Dictionary Methods

Table of contents

01

Access & Iteration

Access & Iteration

01 — `get()` — Safely **retrieve** value of a key

02 — `items()` — Return **key-value pairs** for looping

03 — `keys()` — Return **all keys**

04 — `values()` — Return **all values**

05 — `setdefault()` — Return **value** if key exists, else **set default**

Access & Iteration – .get()

- **Safely retrieve** the value of a given **key**.
- Prevents **KeyError** if the key doesn't exist.

Python	Output

`person = {"name": "Amit", "age": 25}`
`print(person.get("name"))`
`print(person.get("gender", "N/A"))`
	`Amit`
	`N/A`

Access & Iteration - .items()

- returns **key-value** pairs as tuples

Python	Output

`data = {"x": 10, "y": 20}`
`print(data.items())`
 	
`for k, v in data.items():`
`    print(f"{k} = {v}")`
	`dict_items([('x', 10), ('y', 20)])`
	`x = 10`
	`y = 20`

Access & Iteration - .keys()

- returns **all** the dictionary **keys**

Python	**Output**

`data = {"x": 10, "y": 20}`
`print(data.keys())`
 	
`for k in data.keys():`
`    print(f"{k}")`
	`dict_keys(['x', 'y'])`
	`x`
	`y`

Access & Iteration - .values()

- returns **all** the **values**

Python	**Output**

`data = {"x": 10, "y": 20}`
`print(data.values())`
 	
`for v in data.values():`
`    print(v)`
	`dict_values([10, 20])`
	
	`10`
	`20`

Access & Iteration - .setdefault()

- Returns the **value** of a key **if it exists**.
- If not, **inserts** the key with the **given default value**.

Python

```
user = {"username": "amit123"}
print(user.setdefault("email", "not_provided@example.com"))
print(user)
```

Output

```
not_provided@example.com
{'username': 'amit123', 'email': 'not_provided@example.com'}
```

02

Update & Modification

Update & Modification

01	`update()`	**Merge** another dictionary or **overwrite keys**
02	`pop()`	**Remove key** and return its **value**
03	`popitem()`	**Remove** and **return** last inserted item
04	`clear()`	**Remove** all **items**
05	`fromkeys()`	**Create** new **dict** from **sequence** of keys

Update & Modification – .update()

- Adds or updates **key-value pairs** from another dictionary or iterable.

Python

```
info = {"name": "Raj", "age": 30}
info.update({"age": 31, "city": "Mumbai"})
print(info)
```

Output

```
{'name': 'Raj', 'age': 31, 'city': 'Mumbai'}
```

Update & Modification – .pop()

- **Removes** the item with the **specified key** and **returns** its value.
- **Raises error** if key doesn't exist (unless default is given).

Python

```
marks = {"Math": 90, "English": 85}
print(marks.pop("Math"))
print(marks.pop("Science", 0))
```

Output

```
90
0
```

Update & Modification – .popitem()

- **Removes** and **returns** the last inserted item as a **(key, value) tuple** (in Python 3.7+).

Python

```
colors = {"red": "#f00", "green": "#0f0"}
print(colors.popitem())
```

Output

```
('green', '#0f0')
```

Update & Modification – .clear()

- **Removes all items** from the dictionary.

Python

```
config = {"theme": "dark", "font": "Arial"}
config.clear()
print(config)
```

Output

```
{}
```

Update & Modification – .fromkeys()

- Creates a **new dictionary** with the **specified keys and same value** for each.

Python

```
fields = ["name", "email", "phone"]
default_user = dict.fromkeys(fields, "Not Provided")
print(default_user)
```

Output

```
{'name': 'Not Provided', 'email': 'Not Provided', 'phone': 'Not Provided'}
```

03

Utility

Utility Methods

01 — `copy()` — Shallow copy of dictionary

Utility - .copy()

- Returns a **shallow copy** of the dictionary (not for nested structures).

Python

```
original = {"a": 1, "b": 2}
duplicate = original.copy()
duplicate["a"] = 99
print(original)
```

Output

```
{'a': 1, 'b': 2}
```

Summary

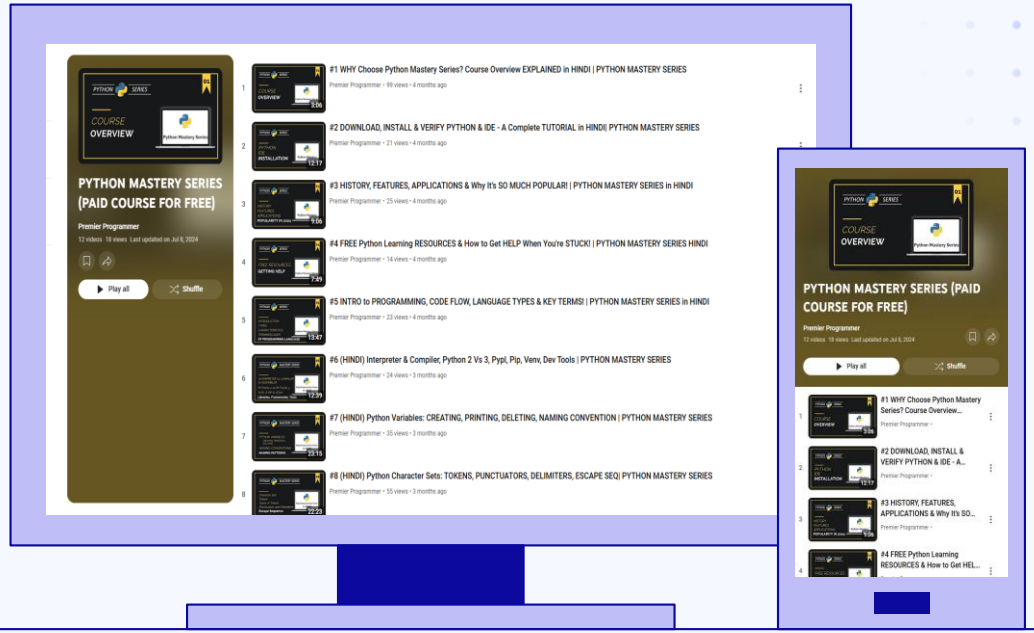
Use Case	Method(s)		
Value retrieval	`.get()`	`.setdefault()`	
Iteration	`.keys()`	`.values()`	`.items()`
Merge dictionaries	`.update()`		
Remove entries	`.pop()`	`.popitem()`	`.clear()`
Clone dictionary	`.copy()`		
Create from keys	`.fromkeys()`		

Practice Set - 3

Download Link in **Description** and **Pinned Comment**

WATCH

Level up your coding with each episode in this focused Python series.



Next Video!

Practice Set - 3
Solution