Práctica 12:
Entrada y Salida con Ficheros

Índice

Objetivos

Objetivos

- Principalmente
  - Conocer las funciones básicas lectura y escritura en ficheros.
- Además
  - Estructuras de datos compuestas.
  - Ordenar, filtrar y agregar elementos de listas.

Aplicación de gestión de clientes

Desarrollar una aplicación de gestión de clientes de un servicio de subscripción.

La aplicación guardará y leerá la información de los clientes de ficheros.

Aplicación de gestión de clientes

El fichero de clientes tiene campos separados por ‘;’ y contiene, por orden:

- Nombre.
- DNI (8 dígitos y una letra).
- Plan de subscripción (9.90 o 14.90).
- Número total de meses suscrito.
- Suscripción activa actualmente (Y / N).

```
María López;45678901D;14.90;3;Y
Juan Pérez;12345678A;9.90;14;Y
Ana García;23456789B;14.90;5;Y
Luis Rodríguez;34567890C;9.90;26;N
Carlos Sánchez;56789012E;14.90;12;Y
Elena Martínez;67890123F;14.90;8;N
```

Aplicación de gestión de clientes

Debe tener las siguientes funcionalidades:

- Añadir nuevos clientes.
- Mostrar el listado de clientes ordenado.
- Actualizar el estado de la suscripción de clientes existentes.
- Generar informes para cada cliente.
- Filtrar, agregar y ordenar

Aplicación de gestión de clientes

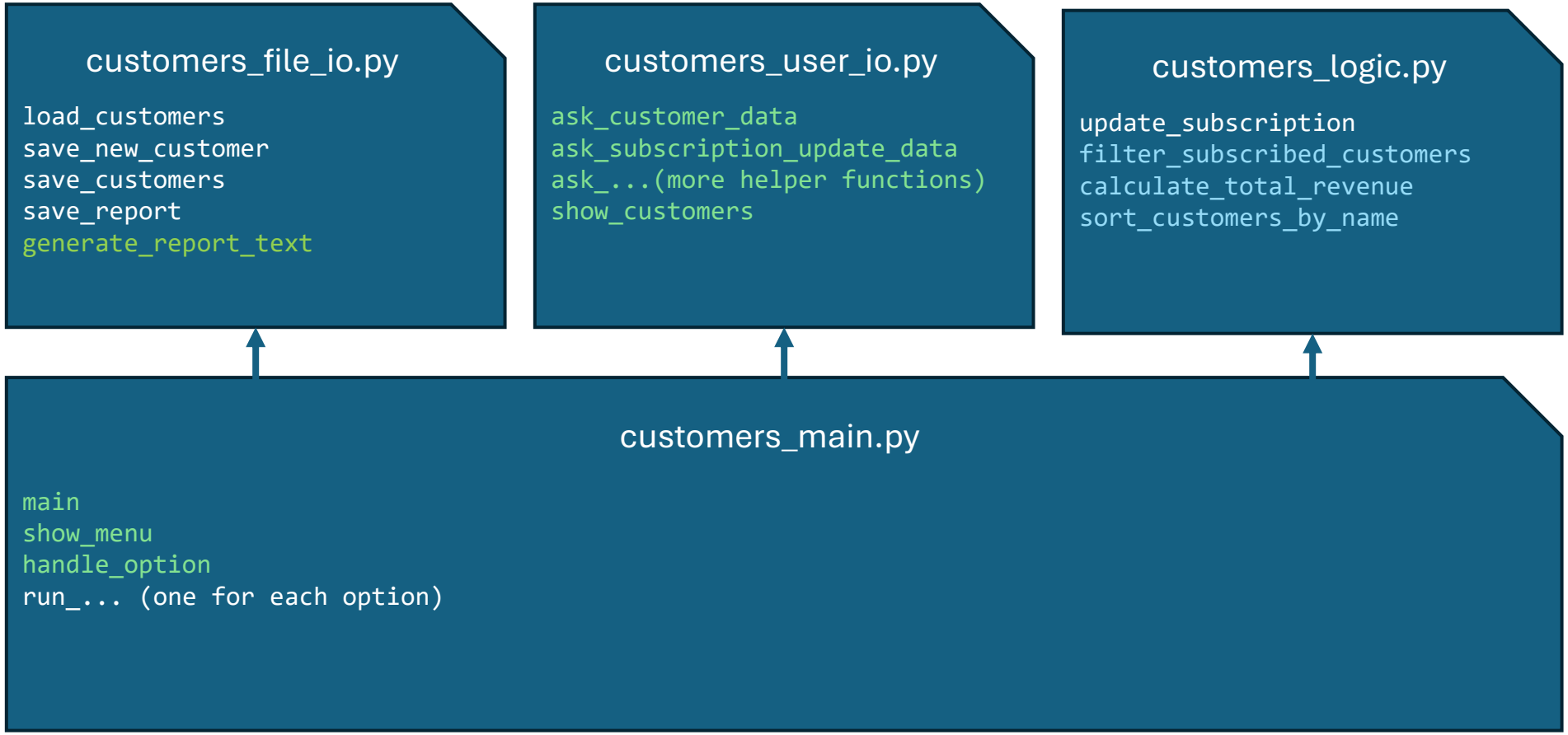
Prueba la aplicación:

