

Dictionaries and Sets

- dict
- set
- frozenset
- set/dict comprehensions

Dictionaries (type dict)

$\{key_1: value_1, \dots, key_k: value_k\}$

- Stores a mutable set of (key, value) pairs, denoted *items*, with distinct keys
- Constructing empty dictionary: `dict()` or `{}`
- `dict[key]` lookup for key in dictionary, and returns associated value. Key must be present otherwise a `KeyError` is raised.
- `dict[key] = value` assigns value to *key*, overriding existing value if present.

key	value
'a'	7
'foo'	'42nd'
5	29
'5'	44
5.5	False
False	True




distinct keys,
i.e. not "=="

Dictionaries (type dict)

Python shell

```
> d = {'a': 42, 'b': 57}
> d
| {'a': 42, 'b': 57}
> d.keys()
| dict_keys(['a', 'b'])
> list(d.keys())
| ['a', 'b']
> d.items()
| dict_items([('a', 42), ('b', 57)])
> list(d.items())
| [('a', 42), ('b', 57)]
```

```
> for key in d:
    print(key)
| a
| b
> for key, val in d.items():
    print("Key", key, "has value", val)
| Key a has value 42
| Key b has value 57
> {5: 'a', 5.0: 'b'}
| {5: 'b'}
```



Python shell

```
> surname = dict(zip(['Donald', 'Mickey', 'Scrooge'], ['Duck', 'Mouse', 'McDuck']))
> surname['Mickey']
| 'Mouse'
```

Dictionary operation	Description
<code>len(d)</code>	Items in dictionary
<code>d[key]</code>	Lookup key
<code>d[key] = value</code>	Update value of key
<code>del d[key]</code>	Deletes an existing key
<code>key in d</code>	Key membership
<code>key not in d</code>	Key non-membership
<code>clear()</code>	Remove all items
<code>copy()</code>	Shallow copy
<code>get(key)</code>	<code>d[key]</code> if key in dictionary, otherwise <code>None</code>
<code>items()</code>	<i>View</i> of the dictionaries items
<code>keys()</code>	<i>View</i> of the dictionaries keys
<code>values()</code>	<i>View</i> of the dictionaries values
<code>pop(key)</code>	Remove key and return previous value
<code>popitem()</code>	Remove and return an arbitrary item
<code>update()</code>	Update key/value pairs from another dictionary

Order returned by `list(d.keys())` ?

The Python (3.6.4) Tutorial 5.5 Dictionaries

“Performing `list(d.keys())` on a dictionary returns a list of all the keys used in the dictionary, in **arbitrary order** (if you want it sorted, just use `sorted(d.keys())` instead).”

docs.python.org/3/tutorial/datastructures.html docs.python.org/3/library/stdtypes.html

The Python (3.6.4) Standard Library 4.10.1. Dictionary view objects

“Keys and values are iterated over in an **arbitrary order** which is non-random, varies across Python implementations, and depends on the dictionary’s history of insertions and deletions.”

Python 3.6.4 shell

```
> d = {'d': 1, 'c': 2, 'b': 3, 'a': 4}
> d['x'] = 5 # new key at end
> d['c'] = 6 # overwrite value
> del d['b'] # remove key 'b'
> d['b'] = 7 # reinsert key 'b' at end
> d
| {'d': 1, 'c': 6, 'a': 4, 'x': 5, 'b': 7}
```



[Raymond Hettinger @ Twitter](#)

See also Raymond’s talk @ PyCon 2017
[Modern Python Dictionaries](#)
[A confluence of a dozen great ideas](#)

Dictionary comprehension

- Similarly to creating a list using list comprehension, one can create a set of key-value pairs:

{key : value for variable in list}


Python shell


```
> names = ['Mickey', 'Donald', 'Scrooge']
> dict(enumerate(names, start=1))
| {1: 'Mickey', 2: 'Donald', 3: 'Scrooge'}
> {name: idx for idx, name in enumerate(names, start=1)}
| {'Donald': 2, 'Mickey': 1, 'Scrooge': 3}
```

Sets (set and frozenset)

