ESCUELA DE INGENIERÍA

Concepto de hash e inmutabilidad

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Cristian Ruz Ruz

Profesor Asistente Adjunto, Departamento de Ciencia de la Computación
Facultad de Ingeniería
Pontificia Universidad Católica de Chile

Índice

- Concepto de hash
- Tablas de hash y búsquedas eficientes
- Elementos hasheables en Python

Introducción

- El concepto de hash.
- Para qué se usan los hash.
- Qué relación hay entre hash y estructuras inmutables.

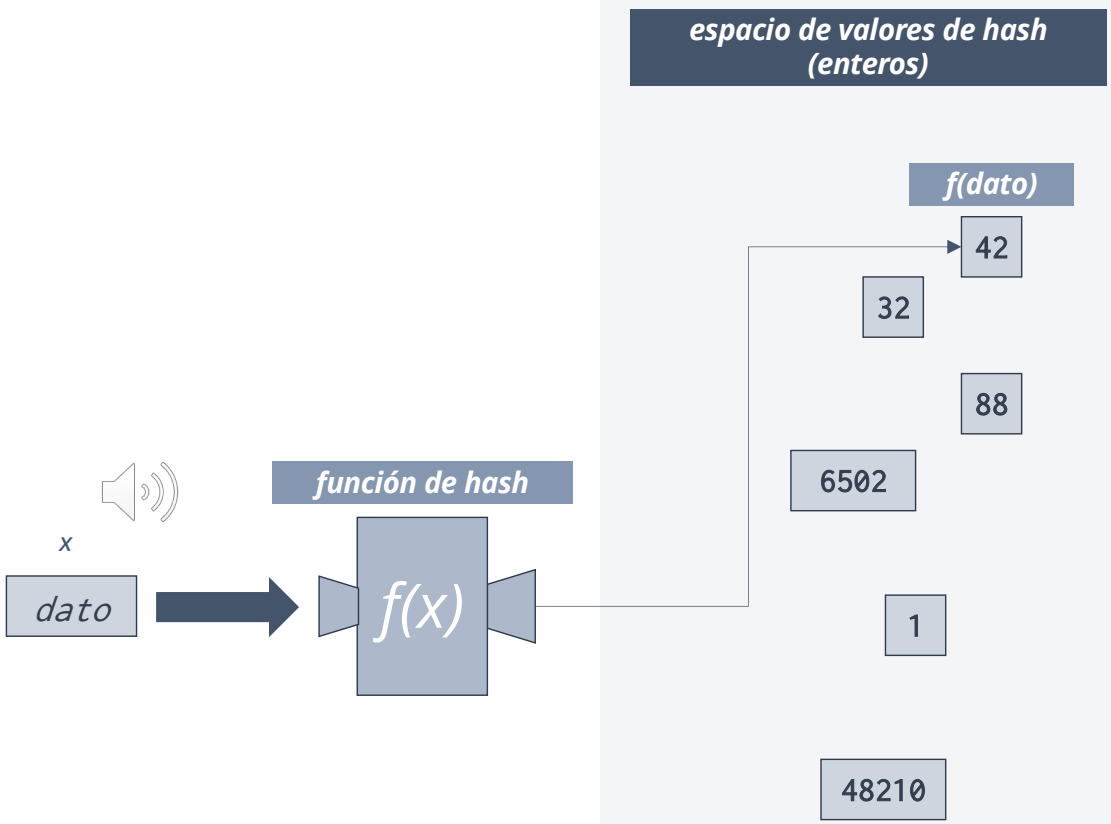
Concepto de *hash*

¿Qué es un *hash*?