- Está disponible en el campus virtual

Organización del código

Organización del código

*Funciones ya implementadas
*Funciones a implementar en clase
*Funciones a implementar en casa



Leer clientes

Leer clientes

En el módulo `customers_file_io` implementa la función `load_clients`:

- Recibe el nombre del fichero y un carácter separador.
- Devuelve una lista con los clientes.
  - Cada cliente es una tupla con:
    - nombre(str), dni(str), tipo de suscripción(float), meses suscrito(int), suscripción activa (bool).

Leer clientes

En el módulo main, implementa la función `run_read_clients`:

- No recibe ningún parámetro.
- Dentro de la función, se usa las constantes `CLIENTS_FILE` y `SEPARATOR` para llamar a la función load_clients.
- La lista obtenida se pasa a `customers_user_io.show_clients` para mostrarla en pantalla.
- Ejecuta el menú y comprueba que funciona (el código del menú ya está implementado).

Añadir clientes

Añadir clientes

Implementa `customers_file_io.save_new_customer`

- Recibe la ruta del fichero, el carácter a usar como separador en el fichero csv, y una tupla con los datos del cliente.
- Puedes crear una función auxiliar para crear el string a partir de la tupla del cliente y el separador.
- Para actualizar el fichero:
  - Leer y escribir con `open` y `write`
  - 'a' para escribir al final del fichero
  - Comprueba que funciona en el main y abre el fichero una vez guardado.

Modificar clientes

Modificar cliente

Crea la función `update_subscription` en `customers_logic`

- Recibe la lista con todos los clientes (tuplas), el dni del subscriptor a actualizar, y un booleano representando si su subscripción se activa (`True`) o desactiva (`False`).
- Devuelve una nueva lista de clientes con el estado de la subscripción del cliente actualizada.

Modificar cliente

Crea la función `update_customers` en el módulo `customers_file_io`.

- Recibe el nombre del fichero y una lista de clientes.
- Reescribe todo el contenido del fichero con la nueva lista recibida.

(Si encuentras que esta función repite código que ya has hecho, es una buena oportunidad de extraer ese código en una función)

Modificar cliente

Crea la función `run_update_subscription` en el módulo main:

- Obtiene la lista de clientes leyendo el fichero de clientes.
- Solicita el dni y la subscripción usando la función `customers_user_io.ask_subscription_update`.
- Actualiza la lista usando la función `customers_logic.update_subscription`.
- Guarda el fichero usando

Modificar cliente

Doctests:

- Para realizar pruebas con listas complejas, podemos declarar variables globales en el main que sean accedidas desde los tests.
- Esto evita tener que escribirlo cada vez en los tests.
- El fichero customers_logic ya dispone del código necesario.

