F210 Programación Aplicada a Finanzas

TP 1: Objetos (Parte 1)

1. **Clock**

   Se le pide que diseñe un reloj usando OOP. No el display de un reloj, sino un verdadero reloj, que internamente tenga un mecanismo para "medir" el paso del tiempo. La idea, como en todo proyecto de objetos, es ir construyendo el modelo en forma bottom-up:

   - Defina una clase `clock(object)` con atributos de instancia de clase `hora, minuto ,segundo.` Si lo desea, puede agregarle dos atributos más indicando si es un reloj que marca hasta 12 horas o si indica las 24 hs, y si es AM o PM.
   - Construya un método `onesec(self)` cuya función sea medir un segundo. No tendrá inputs (más allá de self) ni outputs, e internamente realizará un loop incrementando un contador hasta un determinado numero N. La idea es que al terminar de procesar ese loop, haya transcurrido aproximadamente 1 segundo.
   - Construya un método `update(self)` que actualice el setting del reloj incrementando el tiempo de a 1 segundo cada vez que se lo llama.
   - Obviamente necesitará un método `printCurrentTime(self),` que indique la hora segun el reloj. Trate que la salida se vea algo asi como `The current time is: hh:mm:ss`
   - Un método `SetClock(self,tupla=None)` que en principio reciba una tupla del tipo `(h,m,s)` para inicializar el reloj antes de ponerlo a andar. El input por defecto será `None`, en cuyo caso el método deberá importar el módulo `datetime`, traer la hora actual con la funcion `datetime.datetime.now()` y usar el resultado para inicializar los atributos de clock.
   - Un método `work(self,tupla=None)` que haga correr el reloj. En tupla pondra el input con el que quiere iniciar el reloj, que por defecto tomará el tiempo de máquina usando `SetClock().` Usted querrá que el reloj corra y para ello hara un loop infinito que corra en secuencia `onesec` y `update` y, de tanto en tanto cuando usted lo desee, le muestre la hora y la compare - por ejemplo - con la hora real. Ah, y usted tambien tiene que poder decirle al reloj que termine de correr.

     Para lograr esta interaccion con el reloj le sugiero importar dentro del método work la librería `keyboard` y usar el método `keyboard.is_pressed(tecla)`. Este modulo keyboard pone a la máquina en escucha permanente mientras el reloj esta corriendo y, cuando usted apriete una tecla particular estando en la consola de Spyder, si dicha tecla coincide con la que haya definido en `tecla` usted puede, mediante un if clause, ejecutar un bloque de codigo especifico que se intercala en la ejecucion del loop infinito que este corriendo. En el caso de la figura, al oprimir la tecla "p" se imprime el tiempo del reloj y se lo compara con el tiempo real de la computadora. Si se presiona "q" el reloj se detiene. Si se presiona cualquier

otra tecla el reloj sigue de largo.

```
if keyboard.is_pressed("p"):
    self.printCurrentTime()
    print(datetime.datetime.now().time())
elif keyboard.is_pressed("q"):
    self.printCurrentTime()
    print(datetime.datetime.now().time())
    print("Stopping clock")
    break
```

Si el reloj está bien calibrado, al ejecutar el método work, deberia verse una salida por consola parecida a esta:

```
The current time is:  22:29:3
The current time is:  22:29:14
22:29:14.519482
The current time is:  22:29:22
22:29:22.261000
The current time is:  22:29:32
22:29:32.260376
The current time is:  22:29:37
22:29:37.243080
Stopping clock
```

2. Construya dos subclases de la clase `clock`, `Cronometro(clock)` y `Temporizador(clock)`. `Cronometro(clock)` utiliza la funcionalidad de `clock` pero adaptada para medir el tiempo transcurrido, capaz de medir tiempos parciales y tiempo total transcurrido desde el inicio. Al apretar la tecla "p" ahora el cronometro debe indicar el tiempo transcurrido desde la ultima vez que se apretó "p" (`"Lap time"`) y el tiempo total transcurrido (`"Total time"`). Los tiempos de cada vuelta (`"Lap time"`) se deben ir guardando en una lista `Times` atributo de instancia de clase del cronometro. Si se aprieta la tecla "q" el cronometro entiende que se termino la ultima vuelta, e imprime un reporte de los tiempos de cada vuelta y el tiempo total. Idealmente, para el cronometro nos interesara medir el tiempo con una precision de decima de segundo, pero no es imprescindible.

   `Temporizador(clock)` también hereda a clock pero tiene una funcionalidad distinta, está pensado para programar la duracion de una actividad. Se inicializa introduciendo una tupla con las horas minutos y segundos que deben transcurrir, y el tiempo ahora **debe correr hacia atrás**. Al apretar la tecla "p" ahora el temporizador debe indicar el tiempo transcurrido desde que se inicio el conteo del tiempo, y el tiempo restante. Si se aprieta la tecla "q" el temporizador se interrumpe por indicacion del usuario. Si el temporizador llega a (0,0,0) debe imprimir algo asi como `"Tiempo Cumplido"` y terminar la ejecucion.