Evaluación Final Módulo 1 Data Analytics

Instrucciones:

- Antes de empezar, hay que crear un nuevo repositorio desde GitHub Classroom usando este enlace. Una vez creado, hay que clonar en nuestro ordenador y en la carpeta creada empezaremos a trabajar en el ejercicio.
- Esta evaluación consta de una serie de preguntas que evalúan tu comprensión y habilidades en relación con funciones, clases y regex.
- Puedes usar recursos externos, incluyendo internet y materiales de referencia o tus propias notas.
- Completa los ejercicios en un jupyter notebook.

Ejercicio

A lo largo de esta evaluación tendrás que crear una clase llamada `TiendaOnline` que cumpla los siguientes requisitos:

- La clase `TiendaOnline` debe tener los siguientes atributos:
    1. `inventario` (lista de diccionarios): Un atributo para almacenar los productos en el inventario. Cada producto debe ser representado como un diccionario con las siguientes claves: `'nombre'`, `'precio'`, y `'cantidad'`. Al principio deberá ser una lista vacía. Ejemplo de como debería ser:

        ```
        [{'nombre': 'Camisa', 'precio': 20, 'cantidad': 40},
        {'nombre': 'Pantalón', 'precio': 30, 'cantidad': 30}]
        ```

    2. `ventas_totales` (float): Un atributo para llevar un registro de las ventas totales de la tienda. Inicializado con valor 0.

La clase `TiendaOnline` debe tener los siguientes métodos:

1. `agregar_producto(self, nombre, precio, cantidad)`: Este método agrega un producto al inventario o actualiza su cantidad si ya existe. Debe recibir el nombre, precio y cantidad del producto como parámetros.
    - Itera a través del inventario y compara los nombres de los productos con el nombre proporcionado.
    - Si el producto ya existe, actualiza la cantidad.
    - Si no existe, agrega un nuevo producto al inventario.
2. `ver_inventario(self)`: Muestra el inventario de productos con sus detalles.
    - Utiliza un bucle `for` para recorrer el inventario.

- Imprime los detalles (nombre, precio, cantidad) de cada producto.
- Debería verse:

```
Nombre: Camisa, Precio: $20, Cantidad: 50
Nombre: Pantalón, Precio: $30, Cantidad: 30
Nombre: Zapatos, Precio: $50, Cantidad: 40
Nombre: Camisa, Precio: $20, Cantidad: 50
```

3. `buscar_producto(self, nombre)`: Busca un producto en el inventario por nombre y muestra sus detalles si se encuentra. Debe recibir el nombre del producto como parámetro.
   - Utiliza un bucle `for` para recorrer el inventario.
   - Compara los nombres de los productos con el nombre proporcionado.
   - Si se encuentra el producto, imprime sus detalles.
   - Debe mostrar:

```
Nombre: Camisa, Precio: $20, Cantidad: 40
```

4. `actualizar_stock(self, nombre, cantidad)`: Actualiza el stock de un producto en el inventario. Debe recibir el nombre del producto y la cantidad a agregar o quitar como parámetros.
   - Utiliza un bucle `for` para recorrer el inventario.
   - Busca el producto por nombre.
   - Actualiza la cantidad según la entrada del usuario.
   - Si el producto no esta en el inventario muestra un mensaje indicándolo.
5. `eliminar_producto(self, nombre)`: Elimina un producto del inventario por nombre. Debe recibir el nombre del producto como parámetro.
   - Utiliza un bucle `for` para recorrer el inventario.
   - Busca el producto por nombre.
   - Elimina el producto del inventario si existe.
   - Si el producto no esta en el inventario muestra un mensaje indicándolo.
6. `calcular_valor_inventario(self)`: Calcula y muestra el valor total del inventario.
   - Utiliza un bucle for para calcular el valor total del inventario.
   - Itera a través del inventario y suma el valor de cada producto (precio x cantidad). Es decir, calcula el valor total del inventario. Ejemplo:

```
# si tenemos 5 camisas que valen 5 euros
# y 10 calcetines que valen 1 euro
# este método te tiene que devolver: 35 euros
```

```
valor_camisas = 5 * 5
valor_calcetines = 10 * 1

valor_camisas + valor_calcetines = 35
```

Instrucciones Adicionales:

- Debes crear instancias de la clase `TiendaOnline` y probar cada uno de los métodos para demostrar que funcionan correctamente.
- Para las entrevistas técnicas, asegúrate de tener preparado un código que permita añadir una cantidad significativa de productos al inventario. Esto facilitará comprobar si los métodos para visualizar el inventario, buscar productos, y otras funcionalidades están funcionando correctamente."

BONUS

Dedica tiempo a esta sección solo si has terminado la anterior, entiendes bien lo que has realizado y te sientes lo suficientemente segura para avanzar. Recuerda que el objetivo principal no es completarlo todo, sino disfrutar del proceso y consolidar tu aprendizaje.

