Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2022-1

9 de Junio de 2022
Actividad Formativa

Actividad Formativa 4

Iterables e Iteradores

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AF4/`
- **Hora del *push*:** 16:40

Importante: Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

Introducción

Se viene MasterDCChef Edición 2022-1! Los ~~chefs~~ ayudantes se han estado preparando por meses para hacer los mejores platos y finalmente llegó la semana de la competencia! Por eso han decidido ir a comprar los ingredientes para sus platos al supermercado. Eso si, al revisar el mapa del lugar se encontraron con un supermercado muy confuso! Lleno de iteradores, generadores y funciones built-in de python. Tu tarea será guiar a los ayudantes para que logren realizar bien sus compras y puedan lucir sus dotes culinarios en la competencia.

Flujo del programa

El programa consiste en una simulación del supermercado que está auspiciando a MasterDCChef. Al ejecutar el archivo `main.py`, cada ayudante recorrerá los pasillos del supermercado según los ingredientes que debe comprar para poder cocinar sus platos. El programa termina una vez que todos los ayudantes hayan pasado por la caja y comprado (o no, si no les alcanza el dinero) sus productos.

Deberás definir un generador y un iterador que ayuden a los competidores a recorrer el supermercado y conocer la cantidad total de cada producto que deberán comprar. Primero, deberás implementar el IterablePasillos, y luego el generador. Además, deberás crear tres métodos que facilitarán la planificación de compra de los ayudantes en el supermercado, para los cuales deberás implementar `map`, `filter` y `reduce`.

Archivos

Archivos de datos

- `data/ayudantes.csv`: Contiene la información de las compras de cada ayudante. **No debes modificarlo**
  El formato del archivo es:

  `nombre,plato_1,plato_2,plato_3,plato_4,presupuesto`

- `data/pasillos.csv`: Contiene la información de cada pasillo del supermercado. **No debes modificarlo**
  El formato del archivo es:

  `id_pasillo,nombre_pasillo`

- `data/pasillos_productos.csv`: Contiene la información de todos los ingredientes que se venden en el supermercado. **No debes modificarlo**
  El formato del archivo es:

  `id_pasillo,nombre_ingrediente,precio_ingrediente`

- `data/platos.csv`: Contiene la información de los platos disponibles para preparar. **No debes modificarlo**
  El formato del archivo es:

  `nombre_plato,ing_1;cantidad_ing_1,ing_2;cantidad_ing_2,ing_3;cantidad_ing_3...` [1]

Archivos de código

- `leer_archivos.py`: Sirve para cargar los datos de `ayudantes.csv`, `pasillos.csv`, `pasillos_productos.csv` y `platos.csv`. **No debes modificarlo**
- `parametros.py`: Contiene las rutas de las archivos de texto. **No debes modificarlo**
- `main.py`: Realiza todo el flujo del programa e imprime los resultados en la consola. **No debes modificarlo**
- `ayudantes.py`: Contiene la clase `Ayudante` y la *namedtuple* `Plato`. **Debes modificarlo**
- `supermercado.py`: Contiene las clases `Pasillo`, `ListaPasillos`, `IterablePasillos` y `Supermercado`, las que se utilizan para guardar y recorrer los datos de manera personalizada. **Debes modificarlo**

[1] ing=ingrediente. La cantidad de ingredientes puede variar según cada plato.

Clases implementadas

A continuación se describen las clases con las que deberás trabajar para realizar la actividad, que se encuentran en los archivos `ayudantes.py` y `supermercado.py`.

Namedtuple Plato

Namedtuple que representa un plato. Tiene los siguientes atributos:

- `nombre`: Un `str` con el nombre del plato.
- `ingredientes`: Una `list` de tuplas en el formato `(nombre_ingrediente, cantidad)`

Clase Ayudante

`class Ayudante:` **Debes modificarlo**
Clase que representa a los distintos ayudantes que participarán del MasterDCChef. Tiene los siguientes métodos y atributos:

- `def __init__(self)`: Inicializador de la clase con los siguientes atributos: **No debes modificarlo**
  - `self.nombre`: Un `str` con el nombre del ayudante.
  - `self.platos`: Una `list` que contiene `namedtuples` de los platos que debe preparar el ayudante.
  - `self.dinero`: Un `int` con el dinero que posee el ayudante.
- `def obtener_ingredientes_platos(self)`: Retorna una lista donde cada elemento contiene tuplas con los ingredientes para un plato y su cantidad. **Debes modificarlo**
- `def cantidad_ingredientes(self, lista_ingredientes_platos)`: Generador que recibe la lista de `obtener_ingredientes_platos` y genera la cantidad total de cada ingrediente que se debe comprar. **Debes modificarlo**
- `def total_compra(self, ingredientes_platos, supermercado)`: Calcula el valor total a pagar por los ingredientes. **Debes modificarlo**

Clase Pasillo

`class Pasillo:` **No debes modificarlo**
Clase que representa un pasillo dentro del supermercado. Tiene los siguientes métodos y atributos:

- `def __init__(self, id, nombre, productos)`: Inicializador de la clase con los siguientes atributos:
  - `self.id`: Un `int` con el número identificador del pasillo.
  - `self.nombre`: Un `str` con el nombre del pasillo.
  - `self.productos`: Una `list` con los nombres de los productos que se encuentran en el pasillo.
  - `self.siguiente`: Contiene el siguiente pasillo a recorrer.

Clase ListaPasillos

`class ListaPasillos:` **Debes modificarlo**
Clase que tiene los pasillos del supermercado en forma de lista ligada. Tiene los siguientes métodos y atributos:

- `def __init__(self, primer_pasillo)`: Inicializador de la clase con los siguientes atributos: No debes modificarlo
  - `self.primer_pasillo`: Un `Pasillo` Representando el primer pasillo de la lista ligada.
- `def __iter__(self)`: Método que retorna el iterador correspondiente **Debes modificarlo**

Clase IteradorPasillos

`class IteradorPasillos:` **Debes modificarlo**
Iterador para recorrer los pasillos del supermercado. Debes implementarlo en su totalidad.

Clase Supermercado

`class Supermercado:` **Debes modificarlo**
Clase que representa el supermercado en que los participantes comprarán los ingredientes de sus platos a preparar. Tiene los siguientes métodos y atributos:

- `def __init__(self, lista_pasillos, productos)`: Inicializador de la clase con los siguientes atributos: No debes modificarlo
  - `self.lista_pasillos`: Una lista ligada de `ListaPasillos`.
  - `self.productos`: Un `dict` con los ingredientes y su información, donde la llaves son los nombres de cada ingrediente y los valores son tuplas con el id del pasillo donde se encuentra y su precio.
- `def consulta_precio(self, nombre_producto)`: Retorna el precio de un producto. No debes modificarlo
- `def pasillo_tiene_ingredientes(self, pasillo, ingredientes_platos)`: Método que retorna True cuando el pasillo contiene alguno de los ingredientes a comprar. No debes modificarlo
- `def pasillos_a_recorrer(self, ingredientes_platos)`: Entrega una lista con los pasillos que el ayudante deberá recorrer para comprar los ingredientes. **Debes modificarlo**

Detalles de implementación

IMPORTANTE: No deberás utilizar **loops** como `while` o `for` cuando se explicite. En ese caso, puedes usar más de una de las funciones `map`, `filter` y `reduce` encadenadas, y otras estructuras de datos si lo necesitas, pero lo importante es que no uses loops.

PARTE 1: Ayudantes

En esta primera parte deberás completar tres métodos que permitirán a los ayudantes organizar su lista de compras y presupuesto. En el archivo `ayudantes.py` encontrarás la clase Ayudante cuyos métodos deberás implementar.

