Topics covered in the lecture:

1. Accessing dictionary items
2. Modifying dictionaries
3. in, not in, len(), pop(), popitem(), clear(), del, d.keys(), d.items(), d.values(), d.get()
4. Copy a dictionary
5. Nested dictionary
6. Creating objects using constructor
7. Common functions on objects: type(), all(), any(), len(), reversed(), sorted()
8. enumerate()
9. Mutable and Immutable Objects

Practice/Tutorial Questions:

Some output/error finding/mcq type questions for practice:

1. What will be the output of the following code?

```
d = {'x': 1, 'y': 2, 'z': 3}
print(d.pop('y', 0))
print(d)
```

   A) 2 and {'x': 1, 'z': 3}
   B) 0 and {'x': 1, 'y': 2, 'z': 3}
   C) 2 and {'x': 1, 'z': 3, 'y': 2}
   D) Error

2. What will be the output of the following code?

```
d = {'a': 1, 'b': 2}
d['b'] = d.get('c', 3)
print(d)
```

3. What will be the output of the following code?

```
d = {'x': 10, 'y': 20}
d.update({'y': 30, 'z': 40})
print(d)
```

4. What will the output be for the following code?

```
d = {True: 'yes', False: 'no'}
print(d[1])
```

A) yes
B) no
C) Error
D) None

5. What is the output of the following code?

```
d1 = {'a': [1, 2], 'b': 3, 'c':(4,5), 'd': {5:'x', 7:'y'}}
d2 = d1.copy()
d2['a'].append(4)
d2['b'] = 5
d2['c'] = (7,8)
d2['d'][5] = 'z'
print(d1)
print(d2)
```

Write code for the given problems:

1. Write a program that takes a list of words as input and creates a dictionary where each key is a letter, and the corresponding value is a list of words starting with that letter. Is the following solution correct?

Input: ['apple', 'banana', 'apricot', 'berry', 'blueberry']
Output: {'a': ['apple', 'apricot'], 'b': ['banana', 'berry', 'blueberry']}

2. Write a function to merge two dictionaries. If a key is present in both dictionaries:
    a. If the values are numbers, add them.
    b. If the values are lists, concatenate them.
    c. If the values are strings, concatenate them.
    d. If the values are tuples, concatenate them.
    e. If the values are other types, print incompatible types for those entries and merge the remaining dictionary.

    Inputs:

    ```
    dict1 = {'a': 5, 'b': [1, 2], 'c': 'hello', 'd': 42, 't' : (1,2)}

    dict2 = {'a': 10, 'b': [3, 4], 'c': ' world', 'e': 'new_key', 't': (5,7)}
    ```

    Output:

    ```
    {'a': 15, 'e': 'new_key', 't': (1, 2, 5, 7), 'c': 'hello world', 'd': 42, 'b': [1, 2, 3, 4]}
    ```

3. Given a dictionary where the values are lists, write a function to "invert" the dictionary by making the elements of the lists the keys and the original keys as the values in a list.

    Input: {'a': [1, 2], 'b': [2, 3]}
    Output: {1: ['a'], 2: ['a', 'b'], 3: ['b']}

4. Write a function that takes a list of strings and groups them into a dictionary where each key is a sorted version of a word and the value is a list of all words that are anagrams of each other. Assume that all letters are in the lowercase.

Input: ['eat', 'tea', 'tan', 'ate', 'nat', 'bat']
Output: {'aet': ['eat', 'tea', 'ate'], 'ant': ['tan', 'nat'], 'abt': ['bat']}

5. Write a function that compares two dictionaries and finds which keys are missing in the second dictionary but exist in the first one. The function should return a dictionary of missing keys, along with their values from the first dictionary.

   Input:
   dict1 = {'a': 1, 'b': 2, 'c': 3}
   dict2 = {'a': 1, 'b': 2}

   Output: {'c': 3}

6. Given two dictionaries, find the keys that are present in either dictionary but not both, and return the sum of their values.

   Input:
   dict1 = {'a': 1, 'b': 2, 'c': 3}
   dict2 = {'b': 4, 'c': 5, 'd': 6, 'e': 7}

   Output:
   Unique keys: {'e', 'a', 'd'}
   Sum of values of unique keys: 14

Some scenario based questions:

1. **Inventory Management System**

   Design a simple inventory management system using a dictionary, where each key represents an item in the inventory, and the value represents the quantity of that item available. The system should support the following operations:

   I. **Add Items:** You should be able to add a specified quantity of an item to the inventory. If the item does not exist, it should be added to the inventory with the given quantity.
   II. **Remove Items:** You should be able to remove a specified quantity of an item from the inventory. If the quantity to remove exceeds the current quantity, set the quantity to zero (effectively deleting the item from the inventory).
   III. **Display Inventory:** There should be a way to display the current inventory, showing each item and its available quantity.

   Dictionary structure:

   ```
   inventory = {'apple': 50, 'banana': 30}
   ```

2. **Word Frequency from Text with Custom Stopwords:** Write a program that reads a large text file and counts the frequency of each word. Allow the user to specify a list of "stopwords" that should be excluded from the count. The result should be a dictionary of word frequencies, excluding the stopwords. Assume that the stopwords in the “stopwords” list are in lowercase.

