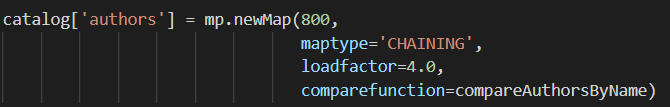
OBSERVACIONES DEL LA PRACTICA

Juan Sebastián Ortega Cod 202021703

Yesid Camilo Almanza Cod 201921773



**Figura 1:** Creación de un mapa para los autores (preguntas 1-4)

1. **¿Qué estructura de datos se usa para este índice?**

Al observar la Figura 1 es posible notar que la función newMap (que como su nombre lo indica crea un nuevo mapa) tiene un parámetro específcio denominado ***maptype*** el cual ha sido establecido como de tipo *‘CHAINING’*. Esto indica que detrás del TAD MAP se está utilizando una Hash Table construida mediante **SEPARATE CHAINING**, por lo que se espera que en cada posición del MAP sea posible almacenar más de un solo autor (Bucket).

1. **¿Cuántos elementos se espera almacenar inicialmente?**

El primer parámetro que recibe la función mp.newMap() está documentado como **numelements** en el módulo mapstructure.py, por lo tanto, es posible deducir que el número **800** (ingresado en este caso específico como el primer parámetro) indica que se espera almacenar 800 autores inicialmente dentro de este mapa.

1. **¿Cuál es el factor de carga?**

El factor de carga está especificado con el valor de **4.0** por el parámetro **loadfactor**. Por lo tanto, se puede obtener el número aporximado de posiciones requeridas siguiento la fórmula:

**loadfactor** = m (elementos que se espera almacenar) / n (posiciones del mapa)

**4.0 = 800 / n**

**n = 200**

Además, es necesario tener en cuenta la regla de que es más eficiente almacenar la información en un mapa cuyo número de posiciones es primo.

**n = nextPrime(200) = 211**

1. **¿Con cuántos elementos serán necesarios agregar para hacer re-hash de la tabla?**

Conociendo el número de posiciones actuales y el factor de carga es posible establecer una desigualdad cuyo resultado nos indique el número de elementos máximo antes de exceder el factor de carga y requerir de un re-hash:

**m / n <= 4.0**

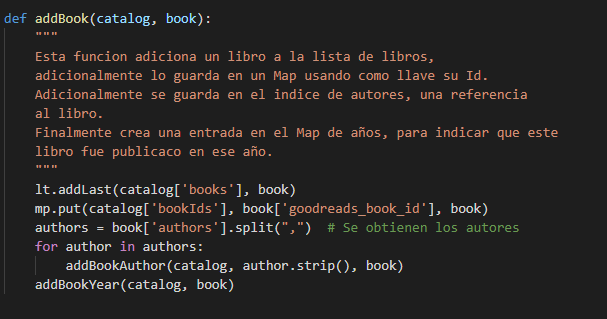
**m / 211 < = 4.0**

**m <= 844**

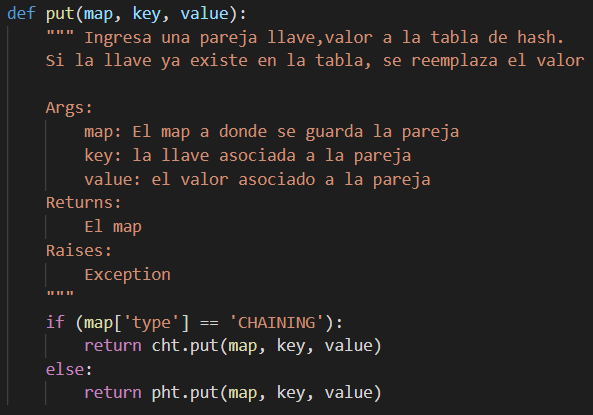
Al encontrar la diferencia entre el número máximo de elementos antes de exceder el factor de carga más un elemento y los elementos ya añadidos al mapa es posible encontrar que para hacer re-hash es necesario agregar:

**844+1 – 800 = 45 elementos**

**De acuerdo a la función addBook del model responder las preguntas 5-7**



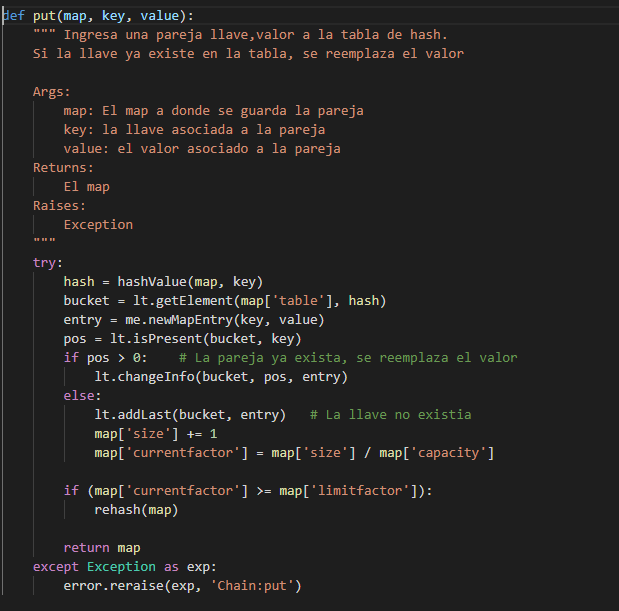
**Figura 2:** Función addBook ()