$\{value_1, \dots, value_k\}$

- Values of type `set` represent mutable sets, where "==" elements only appear once
- Do **not** support: indexing, slicing
- `frozenset` is an immutable version of `set`

```
Python shell
> S = {2, 5, 'a', 'c'}
> T = {3, 4, 5, 'c'}
> S | T
| {2, 3, 4, 5, 'a', 'c'}
> S & T
| {5, 'c'}
> S ^ T
| {2, 3, 4, 'a'}
> S - T
| {2, 'a'}
> {4, 5, 5.0, 5.1}
| {4, 5, 5.1} 
```

Operation	Description
$S \mid T$	Set union
$S \& T$	Set intersection
$S - T$	Set difference
$S \wedge T$	Symmetric difference
<code>set()</code>	Empty set (<code>{}</code> = empty dictionary )
<code>set(L)</code>	Create set from list
<code>x in S</code>	Membership
<code>x not in S</code>	Non-membership
<code>S.isdisjoint(T)</code>	Disjoint sets
$S \leq T$	Subset
$S < T$	Proper subset
$S \geq T$	Superset
$S > T$	Proper superset
<code>len(S)</code>	Size of S

Question – What value has the expression ?

```
sorted( { 5, 5.5, 5.0, '5' } )
```

a) { '5', 5, 5.0, 5.5 }

b) { 5, 5.5 }

c) ['5', 5, 5.0, 5.5]

d) ['5', 5, 5.5]



e) TypeError

f) Don't know

Set comprehension

- Similarly to creating a list using list comprehension, one can create a set of values (also using nested for- and if-statements):

`{value for variable in list}`

- A value occurring multiple times as *value* will only be included once

primes_set.py

```
n = 101
not_primes = {m for f in range(2, n) for m in range(2*f, n, f)}
primes = set(range(2, n)) - not_primes
```

Python shell

```
> L = ['a', 'b', 'c']
> {(x, (y, z)) for x in L for y in L for z in L if x != y and y != z and z != x}
| {('a', ('b', 'c')), ('a', ('c', 'b')), ('b', ('a', 'c')), ..., ('c', ('b', 'a'))}
```

Hash, equality, and immutability

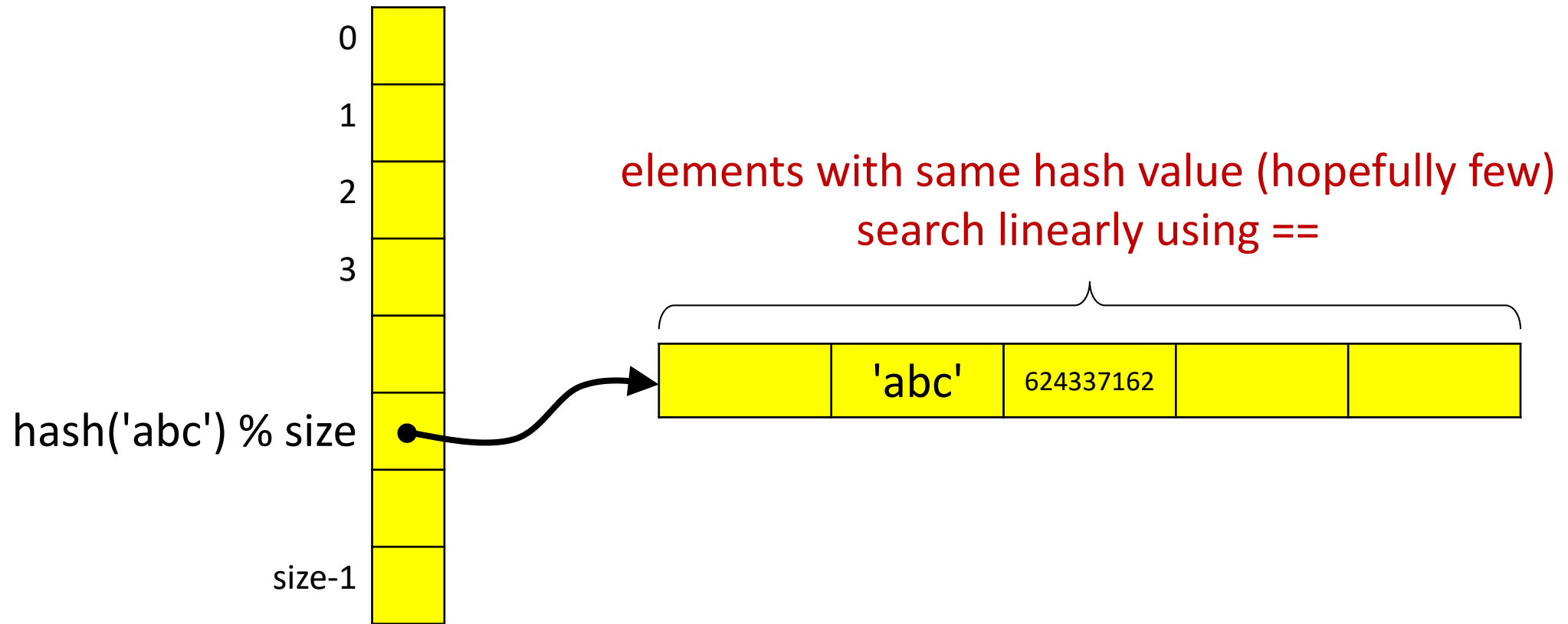
- Keys for dictionaries and sets must be *hashable*, i.e. have a `__hash__()` method returning an integer that does not change over their lifetime and an `__eq__()` method to check for equality with “==”.

