear para que la página web se abra.



3. GIT - Básico

Antes de ingresar en el nivel avanzado del curso, debemos adquirir buenas prácticas de organización en nuestros emprendimientos. Durante el desarrollo de una nueva aplicación solemos trabajar con archivos y modificarlos hasta obtener el resultado esperado, pero:

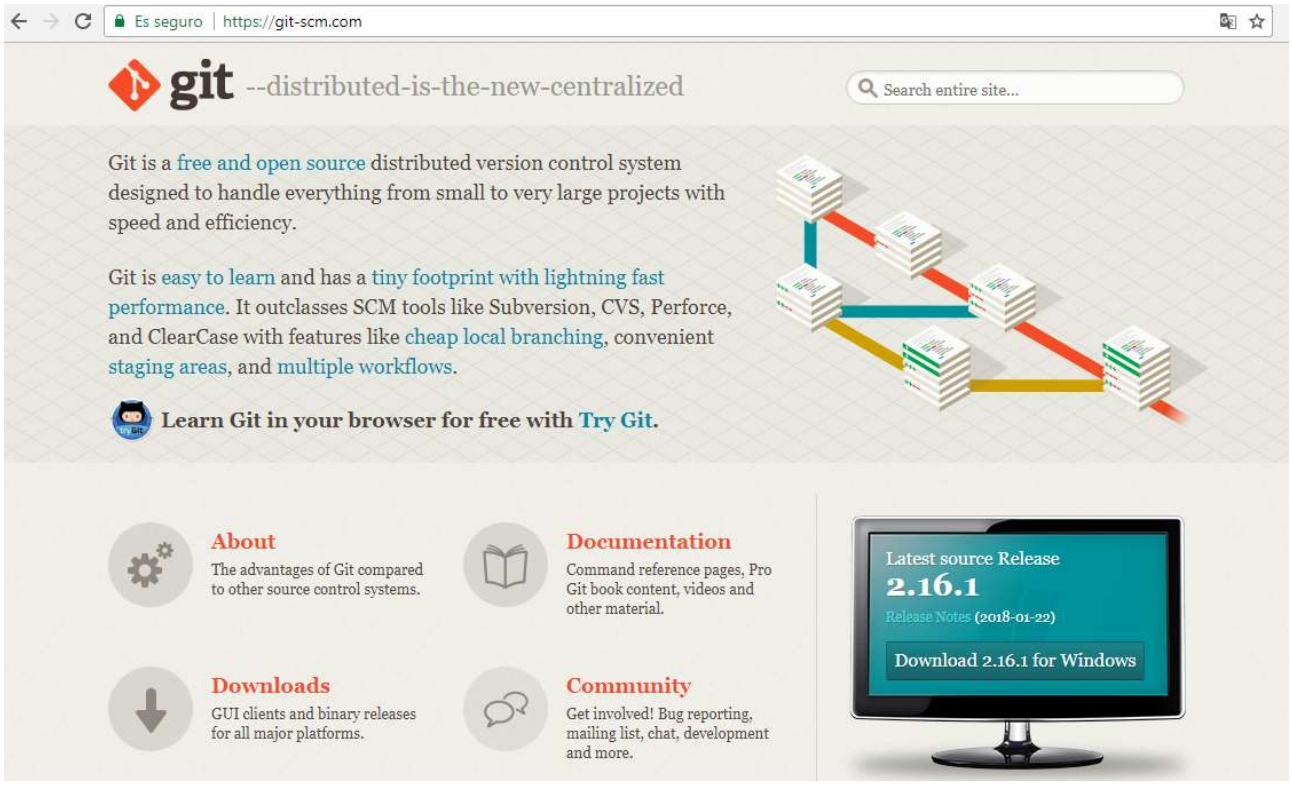
¿Qué pasa si tenemos que interrumpir nuestro pensamiento por un rato e ir a realizar otro trabajo?

¿Qué pasa si al retornar realizamos modificaciones y luego de las mismas queremos retornar a como el archivo se encontraba al inicio?

La respuesta seguramente sería que iríamos realizando copias de las versiones para ir registrando las modificaciones, con el inconveniente de que podemos llegar a tener muchas copias de un mismo archivo con pequeñas modificaciones mal documentadas.

Para evitarnos este problema vamos a trabajar con un controlador de versiones como Git el cual podemos descargar de:

https://git-scm.com/



Forma de uso:

El uso de git es relativamente sencillo, supongamos que tenemos los ejercicios de esta unidad dentro de un directorio que se llama “ejerciciosUnidad”, antes que nada debemos instalar git, lo cual es bastante simple ya que solo debemos descargar el instalador e ir presionando en siguiente hasta finalizar. Una vez que se ha completado la instalación podemos dirigirnos al buscador e iniciar el CMD de git.

PASO 1 - Iniciamos el control con git

Como primer paso nos paramos en el directorio en el cual queremos realizar un control de las versiones, para esto desde el cmd escribo:

cd /

Para ir al directorio raíz, en mi caso el disco C.

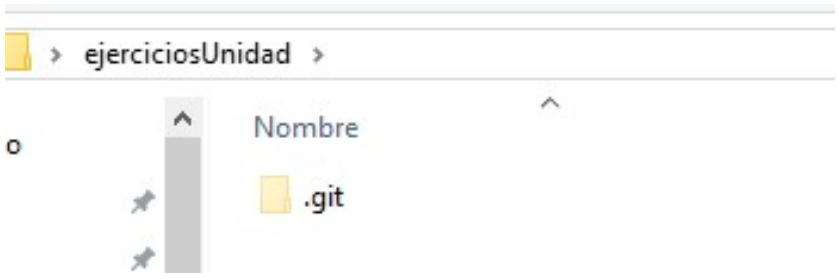
Luego para ir ingresando en los directorios que cada directorio tiene escribo cd seguido del directorio al cual quiero ingresar, por ejemplo si quiero ingresar al directorio "**ejerciciosUnidad**" en el escritorio podría escribir:

```
C:\>cd Users

C:\Users>cd juanb

C:\Users\juanb>cd Desktop

C:\Users\juanb\Desktop>cd ejerciciosUnidad

C:\Users\juanb\Desktop\ejerciciosUnidad>
```

Lo único que nos resta realizar para que git comience a realizar un seguimiento de las versiones es ejecutar:

git init

Si ahora observamos nuestro directorio podemos ver como git a creado un directorio ".git" desde el cual realizará el control de las versiones.



A continuación le indicamos a git quienes somos agregando un mail y un nombre con los siguientes códigos:

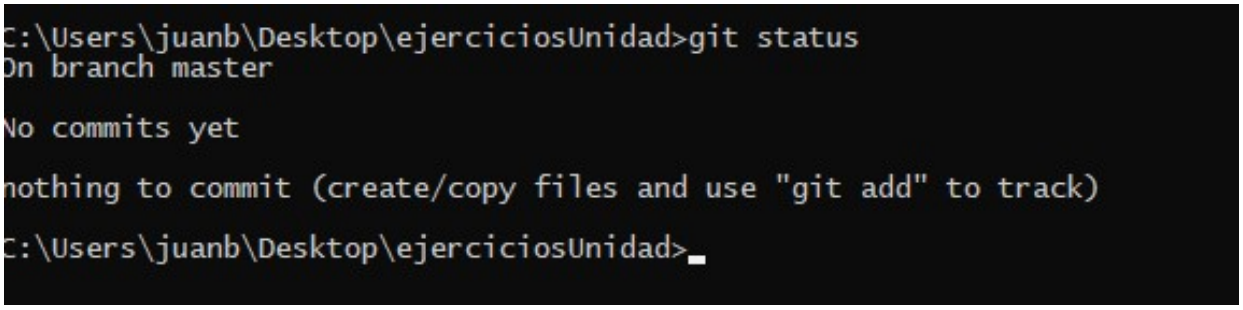
```
git config --global user.email “tu email”
git config --global user.name “tu nombre”
```

PASO 2 – Chequeamos el Status

Si ejecutamos ahora:

```
git status
```

Podemos ver cuál es el estado de nuestras modificaciones en donde git nos informa que aún no hemos guardado nada.



Este comando podemos ejecutarlo cada vez que queramos ver los cambios previos a guardarlos.

PASO 3 – Indicamos archivos a realizar seguimiento

Para realizar el seguimiento de algún archivo o directorio podemos utilizar el comando “git add” seguido del nombre de archivo o agregando un punto para que haga el seguimiento de todo:

```
git add .
```

Si ahora por ejemplo agregamos el archivo “info1.py” y ejecutamos el comando anterior, en el cmd le estaremos indicando a git que le queremos hacer seguimiento al archivo.

Si ahora ejecutamos un “git status” podemos ver cómo nos aparecen todos los archivos a los cuales se les va a realizar un seguimiento ya listados.

```
C:\Users\juanb\Desktop\ejerciciosUnidad>git add .

C:\Users\juanb\Desktop\ejerciciosUnidad>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   info1.py


C:\Users\juanb\Desktop\ejerciciosUnidad>git add .

C:\Users\juanb\Desktop\ejerciciosUnidad>
```

Git CMD

```
C:\xampp\htdocs\curso-php>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   css/bootstrap-grid.css
        new file:   css/bootstrap-grid.css.map
        new file:   css/bootstrap-grid.min.css
```

PASO 4 – Realizamos el seguimiento

Ahora para realizar el seguimiento agregamos el comando (git commit –m “mensaje”) indicando entre comillas el mensaje que va a identificar la modificación realizada.

git commit –m “UNIDAD 8 – Sección 3 - Paso 4 – Aprendiendo a utilizar git.”

Si lo ejecutamos y luego chequeamos el status veremos que nos indica que ya no hay nada que guardar.

```
C:\Users\juanb\Desktop\ejerciciosUnidad>git commit -m "UNIDAD 8 - Sección 3 - Paso 4 - Aprendiendo a utilizar git."
[master (root-commit) aff324c] UNIDAD 8 - Sección 3 - Paso 4 - Aprendiendo a utilizar git.
 1 file changed, 2 insertions(+)
 create mode 100644 info1.py

C:\Users\juanb\Desktop\ejerciciosUnidad>git status
On branch master
nothing to commit, working tree clean

C:\Users\juanb\Desktop\ejerciciosUnidad>
```

PASO 5 – Datos de commit realizado

Finalmente podemos realizar un chequeo de lo realizado mediante el comando:

git log

```
C:\Users\juanb\Desktop\ejerciciosUnidad>git log
commit aff324c60f4b074a507c21dbfa835ce5adbe271f (HEAD -> master)
Author: juan <juanbarretor@gmail.com>
Date:   Sat Dec 22 17:02:03 2018 -0300

    UNIDAD 8 - Secci<C3><B3>n 3 - Paso 4 - Aprendiendo a utilizar git.

C:\Users\juanb\Desktop\ejerciciosUnidad>
```

PASO 6 – Regresar a un punto de la rama principal

Los pasos anteriores los podemos seguir cada vez que realicemos una modificación la cual queramos que sea persistida, y si en algún momento tenemos la necesidad de regresar a un punto dado de la rama (punto en el cual hemos realizado un commit) de forma de recuperar los archivos como los teníamos en un punto dado, debemos indicar el hash del commit correspondiente como sigue:

git checkout numero-de-hash

Es decir que si en algún momento quisiéramos recuperar el proyecto como lo tenemos hasta el primer commit que acabamos de realizar, lo único que tendríamos que hacer es ejecutar:

git checkout aff3

Notemos que solo puse los cuatro primero caracteres del hash y no todos. No hace falta agregar todos los caracteres.

```
C:\Users\juanb\Desktop\ejerciciosUnidad>git log
commit aff324c60f4b074a507c21dbfa835ce5adbe271f (HEAD -> master)
Author: juan <juanbarretor@gmail.com>
Date:   Sat Dec 22 17:02:03 2018 -0300

    UNIDAD 8 - Secci<C3><B3>n 3 - Paso 4 - Aprendiendo a utilizar git.
```

Nota: Git nos brinda muchas más opciones e incluso podemos seleccionar alguna de las interfaces gráficas que se encuentran en la página oficial para realizar los seguimientos. También tenemos la opción de crear proyectos online que sean modificados por varios usuarios a la vez. Para una mejor se recomienda leer el libro oficial que se puede descargar gratis de la página principal de git: ***https://git-scm.com/***

Bibliografía utilizada y sugerida

Libros

Programming Python 5th Edition – Mark Lutz – O'Reilly 2013

Programming Python 4th Edition – Mark Lutz – O'Reilly 2011

Manual online

https://docs.python.org/3.7/tutorial/

https://docs.python.org/3.7/library/index.html

https://www.python.org/dev/peps/pep-0257/

https://www.python.org/dev/peps/pep-0258/

https://git-scm.com/

Lo que vimos

En esta unidad hemos visto la forma de documentar nuestros proyecto y de organizar nuestro trabajo utilizando un controlador de versiones.

Lo que viene:

En la siguiente unidad comenzaremos a trabajar aspectos avanzados del lenguaje de Python.