Question 1:

A weather forecasting company wants to store daily temperature data for different cities. Develop a Python program to prompt users to input temperature data for multiple cities and store it in appropriate variables.

```
# Initialize an empty dictionary to store temperature data
temperature_data = {}

# Prompt users to input temperature data for multiple cities
while True:
    city = input("Enter the name of the city (or type 'quit' to exit): ")
    if city.lower() == 'quit':
        break
    temperature = float(input("Enter the temperature for {} (in Celsius): ".format(city)))
    temperature_data[city] = temperature

print("Temperature data:", temperature_data)

```

Question 2: A gaming company is developing a character creation system for their new role-playing game. Develop a Python program that allows users to input characteristics such as name, race, class, and abilities for their characters.

```
class Character:
    def __init__(self, name, race, char_class, abilities):
        self.name = name
        self.race = race
        self.char_class = char_class
        self.abilities = abilities

# Prompt users to input characteristics for their characters
name = input("Enter character name: ")
race = input("Enter character race: ")
char_class = input("Enter character class: ")
abilities = input("Enter character abilities (comma-separated): ").split(',')

# Create a Character object
character = Character(name, race, char_class, abilities)

print("Character created:")
print("Name:", character.name)
print("Race:", character.race)
print("Class:", character.char_class)
print("Abilities:", character.abilities)

```

Question 3: You are organizing a school event where students will be asked to sign up using a form.Write a Python program that prompts the user to enter their name, age, and grade, and then prints a welcome message including their name and an appropriate message based on their age and grade.

```
# Prompt the user to enter their name, age, and grade
name = input("Enter your name: ")
age = int(input("Enter your age: "))
grade = int(input("Enter your grade: "))

# Print a welcome message based on age and grade
if age <= 10:
    print("Welcome,", name + "! Hope you have a great time in school!")
elif age <= 15:
    print("Welcome,", name + "! Enjoy your time in grade", grade)
else:
    print("Welcome,", name + "! Focus on your studies in grade", grade)

```

Question 4: You are managing inventory for a small store. Write a Python program that simulates adding items to a shopping cart. Allow the user to input the item name, quantity, and price, and then calculate the total cost of the items in the cart.

```
# Initialize an empty dictionary to store items in the shopping cart
shopping_cart = {}

# Prompt the user to input item name, quantity, and price
while True:
    item_name = input("Enter the item name (or type 'done' to finish): ")
    if item_name.lower() == 'done':
        break
    quantity = int(input("Enter the quantity: "))
    price = float(input("Enter the price per item: "))
    shopping_cart[item_name] = {'quantity': quantity, 'price': price}

# Calculate the total cost of items in the cart
total_cost = sum(item['quantity'] * item['price'] for item in shopping_cart.values())
print("Total cost of items in the cart:", total_cost)

```

Question 5: A music streaming service wants to create personalized playlists for users based on their favorite genres and artists. Develop a Python program that uses lists and dictionaries to store user preferences and recommend songs accordingly.

```
# Initialize user preferences using dictionaries
user_preferences = {
    'favorite_genres': ['Rock', 'Pop'],
    'favorite_artists': ['Ed Sheeran', 'Coldplay']
}

# Recommend songs based on user preferences
print("Recommended songs based on your preferences:")
# Your recommendation algorithm goes here

```

Question 6: A logistics company needs to track the inventory levels of different products in multiple warehouses. Develop a Python program that uses nested dictionaries to represent warehouse inventory data and allows users to update and retrieve information.

```
# Nested dictionary to represent warehouse inventory data
warehouse_inventory = {
    'warehouse1': {'product1': 100, 'product2': 200},
    'warehouse2': {'product1': 150, 'product2': 180}
}

# Update and retrieve information
warehouse = input("Enter warehouse name: ")
product = input("Enter product name: ")
quantity = int(input("Enter quantity: "))

# Update inventory
if warehouse in warehouse_inventory and product in warehouse_inventory[warehouse]:
    warehouse_inventory[warehouse][product] += quantity
else:
    print("Invalid warehouse or product.")

# Retrieve inventory
print("Inventory for", warehouse + ":", warehouse_inventory[warehouse])

```

Question 7: You are organizing a team-building activity for a group of students. Write a Python program that randomly assigns students to teams of 3. Ensure that each team has an equal number of members, and print out the list of teams with their members.

```
import random

students = ['Student1', 'Student2', 'Student3', 'Student4', 'Student5', 'Student6', 'Student7', 'Student8', 'Student9']

random.shuffle(students)  # Shuffle the list of students

# Split the shuffled list into teams of 3
teams = [students[i:i+3] for i in range(0, len(students), 3)]

# Print out the list of teams with their members
print("Teams:")
for i, team in enumerate(teams, 1):
    print("Team {}: {}".format(i, ', '.join(team)))

