Newton School of Technology

18

Dictionaries – Part II

by Gladden Rumao

C14: Problem Solving with Programming

Quick Recap:

Newton School of Technology

- Dictionaries in Python
- Creating a Dictionary
- Accessing Dictionary
- Updating Dictionary
- keys() , values() , items()

Newton School of Technology

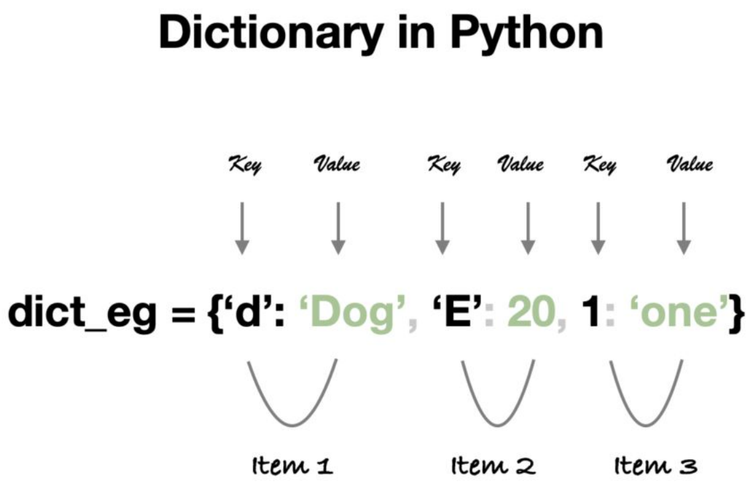
Iterating a Dictionary

Iterating Through Dictionaries

Newton School of Technology

"Dictionaries often contain data that we want to process or analyze."

"Looping through dictionaries allows us to access data stored inside and use it in various ways."



Looping Through Key-Value Pairs

Newton School of Technology

```python
for fruit, price in fruit_prices.items():
    print(f"{fruit} costs ${price}")
# Output:
# apple costs $0.99
# banana costs $0.49
# orange costs $0.79
```

Using **my_dict.items()** returns each **key-value pair** in the form of a tuple, which we can then unpack into key and value in the loop. This gives us access to both elements.