Función de hash

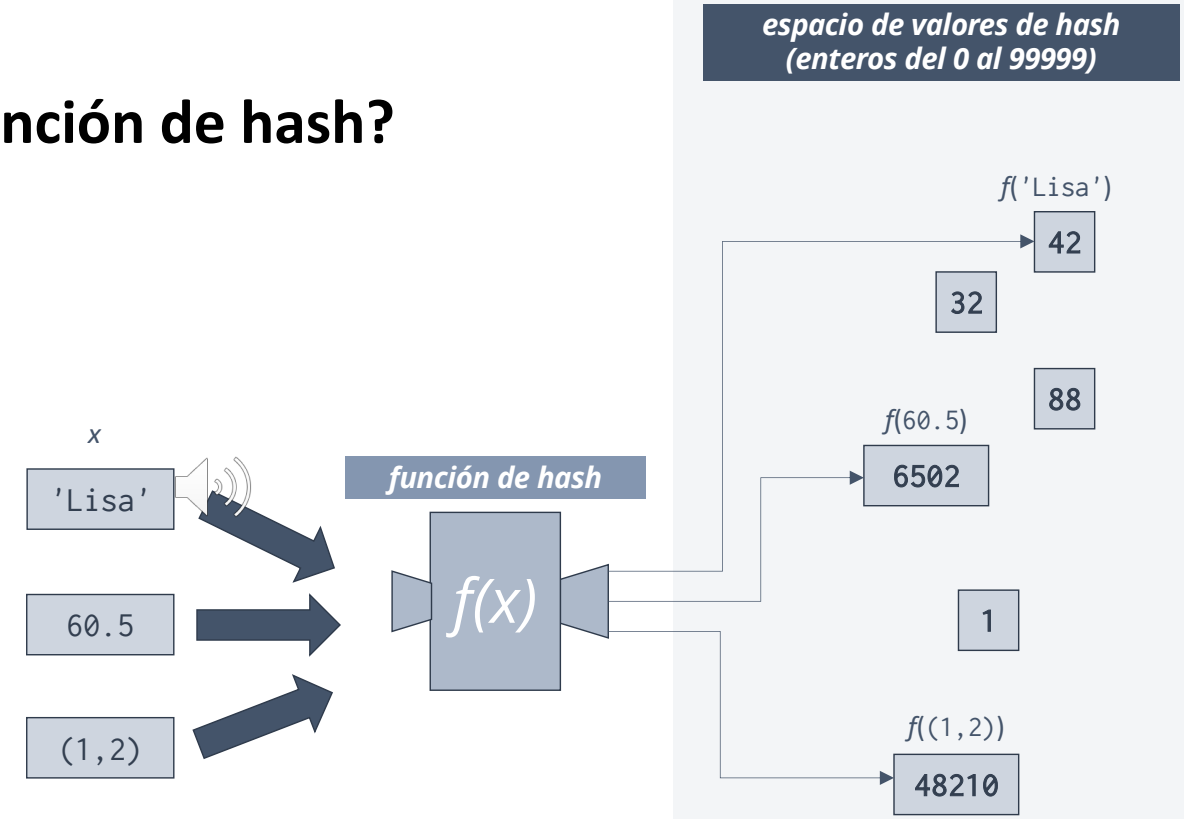
Recibe un dato y devuelve un entero dentro de un rango acotado

