Nested Dictionary

Table of contents

Introduction

- A **nested dictionary** is a dictionary **inside** another dictionary.
- It allows you to represent **complex hierarchical data**—like records, configurations, or structured data tables—where each **key** in the **outer dictionary** maps to **another dictionary** (or even deeper structures).

Syntax

```
nested_dict = {
    "item1": {"key1": value1, "key2": value2},
    "item2": {"key1": value3, "key2": value4}
}
```

Accessing Elements in a Nested Dictionary

- Use **multiple keys** in sequence to access deeper values

Python

```
students = {
    "101": {"name": "Amit", "age": 21, "grade": "A"},
    "102": {"name": "Sara", "age": 22, "grade": "B"},
}

print(students["101"])
print(students["101"]["name"])
```

Output

```
{'name': 'Amit', 'age': 21, 'grade': 'A'}
Amit
```

Accessing Elements in a Nested Dictionary

- The **get() method** is used to **safe-access** the value associated with the specified key. If the key does **not** exist, it returns **None**, or a **default value** if specified

Python

```
students = {
    "101": {"name": "Amit", "age": 21, "grade": "A"},
    "102": {"name": "Sara", "age": 22, "grade": "B"},
}
name = students.get("101", {}).get("name", "Not Found")
print(name)
```

Output

```
Amit
```

Modifying Nested Dictionary Values

- You can update **inner values** just like in a normal dictionary:

Python

```
students = {
    "101": {"name": "Amit", "age": 21, "grade": "A"},
    "102": {"name": "Sara", "age": 22, "grade": "B"},
}
students["102"]["grade"] = "A+"
print(students)
students["101"]["major"] = "Physics"
print(students)
```

Output

```
{'101': {'name': 'Amit', 'age': 21, 'grade': 'A'}, '102': {'name': 'Sara', 'age': 22,
'grade': 'A+'}}
{'101': {'name': 'Amit', 'age': 21, 'grade': 'A', 'major': 'Physics'}, '102': {'name':
'Sara', 'age': 22, 'grade': 'A+'}}
```

Adding New Records (Outer and Inner)

- Add a new **nested entry** by assigning a dictionary

Python

```
students = {
    "101": {"name": "Amit", "age": 21, "grade": "A"},
    "102": {"name": "Sara", "age": 22, "grade": "B"},
}
students["103"] = {"name": "Raj", "age": 20, "grade": "B+"}
print(students)
```

Output

```
{
'101': {'name': 'Amit', 'age': 21, 'grade': 'A'},
'102': {'name': 'Sara', 'age': 22, 'grade': 'B'},
'103': {'name': 'Raj', 'age': 20, 'grade': 'B+'}
}
```

Looping Through Nested Dictionaries

- Loop through **outer** keys

Python

```
students = {
    "101": {"name": "Amit", "age": 21, "grade": "A"},
    "102": {"name": "Sara", "age": 22, "grade": "B"},
}

for roll_no in students:
    print("Roll No:", roll_no)
    print("Details:", students[roll_no])
```

Output

```
Roll No: 101
Details: {'name': 'Amit', 'age': 21, 'grade': 'A'}
Roll No: 102
Details: {'name': 'Sara', 'age': 22, 'grade': 'B'}
```

Looping Through Nested Dictionaries

- Loop through **inner keys**

Python	Output

`students = {`
`  "101": {"name": "Amit", "age": 21, "grade": "A"},`
`  "102": {"name": "Sara", "age": 22, "grade": "B"},`
`}`
`for roll_no, details in students.items():`
`  print(f"Student {roll_no}:")`
`  for key, value in details.items():`
`  print(f"  {key}: {value}")`
	`Student 101:`
	`  name: Amit`
	`  age: 21`
	`  grade: A`
	
	`Student 102:`
	`  name: Sara`
	`  age: 22`
	`  grade: B`

Deleting Items from Nested Dictionaries

- Delete an **inner** key:

Python

```
students = {
    "101": {"name": "Amit", "age": 21, "grade": "A"},
    "102": {"name": "Sara", "age": 22, "grade": "B"},
}

del students["101"]["age"]
print(students)
```

Output

```
{
'101': {'name': 'Amit', 'grade': 'A'},
'102': {'name': 'Sara', 'age': 22, 'grade': 'B'}
}
```

Deleting Items from Nested Dictionaries

- Delete an **entire student record**

Python

```
students = {
    "101": {"name": "Amit", "age": 21, "grade": "A"},
    "102": {"name": "Sara", "age": 22, "grade": "B"},
}

del students["102"]
print(students)
```

Output

```
{
'101': {'name': 'Amit', 'age': 21, 'grade': 'A'}
}
```

Summary

Operation	Syntax Example
Access value	`dict["key1"]["key2"]`
Modify value	`dict["key1"]["key2"] = new_value`
Add new subkey	`dict["key1"]["new_key"] = value`
Add new record	`dict["new_key"] = {...}`
Delete inner value	`del dict["key1"]["key2"]`
Delete full entry	`del dict["key1"]`
Safe access	`dict.get("key1", {}).get("key2", default)`

WATCH

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

Next Video!

Dictionary Copy