- `def obtener_ingredientes_platos(self)`: En este método deberás obtener todos los ingredientes necesarios para los platos del ayudante. Mediante el uso de *map* y otras funciones si así lo requieres, retorna una lista con cada uno de los ingredientes de los platos del ayudante. **no debes utilizar loops** **Debes modificarlo**
- `def cantidad_ingredientes(lista_ingredientes_platos)`: Este generador recibe una lista de listas de ingredientes para hacer platos, y deberá ir generando tuplas de ingredientes que contengan el nombre del ingrediente y la cantidad total que se deberá comprar para hacer todos los platos que requieran ese ingrediente. No se debe repetir el nombre del ingrediente en 2 tuplas generadas

por este método. Más detalladamente, el parámetro `lista_ingredientes_platos` vendrá en este formato: **puedes usar loops** Debes modificarlo

```
[
    [
        (ingrediente_7, cantidad_7),
        (ingrediente_50, cantidad_50),
        (ingrediente_8, cantidad_8)
    ],
    [
        (ingrediente_5, cantidad_5),
        (ingrediente_8, cantidad_8_2),
        (ingrediente_12, cantidad_12)
    ]
]
```

Y tu generador deberá retornar tuplas de esta forma:

```
(ingrediente_7, cantidad_7)
(ingrediente_50, cantidad_50)
(ingrediente_8, cantidad_8 + cantidad_8_2)
...
```

- `def total_compra(self, ingredientes_platos, supermercado)`: En este método, deberás utilizar *reduce* (y otras funciones si así lo requieres) para calcular el total de la compra. Recibe una lista de tuplas de la forma:

```
[
    [
        (ingrediente_7, cantidad_7),
        (ingrediente_50, cantidad_50),
        (ingrediente_8, cantidad_8)
    ],
    [
        (ingrediente_5, cantidad_5),
        (ingrediente_8, cantidad_8_2),
        (ingrediente_12, cantidad_12)
    ]
]
```

y una instancia de Supermercado. Esta función debe retornar el precio total de la compra de todos los ingredientes de los platos del ayudante. Recuerda que `Supermercado` tiene un método para consultar el precio de un producto mediante su nombre. **no debes utilizar loops** Debes modificarlo

PARTE 2: Supermercado

En esta segunda parte deberás implementar clases que permitan iterar sobre los pasillos del supermercado y además completar un método que entregará el recorrido para realizar las compras. En el archivo `supermercado.py` encontrarás las clases `IterablePasillos` y `ListaPasillos` que deberás implementar de manera que `ListaPasillos` sea un iterable. Por último deberás completar un método de la clase Supermercado.

- `class ListaPasillos`: Iterable que representa una lista ligada de los pasillos del supermercado. Debes implementar:
  - `def __iter__(self)`: Deberás modificar este método para que cree y retorne una instancia de IteradorPasillos. **Debes modificarlo**
- `class IteradorPasillos`: Iterador para recorrer los pasillos de tu supermercado. Recuerda que los pasillos del supermercado están implementados como lista ligada. El iterador deberá retornar la clase `Pasillo` que corresponde al siguiente pasillo en cada iteración.
  - `def __init__(self, primer_pasillo)`: Deberás modificar el constructor para que guarde los atributos necesarios para iterar sobre los pasillos del supermercado. **Debes modificarlo**
  - `def __iter__(self)`: Este método debe retornar un Iterador. Debes aplicar tus conocimientos de iterables para saber cuál. **Debes modificarlo**
  - `def __next__(self)`: Retorna el siguiente pasillo. Deberás modificar este método para que tu iterador funcione correctamente. **Debes modificarlo**
- `def pasillos_a_recorrer(self, ingredientes_platos)`: Este método permitirá obtener la lista de pasillos que debe recorrer un ayudante. Recibe la lista de ingredientes que requiere el ayudante y retorna una lista (normal, no ligada) con los pasillos que tienen los ingredientes necesarios. **no debes utilizar loops** **Debes modificarlo**

Notas

- Recuerda que para las funciones `map`, `reduce` y `filter`, es necesario que entregues una función. Esta función puede ser un `lambda`, aunque también puedes definir una función auxiliar dentro de otra función.
- Siéntete libre de agregar nuevos `print()` en cualquier lugar de tu código para encontrar errores. Es una herramienta muy útil para comprobar tu desarrollo, pero recuerda borrarlos antes de hacer el commit final, así evitas confundir a tu corrector.
- A pesar de que no debes modificar todos los archivos, puede ser útil verlos para entender que está pasando por detrás del programa. Se recomienda especialmente ver el flujo en el archivo `main.py` para entender qué funciones se llaman primero.

Requerimientos

- (0.5 pts) Parte 1: Ayudantes
- (0.5 pts) Parte 2: Supermercado