Tarea 2: Andrés Calderón Guardia

Primero para el archivo ***ingredientes.py*** creé la estructura de un ingrediente añadiendo las componentes de nombre, precio (por 100gr) y el identificador, que lo separé en 2 para tener un mejor orden, una letra (categoria) que indica el tipo de ingrediente que es (pan, queso, proteína, verdura o salsa) y un número (index) para ordenar los ingredientes de un mismo tipo. Con esto hecho hice los ingredientes dados en la tarea dentro del archivo y también una lista que los agrupa según su tipo. Tras esto hice las funciones *esIngrediente* para los asserts y *precioId* que devuelve el precio de un ingrediente según su identificador (aunque no terminé usando esta última pero la hice porque se pedía en la tarea).

Luego empecé con el archivo ***panway.py*** en donde creé la estructura de un sándwich en base a lo pedido, 5 componentes cada una siendo una lista de ingredientes e hice los 4 sándwiches predefinidos de la tarea además de una lista con los gramos correspondientes de cada ingrediente y establecí la función correspondiente *esSandwich* para usar también en futuros asserts.

Finalmente empecé con el archivo ***T2.py*** en donde parto imprimiendo en pantalla la bienvenida a la tienda con el input inicial para revisar si se desea editar la lista de ingredientes (creé la función *errorInicial* con el fin de asegurarme que este input se haga correctamente), para luego crear la función *tienda* que maneja toda esta parte, para esto le integré un parámetro llamado *estado* siendo un string que usé para diferenciar si debía agregar, quitar o finalizar esta parte del programa interactivo, y 5 parámetros por omisión correspondientes a las 5 listas de ingredientes clasificadas según el tipo para ir actualizando en cada iteración según la acción que se realizara:

1. Para la parte de agregar pide a través de inputs el nombre, tipo de ingrediente y el precio por 100gr para así crear el ingrediente y pasarlo a la función *agregarIngrediente* (toma una lista de ingredientes y le añade un ingrediente creado con los parámetros que se recibieron e integrándole el mayor índice disponible automáticamente), para luego pedirle al usuario nuevamente si quiere seguir editando los ingredientes de modo que tras esto se utiliza la función *parametroLista* para actualizar correctamente el parámetro por omisión correspondiente según el ingrediente agregado sin alterar las demás listas.
2. Ahora en quitar se le pide al usuario que indique a cuál categoría desea quitarle un ingrediente, tras esto se verifica si la lista deseada tiene al menos un ingrediente que quitar, si no es el caso se vuelve a llamar a la función hasta que se ingrese una categoría válida, y en caso contrario usa la función *imprimirListaIngredientes* (imprime en pantalla todos los ingredientes de una lista y su precio respectivo usando la función *textoIngrediente* que recibe un único ingrediente y devuelve un string con la información a imprimir en la línea) para que el usuario indique cual ingrediente eliminar. Luego se crea una nueva lista de forma similar que en el apartado de agregar, primero se usa la función *parametroPorOmision* que recibe la categoría y las listas de la función *tienda* para devolver la lista de ingredientes correspondiente, luego de recibir esta lista usa la función *quitarIngrediente* que precisamente le quita un ingrediente a la lista dada y por último se usa la función *reindexarLista* que sirve para que los index de los ingredientes de la lista queden ordenados y lo más bajos posibles, ya que al quitar un ingrediente puede ser uno del medio

arruinando el orden, y luego se hace lo mismo que al final de la parte en que se agrega un ingrediente.

3. Cuando se llega al estado 'fin' se pide por input la cantidad de sándwiches y se le ingresa a la segunda gran función del programa interactivo *programaListaSandwiches*.

En esta función se gestiona la lista de sándwiches pedidos usando 5 parámetros que son las listas de *tienda* y otros 5 parámetros por omisión, el *tipo* que indica si se quiere un sándwich predefinido, personalizado o si hay que preguntar por el siguiente sándwich, *cantidad* obtenida de *tienda*, *precio* que maneja el precio total del pedido, *listaSandwiches* que enlista los sándwiches pedidos y *listaPrecios* que guarda los precios de cada sándwich:

1. En el estado *pregunta* se pide un nuevo input indicando sándwich predefinido o personalizado.
2. En predefinido se divide en 2 casos, cuando no hay ingredientes para poder armar alguno de los 4 sándwiches predefinidos (que en cuyo caso pide que se pida uno personalizado) y cuando si se puede pedir alguno en donde parte imprimiendo la lista de los sándwiches predefinidos y se pide un input (como este input es un número usa la función *sandwichBuscado* para identificar cual sándwich se pidió), luego imprime una línea diciendo que fue añadido al pedido y el coste de este hasta el momento usando la función *infoPredefinido*, tras esto se calcula el nuevo precio del pedido con la función *precioSandwich* (calcula el precio de un sándwich usando la lista de los gramos de cada ingrediente que le compone (a su vez esta usa la función *precioLista* que recibe una lista de ingredientes y los gramos de cada uno para calcular el precio total (y a su vez usa la función *precioIngrediente* para calcular el precio de cada ingrediente individualmente según la cantidad de gramos deseada)) que para este caso se usa la función *gramosBuscados* para identificar la lista de gramos del sándwich predefinido correspondiente) y entonces la función se llama a sí misma actualizando los parámetros correspondientes. Si se diese el caso de que no se puede armar el sándwich predefinido elegido, usando la función *errorPredefinido* para identificar que se llegó a este caso y que la misma usa las funciones *existeR1*, *existeR2*, *existeR3* y *existeR4* para lograrlo (verifican que se puede armar el sándwich R1, R2, R3 y R4 respectivamente dadas 5 listas de cada tipo de ingrediente), entonces se pide que ingrese un sándwich predefinido que si se pueda armar con los ingredientes disponibles en la tienda.
3. En personalizado se llama a la función *armarSandwich* para pedir el sándwich personalizado.
4. Y luego hay un if al principio para verificar si el pedido ha terminado para así usar la función *imprimirSandwiches* que muestra un resumen final de los sándwiches junto al precio total.

Por último queda la tercera gran función del programa que es *armarSandwich* la que se encarga de los inputs e interfaz para crear un sándwich personalizado, posee 9 parámetros, las 5 listas de tienda, *cantidad*, *listaPrecios*, *precio* y *listaSandwichesPedidos*, y 11 parámetros por omisión, *estado* que sirve para identificar que hacer dentro de la función, *precio* para ir almacenando el precio del pedido, *largoI* que es usado solamente en el input inicial con la función *errorEntre* que explico más adelante, 5 listas correspondientes con los 5 tipos de ingredientes para tenerlos al momento de crear el sándwich, *sandwFinal* para ir guardando los ingredientes del sándwich, *listaFinal* que guarda los gramos de cada

ingrediente del sándwich y *gramosSalsaFinal* que es utilizado para un caso particular al terminar de pedir salsas.

Tras esto le asigno a una variable el valor de *listaFinal*, luego la siguiente acción depende del parámetro *estado*, que es bastante confusa su lógica pero básicamente hay 2 grandes partes dentro de esta función, la que gestiona que hay que hacer a continuación y la que hace propiamente tal estas cosas, la primera logra que el usuario pueda agregar uno o varios ingredientes de un mismo tipo o no agregar ingredientes de un tipo si así lo desea el usuario e identificar cuando se terminó de añadir ingredientes para así crear el sándwich, la segunda sección se subdivide en 6, cada una por tipo de ingrediente que en su generalidad primero indican si desea pedir el ingrediente, seguido de la lista de los mismos, a lo cual si la respuesta es fin entonces se pasa al siguiente tipo de ingrediente y si se escoge un ingrediente pide cuantos gramos desea, imprime el costo del ingrediente que pidió y cuánto cuesta el sándwich hasta el momento para asignar todos estos valores a *sandwFinal*, esto se hace en cada subdivisión (excepto en la final porque es un caso final particular) pues el usuario podría pedir un único tipo de ingrediente el que no se sabe cuál sería, por lo tanto hay que asignar este valor siempre en cada uno, y tras esto vuelve a preguntar si quiere añadir un ingrediente del mismo tipo de nuevo para que la función vuelva a llamarse a sí misma recursivamente haciendo nuevamente el recorrido anteriormente mencionado (en medio hay ciertos returns para finalizar llamadas recursivas innecesarias), que acaba finalmente cuando se ingresa fin estando en las salsas (pues es el último tipo de ingrediente a pedir para el sándwich) para así mostrar información relevante del sándwich y también identificar si es que el pedido terminó para mostrar un resumen de la orden final.

Finalmente se añade en la última línea del código *tienda(editar)* para que el archivo sea un programa (con editar siendo el input inicial para saber si el usuario quiere editar la lista de ingredientes.

Como añadido hice funciones para mejorar la experiencia del usuario tales como *errorEntre*, *errorCantidad*, *errorTipoSandwich* y *errorInput* que hacen básicamente lo mismo que *errorInicio*, utilizadas en la mayoría de inputs del archivo, y varias funciones útiles que utilice en la tarea como lo son *invertirLista* que básicamente hace lo que dice su nombre y usé a conveniencia en variadas ocasiones, *categoriaBuscada* que según el número entregado devuelve la letra correspondiente, *buscarIndex* y *buscarIngrediente* que toman un índex y un ingrediente respectivamente y lo buscan dentro de la lista dada, y *esEntero* que usa la función isdigit() y se que no hemos visto pero usé para poder verificar que se pueda castear un input a int y utilicé pues no encontré forma de lograrlo con lo enseñado hasta el momento (como mucho logré diferenciar si se podía pasar a entero cuando se entregaba un racional pero no si se entregaba cualquier otro texto, igualmente asumo lo que implique usar esta función pero quería utilizarla para que no pudiera generarse absolutamente ningún error durante la ejecución del archivo además de que es algo extra a los requisitos de la tarea ya que no pedían cubrir estos casos).