```

Question 8: You are building a password manager application. Write a Python program that stores passwords for different accounts in a dictionary, where the keys are the account names and the values are the passwords. Allow the user to add, retrieve, and delete passwords.

```
# Initialize an empty dictionary to store passwords
passwords = {}

while True:
    print("\n1. Add Password")
    print("2. Retrieve Password")
    print("3. Delete Password")
    print("4. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
        account_name = input("Enter account name: ")
        password = input("Enter password: ")
        passwords[account_name] = password
        print("Password added successfully.")
    elif choice == 2:
        account_name = input("Enter account name: ")
        if account_name in passwords:
            print("Password:", passwords[account_name])
        else:
            print("Account not found.")
    elif choice == 3:
        account_name = input("Enter account name: ")
        if account_name in passwords:
            del passwords[account_name]
            print("Password deleted successfully.")
        else:
            print("Account not found.")
    elif choice == 4:
        print("Exiting...")
        break
    else:
        print("Invalid choice. Please try again.")

```

Question 9: A fitness app wants to calculate the user's body mass index (BMI) based on their weight and height. Develop a Python program that prompts users to input their weight (in kilograms) and height (in meters) and calculates their BMI using arithmetic operators.

```
# Prompt users to input weight (in kilograms) and height (in meters)
weight = float(input("Enter your weight (in kilograms): "))
height = float(input("Enter your height (in meters): "))

# Calculate BMI
bmi = weight / (height ** 2)

print("Your BMI is:", bmi)

```

Question 10: A financial institution wants to calculate compound interest for different investment accounts. Develop a Python program that prompts users to input the principal amount, interest rate, and time period, and calculates the compound interest using arithmetic operators.

```
# Prompt users to input principal amount, interest rate, and time period
principal = float(input("Enter the principal amount: "))
interest_rate = float(input("Enter the interest rate (in percentage): ")) / 100
time_period = int(input("Enter the time period (in years): "))

# Calculate compound interest
compound_interest = principal * (1 + interest_rate) ** time_period - principal

print("Compound interest:", compound_interest)

```

Question 11: You are designing a game where players roll two dice and sum the results. Write a Python program that simulates rolling two dice and calculates the sum. Allow the user to roll the dice multiple times and display the results.

```
import random

def roll_dice():
    return random.randint(1, 6) + random.randint(1, 6)

# Prompt the user to roll the dice multiple times
while True:
    input("Press Enter to roll the dice...")
    result = roll_dice()
    print("You rolled:", result)
    play_again = input("Roll again? (yes/no): ").lower()
    if play_again != 'yes':
        break

```

Question 12: You are developing a program to calculate shipping costs for an online store. Write a Python program that prompts the user to enter the weight of their package and the distance it needs to be shipped. Calculate the shipping cost based on a flat rate per pound and a per-mile charge.

```
# Prompt the user to enter the weight of the package and the distance
weight = float(input("Enter the weight of the package (in pounds): "))
distance = float(input("Enter the distance it needs to be shipped (in miles): "))

# Calculate shipping cost based on a flat rate per pound and a per-mile charge
flat_rate_per_pound = 0.5
per_mile_charge = 0.1

shipping_cost = weight * flat_rate_per_pound + distance * per_mile_charge

print("Shipping cost:", shipping_cost)

```

Question 13: An e-commerce platform wants to offer discounts on purchases during a sale event. Develop a Python program that applies different discount rates based on the total purchase amount and displays the final price after applying the discount.

```
# Prompt the user for the total purchase amount
total_purchase_amount = float(input("Enter the total purchase amount: "))

# Apply different discount rates based on the total purchase amount
if total_purchase_amount >= 100:
    discount_rate = 0.2
elif total_purchase_amount >= 50:
    discount_rate = 0.15
elif total_purchase_amount >= 20:
    discount_rate = 0.1
else:
    discount_rate = 0

# Calculate the final price after applying the discount
final_price = total_purchase_amount * (1 - discount_rate)

print("Final price after discount:", final_price)

```

Question 14: A game developer wants to implement a scoring system for a multiplayer game. Develop a Python program that calculates the total score for each player based on their performance in the game and determines the winner.

```
# Define a dictionary to store players' scores
players_scores = {
    'Player1': 100,
    'Player2': 150,
    'Player3': 80,
    # Add more players and their scores as needed
}

# Calculate the total score for each player
total_scores = sum(players_scores.values())

# Determine the winner
winner = max(players_scores, key=players_scores.get)

print("Total scores:")
for player, score in players_scores.items():
    print(player + ":", score)
print("Winner:", winner)

