Advanced Dictionary Concepts

Introduction to Dictionary

A dictionary is a collection of key-value pairs, where each key is associated with a value. It is an unordered, mutable, and indexed data structure.

1. Basic Structure of a Dictionary

A dictionary is created by placing curly braces `{}` and separating keys and values with a colon `:`.

Example:

```
student = {'name': 'Hemant', 'age': 23, 'major': 'Computer Science'}
print(student)
```

```
Output:
{'name': 'Hemant', 'age': 23, 'major': 'Computer Science'}
```

2. Key Features of Dictionaries

- **Keys are unique:** No duplicate keys are allowed.
- **Values can be of any data type:** Including numbers, lists, tuples, or even another dictionary.
- **Keys must be immutable:** Only hashable data types like strings, numbers, or tuples can be keys. Mutable types like lists or other dictionaries cannot be used as keys.

3. Built-in Dictionary Methods

- `d.clear()`: Clears a dictionary.
- `d.get(<key>[, <default>])`: Returns the value for a key if it exists in the dictionary.
- `d.items()`: Returns a list of key-value pairs in a dictionary.
- `d.keys()`: Returns a list of keys in a dictionary.
- `d.values()`: Returns a list of values in a dictionary.

- `d.pop(<key>[, <default>])`: Removes a key from a dictionary, if present, and returns its value.
- `d.popitem()`: Removes a key-value pair from a dictionary.
- `d.update(<obj>)`: Merges a dictionary with another dictionary or with an iterable of key-value pairs.

Examples of Built-in Methods

Using `get()` Method

The `get()` method is used to retrieve the value for a specific key. If the key is not found, it returns the specified default value.

Example:

```
student = {'name': 'Hemant', 'age': 23}
print(student.get('name'))  # Retrieves the value for 'name'
print(student.get('major', 'Not Specified'))  # Returns default value
```

```
Output:
Hemant
Not Specified
```

Using `items()` Method

The `items()` method returns all key-value pairs as a list of tuples.

Example:

```
student = {'name': 'hemant', 'age': 23}
print(student.items())
```

```
Output:
dict_items([('name', 'Hemant'), ('age', 23)])
```

Using `keys()` and `values()` Methods

These methods retrieve all keys or values from the dictionary.

Example:

```
student = {'name': 'Hemant', 'age': 23}
print(student.keys())   # Retrieves all keys
print(student.values()) # Retrieves all values
```

```
Output:
dict_keys(['name', 'age'])
dict_values(['Hemant', 23])
```

Using Nested Dictionaries

Dictionaries can contain other dictionaries as values. This is useful for representing complex structures.

Example:

```
company = {
    'employee1': {'name': 'John', 'age': 30},
    'employee2': {'name': 'Jane', 'age': 25}
}
print(company['employee1']['name'])
```

```
Output:
John
```

Using `update()` Method

This method merges dictionaries or updates keys with new values.

Example:

```
student = {'name': 'Hemant', 'age': 23}
student.update({'major': 'Mathematics'})
```

```
print(student)
```

```
Output:
{'name': 'Hemant', 'age': 23, 'major': 'Mathematics'}
```