PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2233 - PROGRAMACIÓN AVANZADA

Actividad 06

1^o **semestre 2018**
26 de abril

Introducción

Jeff Musk, CEO del *e-commerce* internacional *Programazon*, se fue de viaje al Caribe y volvió para encontrar que la base de datos de su sistema de ventas *online* había sido modificado por error. Lamentablemente, como CEO de la compañía, no programa *software* hace mucho tiempo; por esto, decidió llamar a los estudiantes de *Programación avanzada*, pidiendo ayuda. Para que Jeff Musk reconozca las funciones arregladas, deberás modificarlas utilizando **decoradores** procurando que todo vuelva a la normalidad.

Cuando todo iba bien con *Programazon*, el sistema pasaba todos los *tests* de `testing.py`. Tu misión al final de este desafío no es sólo pasar esos *tests*, sino también solucionar el resto de los problemas no verificados. No es necesario modificar los *tests*; basta con sólo pasarlos.

Funciones de *Programazon*

Estas funciones ya son parte de *Programazon* (definidas en el archivo `programazon.py`) y **no puedes modificarlas**, si no que sólo decorarlas.

- Para no perder ganancias debes registrar en un *log* de eventos cada vez que una función relacionada a ventas o movimientos de dinero es ejecutada. Estas funciones las realiza cada `Cliente`: `abonar` (aumenta saldo disponible), `agregar_al_carro` y `pagar`.
- El sistema no acepta abonos de dinero en montos si es que no son números enteros[1]. Por lo tanto, es necesario controlar que el `monto` que recibe la función `abonar` siempre sea del tipo `int`.
- Al momento de `pagar` es necesario verificar que el pago se esté haciendo a una `TiendaOnline` de *Programazon*. Además, el método `pagar` certifica al cliente mediante su contraseña, la que tiene que ser del tipo `str`.
- La base de datos de productos ha sido modificada. Cada línea de la base de datos debería ser de la forma `producto;precio`, sin embargo ha quedado de la forma `otcudorp;precio`. Actualmente al querer comprar un libro, el cliente debe buscar por "orbil", pero (casi) nadie busca así por un libro. Por lo tanto, el programa debe ser modificado para que la función `_procesar` de `TiendaOnline` retorne el nombre del producto de forma correcta.

Decoradores

Para asegurar que las funciones anteriores sean correctas, debes crear los decoradores indicados a continuación. Cada uno de ellos debe ser implementado en el archivo `decoradores.py`.

[1] Los centavos dejaron de existir en Chile.

- `registrar`: Cada vez que una función decorada por `registrar` sea ejecutada, se debe guardar en el archivo llamado `registro.txt` un registro con: el nombre de la función a la que se le aplica el decorador, los argumentos que esta recibió y el resultado retornado. Si el archivo ya existiera, el registro debe agregarse como una nueva línea.
- `verificar_tipos(tipo_1, ..., tipo_i, ..., tipo_n)`: Dada una función, se verifica que los argumentos recibidos sean de los tipos indicados en los parámetros del decorador, respectivamente. No es necesario revisar el tipo de `self`.
  - Si algún argumento no es del tipo especificado, se debe levantar una excepción del tipo `TypeError` con el mensaje `"El <argumento_i> no es del tipo <tipo_i>"`.
  - Si la cantidad de argumentos ingresados en el decorador no coincide con la cantidad de argumentos que recibe la función decorada, se debe levantar una excepción del tipo `TypeError` con el mensaje `"La cantidad de argumentos de la función no coincide con la cantidad de argumentos entregados"`.
- `invertir_string`: Corrige el comportamiento de la función `_procesar`. Una forma de hacerlo es invirtiendo el orden del primer elemento que retorna la función decorada. Por ejemplo, dado un elemento `ejemplo` retorna `olpmeje`.

Bonus

Los clientes de *Programazon* son cada día más exigentes, quieren que su compra sea rápida y fácil. El área de *Experiencia de clientes* te ha pedido registrar cada vez que la etapa de `pago` toma más de un tiempo determinado (por defecto 3 segundos). Para solucionar esto debes crear el decorador `temporizador`.

- `temporizador(tiempo_límite)`: Dada una función, imprime el tiempo, en segundos, que esta demora en ejecutarse. Si la función demora más de `tiempo_límite`, se imprime en consola una notificación con el mensaje `"Función excede tiempo esperado"`, además del nombre de la función a la que se le aplica el decorador, el tiempo límite a esperar, y el tiempo total que tomó su ejecución.

Notas importantes

- Sólo puedes modificar el código original de `programazon.py` decorando funciones. **Cualquier otro cambio en la estructura del sistema significará un 1.0 inmediato.**
- Recuerda que puedes utilizar más de un decorador por función.
- Dentro de tus decoradores puedes usar el decorador `wraps(nombre_función)` del módulo `functools` para que tu decorador sepa el nombre de la función que decora.
- No olvides crear el archivo `.gitignore` para no subir el *log* con el registro.
- En el bonus, puedes usar `time()` de la librería `time` para tomar el tiempo.

Requerimientos

- (4.5 pts) Decoradores funcionan correctamente.
  - (1.5 pts) Decorador `registrar` funciona correctamente.
  - (1.5 pts) Decorador `verificar_tipos` funciona correctamente.
  - (1.5 pts) Decorador `invertir_string` funciona correctamente.
- (1.5 pts) Decorar correctamente el programa.

- (1.0 pts) **Bonus**
  - (0.8 pts) Decorador `temporizador` funciona correctamente.
  - (0.2 pts) Decorar correctamente con `temporizador`.

Entrega

- **Lugar:** En su repositorio de Github en la **carpeta** `Actividades/AC06/`
- **Hora:** 16:30 horas.