```

Question 15: A restaurant offers different discounts based on the time of day. Write a Python program that prompts the user for the current time and calculates the appropriate discount based on the following criteria: 10% discount for breakfast (before 10 AM), 20% discount for lunch (between 12 PM and 3 PM), and 15% discount for dinner (after 6 PM). Display the calculated discount to the user.

```
# Prompt the user for the current time
current_time = int(input("Enter the current time (in 24-hour format, without colon): "))

# Calculate the appropriate discount based on the time of day
if current_time < 1000:
    discount = 0.10  # 10% discount for breakfast
elif 1200 <= current_time <= 1500:
    discount = 0.20  # 20% discount for lunch
elif current_time >= 1800:
    discount = 0.15  # 15% discount for dinner
else:
    discount = 0

print("Applicable discount:", discount * 100, "%")

```

Question 16: You are organizing a tournament with a single-elimination format. Write a Python program that takes a list of participants and randomly generates the matchups for each round. Print out the matchups for each round until a winner is determined.

```
import random

participants = ['Participant1', 'Participant2', 'Participant3', 'Participant4', 'Participant5', 'Participant6', 'Participant7', 'Particip

while len(participants) > 1:
    matchups = []
    random.shuffle(participants)
    for i in range(0, len(participants), 2):
        matchups.append((participants[i], participants[i+1]))
    print("Matchups for this round:")
    for matchup in matchups:
        print(matchup[0], "vs", matchup[1])
    participants = [winner for matchup in matchups for winner in random.choice(matchup)]
print("Winner:", participants[0])

```

Question 17: A data analysis company wants to perform statistical analysis on a dataset containing sales data. Develop a Python program that defines functions for calculating the mean, median, and standard deviation of the dataset and uses them to analyze the data.

```
def calculate_mean(data):
    return sum(data) / len(data)

def calculate_median(data):
    sorted_data = sorted(data)
    n = len(sorted_data)
    if n % 2 == 0:
        return (sorted_data[n // 2 - 1] + sorted_data[n // 2]) / 2
    else:
        return sorted_data[n // 2]

def calculate_standard_deviation(data):
    mean = calculate_mean(data)
    variance = sum((x - mean) ** 2 for x in data) / len(data)
    return variance ** 0.5

# Example usage
sales_data = [100, 120, 150, 180, 200]
print("Mean:", calculate_mean(sales_data))
print("Median:", calculate_median(sales_data))
print("Standard Deviation:", calculate_standard_deviation(sales_data))

```

Question 18: A social media platform wants to implement user authentication and authorization functionality. Develop a Python program that defines functions for user login, registration, and access control, and uses them to manage user accounts.

```
class UserAuthentication:
    def __init__(self):
        self.users = {}

    def register(self, username, password):
        if username not in self.users:
            self.users[username] = password
            print("User registered successfully.")
        else:
            print("Username already exists.")

    def login(self, username, password):
        if username in self.users and self.users[username] == password:
            print("Login successful.")
        else:
            print("Invalid username or password.")

# Example usage
auth = UserAuthentication()
auth.register("user1", "password123")
auth.login("user1", "password123")

```

Question 19: You are building a calculator application with various mathematical functions. Write a Python program that defines functions for addition, subtraction, multiplication, and division. Allow the user to choose a function and input the numbers to perform the calculation.

```
def addition(x, y):
    return x + y

def subtraction(x, y):
    return x - y

def multiplication(x, y):
    return x * y

def division(x, y):
    if y != 0:
        return x / y
    else:
        return "Cannot divide by zero."

# Example usage
print("Addition:", addition(5, 3))
print("Subtraction:", subtraction(5, 3))
print("Multiplication:", multiplication(5, 3))
print("Division:", division(5, 3))

```

Question 20: A company is organizing a team-building exercise where employees will participate in various activities. Write a Python program to randomly assign employees to teams for each activity, ensuring that each team has an equal number of members.

```
import random

employees = ['Employee1', 'Employee2', 'Employee3', 'Employee4', 'Employee5', 'Employee6', 'Employee7', 'Employee8', 'Employee9']

activities = ['Activity1', 'Activity2', 'Activity3']

for activity in activities:
    random.shuffle(employees)
    teams = [employees[i:i+3] for i in range(0, len(employees), 3)]
    print("Teams for", activity + ":")
    for i, team in enumerate(teams, 1):
        print("Team {}: {}".format(i, ', '.join(team)))

```

Question 21: A ticket booking system encounters errors when processing user requests due to invalid input. Develop a Python program that handles errors such as invalid ticket quantities, incorrect payment information, and expired sessions.

```
# Here's a basic example of error handling in Python for a ticket booking system:

try:
    ticket_quantity = int(input("Enter the number of tickets you want to purchase: "))
    if ticket_quantity <= 0:
        raise ValueError("Invalid ticket quantity.")
except ValueError as e:
    print("Error:", e)
else:
    print("Tickets booked successfully.")

