# Exercise 1: List Operations  
# 1. Create a list called `numbers` containing the numbers `1`, `2`, `3`, `4`, and `5`.  
  
numbers = [1, 2, 3, 4, 5]  
# 2. Append the number `6` to the list.  
numbers.append(6)  
# 3. Remove the number `3` from the list.  
numbers.remove(3)  
# 4. Insert the number `0` at the beginning of the list.  
numbers.insert(0,0)  
# 5. Print the final list  
print(numbers)  
  
# Exercise 2: Tuple Operations  
# 1. Create a tuple called `coordinates` containing the elements `10.0`, `20.0`, and `30.0`.  
coordinates = (10.0,20.0,30.0)  
# 2. Access and print the second element of the tuple.  
print(coordinates[1])  
# 3. Try to change the third element of the tuple to `40.0`. What happens?  
#coordinates[2] = 40.0  
# Throws error  
  
# Exercise 3: Set Operations  
# 1. Create a set called `fruits` containing `"apple"`, `"banana"`, `"cherry"`.  
fruits = {"apple", "banana", "cherry"}  
# 2. Add `"orange"` to the set.  
fruits.add("orange")  
# 3. Remove `"banana"` from the set.  
fruits.remove("banana")  
# 4. Check if `"cherry"` is in the set and print a message based on the result.  
if "cherry" in fruits:  
 print("cherry is present")  
else:  
 print("cherry is not present")  
# 5. Create another set called `citrus` with elements `"orange"`, `"lemon"`, `"lime"`.  
citrus = {"orange", "lemon", "lime"}  
# 6. Perform a union of `fruits` and `citrus` and print the result.  
print(fruits.union(citrus))  
# 7. Perform an intersection of `fruits` and `citrus` and print the result.  
print(fruits.intersection(citrus))  
  
# Exercise 4: Dictionary Operations  
# 1. Create a dictionary called `person` with keys `"name"`, `"age"`, and `"city"`, and values `"John"`, `30`, and `"New York"`, respectively.  
person = {  
 "name":"John",  
 "age":30,  
 "city":"New York"  
}  
# 2. Access and print the `"name"` key from the dictionary.  
print(person["name"])  
# 3. Update the `"age"` key to `31`.  
person["age"] = 31  
# 4. Add a new key-value pair `"email": "john@example.com"` to the dictionary.  
person["email"]= "john@example.com"  
# 5. Remove the `"city"` key from the dictionary.  
del person["city"]  
# 6. Print the final dictionary  
print(person)  
  
# Exercise 5: Nested Dictionary  
# 1. Create a dictionary called `school` where the keys are student names and the values are dictionaries containing the subjects and their corresponding grades. Example structure:  
school = {  
 "Alice": {"Math": 90, "Science": 85},  
 "Bob": {"Math": 78, "Science": 92},  
 "Charlie": {"Math": 95, "Science": 88}  
}  
# 2. Print the grade of `"Alice"` in `"Math"`.  
print(school["Alice"]["Math"])  
# 3. Add a new student `"David"` with grades `"Math": 80` and `"Science": 89`.  
school["David"] = {"Math":80,"Science":89}  
# 4. Update `"Bob"`'s `"Science"` grade to 95.  
school["Bob"]["Science"] = 95  
# 5. Print the final `school` dictionary.  
print(school)  
  
# Exercise 6: List Comprehension  
# 1. Given a list of numbers `[1, 2, 3, 4, 5]`, use list comprehension to create a new list where each number is squared.  
numbers = [1,2,3,4,5]  
new\_list = [i\*\*2 for i in numbers]  
# 2. Print the new list.  
print(new\_list)  
  
# Exercise 7: Set Comprehension  
# 1. Create a set comprehension that generates a set of squared numbers from the list `[1, 2, 3, 4, 5]`.  
numbers2 = [1,2,3,4,5]  
result\_set = {i\*\*2 for i in numbers2}  
# 2. Print the resulting set  
print(result\_set)  
  
# Exercise 8: Dictionary Comprehension  
# 1. Create a dictionary comprehension that generates a dictionary where the keys are the numbers from `1` to `5`, and the values are the cubes of the keys.  
cube\_dict = {i:i\*\*3 for i in range(1,6)}  
# 2. Print the resulting dictionary  
print(cube\_dict)  
  
# Exercise 9: Combining Collections  
# 1. Create two lists: `keys = ["name", "age", "city"]` and `values = ["Alice", 25, "Paris"]`.  
keys = ["name", "age", "city"]  
values = ["Alice", 25, "Paris"]  
# 2. Use the `zip()` function to combine the `keys` and `values` lists into a dictionary.  
combine = {keys:values for keys,values in zip(keys,values)}  
# 3. Print the resulting dictionary  
print(combine)  
  
# Exercise 10: Count Word Occurrences (Using a Dictionary)  
# 1. Write a Python program that takes a string as input and counts the occurrences of each word in the string using a dictionary. Example input:  
  
sentence = "the quick brown fox jumps over the lazy dog the fox"  
words = sentence.split()  
count\_word = {}  
for word in words:  
 count = count\_word.get(word,0)  
 count\_word[word] = count + 1  
# 2. Print the resulting dictionary with word counts  
print(count\_word)  
  
# Exercise 11: Unique Elements in Two Sets  
# 1. Create two sets: `set1 = {1, 2, 3, 4, 5}` and `set2 = {4, 5, 6, 7, 8}`.  
set1 = {1, 2, 3, 4, 5}  
set2 = {4, 5, 6, 7, 8}  
# 2. Find and print the unique elements in both sets combined.  
print(set1.union(set2))  
# 3. Find and print the common elements between the two sets.  
print(set1.intersection(set2))  
# 4. Find and print the elements that are only in `set1` but not in `set2`.  
print(set1.difference(set2))  
  
# Exercise 12: Tuple Unpacking  
# 1. Create a tuple with three elements: `("Alice", 25, "Paris")`.  
tuple1 = ("Alice", 25, "Paris")  
# 2. Unpack the tuple into three variables: `name`, `age`, and `city`.  
name, age, city = tuple1  
# 3. Print the variables to verify the unpacking  
print(name," ",age," ",city)  
  
#Exercise 13: Frequency Counter with Dictionary  
# 1. Write a Python program that counts the frequency of each letter in a given string using a dictionary. Example string:  
  
text = "hello world"  
text\_list = text.split()  
letter\_count = {}  
for i in text\_list:  
 for j in range(0,len(i)):  
 count = letter\_count.get(i[j],0)  
 letter\_count[i[j]] = count + 1  
  
# 2. Print the resulting dictionary with letter frequencies.  
print(letter\_count)  
  
# Exercise 14: Sorting a List of Tuples  
# 1. Given a list of tuples representing students and their grades:  
  
students = [("Alice", 90), ("Bob", 80), ("Charlie", 85)]  
  
# 2. Sort the list by grades in descending order and print the sorted list  
students.sort(key=lambda x: x[1], reverse=True)  
print(students)