Resultado es el **hash**, o **valor de hash**



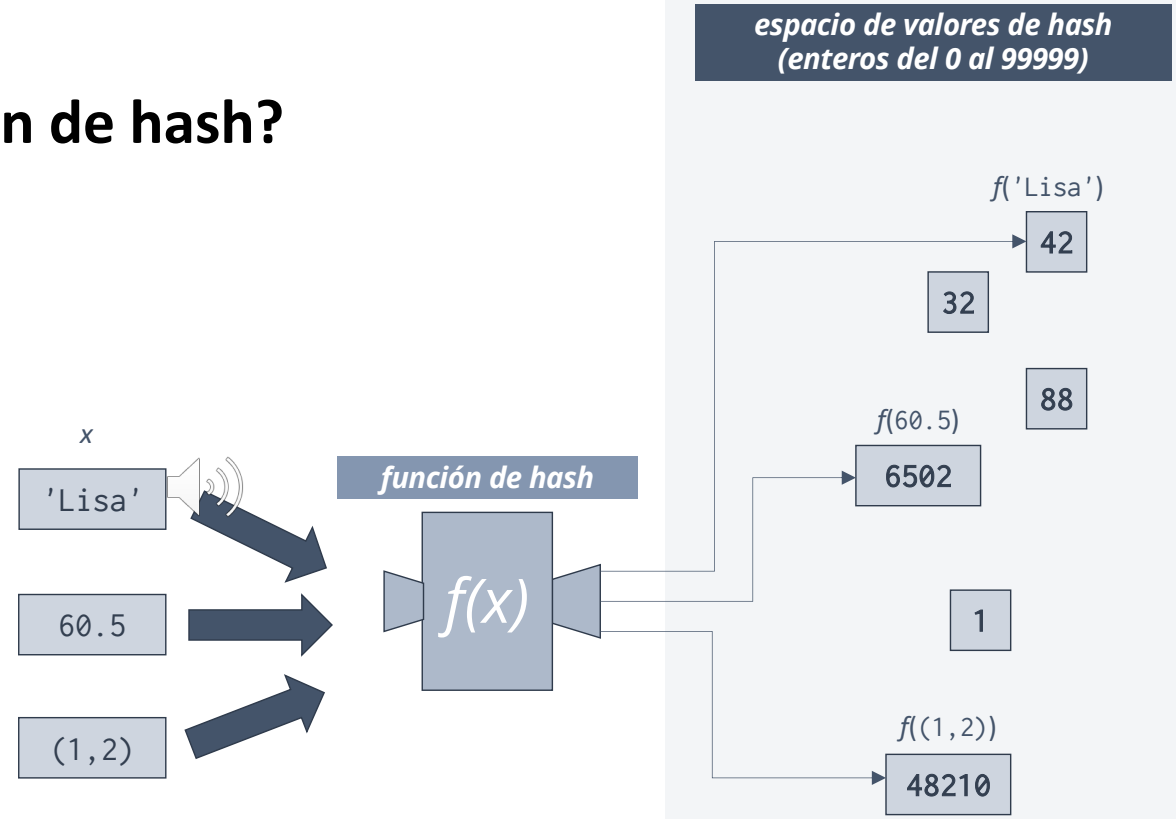
¿Qué debe cumplir una función de hash?

1. Debe poder calcularse de manera **eficiente.**
2. Debe entregar resultados dentro de un **rango acotado.**
3. Debe provocar **pocas colisiones.**



¿Para qué sirve una función de hash?

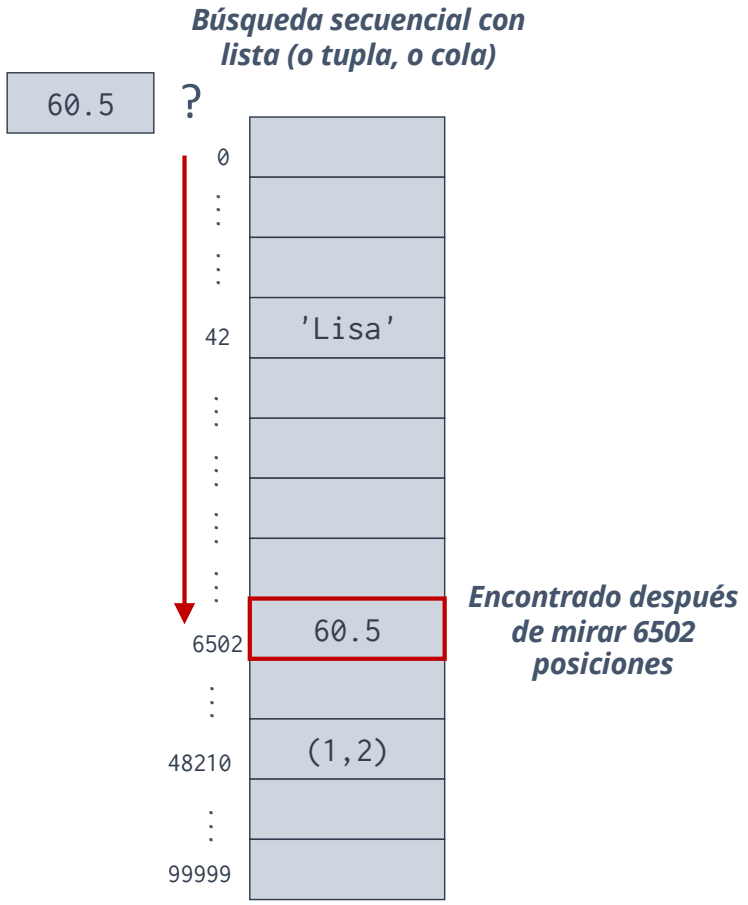
1. Búsqueda eficiente de elementos.
2. Comparaciones eficientes entre datos.
3. Criptografía, verificaciones de integridad, etc.



