**TASK(27.08.2024)**

**BY AMITA C**

**### 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]

1. **Append the number `6` to the list.**

numbers.append(6)

1. **Remove the number `3` from the list.**

numbers.remove(3)

1. **Insert the number `0` at the beginning of the list.**

numbers.insert(0, 0)

1. **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)

1. **Access and print the second element of the tuple.**

print(coordinates[1])

1. **Try to change the third element of the tuple to `40.0`. What happens?**

try: coordinates[2] = 40.0 except TypeError as e: print(f"Error: {e}")

**### Exercise 3: Set Operations**

1. **Create a set called `fruits` containing `"apple"`, `"banana"`, `"cherry"`.**

fruits = {"apple", "banana", "cherry"}

1. **Add `"orange"` to the set.**

fruits.add("orange")

1. **Remove `"banana"` from the set.**

fruits.remove("banana")

1. **Check if `"cherry"` is in the set and print a message based on the result.**

if "cherry" in fruits: print("Cherry is in the set.") else: print("Cherry is not in the set.")

1. **Create another set called `citrus` with elements `"orange"`, `"lemon"`, `"lime"`.**

citrus = {"orange", "lemon", "lime"}

1. **Perform a union of `fruits` and `citrus` and print the result.**

union\_set = fruits.union(citrus) print("Union:", union\_set)

1. **Perform an intersection of `fruits` and `citrus` and print the result.**

intersection\_set = fruits.intersection(citrus) print("Intersection:", intersection\_set)

**### 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" }

1. **Access and print the `"name"` key from the dictionary.**

print(person["name"])

1. **Update the `"age"` key to `31`.**

person["age"] = 31

1. **Add a new key-value pair `"email": "john@example.com"` to the dictionary.**

person["email"] = "john@example.com"

1. **Remove the `"city"` key from the dictionary.\**

person.pop("city")

1. **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:**

**```python**

**school = {**

**"Alice": {"Math": 90, "Science": 85},**

**"Bob": {"Math": 78, "Science": 92},**

**"Charlie": {"Math": 95, "Science": 88}**

**}**

**```**

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] squared\_numbers = [x\*\*2 for x in numbers]

1. **Print the new list.**

print(squared\_numbers)

**### Exercise 7: Set Comprehension**

1. **Create a set comprehension that generates a set of squared numbers from the list `[1, 2, 3, 4, 5]`.**

squared\_set = {x\*\*2 for x in [1, 2, 3, 4, 5]}

1. **Print the resulting set.**

print(squared\_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.**

cubes = {x: x\*\*3 for x in range(1, 6)}

1. **Print the resulting dictionary.**

print(cubes)

**### 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"]

1. **Use the `zip()` function to combine the `keys` and `values` lists into a dictionary.**

combined\_dict = dict(zip(keys, values))

1. **Print the resulting dictionary.**

print(combined\_dict)

**### 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:**

**```python**

**sentence = "the quick brown fox jumps over the lazy dog the fox"**

**```**

sentence = "the quick brown fox jumps over the lazy dog the fox"

words = sentence.split()

word\_count = {}

for word in words:

word\_count[word] = word\_count.get(word, 0) + 1

1. **Print the resulting dictionary with word counts.**

print(word\_count)

**### 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}

1. **Find and print the unique elements in both sets combined.**

unique\_elements = set1.union(set2)

print("Unique elements:", unique\_elements)

1. **Find and print the common elements between the two sets.**

common\_elements = set1.intersection(set2)

print("Common elements:", common\_elements)

1. **Find and print the elements that are only in `set1` but not in `set2`.**

only\_in\_set1 = set1.difference(set2)

print("Elements only in set1:", only\_in\_set1)

**### Exercise 12: Tuple Unpacking**

1. **Create a tuple with three elements: `("Alice", 25, "Paris")`.**

info = ("Alice", 25, "Paris")

1. **Unpack the tuple into three variables: `name`, `age`, and `city`.**

name, age, city = info

1. **Print the variables to verify the unpacking.**

print(f"Name: {name}, Age: {age}, City: {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:**

**```python**

**text = "hello world"**

**```**

text = "hello world"

letter\_frequency = {}

for letter in text: if letter != " ": # Skip spaces

letter\_frequency[letter] = letter\_frequency.get(letter, 0) + 1

1. **Print the resulting dictionary with letter frequencies.**

print(letter\_frequency)

**### Exercise 14: Sorting a List of Tuples**

**1. Given a list of tuples representing students and their grades:**

**```python**

**students = [("Alice", 90), ("Bob", 80), ("Charlie", 85)]**

**```**

students = [("Alice", 90), ("Bob", 80), ("Charlie", 85)]

1. **Sort the list by grades in descending order and print the sorted list.**

sorted\_students = sorted(students, key=lambda student: student[1], reverse=True) print(sorted\_students)