75.40 Algoritmos y Programación I

Cátedra Pablo Guarna

2º Cuatrimestre 2019 - Trabajo Práctico Grupal

Revisión v1.0

Agregando funcionalidad

El trabajo práctico se basa en la versión 1.0 entregada recientemente y agrega algunas funcionalidades, centrándose en la persistencia de datos.

Funcionalidades a implementar

1) Carga de información predefinida:

   La función de carga de información predefinida debe ser extraída (y eliminada) del programa principal, y agregada en un programa separado llamado info_predefinida.py donde se guarde esa información en uno o más archivos Pickle.

   Dentro del programa principal se debe modificar la función de carga predefinida por una importación de los archivos Pickle generados desde info_predefinida.py hacia las estructuras de memoria correspondientes. En caso de existir información en memoria, el programa debe advertir sobre esta condición y preguntar si se desea sobreescribir toda la información actual con la información almacenada en los archivos.

2) Guardado y recupero de información

   Al salir del programa toda la información actual deberá ser guardada en archivos con formato a definir por el grupo. Al ejecutar nuevamente el programa se deberá detectar si existe información de una corrida previa y se la cargará automáticamente para retomar las operaciones desde donde se había dejado la última vez.

3) Modificación de las entidades

   - En cada restaurante se agrega un radio en km, que es el radio de entrega válido.
   - En cada Rappitendero se agrega un campo con la distancia recorrida en km, que se incrementa con cada pedido entregado. Tener en cuenta el

viaje desde la última posición hasta el restaurante y desde el restaurante hasta el domicilio de entrega.

4) Pedido manual:

Se debe implementar la siguiente secuencia de pasos:

a) Se pide al Cliente iniciar sesión ingresando nombre y contraseña (validar que exista primero el nombre y luego la contraseña)
b) Enumerar solo los restaurantes con radio de entrega que abarquen el domicilio del cliente. Calcular la distancia del restaurante al domicilio usando la fórmula de Haversine. Indicar en el código la fuente de esta fórmula. Es también válido utilizar bibliotecas de Python para realizar este cálculo.
c) Permitir al usuario elegir un restaurante y luego enumerar los platos que este ofrece.
d) Permitir al usuario elegir qué plato quiere y cuantas unidades, validando que la cantidad pedida sea positiva.
e) Preguntarle al usuario si desea algo extra o si ya terminó con su pedido y repetir el ciclo, volviendo a enumerar los platos disponibles.
f) Cuando el usuario terminó, generar el pedido y asignarlo al Rappitendero que más cerca se encuentre del Restaurante, usando la misma fórmula de distancia antes mencionada.
g) Asignar la latitud y longitud del restaurante al Rappitendero.
h) Asumir que el Rappitendero se mueve a una velocidad de 15 km/h. Calcular el tiempo de entrega e informarlo.
i) Actualizar la posición del Rappitendero al domicilio del Cliente. Ahora que tenemos más adopción de la aplicación, el cliente solo gana un 5% del pedido en Rappicréditos para pedidos de hasta $200, 10% en pedidos entre $200 y $1000, y 15% en pedidos de mayor valor. El Rappitendero gana el equivalente al 5% del pedido como propina.
j) Se cierra la sesión del Cliente

5) Simulación de pedidos:

a) Se pide al usuario ingresar una cantidad de simulaciones entre 1 y 100, un máximo de platos por pedido.
b) Se toma un Cliente al azar.
c) Se elige un Restaurante al azar con radio de entrega abarcando al domicilio del cliente. En caso de no haber ninguno, se contabiliza la simulación y se empieza de nuevo desde el punto b.
d) Se elige un número de platos para el pedido entre 1 y el máximo establecido o la cantidad de platos del restaurante.
e) Se eligen cantidades al azar para cada plato.
f) Se busca al Rappitendero más próximo al domicilio.
g) Se informa cuánto tarda el Rappitendero, sabiendo que la velocidad promedio es 15 km/h.
h) Se hace la entrega y se actualiza la posición del Rappitendero a la posición del domicilio del cliente.
i) Se actualizan los Rappicréditos del cliente y se asigna la ganancia al Rappitendero siguiendo las mismas reglas que para el pedido manual.

6) Informes

Se agrega un cuarto informe que debe generar un archivo rappitenderos.csv con el nombre del Rappitendero y la distancia recorrida, sin ningún orden en particular. El archivo se sobreescribe si ya existía, sin necesidad de advertir al usuario.

7) Punto extra: Pruebas unitarias

Implementar pruebas unitarias para la función que calcula los Rappicréditos ganados por el usuario y la propina ganada por el Rappitendero (ver punto i del pedido manual) usando el módulo unittest explicado en clase.