Tablas de hash y búsquedas eficientes

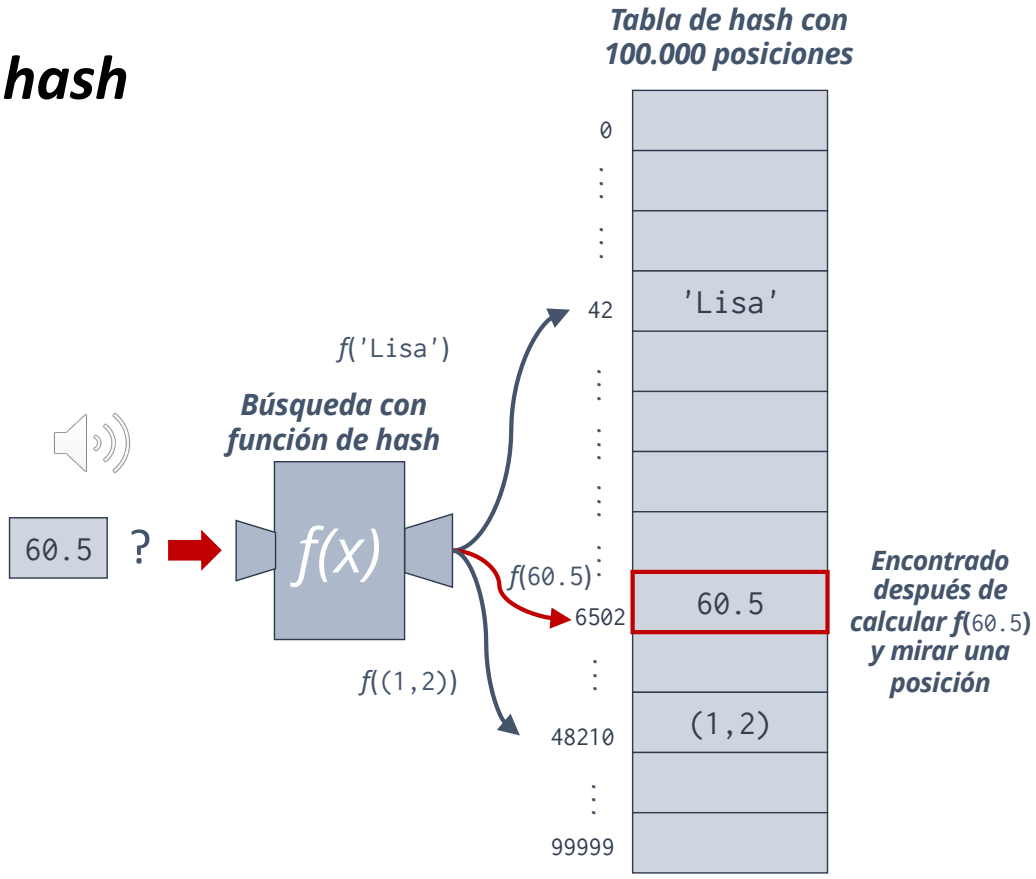
Buscando elementos: `list`

1. Lista con 100.000 posiciones. Queremos **saber si 60.5 está en la lista.**
2. Si está, lo encontraremos, en promedio, **después de recorrer 50.000 posiciones.**
3. Si **no está**, debemos mirar **todas las posiciones.**
4. Mientras **más posiciones hay**, más nos demoraremos.