Modificar cliente

```python
if __name__ == "__main__":
    import doctest
    # Define test data only when running doctests
    # This data represents: (name, dni, monthly_subscription, months_subscribed, is_active)
    test_customers = [
        ("Juan Pérez", "12345678A", 10.0, 12, True),
        ("María López", "23456789B", 15.0, 24, False),
        ("Carlos García", "34567890C", 12.5, 6, True),
        ("Ana Fernández", "45678901D", 20.0, 18, True),
        ("Luis Martínez", "56789012E", 8.0, 3, False),
    ]
    # Inject the test data into doctest namespace without creating global variables
    doctest.testmod(extraglobs={"customers": test_customers}, verbose=True)
```

Generar informes

Generar informes

Se desea generar un fichero .txt para cada cliente:

- El nombre del fichero seguirá el patrón `<DNI>_<nombre>.txt`

- Ejemplo:
  - El tiempo se muestra en años y meses.

```
Report for Lucía López, 22334455P
    -Time subscribed: 2 years, 10 months
    -Active subscription : Yes
    -Total expenses: 506.6 €
```

Generar informes

En `customers_file_io` crea la función `save_report`:

- Recibe un cliente.
- Genera el fichero para dicho cliente.

Puedes usar la función `generate_report_text` para generar el texto del fichero.

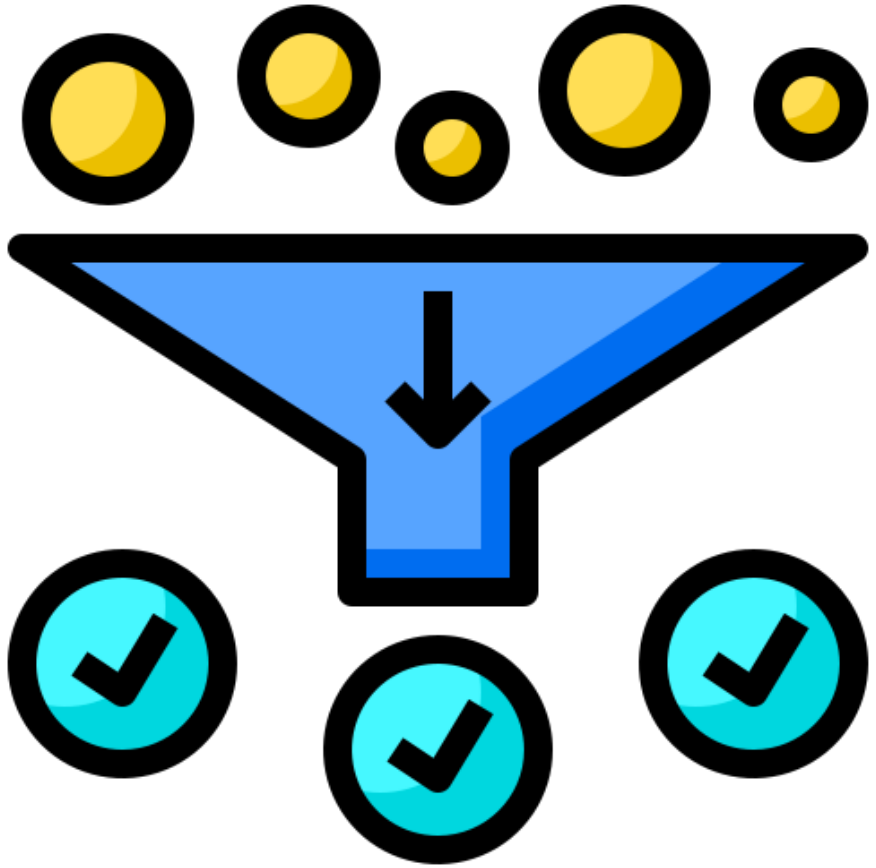
Generar informes

En el `customers_main` crea la función `run_generate_customer_reports`:

- Lee la lista de clientes.
- Para cada cliente:
  - Escribe el informe.

Filtrar, agregar y ordenar

Trabajo para casa: Filtrar clientes



Trabajo para casa: Filtrar clientes

En el módulo `customers_logic` crea la función `filter_subscribed_clients`:

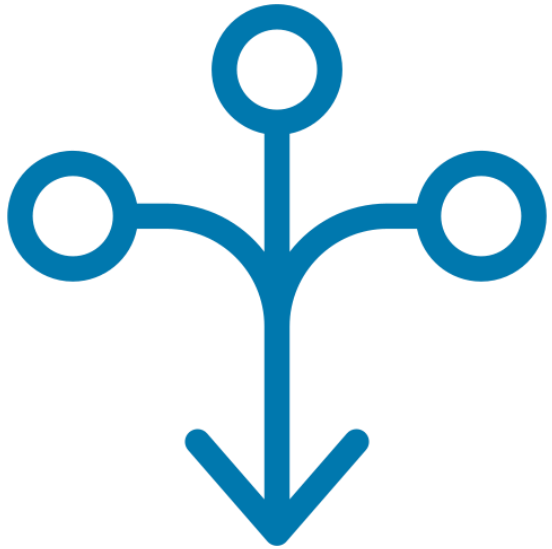
- Recibe una lista de clientes.
- Devuelve una nueva lista solo con aquellos que tienen una subscripción activa.