   Input:
   text = "This is a test. This test is only a test."
   stopwords = ['is', 'a', 'only']

Output:
Output: {'this': 2, 'test': 3}

3. Write a program that processes a list of data points representing items sold in various cities. For each city, count how many of each item has been sold. Use a multi-level dictionary (e.g., `city -> item -> count`) to store the results.

   Input:
   data = [
   ('New York', 'item_A'),
   ('New York', 'item_B'),
   ('Los Angeles', 'item_A'),
   ('New York', 'item_A'),
   ('Los Angeles', 'item_B'),
   ('New York', 'item_B'),
   ('Los Angeles', 'item_B'),
   ('Chicago', 'item_A')
   ]
   Output:
   {'New York': {'item_A': 2, 'item_B': 2}, 'Los Angeles': {'item_A': 1, 'item_B': 2}, 'Chicago': {'item_A': 1}}

4. **Library Management System:** You are developing a library management system. Create a dictionary where the keys are book titles and the values are another dictionary containing the author, publication year, and availability status (True for available, False for checked out). Write functions to:
   a. Add a new book to the library.
   b. Check out a book (update its availability status).
   c. Return a book (update its availability status).
   d. Display all available books.

   Dictionary structure:

```
library = {
    "The Great Gatsby": {
        "author": "F. Scott Fitzgerald",
        "year": 1925,
        "available": True
    },
    "1984": {
        "author": "George Orwell",
        "year": 1949,
        "available": True
    },
    "To Kill a Mockingbird": {
        "author": "Harper Lee",
        "year": 1960,
        "available": True
    }
}
```

5. **E-commerce Inventory Management:** You are building an inventory management system for an e-commerce platform. Create a dictionary where the keys are product IDs and the values are another dictionary containing product name, price, and stock quantity. Write functions to:
    - Add a new product to the inventory.
    - Update the stock quantity when a sale occurs.
    - Get the total value of the inventory.
    - Display all products in stock.

    Dictionary structure:

```
inventory = {
    'P001': {
        'name': 'Laptop',
        'price': 1000,
```

```
        'stock': 50
    },
    'P002': {
        'name': 'Smartphone',
        'price': 500,
        'stock': 0
    }
}
```

6. **Recipe Management System:** You are developing a recipe management system. Create a dictionary where the keys are recipe names and the values are another dictionary containing ingredients and their quantities. Write functions to:
    a. Add a new recipe.
    b. Update an ingredient quantity in a recipe.
    c. Find all recipes that use a specific ingredient.
    d. Remove a recipe from the collection.

Dictionary structure:

```
recipes = {
    'Pancakes': {
        'flour': '2 cups',
        'milk': '1 cup',
        'eggs': '2'
    },
    'Omelette': {
        'eggs': '3',
        'cheese': '1 cup',
        'milk': '1/4 cup'
    }
```

```
}
```

7. **Employee Database:** You are building an employee database for a company. Create a dictionary where each key is an employee ID and the value is another dictionary containing employee details such as name, department, and salary. Write functions to:
    a. Add a new employee.
    b. Update the salary of an existing employee
    c. List all employees in a specific department.
    d. Remove an employee from the database.

    Dictionary structure:

```
employee_database = {
    'E001': {
        'name': 'John Doe',
        'department': 'Engineering',
        'salary': 70000
    },
    'E002': {
        'name': 'Jane Smith',
        'department': 'Marketing',
        'salary': 60000
    },
    'E003': {
        'name': 'Emily Johnson',
        'department': 'Engineering',
        'salary': 75000
    }
}
```

8. **Fitness Tracker Application:** You are creating a fitness tracker application. Create a dictionary where the keys are users and the values are another dictionary containing their daily activity data (steps, calories burned, and active minutes). Write functions to:

a. Add a new day's activity for a user.
b. Calculate total steps and calories burned for a user over a week.
c. Find the user with the highest total steps.
d. Display the activity data for a specific user.

Dictionary structure:

```
fitness_data = {
    'Alice': {
        '2024-10-01': {'steps': 10000, 'calories': 500,
'active_minutes': 30},
        '2024-10-02': {'steps': 8000, 'calories': 400,
'active_minutes': 25}
    },
    'Bob': {
        '2024-10-01': {'steps': 12000, 'calories': 600,
'active_minutes': 40}
    }
}
```

9. **Shopping Cart System:** You are implementing a shopping cart system for an online store. Create a dictionary where each key is a product ID and the value is another dictionary containing product name, price, and quantity in the cart. Write functions to:
    a. Add a product to the cart.
    b. Update the quantity of a product.
    c. Remove a product from the cart.
    d. Calculate the total cost of items in the cart.

    Structure of the shopping cart:
    cart = { 'P001': {'name': 'Laptop', 'price': 1000, 'quantity': 1}, 'P002': {'name': 'Smartphone', 'price': 500, 'quantity': 2} }

10. **Event Management System:** You are creating an event management system. Create a dictionary where the keys are event names and the values are another dictionary containing event details such as date, location, and the number of attendees. Write functions to:
    a. Add a new event.
    b. Update the number of attendees for an event.
    c. Find all events occurring on a specific date.
    d. Remove an event from the system

    Example structure:

```
events = {
    'Music Concert': {
        'date': '2024-11-05',
        'location': 'City Hall',
        'attendees': 200
    },
    'Art Exhibition': {
        'date': '2024-11-10',
        'location': 'Art Gallery',
        'attendees': 150
    }
}
```