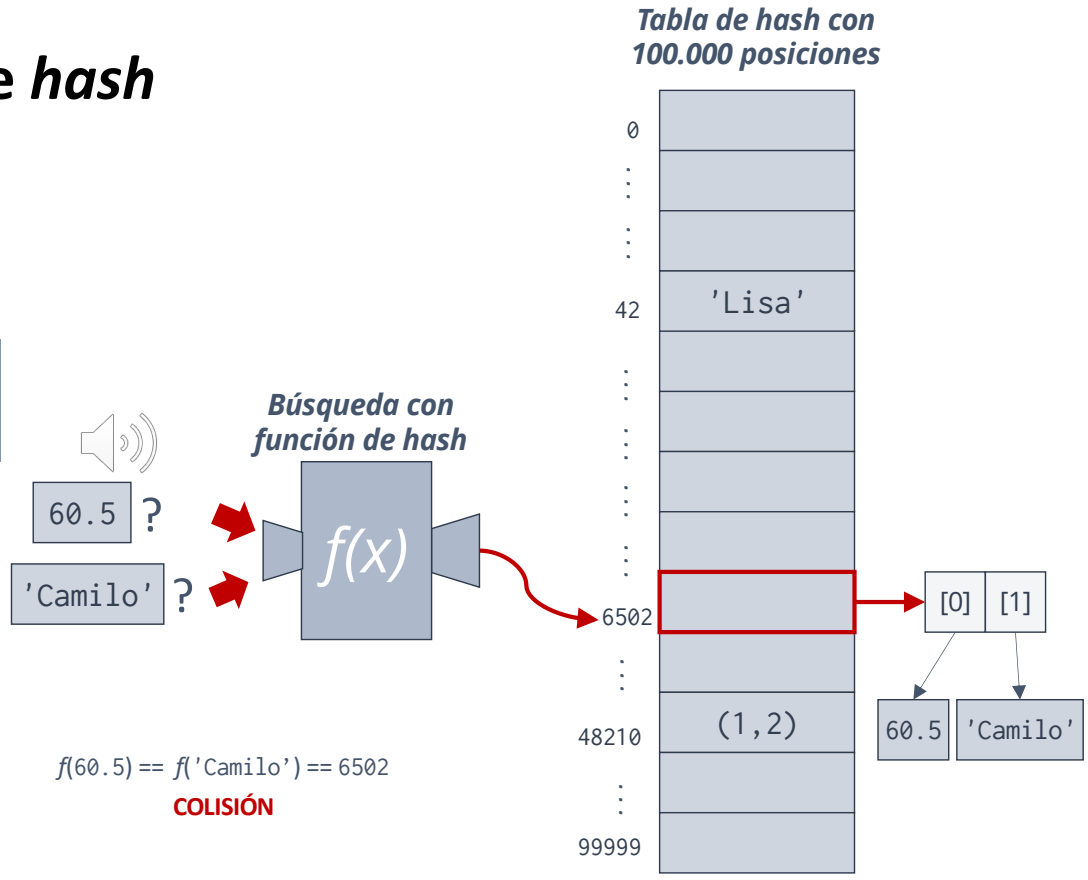
Buscando elementos: tabla de *hash*

1. Estructura para **encontrar elementos de manera eficiente.**
2. **El *hash* de cada valor indica su posición.**
3. Si **dos valores tienen el mismo *hash***, se encadenan en una lista.



Buscando elementos: tabla de *hash*

1. Estructura para **encontrar elementos de manera eficiente.**
2. **El *hash* de cada valor indica su posición.**
3. Si **dos valores tienen el mismo *hash***, se encadenan en una lista.



Elementos *hasheables* en Python

Elementos *hasheables*

1. Función `hash()`.
2. **Hash para** `int`, `float`, `bool`, `string`, `tuple`.
3. **El mismo hash** para cada valor **durante toda la ejecución.**

Obtener el *hash* de un valor

```
print(hash(10))
```

```
10
```

```
print(hash(60.5))
```

```
1152921504606847036
```

```
print(hash('Lisa'))
```

```
-2142938998658150559
```

```
print(hash((1,2)))
```

```
3713081631934410656
```

```
print(hash('Lisa'))
```

```
-2142938998658150559
```

Elementos NO *hasheables*

1. No hay **hash** para `list`, ni para `deque`.
2. Tampoco para `tuple` que contengan elementos no **hasheables.**

No podemos obtener *hash* para `list`

```
print(hash([1,2]))
```

```
TypeError: unhashable type: 'list'
```

```
print(hash((1,[2,3],4)))
```

```
TypeError: unhashable type: 'list'
```

```
print(hash((1,(2,[3,4],5))))
```

