99 | GROUP

Python Dictionary (or Hash tables)

Ye Lin Aung | yelin@99.co

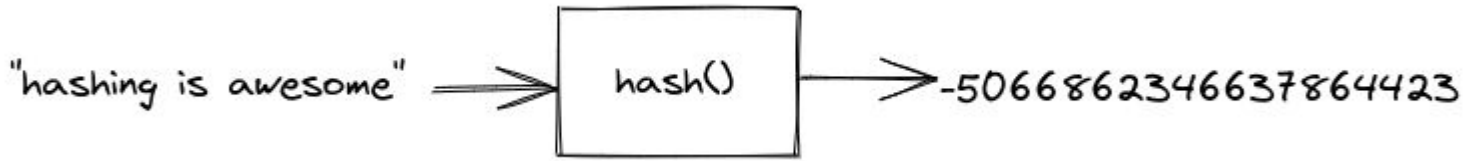
Agenda

1. What is `hash()` ? (baby don't hurt me ..)
2. Hashable types
3. Basic hash table examples
    a. Function based
    b. Namespace based
    c. More namespace based
    d. Classes
    e. More Classes
    f. More .. Classes

What is hashing ?

What is hashing ?

- It's a process that takes some input of data (any length) and return a fixed-length value (hash value).
- It is **fast**. Calculating the hash value is very fast.
- It is **deterministic**. The same input will always produce the same hash.
- It produces **fixed length hash**. No matter the input, the returning hash will always be of a fixed, predetermined length.
- They are "one way". You cannot recalculate `-5066862346637864423` back to `"hashing is awesome"`



What is hashing ?

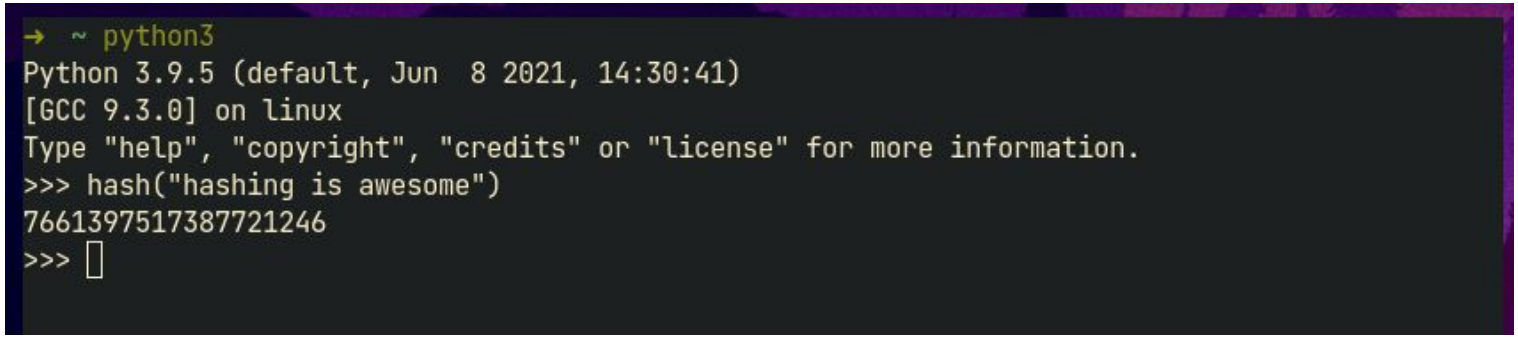
- There are a lot of hashing algorithms. Some popular ones are
  - MD5 - MD5 Hash Generator
  - Secure Hash Algorithms
    - SHA-1
    - SHA-2 etc
  - MurmurHash

Where do we use this hashing thing?

- Storing the passwords!
  - Our user service DO NOT STORE the passwords as plain texts. We hash the password and store the hash. When user enters the password, we hash the input and compare it with the value we've stored in the database.
- "Forgot Password"
  - When you request forgot password, you don't get back your original passwords and have to reset with a new one because we don't have the original password!

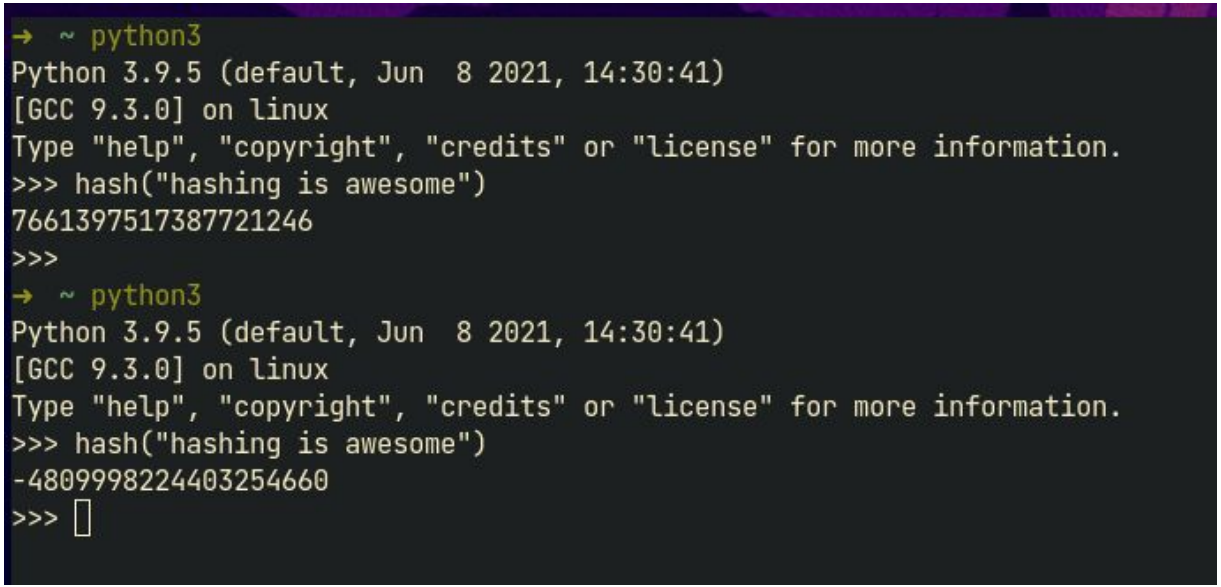
What is hashing ?

- Try it in your console



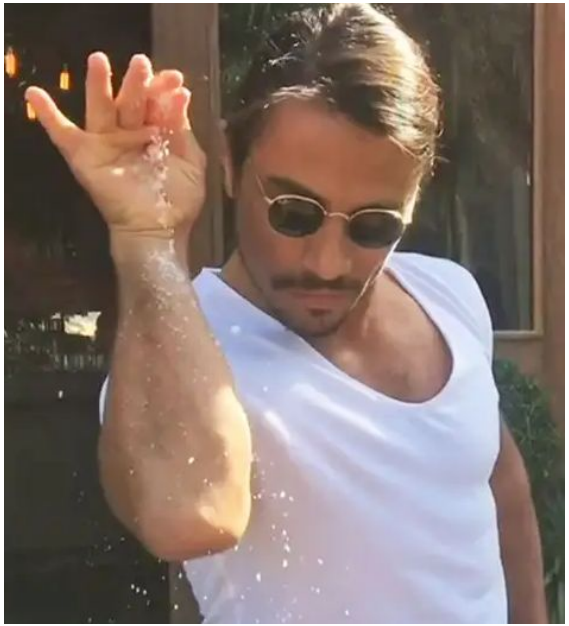
What is hashing ?

- Wait.. why are they different in a new session? Didn't you say they are *deterministic* ?



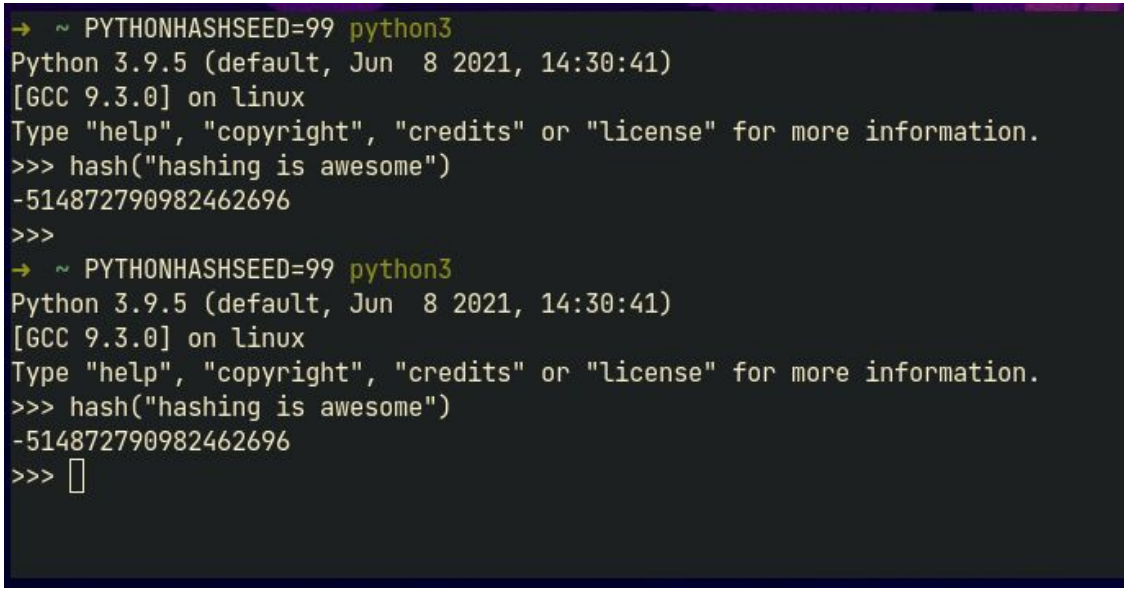
Salt!

- Because starting from Python 3.3, inputs are “salted” with a random value before the hashing process.



Can we make it deterministic?

- We can override this behaviour by setting `PYTHONHASHSEED` environmental variable.



But why do we need the salt, you ask

- This is a security feature. We *truly* want things to be a one-way street.
- If someone got the access to our user database…
  - They could get a big list of commonly used passwords (`123456 ??`)
  - Compare the hash strings between these and the ones from the database and guess the password! (known as Rainbow table)
- That's why we need to "salt" the password before hashing them.

Can we **hash()** everything in Python?

- Only for the “immutable” types!

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

From Mutable vs Immutable Objects in Python | by megha mohan

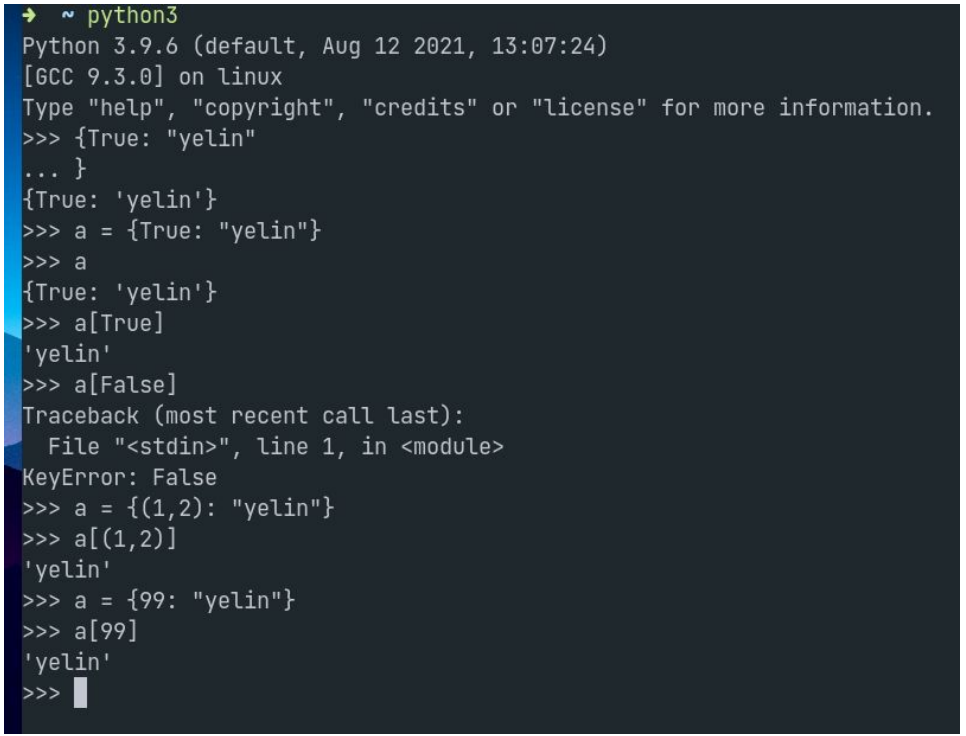
Can we **hash()** everything in Python?

- Only for the “immutable” types!

```
→  ~ python3
Python 3.9.5 (default, Jun  8 2021, 14:30:41)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> hash(True)
1
>>> hash(False)
0
>>> hash(False)
0
>>> hash([1,2])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
>>>
```

Enter Dictionary! (Dict)

- A key / value pair data structure
- They “key” must be hashable. I.e bool, str, tuple
- Hash tables are very useful
  - Cache
  - `Set()`



A Hashtable - Function based

- A simple function with
  - `Setup`
  - `Insert (k, v)`
  - `Lookup (k)`

- These awesome examples are from this talk [#PyConEstonia 2020: 'Object Oriented Programming from scratch (four times)' by Raymond Hettinger] . Recommend to watch the whole talk!

A Hashtable - Namespace!

- Let's add "namespace" ability to have more hashtables
- Using a dictionary as a namespace (the irony!)

A Hashtable - Simplenamespace

- If we replace the dictionary with Simplenamespace ....

A Hashtable - Class

- We are just moving the functions into a Class
- Notice about `self`

A Hashtable - Class Magic

- Dunder/Magic Methods
  - `__init__`
  - `__str__`
  - `__repr__`
  - `__len__`

Example source codes

- https://github.com/yelinaung/hashtable-examples

What we didn’t dive deep?

1. How do we solve if the two hashes are the same (aka Hash collision) ?
2. What are the ways to solve that?

Maybe in part 2 … !


99 GROUP
THANK YOU
99.co
rumah123.com
iProperty.com.sg
SRX