DICTIONARIES, BEHIND THE SCENES

Gisela Rossi

“Any running Python program has many dictionaries active at the same time, even if the user's program code doesn't explicitly use a dictionary" - A. Kuchling

1.

HASHING & DICTS

An overview of the engine behind dicts

WHAT IS HASHING?

A hash function is a function that can be used to map data of arbitrary size onto an integer.

A hash function must satisfy:

- If two objects are equal, then their hashes should be equal

A good hash function should satisfy:

- The hash of an object should be cheap to compute
- Minimise collisions

HASH TABLES

We use hashing functions to create hash tables. All insertion, search, and deletion are **`O(1)`** on average

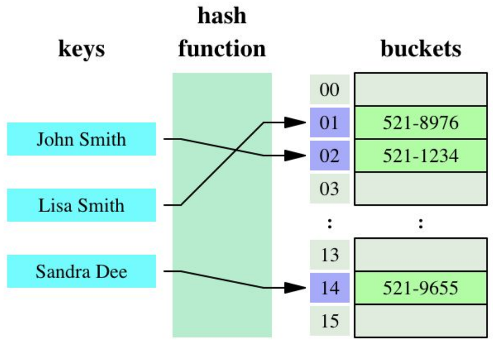
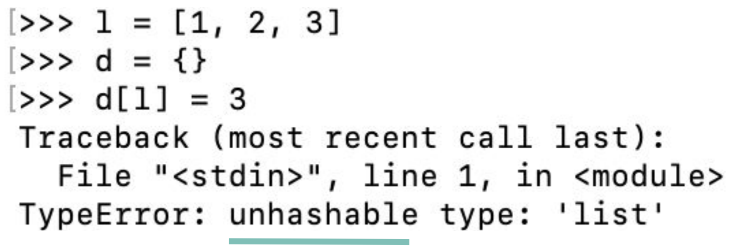


Image by Jorge Stolfi, shared under CC BY-SA 3.0

Python dictionaries and sets are implemented using hash tables. This is why you get this kind of errors:



```
[>>> l = [1, 2, 3]
[>>> d = {}
[>>> d[l] = 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

HASHABLE

An object is hashable if it has a hash value which never changes during its lifetime and can be compared to other objects. Hashable objects which compare equal must have the same hash value.

Hashability makes an object usable as a dictionary key or set member.

it needs a
`__hash__()`
method

HASHABLE

An object is hashable if it has a hash value which never changes during its lifetime and can be compared to other objects. Hashable objects which compare equal must have the same hash value.

Hashability makes an object usable as a dictionary key or set member.

The hash can't depend on mutable attributes

HASHABLE

An object is hashable if it has a hash value which never changes during its lifetime and can be compared to other objects. Hashable objects which compare equal must have the same hash value.

Hashability makes an object usable as a dictionary key or set member.

it needs a

`__eq__()`

method

HASHABLE

An object is hashable if it has a hash value which never changes during its lifetime and can be compared to other objects. Hashable objects which compare equal must have the same hash value.

Hashability makes an object usable as a dictionary key or set member.

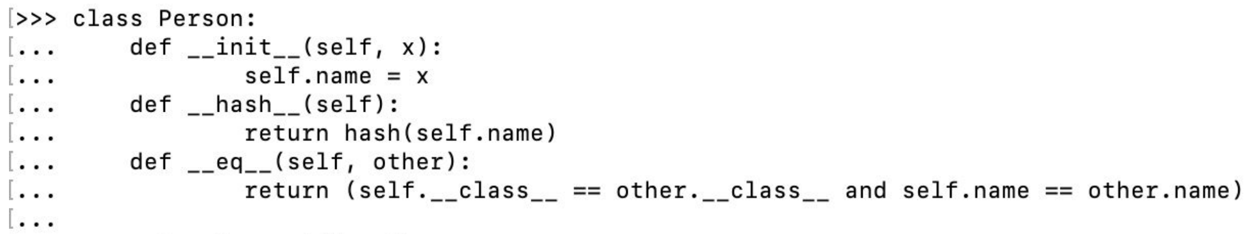
ARE USER-DEFINED OBJECTS HASHABLE?

- By default: YES. All user-defined classes have `__eq__()` and `__hash__()` methods.
- You can override them
- If you override `__eq__()` but you don't override `__hash__()` → automatically unhashable
- If you don't override `__eq__()` and want unhashability → set `__hash__ = None`

USER DEFINED HASH FUNCTIONS

- If you decide to define the equality and hash of your object by the equality and hash of some of the instance's attributes, you have to make sure those attributes never change

- The standard way to override __hash__ is to use the built-in hash() over a tuple of your relevant immutable attributes:



```
[>>> class Person:
[...     def __init__(self, x):
[...             self.name = x
[...     def __hash__(self):
[...             return hash(self.name)
[...     def __eq__(self, other):
[...             return (self.__class__ == other.__class__ and self.name == other.name)
[...
```

2.

DICT SECURITY

An evolution on hashing security

AN EVOLUTION ON HASHING SECURITY

Version 1

- non-cryptographic xor hash algorithm similar to FNV for string hashing
- efficient, but prone to a denial of service (DOS) vulnerability by sending you keys designed to collide
- attackers could exploit worst case performance of a dict insertion, O(n^2)

AN EVOLUTION ON HASHING SECURITY

Version 2

- still FNV, a random prefix and sufix were added as an attempt to randomize the hash values
- configurable with PYTHONHASHSEED env. variable
- performance issues
- although it made it harder for attackers, it didn't solved the vulnerability

AN EVOLUTION ON HASHING SECURITY

Version 3

- Introduced with PEP 456: Siphash
- Siphash is family of pseudorandom functions optimized for speed on short messages
- Siphash24 is about as fast as the old code with FNV

SNAPSHOT OF TODAY

Where do currently supported Python versions stand with regards to hashing?

Original Hash

2.7.1 - 2.7.2

Random Hash

2.7.3 - 2.7.13

Siphash

All currently supported python 3 versions (>3.4)

3.

Misc optimizations

Examples of fine tuning

MORE COMPACT DICTIONARIES

Example: d = {'timmy': 'red', 'barry': 'green', 'guido': 'blue'}

Before

- sparse hash table that would re-size if too full

Entries =

-8522787127447073495	'barry'	'green'
-9092791511155847987	'timmy'	'red'
-6480567542315338377	'guido'	'blue'

Now

- dense table referenced by a sparse table of indices.

Indices =

	1				0		2

Entries =

-9092791511155847987	'timmy'	'red'
-8522787127447073495	'barry'	'green'
-6480567542315338377	'guido'	'blue'

MORE COMPACT DICTIONARIES

- This change only affected data layout
- Significant memory savings (small dicts get the most benefit)
- Iteration is faster
- Resizing is faster (and touches fewer pieces of memory)
- Dictionaries remember the order of items inserted!

KEY-SHARING DICTIONARY

- Python uses dictionaries behind a class' attributes. (keys are strings)
- Key sharing (PEP 412) allowed attribute dictionaries to share keys with other attribute dictionaries of instances of the same class.
- Memory use is reduced up to 20%

STRING KEYS OPTIMIZATION

- Dictionaries with string keys are handled differently
- This is better for two reasons:
  - Comparing strings instead of python objects is simpler and faster
  - Comparing strings can't raise an exception

THANKS!

I am a Software Engineer and co-leader of PyLadies London.

You can find me at:

@gise_rossi

(where I post a lot of pics of my kitten)

REFERENCES

- https://docs.python.org/3/reference/datamodel.html#object.__hash__
- https://www.python.org/dev/peps/pep-0456/#sip
- https://learning.oreilly.com/library/view/beautiful-code/9780596510046/ch18.html
- https://morepypy.blogspot.com/2015/01/faster-more-memory-efficient-and-more.html
- https://hynek.me/articles/hashes-and-equality/
- Brandon Rhodes talk The Dictionary Even Mightier PyCon 2017
- Fluent Python, Luciano Ramalho, 2015
- https://www.asmeurer.com/blog/posts/what-happens-when-you-mess-with-hashing-in-python/
- https://mail.python.org/pipermail/python-dev/2012-December/123028.html
- https://www.python.org/dev/peps/pep-0412/
- https://github.com/python/cpython/blob/master/Objects/dictobject.c