`'abc'.__hash__()` could e.g. return 624337162

`(624337162).__hash__()` would also return 624337162

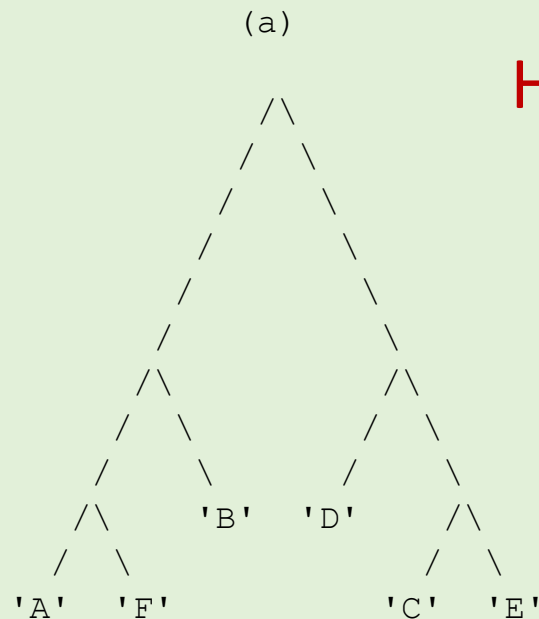
- **All built-in immutable types are hashable.** In particular tuples of immutable values are hashable. I.e. trees represented by nested tuples like `((('a'), 'b'), ('c', ('d', 'e')))` can be used as dictionary keys or stored in a set.

Sketch of internal set implementation

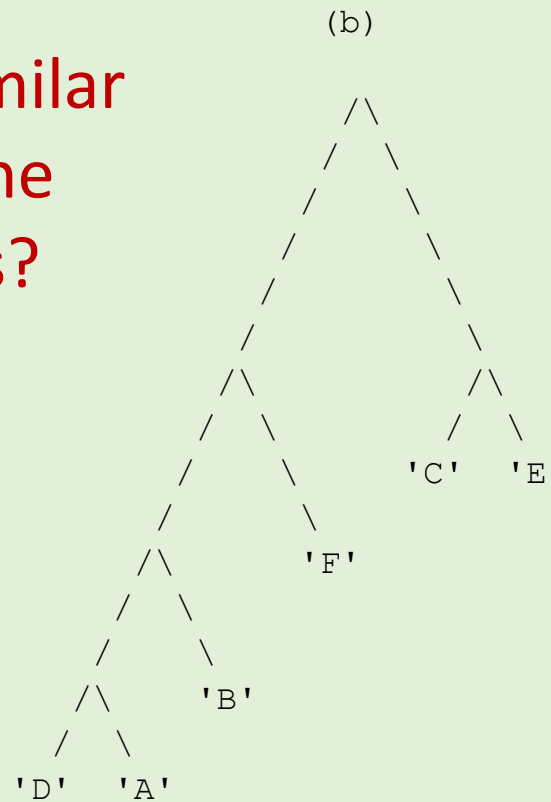


Handin 3 & 4 – Triplet distance (Dobson, 1975)

((('A', 'F'), 'B'), ('D', ('C', 'E')))) (((('D', 'A'), 'B'), 'F'), ('C', 'E')))



How similar
are the
trees?



Handin 3 & 4 – Triplet distance (Dobson, 1975)

Consider all $\binom{n}{3}$ subsets of size three, and count how many do not have identical substructure (topology) in the two trees.