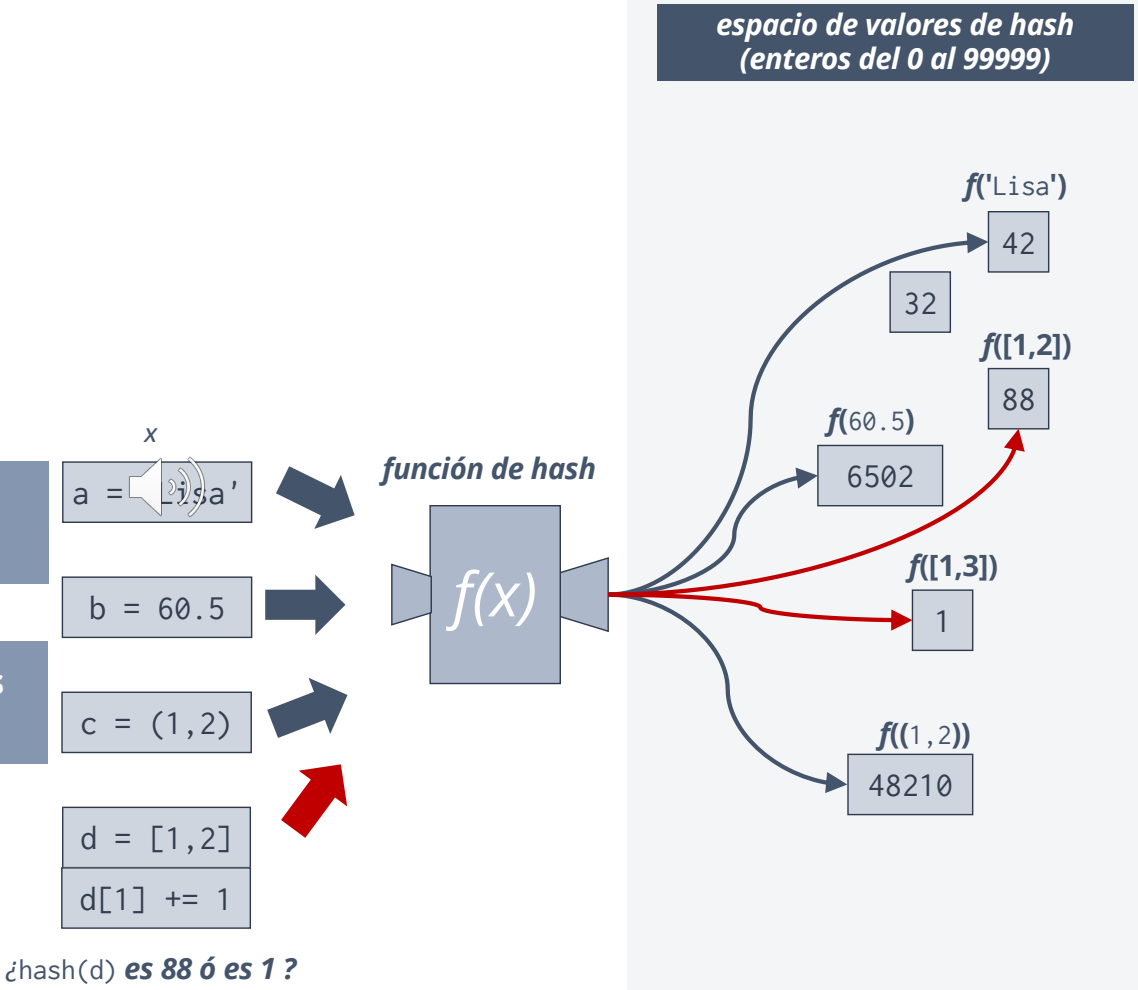
```
TypeError: unhashable type: 'list'
```

```
print(hash(deque([1,2])))
```

```
TypeError: unhashable type: 'collections.deque'
```

¿Qué significa ser *hasheable*?

1. Listas y deque son **mutables**.
2. **No podemos usar elementos mutables** como entrada para una función de hash.



¿hash(d) ***es 88 ó es 1 ?***

Síntesis

- Concepto de hash.
- Estructura **tabla de *hash*.**
- Búsqueda eficientes.
- Hashes e inmutabilidad.

Referencias bibliográficas

- Glossary. Documentación Python 3.8. Glosario de términos. Recuperado de: https://docs.python.org/3.8/glossary.html#term-hashable

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE