Docstrings:

En programación la documentación es uno de los aspectos mas importantes, tanto para las funciones, como para el código, para otros y para nosotros mismos. Cada una de las funciones que ya vienen com los programas están documentadas en lo que se conoce como **docstrings**. Estos docstrings son realmente fáciles de hacer si queremos comentar una funcion y explicar como se usa, o para que se usa. Simplemente tenemos que agregar tripes comillas delante y detrás y el interprete lo interpretara como que este es el docstring de la funcion, y si usamos el metodo **help()** , este nos devolverá el docstring que existe dentro de la funcion.

El metodo help lo podemos usar para saber como se usa una clase, metodo o funcion sin tener que ir a la documentación oficial en línea.

Pydoc:

La funcion integrada help hace uso del módulo Pydoc para generar la documentación en tiempo real de la documentación. Desde la terminal no se puede hacer uso del metodo help() sino que tenemos que hacer uso de Pydoc. Pydoc se usa de manera similar a como se usa el interprete de python en la terminal hay que escribir Pydoc, y después de este escribimos el nombre del paquete, modulo o funcion que deseamos ver.

El punto fuerte de Pydoc es que no solo podemos leer, sino que también podemos escribir en el. Haciendo uso de Pydoc -w y luego de esto el nombre del paquete o modulo. Esto lo que hace es que cresa un fichero tipo HTML. Si estamos haciéndolo en un paquete, que tiene subpaquetes, que tienen módulos, que tienen funciones, se crearan archivos HTML para cada uno de ellos los cuales están relacionados en una estructura de tipo árbol.

Doctest:

Doctests como su nombre lo indica son tests que están incluidos en los docstrings, pero como agregamos un test dentro de una cadena e texto?? La respuesta es sencilla, todo lo que este en una cadena de texto con triple comillas pero que esa línea comience con “>>>” será tomado como código para el módulo Doctest y su metodo testmod(). Esto visualmente luce de la manera siguiente:

def suma (a, b):

“””

La funcion suma (a+b) recibe dos parámetros a y b.

Devuelve la suma de ambos

>>> suma(5,10)

15

return a+b

import doctest

doctest.testmod()

>>>TestResults (failed=0, attempted=1)

Unitests o Pyunit:

Los Unitests se crean esencialmente cuando de la clase unittest.TestCase se crea una clase hijo que tiene todos sus metodos esto por su puesto se hace de la manera siguiente:

Import unittest

class Pruebas (unittest.TestCase):

def Test():

assertEqual(ellmnt1,ellmnt2)

if \_\_name\_\_ == “\_\_main\_\_”:

unittest.main()

Lo más importante sobre los unittest es que podemos usar los tests que están dentro de estos como assertEqual o assertTrue y así, estos al ser metodos ya construidos en la clase no tenemos mas que hacer que escribir self.assertEqual por ejemplo y ya ahí tenemos una prueba. La manera de ejecutar estas pruebas es escribiendo la línea de código que se muestra al final del script. O simplemente podríamos correr el unittest.main() en cualquier momento del final del script.