```

Question 22: A data processing pipeline encounters errors when reading and writing data files due to file permissions and network issues. Develop a Python program that handles errors such as file not found, permission denied, and connection timeout.

```
# Here's a basic example of error handling in Python for a data processing pipeline:

try:
    with open('data.txt', 'r') as file:
        data = file.read()
    # Process the data...
except FileNotFoundError:
    print("Error: File not found.")
except PermissionError:
    print("Error: Permission denied.")
except Exception as e:
    print("Error:", e)

```

Question 23: A bank is developing an ATM system. Develop a Python program that simulates withdrawing money from an ATM. Handle errors such as insufficient funds, incorrect PIN entry, and invalid withdrawal amounts. Display appropriate error messages to the user and allow them to retry or cancel the transaction.

```
class ATM:
    def __init__(self, balance, pin):
        self.balance = balance
        self.pin = pin

    def withdraw(self, amount, entered_pin):
        if entered_pin != self.pin:
            raise ValueError("Incorrect PIN.")
        if amount > self.balance:
            raise ValueError("Insufficient funds.")
        self.balance -= amount
        return "Withdrawal successful. Remaining balance: {}".format(self.balance)

# Example usage
atm = ATM(1000, 1234)
try:
    print(atm.withdraw(500, 1234))  # Correct PIN
    print(atm.withdraw(800, 9999))  # Incorrect PIN
    print(atm.withdraw(1200, 1234)) # Insufficient funds
except ValueError as e:
    print("Error:", e)

```

Question 24: A small business wants to track inventory using a text file. Write a Python program that allows users to input item details, display all inventory records, and calculate the total value of the inventory.

Challenge:

Ensure that the program handles input errors gracefully and provides clear feedback to the user in case of invalid input.

```
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, item_name, quantity, price):
        self.items[item_name] = {'quantity': quantity, 'price': price}

    def display_inventory(self):
        for item, details in self.items.items():
            print("Item:", item)
            print("Quantity:", details['quantity'])
            print("Price:", details['price'])

    def calculate_total_value(self):
        total_value = sum(details['quantity'] * details['price'] for details in self.items.values())
        print("Total value of inventory:", total_value)

# Example usage
inventory = Inventory()
inventory.add_item("Item1", 10, 5.0)
inventory.add_item("Item2", 20, 7.0)
inventory.display_inventory()
inventory.calculate_total_value()

```

Question 25: A scientific research project generates experimental data that needs to be stored in files for analysis. Develop a Python program that writes experimental data to text files, including metadata such as timestamps and experiment parameters.

```
import datetime

# Experimental data
experimental_data = {
    'experiment_name': 'Experiment 1',
    'timestamp': datetime.datetime.now(),
    'parameters': {
        'temperature': 25,
        'pressure': 101.3,
        'time_duration': 60
    },
    'results': {
        'measurement1': 10,
        'measurement2': 15,
        'measurement3': 20
    }
}

# Write experimental data to a text file
with open('experimental_data.txt', 'w') as file:
    for key, value in experimental_data.items():
        file.write('{}: {}\n'.format(key, value))

```

Question 26: A research institute collects data from experiments conducted by scientists. Develop a Python program that reads experimental data from a text file, performs basic analysis (e.g., calculating averages, finding maximum values), and saves the results to an output file. Allow users to specify the input and output file paths.

```
# Read experimental data from a text file
def read_experimental_data(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
    experimental_data = {}
    for line in lines:
        key, value = line.strip().split(': ')
        if key in ('timestamp',):
            value = datetime.datetime.strptime(value, '%Y-%m-%d %H:%M:%S.%f')
        elif key in ('parameters', 'results'):
            value = eval(value)
        experimental_data[key] = value
    return experimental_data

# Example usage
experimental_data = read_experimental_data('experimental_data.txt')
print(experimental_data)

```

Question 27: An online bookstore wants to manage information about books, including title, author, genre, and price. Develop a Python program that defines a Book class with attributes and methods for managing book data.

```
class Book:
    def __init__(self, title, author, genre, price):
        self.title = title
        self.author = author
        self.genre = genre
        self.price = price

# Example usage
book1 = Book("Title1", "Author1", "Fiction", 20.0)
print("Book Title:", book1.title)
print("Author:", book1.author)
print("Genre:", book1.genre)
print("Price:", book1.price)

