```python
# ✅ Looping Through Dictionary

# Sample dictionary
my_dict = {'a': 10, 'b': 25, 'c': 15, 'd': 5}

# 1. Loop through a dictionary and print each key and value
for key, value in my_dict.items():
    print(key, ":", value)

# 2. Find and print all keys that have values greater than a given number
threshold = 10
for key, value in my_dict.items():
    if value > threshold:
        print(f"Key with value > {threshold}:", key)

# 3. Count the occurrences of each value in a dictionary
from collections import Counter
value_count = Counter(my_dict.values())
print("Value occurrences:", value_count)

# 4. Filter out dictionary items where values are less than a threshold
filtered_dict = {k: v for k, v in my_dict.items() if v >= threshold}
print("Filtered dictionary:", filtered_dict)

# 5. Print dictionary items in sorted order of their keys
for key in sorted(my_dict):
    print(f"{key}: {my_dict[key]}")
# ✅ String Slicing & Iteration

text = "HelloWorldPython"

# 1. Extract the first 5 and last 5 characters
print("First 5:", text[:5])
print("Last 5:", text[-5:])

# 2. Remove every alternate character
print("Alternate characters removed:", text[::2])

# 3. Extract the substring from index 2 to 7
print("Substring (2 to 7):", text[2:8])

# 4. Iterate through a string and print each character
for char in text:
    print(char)

# 5. Reverse a string using slicing
print("Reversed string:", text[::-1])
# ✅ String Strip Functions

sample = "   Hello World   "
special = "###Welcome###"

# 1. Remove leading spaces
print("No leading space:", sample.lstrip())

# 2. Remove trailing spaces
print("No trailing space:", sample.rstrip())

# 3. Strip both leading and trailing spaces
print("Stripped:", sample.strip())

# 4. Remove specific characters (#) from the start
print("Remove #: start:", special.lstrip('#'))

# 5. Remove specific characters (#) from the end
print("Remove #: end:", special.rstrip('#'))
```

```python
# 6. List comprehension: squares of numbers from 1 to 10
squares = [x**2 for x in range(1, 11)]
print("Squares 1 to 10:", squares)

# 7. List comprehension: even numbers from 1 to 20
evens = [x for x in range(1, 21) if x % 2 == 0]
print("Even numbers:", evens)
# ✅ List Operations (Homework)

nums = [10, 20, 30, 40, 50, 60, 70]

# 1. Slice to extract first 3 and last 3 elements
print("First 3:", nums[:3])
print("Last 3:", nums[-3:])

# 2. Add, remove, and change an item
nums.append(80)
print("After append:", nums)
nums.remove(40)
print("After remove:", nums)
nums[2] = 99
print("After change:", nums)

# 3. Copy a list and modify
copy_nums = nums.copy()
copy_nums[0] = 100
print("Original list:", nums)
print("Modified copy:", copy_nums)

# 4. Join a list into sentence
words = ["Python", "is", "awesome"]
sentence = " ".join(words)
print("Sentence:", sentence)

# 5. Iterate through list
for item in nums:
    print("Element:", item)

# 6. Print only even numbers
evens = [x for x in nums if x % 2 == 0]
print("Even numbers:", evens)

# 7. Use all() to check if all numbers are positive
print("All positive:", all(x > 0 for x in nums))

# 8. Use any() to check if any number > 50
print("Any > 50:", any(x > 50 for x in nums))

# 9. all() and any() for strings
names = ["apple", "banana", "apricot"]
print("All start with a:", all(name.startswith("a") for name in names))
print("Any ends with e:", any(name.endswith("e") for name in names))
# ✅ Built-in Functions

numbers = [4, 9, 16, 25, 36]

# 1. min, max, length, and sum
print("Min:", min(numbers))
print("Max:", max(numbers))
print("Length:", len(numbers))
print("Sum:", sum(numbers))

# 2. Find the longest word
words = ["apple", "banana", "cherry", "grapefruit"]
longest = max(words, key=len)
```

```python
print("Longest word:", longest)

# 3. Sum of even numbers
even_sum = sum(x for x in numbers if x % 2 == 0)
print("Sum of evens:", even_sum)

# 4. Shortest string
shortest = min(words, key=len)
print("Shortest word:", shortest)

# 5. Sort list of tuples by second element
pairs = [(1, 3), (2, 1), (5, 2)]
sorted_pairs = sorted(pairs, key=lambda x: x[1])
print("Sorted tuples:", sorted_pairs)
# ✅ Dictionary Operations

student = {"name": "Alice", "age": 22}

# 1. Access an item by key
print("Name:", student["name"])

# 2. Add new key-value
student["grade"] = "A"
print("Updated dict:", student)

# 3. Remove an item
del student["age"]
print("After removal:", student)

# 4. Copy dictionary and modify
copy_student = student.copy()
copy_student["name"] = "Bob"
print("Original:", student)
print("Modified copy:", copy_student)

# 5. Loop through and print
for key, value in student.items():
    print(key, "->", value)
```

```python
# ✅ 1. Analyzing Sales Data

from functools import reduce

sales = [120, 75, 300, 450, 80, 150]

# 1. Filter sales above threshold (e.g., $100)
high_sales = list(filter(lambda x: x > 100, sales))
print("Sales > $100:", high_sales)

# 2. Apply 10% discount to all sales
discounted_sales = list(map(lambda x: x * 0.9, sales))
print("Discounted sales:", discounted_sales)

# 3. Compute total sales
total_sales = reduce(lambda x, y: x + y, sales)
print("Total sales amount:", total_sales)
# ✅ 2. Student Grades Processing

from functools import reduce

scores = [45, 78, 88, 59, 92, 30, 66]

# 1. Filter passing grades (pass mark = 60)
passing = list(filter(lambda x: x >= 60, scores))
print("Passing grades:", passing)

# 2. Curve scores by adding 5 points
curved = list(map(lambda x: x + 5, scores))
print("Curved scores:", curved)

# 3. Find highest score
highest = reduce(lambda x, y: x if x > y else y, scores)