Trabajo para casa: Filtrar clientes

En el módulo main, implementa `run_filter_suscribed_clients`:

- Lee el fichero de clientes.
- Crea una lista filtrada.
- La muestra en consola.

Trabajo para casa: Agregación



1 result!

Trabajo para casa: Agregación

La agregación es un proceso en el que a partir de una lista de valores se calcula un solo resultado.

En este caso, vamos a recorrer todos los clientes para calcular la recaudación total de todos ellos, teniendo en cuenta el tipo de suscripción que tienen y el número de meses suscritos.

Trabajo para casa: Agregación

En el módulo `customers_logic` crea la función:

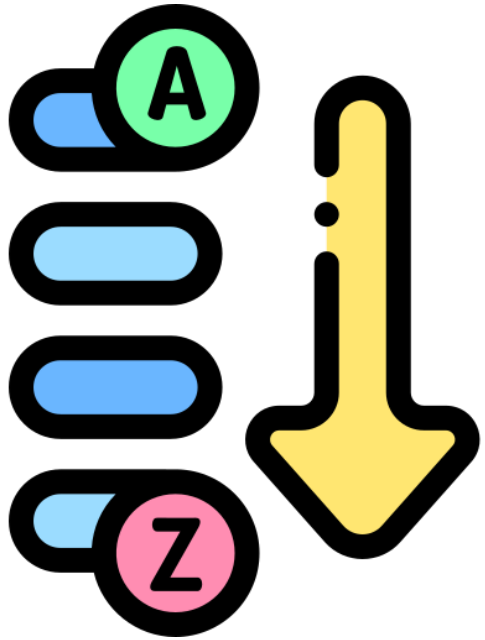
- `calculate_total_revenue`:
  - Recibe una lista de clientes y calcula la suma de los gastos totales de todos los clientes.

Trabajo para casa: Agregación

En el módulo main implementa la función `run_calculate_total_revenue`:

- Lee el fichero de clientes.
- Calcula el gasto total de todos los clientes.
- Lo muestra por consola.

Trabajo para casa: Ordenar clientes



Trabajo para casa: Ordenar clientes

En el módulo `customers_logic`, implementa la función `sort_customers_by_name`.

- Recibe una lista de clientes.
- Devuelve una nueva lista ordenada por orden alfabético según su nombre.

Trabajo para casa: Ordenar clientes

Comparación de strings:

- Al usar operadores relacionales (<, <=, >=, >) con strings, las compara alfabéticamente:
  - `“Álvaro” < “Alberto” == False`

Trabajo para casa: Ordenar clientes

Algoritmo de ordenación por inserción directa

Crea una lista nueva vacía para ordenar los clientes.

- Recorre la lista original de clientes (desordenada).
    - Por cada cliente desordenado, recorre la lista ordenada.
        - -Crea una variable ‘ordenado’ con valor False
        - -Si el nombre del cliente desordenado es menor alfabéticamente que el cliente ordenado que visitas
            - -Insértalo en esa posición.
            - -Cambia ‘ordenado’ a True.
            - -Sal del bucle.

    - -Si tras recorrer toda la lista de ordenados aún no se ha insertado el elemento, añadir al final.

- Retornar la nueva lista de clientes ordenada

Trabajo para casa: Ordenar clientes

Implementa `run_show_customers_sorted`, utilizando las funciones:

- Lee el fichero de clientes.
- Obtén la lista ordenada de clientes.
- Muestra la lista ordenada.

Trabajo para casa: Salida a fichero

Una vez implementadas las 3 funcionalidades previas.

En las 3 funciones '`run...`', pregunta al usuario si quieren guardar el resultado en un fichero. En ese caso, además de imprimir en consola deben crear un fichero con el resultado (con distinto nombre al fichero de clientes).

Probablemente encuentres oportunidades para reutilizar y refactorizar código que ya tienes.