Python Dictionaries

Table of contents

Creating and indexing 1
`keys`, `values`, `items` 2
`clear` and `copy` 2
`get` 2
Warning: keys must be immutable 3

A **dictionary** is a data structure that stores `(key, value)` pairs. Python uses a "magic" hash function to find a key among the stored pairs quickly (see 2025-08-18-non-primitive-data[1] for how a hash function generalizes indexing-by-position to indexing-by-any-key). Dictionaries are **not ordered** (keys can be of mixed type) and are **mutable**.

Hash tables are a candidate for the most important/useful data structure in computer science.

Creating and indexing

```
>>> h = {
...     "red": ["apple", "firetrucks", "cars"],
...     "yellow": ["banana", "cars"],
...     "blue": ["sky", "cars"]
... }
>>> h["blue"]
['sky', 'cars']
>>> h["green"]
KeyError: 'green'
>>> h["green"] = ["leaves"]   # fine -- we're assigning, not retrieving
```

[1] courses/csse1001/2025-08-18-non-primitive-data.pdf

keys, values, items

```
>>> h.keys()
dict_keys(['red', 'yellow', 'blue'])
>>> h.values()
dict_values([['apple', 'firetrucks', 'cars'], ['banana', 'cars'], ['sky', 'cars']])
>>> h.items()
dict_items([('red', ['apple', 'firetrucks', 'cars']), ('yellow', ['banana', 'cars']), ('blue
```

clear and copy

```
>>> f = h.copy()
>>> h.clear()
>>> h["blue"]
KeyError: 'blue'
>>> f["blue"]
['sky', 'cars']
```

Like list's `.copy()` (see python-lists[2]), dict's `.copy()` is only a **shallow copy** — mutable values are still shared:

```
>>> f = h.copy()
>>> h["blue"].append("windex")
>>> h["blue"]
['sky', 'cars', 'windex']
>>> f["blue"]
['sky', 'cars', 'windex']   # shallow copy -- same underlying list
```

get

`.get(key)` is a safer alternative to `h[key]` — it returns `None` instead of raising `KeyError` if the key is missing:

```
>>> h.get("red")
['apple', 'firetrucks', 'cars']
>>> h["green"]
KeyError: 'green'
>>> h.get("green")
None
```

[2] courses/csse1001/python-lists.pdf

Without a method, the same "avoid a `KeyError`" pattern can be written explicitly:

```
>>> if key in table:
...     table[key] += 1
... else:
...     table[key] = 0
```

Warning: keys must be immutable

The keys of a dictionary must be an immutable type (see python-primitive-data-types[3] and python-lists[4]):

```
>>> d = dict()
>>> d[[1, 2, 3]] = 1
TypeError: unhashable type: 'list'
>>> d[(1, 2, 3)] = 1   # tuples are immutable -- fine
>>> d[dict()] = 1
TypeError: unhashable type: 'dict'
```

[3] courses/csse1001/python-primitive-data-types.pdf
[4] courses/csse1001/python-lists.pdf