Looping Through Values Only

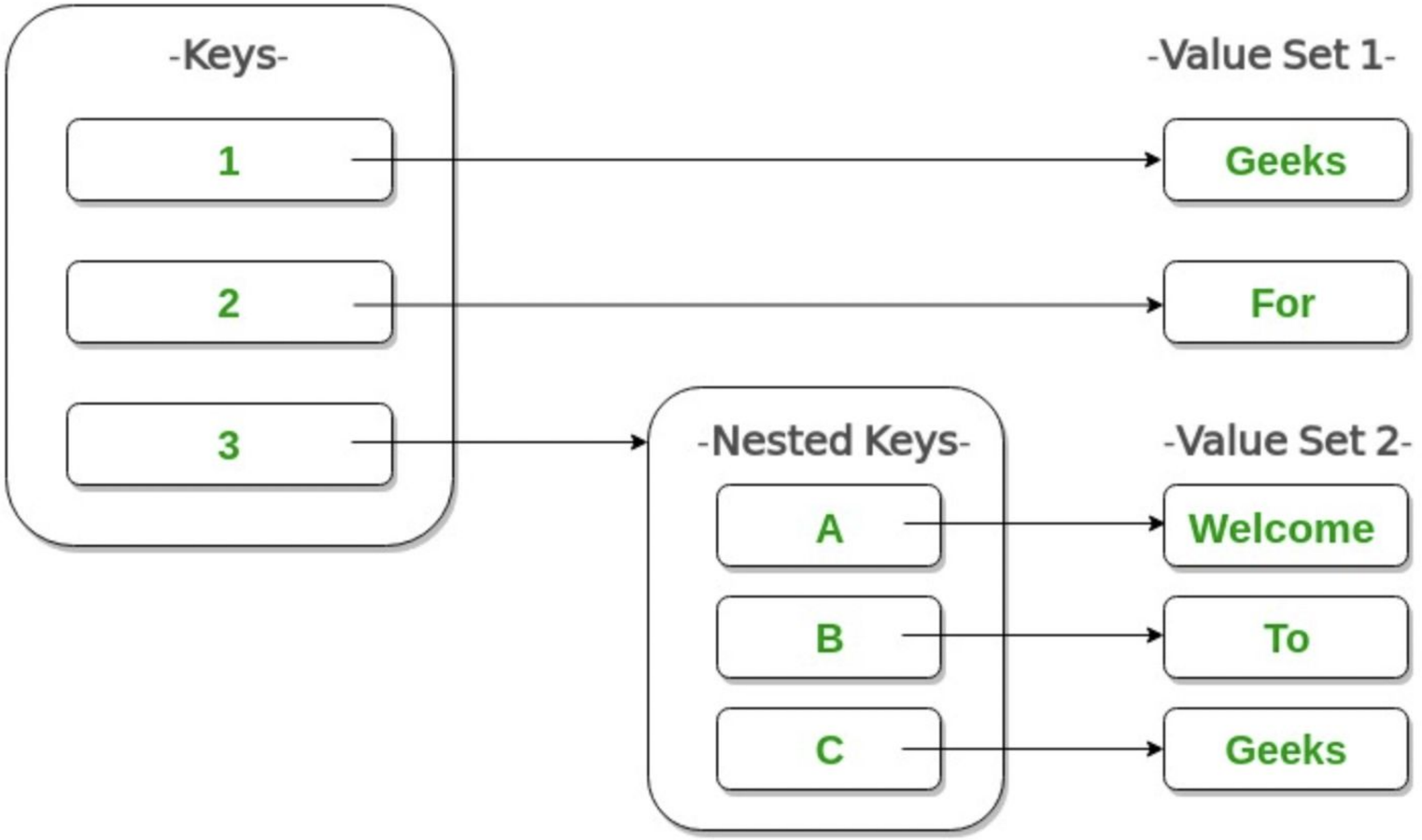
Newton School of Technology

```python
for price in fruit_prices.values():
    print(price)
# Output:
# 0.99
# 0.49
# 0.79
```

Using **my_dict.values()** retrieves only the values from the dictionary, leaving the keys out.

Nested Dictionaries

A **nested dictionary** is a dictionary that contains other dictionaries as values



Syntax of a Nested Dictionary

Newton School of Technology

The syntax for creating a nested dictionary is similar to a regular dictionary, but the values within it are also dictionaries.

```
# Basic syntax of a nested dictionary
nested_dict = {
    "key1": {"sub_key1": "value1", "sub_key2": "value2"},
    "key2": {"sub_key1": "value3", "sub_key2": "value4"}
}
```

