Python Dictionaries: Key Notes

1. Definition:
- A **dictionary** is a collection of **key-value pairs**.
- Each **key** is unique and maps to a **value**.

2. Key Characteristics:
- **Unordered**: No specific order (insertion order maintained in Python 3.7+).
- **Mutable**: Can modify (add, update, delete elements).
- **Fast Access**: O(1) time for search, insert, delete.
- **Keys**: Must be immutable (e.g., strings, numbers).
- **Values**: Can be of any type (int, string, list, etc.).

3. Creating a Dictionary:

```python
# Empty dictionary
my_dict = {}

# Dictionary with initial key-value pairs
my_dict = {
"name": "Alice",
"age": 25,
"city": "New York"
}
```

4. Accessing Values:

```python
# By key
```

```
print(my_dict["name"]) # Output: Alice

# Using get() to avoid KeyError
print(my_dict.get("age", "Not Found")) # Output: 25
```

5. Adding or Updating Values:

```python
# Add new key-value pair
my_dict["gender"] = "Female"

# Update existing key-value
my_dict["age"] = 26
```

6. Deleting Elements:

```python
# Using del to remove a key-value pair
del my_dict["city"]

# Using pop() to remove and return a value
age = my_dict.pop("age") # Output: 26

# Clear all elements from dictionary
my_dict.clear() # Empties dictionary
```

---

7. Iterating Through a Dictionary:

```python
# Iterate through keys
for key in my_dict:
print(key)

# Iterate through values
for value in my_dict.values():
print(value)

# Iterate through key-value pairs
for key, value in my_dict.items():
print(f"{key}: {value}")
```

---

8. Common Dictionary Methods:

- `get(key, default)`: Returns the value for `key`. Returns `default` if `key` not found.
- `keys()`: Returns a view object of all the keys.
- `values()`: Returns a view object of all the values.
- `items()`: Returns a view object of all key-value pairs.
- `pop(key)`: Removes the item with the specified key and returns its value.
- `update(other_dict)`: Updates the dictionary with key-value pairs from another dictionary.
- `clear()`: Removes all key-value pairs from the dictionary.

---

9. Example Code:

```python
```

```
# Dictionary storing student information
student = {
"name": "John",
"age": 22,
"major": "CS",
"courses": ["Math", "CS", "Physics"]
}

# Access values
print(student["name"]) # Output: John

# Add new key-value pair
student["graduation_year"] = 2025

# Update existing value
student["age"] = 23

# Iterate through dictionary
for key, value in student.items():
print(f"{key}: {value}")
```

---

10. Performance Considerations:
- **Time Complexity**:
- **O(1)** for access, insert, and delete operations on average.