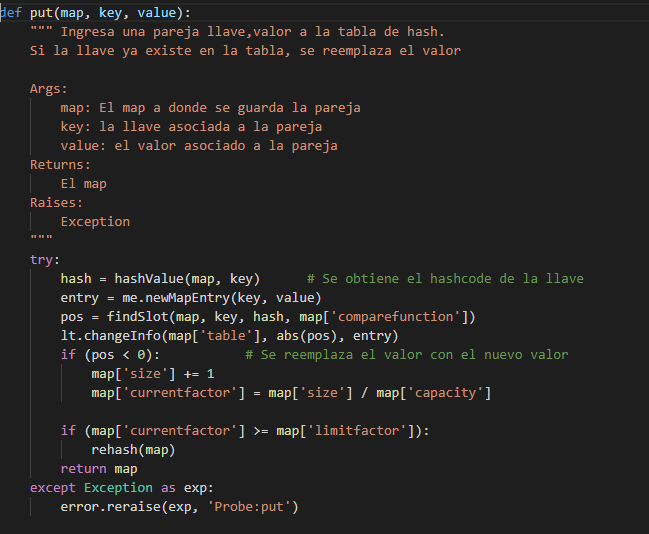
**Figura 3:** Función put() en mapstructure.py (preguntas 5-7)

1. ¿Qué hace la instrucción “**mp.put(...)”**?

En la Figura 3 es posible observar que la función **mp.put()** recibe tres argumentos: map, key y value. El argumento **map** hace referencia a un TAD MAP, el argumento **key** hace referencia a la llave de una pareja de datos y el argumento **value** hace referencia al valor de esta pareja. De esta forma, esta función busca añadir una llave (**key**) al TAD MAP (**map**) y a su vez agregar el valor (**value**) asociado a dicha llave. Además, esta función evalua el tipo de estructura de datos sobre la que está construido el TAD MAP y procede a añadir la pareja **key,value** ya sea a un mapa construido mediante *SEPARATE\_CHAINING* o mediante *LINEAR PROBING*:



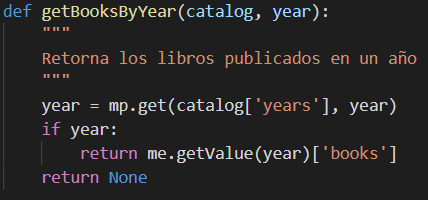
**Figura 4:** Función cht.put() | SEPARATE CHAINING



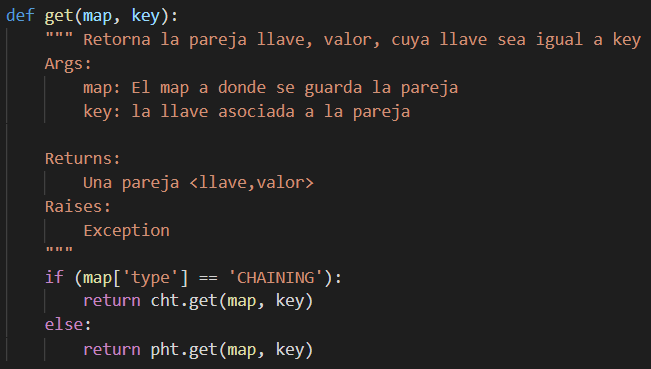
**Figura 5:** Función pht.put() | LINEAR PROBING

1. **¿Qué papel cumple “book[‘goodreads\_book\_id’]” en esa instrucción?**
2. **¿Qué papel cumple el tercer parámetro “book” en esa instrucción?**

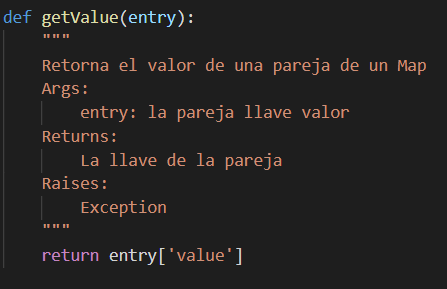
**En respuesta a la pregunta 6 y 7:** Para nuestro ejercicio en Books evidenciamos como la función **put** pasa de parametros **catalog['bookIds']** que es un TAD MAP con la información de los libros, **“book[‘goodreads\_book\_id’]**  es la la id de nuestro libro la cual será una llave que sera añadida a nuestro mapa y para finalizar **book** es el value que sera añadido a nuestro mapa y que contiene la información del libro



**Figura 6:** Función getBooksByYear (Pregunta 8-10)



**Figura 7:** Función get() en mapstructure.py



**Figura 8:** Función getValue() en mapentry.py

1. ¿Qué hace la instrucción **“mp.get(…)”**?

En la Figura 7 es posible observar que la función **mp.get()** recibe dos argumentos: **map** y **key.** De acuerdo a su documentación, esta función recibe un TAD MAP (map) y una llave específica (key), luego procede a revisar el tipo de estructura de datos utilizada y en base a esto utiliza una función diferente para obtener como retorno a la pareja **key, value** en ambos casos.

1. ¿Qué papel cumple **“*year”*** en esa instrucción?

Al relacionar la Figura 6 con la Figura 7 es posible observar que **year** entra como parametro a la función get() haciendo referencia a la llave respecto a la cual se desea realizar una búsqueda. Por lo tanto, cumple con la función de encontrar todos los libros (values) que estén identificados con ese **año específico a forma de llave (key)** .

1. ¿Qué hace la instrucción **“me.getValue(…)”**?

En la Figura 8 es posible observar que getValue() recibe un único argumento: **entry**. Este argumento hace referencia a un objeto compuesto por la pareja **llave, valor** respecto a la cual retorna únicamente el **valor.** Al relacionar esta función con su aparición en la Figura 6 es posible observar como se pasa la variable *year* (la cual contiene la pareja llave, valor obtenida por la función get()) como parámetro, siendo el resultado de esta la información del libro (value) asociada a esta llave.