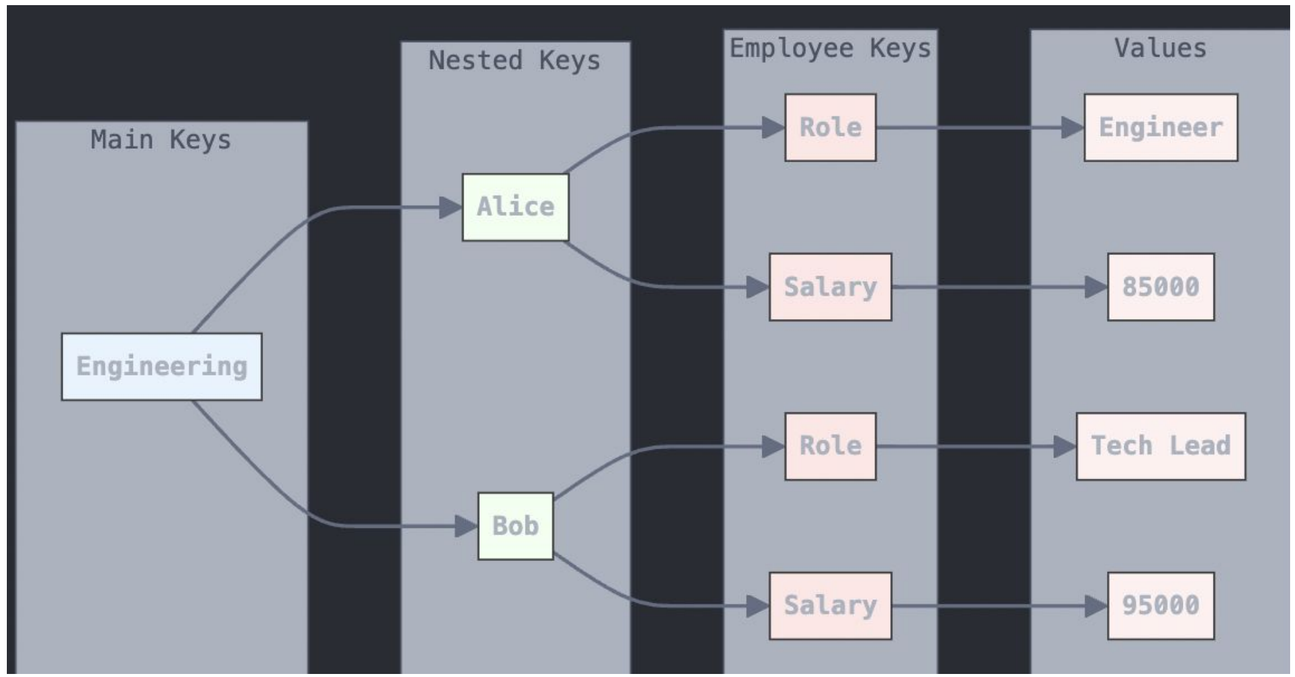
- **Outer Dictionary**: Contains main keys ("**key1**" and "**key2**" in the example above).
- **Inner Dictionaries**: Each key in the outer dictionary maps to another dictionary, containing its **own key-value pairs**.

Creating the Nested Dictionary

Newton School of Technology

```
company_data = {
    "Engineering": {
        "Alice": {"Role": "Engineer", "Salary": 85000},
        "Bob": {"Role": "Tech Lead", "Salary": 95000}
    },
    "Marketing": {
        "Charlie": {"Role": "Analyst", "Salary": 65000},
        "Daisy": {"Role": "Manager", "Salary": 78000}
    }
}
```

Visualizing the elements in Nested



- The **outer dictionary** has department names as keys, like "**Engineering**" and "**Marketing**".
- Each department **key maps to another dictionary**, which contains employees as **keys** (e.g., "Alice", "Bob").
- The **keys "Role" and "Salary"** describe the employee's job position and compensation, respectively, while the **values** are the actual data (like "Engineer" and 85000)

Accessing Elements in Nested

Newton School of Technology

To access elements in a nested dictionary, specify each key level to drill down to the specific value you need.

python

```python
role_bob = company_data["Engineering"]["Bob"]["Role"]
print(role_bob)  # Output: Tech Lead
```

python

```python
salary_daisy = company_data["Marketing"]["Daisy"]["Salary"]
print(salary_daisy)  # Output: 78000
```

Newton School of Technology

Dictionary Comprehension

Dictionary Comprehension

Newton School of Technology

Dictionary comprehension is a concise way to create dictionaries by applying an expression to an iterable. It is similar to list comprehensions but produces dictionaries instead of lists.

```python
{key_expression: value_expression for item in iterable if condition}
```

Where:

- **key_expression**: The key you want in the dictionary.
- **value_expression**: The value associated with that key.
- **iterable**: An iterable (like a list or range) over which the comprehension is executed.
- **condition (optional)**: A condition that filters which items to include in the dictionary.

Creating a Dictionary Comprehension

Newton School of Technology

```python
numbers = [1, 2, 3, 4, 5]
squares = {num: num ** 2 for num in numbers}
print(squares)
```

The comprehension loops through each number in the numbers list, computes the square, and stores it in a dictionary with the number as the key.

```python
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Thank You!