Para realizar esta parte, debes añadir los siguientes atributos a la clase `TiendaOnline`:

- `clientes` (diccionario): Un atributo para llevar un registro de los clientes de la tienda. Cada cliente debe ser representado como un diccionario con las siguientes claves: `'nombre'` y `'email'`. Al inicio deberá ser un diccionario vacío. Además, cada cliente debe tener un historial de compras. Deberá parecerse a:

```
{'Cliente1': {'email': 'cliente1@email.com', 'compras': []},
'Cliente2': {'email': 'cliente2@email.com', 'compras': []}}
```

Y posteriormente, debes añadir los siguientes métodos a la clase `TiendaOnline`:

1. `agregar_cliente(self, nombre, email)`: Agrega un nuevo cliente al registro de clientes. Debe recibir el nombre y el correo electrónico del cliente como parámetros.
    - Agrega un cliente al diccionario de clientes con su nombre y correo electrónico.
2. `ver_clientes(self)`: Muestra la lista de clientes registrados con sus nombres y correos electrónicos.
    - Utiliza un bucle `for` para recorrer la base de datos de clientes.
    - Imprime los detalles de cada cliente (nombre y correo electrónico).
3. `realizar_compra(self)`: Permite a un cliente realizar una compra seleccionando productos del inventario. Debe interactuar con el cliente para seleccionar productos y calcular el costo total de la compra.
    - Utiliza un bucle `while` para permitir al cliente realizar múltiples compras.
    - Muestra el inventario y solicita al cliente ingresar el nombre del producto que desea comprar.
    - Registra los productos seleccionados en un carrito y actualiza el inventario.

- Calcula el costo total de la compra.

4. `procesar_pago(self)`: Procesa el pago de una compra, calcula el cambio y muestra un mensaje de confirmación.
    - Utiliza un bloque `try...except` para manejar excepciones.
    - Solicita al cliente ingresar la cantidad total y la cantidad de pago usando un input.
    - Calcula el cambio y muestra un mensaje de pago exitoso o un error en caso de monto insuficiente.

5. `registrar_compra(self, nombre_cliente, carrito)`: Registra una compra para un cliente, actualiza las ventas totales y agrega la compra al historial del cliente. Debe recibir el nombre del cliente y el carrito de compras como parámetros.
    - Busca al cliente en el diccionario de clientes.
    - Si el cliente no esta en el diccionario de clientes, muestra que no se puede realizar la acción por que el cliente no está en el diccionario.
    - Calcula el total de la compra y registra la compra, incluyendo los productos y el total.
    - Ejemplo:

```
carrito_cliente1 = {"Camisa": {"precio": 20, "cantidad": 3}}
tienda.registrar_compra("Cliente1", carrito_cliente1)
```

6. `ver_compras_cliente(self, nombre_cliente)`: Muestra el historial de compras de un cliente. Debe recibir el nombre del cliente como parámetro.
    - Busca al cliente en el diccionario de clientes.
    - Muestra las compras realizadas por el cliente, incluyendo detalles de productos y totales.

7. `calcular_ventas_totales(self)`: Muestra las ventas totales de la tienda.
    - Suma los totales de todas las compras realizadas y muestra el total de ventas totales en la tienda.

Normas

Este ejercicio está pensado para que lo realices de forma individual en clase, pero podrás consultar tus dudas con la profesora y tus compañeras si lo consideras necesario. Ellas no te darán directamente la solución de tu duda, pero sí pistas para poder solucionarla. Aún facilitando la comunicación entre compañeras, durante la prueba no debes copiar código de otra persona ni acceder a su portátil. Confiamos en tu responsabilidad.

La evaluación es una buena oportunidad para conocer cómo estás progresando, saber qué temas debes reforzar durante las siguientes semanas y cuáles dominas. Te recomendamos que te sientas cómoda con el ejercicio que entregues y no envíes cosas copiadas que no entiendas.

Si detectamos que has entregado código que no es tuyo, no entiendes y no lo puedes defender, pasarás directamente a la re-evaluación del módulo. Tu objetivo no debería ser pasar la evaluación sino convertirte

en analista de datos, y esto debes tenerlo claro en todo momento.

Una vez entregado el ejercicio realizarás una revisión del mismo con la profesora (20 minutos), que se asemejará a una entrevista técnica: te pedirá que expliques las decisiones tomadas para realizarlo.

Es una oportunidad para practicar la dinámica de una entrevista técnica donde te van a proponer cambios sobre tu código que no conoces a priori. Si evitas que otras compañeras te den pistas sobre la dinámica de feedback, podrás aprovecharlo como una práctica y pasar los nervios con la profesora en lugar de en tu primera entrevista de trabajo.

Al final tendrás un feedback sobre aspectos a destacar y a mejorar en tu ejercicio, y sabrás qué objetivos de aprendizaje has supera

Criterios de evaluación

Vamos a listar los criterios de evaluación de este ejercicio. Si no superas al menos el 80% de estos criterios o no has superado algún criterio clave (marcados con *) te pediremos que realices una re-evaluación con el fin de que termines el curso mejor preparada y enfrentes tu primera experiencia profesional con más seguridad. En caso contrario, estás aprendiendo al ritmo que hemos pautado para poder afrontar los conocimientos del siguiente módulo.

Python Básico

- Condicionales (if, elif, else)*
- Bucles (for y while). Importante entender cómo iterar por una lista, o un diccionario. *
- Funciones (creación de funciones con parámetros, parámetros por defecto)*
- Conceptos básicos de regex
- Clases
- Diferencias entre los distintos tipos de datos en python (strings, listas, tuplas, sets y diccionarios)

Otros criterios a tener en cuenta

- Usar inglés para nombres de variables, funciones, clases, mensajes de commit, nombres de ficheros.
- El repositorio de GitHub debe tener README explicando muy brevemente cómo arrancar el proyecto.

¡Al turrón!