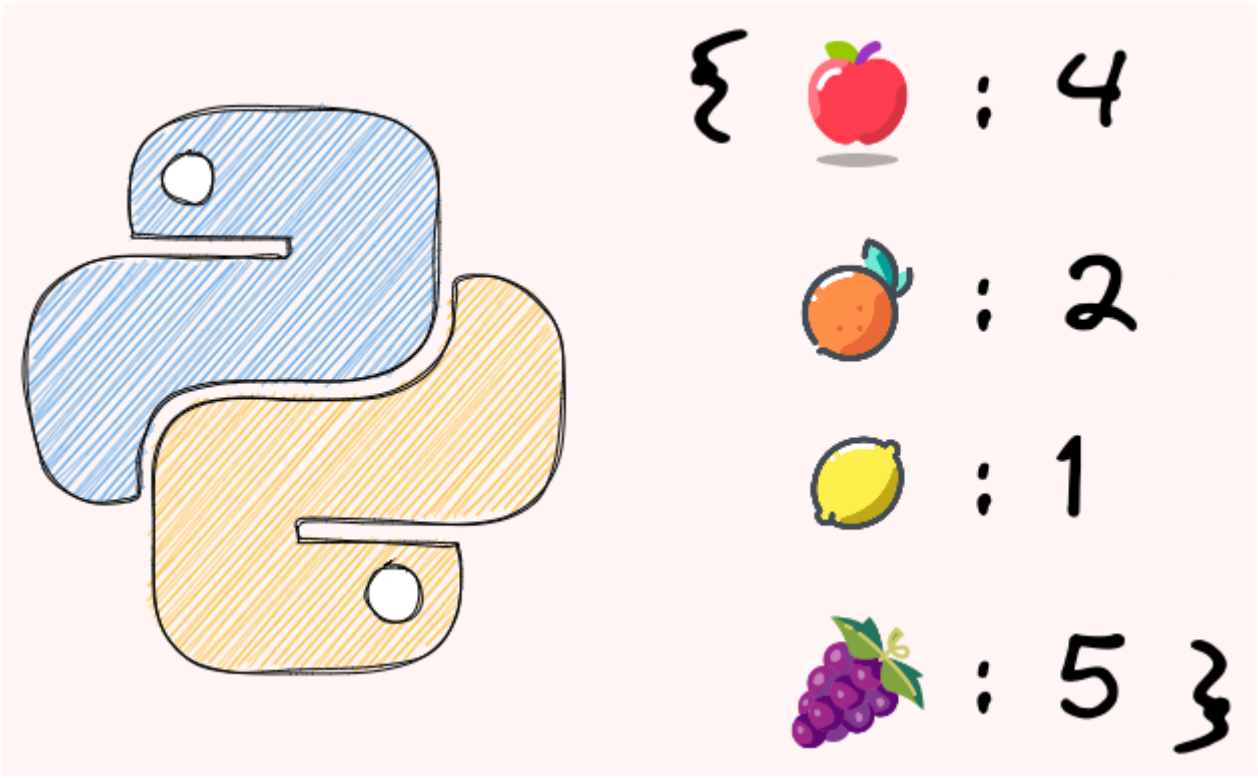
Efficient Python Tricks and Tools for Data Scientists - By Khuyen Tran

Dictionary



Update: Update a Dictionary With Items From Another Dictionary

If you want to update a dictionary with items from another dictionary or from an iterable of key/value pairs, use the `update` method.

```
birth_year = {"Ben": 1997}
new_birth_year = {"Michael": 1993, 'Lauren': 1999}
birth_year.update(new_birth_year)
```

```
birth_year.update(Josh=1990, Olivia=1991)
```

```
birth_year
```

```
{'Ben': 1997, 'Michael': 1993, 'Lauren': 1999, 'Josh': 1990, 'Olivia': 1991}
```

Key Parameter in Max(): Find the Key with the Largest Value

Apply max on a Python dictionary will give you the largest key, not the key with the largest value. If you want to find the key with the largest value, specify that using the `key` parameter in the `max` method.

```
birth_year = {"Ben": 1997, "Alex": 2000,
"Oliver": 1995}

max(birth_year)
```

```
'Oliver'
```

```
max_val = max(birth_year, key=lambda k:
birth_year[k])
max_val
```

```
'Alex'
```

dict.get: Get the Default Value of a Dictionary if a Key Doesn't Exist

If you want to get the default value when a key doesn't exist in a dictionary, use `dict.get`. In the code below, since there is no key `meeting3`, the default value `online` is returned.

```
locations = {'meeting1': 'room1', 'meeting2': 'room2'}
```

```
locations.get('meeting1', 'online')
```

```
'room1'
```

```
locations.get('meeting3', 'online')
```

```
'online'
```

Double dict.get: Get Values in a Nested Dictionary with Missing Keys

It can be challenging to get values in a nested dictionary with missing keys.

```
fruits = [
    {"name": "apple", "attr": {"color": "red",
"taste": "sweet"}},
    {"name": "orange", "attr": {"taste":
"sour"}},
    {"name": "grape", "attr": {"color":
"purple"}},
    {"name": "banana"},
]
```

You can use an if-else statement but it is long and hard to read.

```
colors = [
    fruit["attr"]["color"]
    if "attr" in fruit and "color" in
fruit["attr"]
    else "unknown"
    for fruit in fruits
]
colors
```

```
['red', 'unknown', 'purple', 'unknown']
```

A better way is to use the `get` method twice like below. The first `get` method will return an empty dictionary if the key `attr` doesn't exist. The second `get` method will return `unknown` if the key `color` doesn't exist.

```
colors = [fruit.get("attr", {}).get("color",
"unknown") for fruit in fruits]
colors
```

```
['red', 'unknown', 'purple', 'unknown']
```

dict.fromkeys: Get a Dictionary From a List and a Value

If you want to get a dictionary from a list and a value, try `dict.fromkeys`.

```
furnitures = ['bed', 'table', 'chair']
food = ['apple', 'pepper', 'onion']
loc1 = 'IKEA'
loc2 = 'ALDI'
```

For example, we can use `dict.fromkeys` to create a dictionary of furnitures' locations:

```
furniture_loc = dict.fromkeys(furnitures,
loc1)
furniture_loc
```

```
{'bed': 'IKEA', 'table': 'IKEA', 'chair':
'IKEA'}
```

... or create a dictionary of food's locations:

```
food_loc = dict.fromkeys(food, loc2)
food_loc
```

```
{'apple': 'ALDI', 'pepper': 'ALDI', 'onion':
'ALDI'}
```

These 2 results can be combined into a location dictionary like below:

```
locations = {**food_loc, **furniture_loc}
locations
```

```
{'apple': 'ALDI',
 'pepper': 'ALDI',
 'onion': 'ALDI',
 'bed': 'IKEA',
 'table': 'IKEA',
 'chair': 'IKEA'}
```