print("Highest score:", highest)
# ✅ 3. E-commerce Product Price Analysis

from functools import reduce

prices = [250, 100, 499, 50, 700, 120]

# 1. Apply 20% discount to all
discounted = list(map(lambda x: x * 0.8, prices))
print("Discounted prices:", discounted)

# 2. Filter expensive items (above 300 after discount)
expensive = list(filter(lambda x: x > 300, discounted))
print("Expensive items:", expensive)

# 3. Calculate total revenue
total_revenue = reduce(lambda x, y: x + y, discounted)
print("Total revenue:", total_revenue)
# ✅ 4. Word Processing: Sentence Transformation

from functools import reduce

words = ["python", "is", "powerful", "and", "flexible"]

# 1. Filter words longer than 5 characters
long_words = list(filter(lambda w: len(w) > 5, words))
print("Long words:", long_words)

# 2. Capitalize all words
capitalized = list(map(lambda w: w.capitalize(), words))
print("Capitalized:", capitalized)

# 3. Count total characters in all words
```

```python
total_chars = reduce(lambda x, y: x + len(y), words, 0)
print("Total characters:", total_chars)
# ✅ 5. Employee Salary Processing

from functools import reduce

salaries = [3000, 4500, 6000, 2800, 7500]

# 1. Apply 10% bonus
bonus_salaries = list(map(lambda s: s * 1.1, salaries))
print("With bonus:", bonus_salaries)

# 2. Filter high earners (above $5000)
high_earners = list(filter(lambda s: s > 5000, bonus_salaries))
print("High earners:", high_earners)

# 3. Find the highest salary
highest_salary = reduce(lambda x, y: x if x > y else y, bonus_salaries)
print("Highest salary:", highest_salary)
# ✅ 6. List of Numbers: Even Filtering and Summation

from functools import reduce

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# 1. Filter even numbers
evens = list(filter(lambda x: x % 2 == 0, numbers))
print("Even numbers:", evens)

# 2. Square all numbers
squared = list(map(lambda x: x ** 2, numbers))
print("Squared numbers:", squared)

# 3. Sum all squared numbers
total_sum = reduce(lambda x, y: x + y, squared)
print("Sum of squares:", total_sum)
```

```python
# ✅ Class Assignment Problems
# 1. Bank Account Management System

class BankAccount:
    def __init__(self, account_name, balance):
        self.account_name = account_name
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print(f"{amount} deposited. New balance is {self.balance}")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Not enough balance.")
        else:
            self.balance -= amount
            print(f"{amount} withdrawn. New balance is {self.balance}")

# Example usage
acc = BankAccount("Alice", 1000)
acc.deposit(500)
acc.withdraw(300)
# 2. Hospital Management System (Inheritance)

# Base class
class Person:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender

# Doctor class inherits from Person
class Doctor(Person):
    def __init__(self, name, age, gender, speciality, salary):
        super().__init__(name, age, gender)
        self.speciality = speciality
        self.salary = salary

    def show_info(self):
        print(f"Doctor: {self.name}, Speciality: {self.speciality}, Salary: {self.salary}")

# Patient class inherits from Person
class Patient(Person):
    def __init__(self, name, age, gender, disease, fee):
        super().__init__(name, age, gender)
        self.disease = disease
        self.fee = fee

    def show_info(self):
        print(f"Patient: {self.name}, Disease: {self.disease}, Fee: {self.fee}")

# Example usage
doc = Doctor("Dr. John", 45, "Male", "Cardiologist", 100000)
pat = Patient("Emma", 30, "Female", "Flu", 500)

doc.show_info()
pat.show_info()
# ✅ Home Assignment Problems
# 1. Smart Device System

class SmartDevice:
    def __init__(self, name):
        self.name = name
        self.status = "Off"
```

```python
    def turn_on(self):
        self.status = "On"
        print(f"{self.name} is now On")

    def turn_off(self):
        self.status = "Off"
        print(f"{self.name} is now Off")

# Example usage
light = SmartDevice("Living Room Light")
light.turn_on()
light.turn_off()
# 2. Book Management System

class Book:
    def __init__(self, title, author, available=True):
        self.title = title
        self.author = author
        self.available = available

    def borrow(self):
        if self.available:
            self.available = False
            print(f"You borrowed '{self.title}'")
        else:
            print(f"'{self.title}' is not available")

    def return_book(self):
        self.available = True
        print(f"You returned '{self.title}'")

# Example usage
book1 = Book("Python Basics", "John Doe")
book1.borrow()
book1.borrow()  # try again
book1.return_book()
# 3. Vehicle Management System (Inheritance)

# Base class
class Vehicle:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def show_info(self):
        print(f"Brand: {self.brand}, Model: {self.model}")

# Car class
class Car(Vehicle):
    def __init__(self, brand, model, doors):
        super().__init__(brand, model)
        self.doors = doors

    def show_info(self):
        super().show_info()
        print(f"Doors: {self.doors}")

# Bike class
class Bike(Vehicle):
    def __init__(self, brand, model, type_bike):
        super().__init__(brand, model)
        self.type_bike = type_bike

    def show_info(self):
        super().show_info()
        print(f"Type: {self.type_bike}")
```

```python
# Example usage
car1 = Car("Toyota", "Corolla", 4)
bike1 = Bike("Yamaha", "FZ", "Sports")

car1.show_info()
bike1.show_info()
```