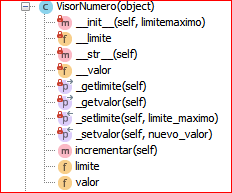
**Recursos:**

* **Métodos especiales (** (\_\_str\_\_, \_\_add\_\_, \_\_cmp\_\_**)** 
  + <http://python-para-impacientes.blogspot.com.es/2015/06/programacion-orientada-objetos-y-iii.html>
  + Archivo **Clases en Python.pdf**, página 147.
* **Delegación:**
  + <http://librosweb.es/libro/algoritmos_python/capitulo_15/delegacion.html>

Actividades:

1. Realiza un programa con la siguiente clase, **VisorNumero** que visualiza un número (futura hora de un reloj horas o minutos):

* **Atributos superocultos** , \_\_limite y \_\_valor, con sus propiedades **limite** y **valor** respectivamente.
* El **constructor** asignará un limitemaximo y valor lo inicializa a 0.
* El **método especial** \_\_str\_\_ permite imprimir el objeto, en este caso devuelve un string con el valor formateado a número (00).
* **método** incrementar(): acumula 1 a valor y si llega al limite, valor vuelve a 0.

Para probar dicha clase,en el método main() escribe:

hora = VisorNumero (24)

minuto= VisorNumero(60)

print (**"Hora: "**)

**for** i **in** range(90):

hora.incrementar()

print (hora, end=**"-"**)

print (**"\n"**)

print (**"Minuto: "**,)

**for** i **in** range(90):

minuto.incrementar()

print( minuto , end=**"-"**)

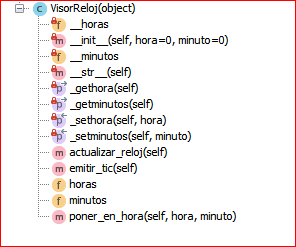
Este será el resultado:

Hora:

01-02-03-04-05-06-07-08-09-10-11-12-13-14-15-16-17-18-19-20-21-22-23-00-01-02-03-04-05-06-07-08-09-10-11-12-13-14-15-16-17-18-19-20-21-22-23-00-01-02-03-04-05-06-07-08-09-10-11-12-13-14-15-16-17-18-19-20-21-22-23-00-01-02-03-04-05-06-07-08-09-10-11-12-13-14-15-16-17-18-

Minuto:

01-02-03-04-05-06-07-08-09-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-00-01-02-03-04-05-06-07-08-09-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-

1. Realiza una clase VisorReloj que represente los dígitos de una hora (HH:MM) con la ayuda de la la clase del ejercicio anterior, que la puedes importar o copiar en este ejercicio. El reloj al emitir un tic (emitir\_tick) se incrementa en un minuto. Si el minuto llega a 60 que es el límite se incrementa una hora. Como funciona un reloj normalmente.

* En el **constructor** se le pasa una hora y un minuto (opcionales). Sus atributos son **horas** y **minutos** (objetos de la clase VisorNumero, los cuales se crean con los límites de 24 y 60 respectivamente. Se pone en hora (poner\_en\_hora) y se actualiza el reloj (actualizar\_reloj) .
* Todos los atributos son ocultos y tendrán su correspondiente propiedad de lectura/escritura.
* El método especial \_\_str\_\_() devuelve un string con la hora en formato **hh:mm**. Coge el valor formateado de horas y minutos, o sea utiliza el método especial \_\_str\_\_() de VisorNumero
* Método actualizar\_reloj: devuelve la hora formateada, o sea lo que devuelve \_\_str\_\_() de esta clase.
* Método emitir\_tic: incrementa en uno los minutos con el método incrementar() de VisorNumero. Si el minuto se ha quedado a cero se aumenta en uno las horas. Se actualiza reloj, actualizar\_reloj().
* Método poner\_en\_hora(): se actualiza la hora (xxx.valor) con la hora y minuto pasados como parámetros. Se actualiza el reloj (actualizar\_reloj())

Para probarlo puedes poner las 4 horas que muestro a continuación con sus resultados emitiendo cada una de ellas un tic.

Hora 1: 00:00

Hora 1 + 1 minuto: 00:01

Hora 2: 23:00

Hora 2 + 1 minuto: 23:01

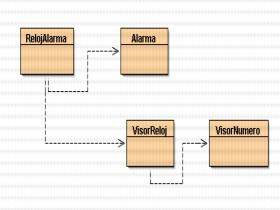
Hora 3: 12:59

Hora 3 + 1 minuto: 13:00

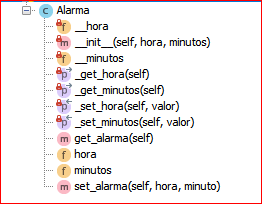
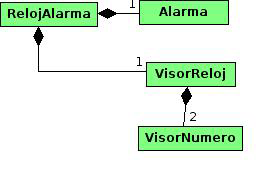
Hora 4: 23:59

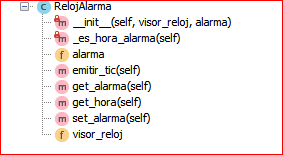
Hora 4 + 1 minuto: 00:00

1. Utilizando las clases anteriores VisorNumero y VisorReloj. Crea las clases RelojAlarma y Alarma. La clase RelojAlarma va a modelar un reloj como el que hemos estudiado pero que incorpora una alarma, así que cada vez que el reloj avanza, si la nueva hora es la misa que la establecida en la alarma se hará sonar un RINGGGGG.



El diagrama de clases es el siguiente:



1. La clase Alarmaguarda la hora de la alarma, permite modificar esta hora con el método set\_alarma**()** y obtiene la hora de alarma como un String de la forma “XX:XX””. El constructor de esta clase establece la alarma, por defecto a las 7:00 horas.
2. La claseRelojAlarmatiene dos atributos: reloj de tipoVisorReloj **y** alarma de tipoAlarma**.** Esta clase incluye:

* un constructor con dos parámetros, la hora y minutos del reloj, que crea el reloj a partir de estos dos parámetros y la alarma.
* el método set\_alarma() permite al reloj cambiar la hora de la alarma.
* los métodosget\_hora() y get\_alarma() devuelven la hora actual y la de la alarma en formato string.
* el método \_es\_hora\_alarma() es un método privado. Este método compara dos cadenas, la que representa la hora actual y la que representa la hora de alarma. Devuelve True si las dos cadenas son iguales.
* el método emitir\_tic() que avanza el reloj. Después de que la hora del reloj avanza se comprueba utilizando el método es\_hora\_alarma**()** si la hora actual y la de la alarma coinciden. Si es así se hace sonar la alarma emitiendo un “RRRRIIIIIIINNNNG” en la pantalla.