**Dictionary and Sets (Fluent Python) chapter 3**

**Dict comprehension**

>>> dial\_codes = [

... (880, 'Bangladesh'),

... (55, 'Brazil'),

... (86, 'China'),

... (91, 'India'),

... (62, 'Indonesia'),

... (81, 'Japan'),

... (234, 'Nigeria'),

... (92, 'Pakistan'),

... (7, 'Russia'),

... (1, 'United States'),

... ]

>>> country\_dial = {country: code for code, country in dial\_codes}

>>> country\_dial

{'Bangladesh': 880, 'Brazil': 55, 'China': 86, 'India': 91, 'Indonesia': 62,

'Japan': 81, 'Nigeria': 234, 'Pakistan': 92, 'Russia': 7, 'United States': 1}

>>> {code: country.upper()

... for country, code in sorted(country\_dial.items())

... if code < 70}

{55: 'BRAZIL', 62: 'INDONESIA', 7: 'RUSSIA', 1: 'UNITED STATES'}

## Merging Mappings with |

>>> d1 = {'a': 1, 'b': 3}

>>> d2 = {'a': 2, 'b': 4, 'c': 6}

>>> d1 | d2

{'a': 2, 'b': 4, 'c': 6}

>>> d1

{'a': 1, 'b': 3}

>>> d1 |= d2

>>> d1

{'a': 2, 'b': 4, 'c': 6}

# Pattern Matching with Mappings

The match/case statement supports subjects that are mapping objects. Patterns for mappings look like dict literals, but they can match instances of any actual or virtual subclass of collections.abc.Mapping.[1](https://learning.oreilly.com/library/view/fluent-python-2nd/9781492056348/ch03.html" \l "idm46582495507536)

##### Example 3-2. creator.py: get\_creators() extracts names of creators from media records

def get\_creators(record: dict) -> list:

match record:

case {'type': 'book', 'api': 2, 'authors': [\*names]}: [](https://learning.oreilly.com/library/view/fluent-python-2nd/9781492056348/ch03.html#callout_dictionaries_and_sets_CO2-1)

return names

case {'type': 'book', 'api': 1, 'author': name}: [](https://learning.oreilly.com/library/view/fluent-python-2nd/9781492056348/ch03.html#callout_dictionaries_and_sets_CO2-2)

return [name]

case {'type': 'book'}: [](https://learning.oreilly.com/library/view/fluent-python-2nd/9781492056348/ch03.html#callout_dictionaries_and_sets_CO2-3)

raise ValueError(f"Invalid 'book' record: {record!r}")

case {'type': 'movie', 'director': name}: [](https://learning.oreilly.com/library/view/fluent-python-2nd/9781492056348/ch03.html#callout_dictionaries_and_sets_CO2-4)

return [name]

case \_: [](https://learning.oreilly.com/library/view/fluent-python-2nd/9781492056348/ch03.html#callout_dictionaries_and_sets_CO2-5)

raise ValueError(f'Invalid record: {record!r}')

Now let’s see how get\_creators handles some concrete doctests:

>>> b1 = dict(api=1, author='Douglas Hofstadter',

... type='book', title='Gödel, Escher, Bach')

>>> get\_creators(b1)

['Douglas Hofstadter']

>>> from collections import OrderedDict

>>> b2 = OrderedDict(api=2, type='book',

... title='Python in a Nutshell',

... authors='Martelli Ravenscroft Holden'.split())

>>> get\_creators(b2)

['Martelli', 'Ravenscroft', 'Holden']

>>> get\_creators({'type': 'book', 'pages': 770})

Traceback (most recent call last):

...

ValueError: Invalid 'book' record: {'type': 'book', 'pages': 770}

>>> get\_creators('Spam, spam, spam')

Traceback (most recent call last):

...

ValueError: Invalid record: 'Spam, spam, spam'

## What Is Hashable

## An object is hashable if it has a hash code which never changes during its lifetime (it needs a \_\_hash\_\_() method), and can be compared to other objects (it needs an \_\_eq\_\_() method).

>>> tt = (1, 2, (30, 40))

>>> hash(tt)

8027212646858338501

>>> tl = (1, 2, [30, 40])

>>> hash(tl)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unhashable type: 'list'

>>> tf = (1, 2, frozenset([30, 40]))

>>> hash(tf)

-4118419923444501110