DATA SCIENCE COURSE TUTORIAL # 29

3.17.5 Nested Dictionaries

What is a Nested Dictionary?

A **nested dictionary** in Python is a dictionary that contains one or more dictionaries as its values. It is used to store complex, structured data in a hierarchical form. This allows multiple levels of key-value storage, making it easier to organize and manage related information.

Example:

```
students = {
    "student1": {
        "name": "Ali",
        "age": 22,
        "grade": "A"
    },
    "student2": {
        "name": "Sara",
        "age": 21,
        "grade": "B"
    }
}
print(students)
```

Accessing Items in a Nested Dictionary

You can access values inside a nested dictionary by chaining keys.

Example:

```
print(students["student1"]["name"])   # Ali
print(students["student2"]["grade"])  # B
```

If you try to access a key that does not exist, Python will raise a **KeyError**.

Adding New Dictionaries or Items

You can add new inner dictionaries or new key-value pairs inside existing dictionaries.

Example:

```
students["student3"] = {
    "name": "Ahmed",
    "age": 23,
    "grade": "A+"
}

students["student1"]["country"] = "Pakistan"
print(students)
```

Changing Values in a Nested Dictionary

You can modify specific values by referring to their nested keys.

Example:

```
students["student2"]["grade"] = "A"
print(students["student2"])
```

Removing Items from a Nested Dictionary

You can remove an entire nested dictionary or a specific key-value pair.

Example:

```
del students["student3"]                # Removes the whole student3 dictionary
del students["student1"]["age"]         # Removes only 'age' key from student1
print(students)
```

Looping through a Nested Dictionary

You can use nested loops to iterate through keys and values inside nested dictionaries.

Example:

```
for key, value in students.items():
    print(key)               # Prints outer dictionary keys (student1, student2)
    for inner_key, inner_value in value.items():
        print(inner_key, ":", inner_value)   # Prints inner key-value pairs
```

Using Dictionary Methods with Nested Dictionaries

You can still use dictionary methods like `.keys()`, `.values()`, and `.items()` for outer or inner dictionaries.

Example:

```
print(students.keys())                # dict_keys(['student1', 'student2'])
print(students["student1"].keys())    # dict_keys(['name', 'grade', 'country'])
```

Copying Nested Dictionaries

When copying nested dictionaries, use the `copy()` method carefully. The `copy()` method creates a **shallow copy**, which means changes inside nested dictionaries may affect both copies.

To create a **deep copy** (completely independent copy), use the `copy` module.

Example:

```
import copy

new_students = copy.deepcopy(students)
new_students["student1"]["name"] = "Hassan"

print(students["student1"]["name"])   # Ali (unchanged)
print(new_students["student1"]["name"])  # Hassan
```