```

Question 28: A car rental company wants to track information about rental vehicles, including make, model, year, and availability. Develop a Python program that defines a Car class with attributes and methods for managing rental vehicles.

```
class Car:
    def __init__(self, make, model, year, availability=True):
        self.make = make
        self.model = model
        self.year = year
        self.availability = availability

# Example usage
car1 = Car("Toyota", "Camry", 2022)
print("Make:", car1.make)
print("Model:", car1.model)
print("Year:", car1.year)
print("Availability:", "Available" if car1.availability else "Not Available")

```

Question 29: Suppose you're building an online shopping platform. How would you implement an operator-based discount system that applies different discount rates based on the total purchase amount? Provide a Python code snippet to calculate the discounted price for a given purchase amount.

```
# Define discount rates based on the total purchase amount
discount_rates = {
    20: 0.05,  # 5% discount for purchases >= $20
    50: 0.1,   # 10% discount for purchases >= $50
    100: 0.15  # 15% discount for purchases >= $100
}

def calculate_discounted_price(purchase_amount):
    for amount, discount_rate in discount_rates.items():
        if purchase_amount >= amount:
            return purchase_amount * (1 - discount_rate)
    return purchase_amount

# Example usage
purchase_amount = float(input("Enter the total purchase amount: "))
discounted_price = calculate_discounted_price(purchase_amount)
print("Discounted price:", discounted_price)

```

Question 30: A travel agency wants to automate the process of booking flights for customers. Develop a Python program that prompts users to input travel preferences, searches for available flights, and generates booking confirmations.

```
class FlightBookingSystem:
    def __init__(self):
        self.flights = {}

    def add_flight(self, flight_number, source, destination, departure_time):
        self.flights[flight_number] = {
            'source': source,
            'destination': destination,
            'departure_time': departure_time
        }

    def search_flights(self, source, destination):
        matching_flights = [flight for flight, details in self.flights.items() if details['source'] == source and details['destination']
        return matching_flights

# Example usage
flight_system = FlightBookingSystem()
flight_system.add_flight("FL001", "New York", "Los Angeles", "08:00")
flight_system.add_flight("FL002", "Los Angeles", "New York", "12:00")
flight_system.add_flight("FL003", "Chicago", "Miami", "10:00")
print("Available flights from New York to Los Angeles:", flight_system.search_flights("New York", "Los Angeles"))

```

Question 31: You're developing a program to analyze user comments on a social media platform. How would you handle the storage of user comments, considering that each comment may contain a mix of text, emojis, and hashtags? Recommend a Python data type or combination of data types to efficiently store this heterogeneous data and provide an example of how you would parse and process a sample comment.

```
# Python data type to store user comments efficiently: Dictionary
# Each comment can be stored as a dictionary with keys for text, emojis, and hashtags
comment = {
    'text': "Great product! 😊 #happy #satisfied",
    'emojis': ['😊'],
    'hashtags': ['happy', 'satisfied']
}

# Example of parsing and processing a sample comment
print("Comment text:", comment['text'])
print("Emojis:", comment['emojis'])
print("Hashtags:", comment['hashtags'])

```

Question 32: You're tasked with designing a program to track financial transactions for a small business. Each transaction includes details such as transaction type (e.g., sale, expense), amount, and date. How would you structure the data to efficiently perform operations like calculating total revenue, expenses, and profit? Recommend a Python data type or structure and outline the process for storing and analyzing transaction data.

```
class Transaction:
    def __init__(self, transaction_type, amount, date):
        self.transaction_type = transaction_type
        self.amount = amount
        self.date = date

# Python data type to efficiently store transaction data: List of Transaction objects
# Each transaction is represented as an instance of the Transaction class
transactions = [
    Transaction("sale", 100, "2024-04-20"),
    Transaction("expense", 50, "2024-04-21"),
    Transaction("sale", 150, "2024-04-21")
]

# Example of analyzing transaction data
total_revenue = sum(transaction.amount for transaction in transactions if transaction.transaction_type == "sale")
total_expenses = sum(transaction.amount for transaction in transactions if transaction.transaction_type == "expense")
profit = total_revenue - total_expenses
print("Total Revenue:", total_revenue)
print("Total Expenses:", total_expenses)
print("Profit:", profit)

```

Question 33: You're designing a smart home system where certain appliances are activated based on environmental conditions. How would you use bitwise operators to represent different sensor readings (e.g., temperature, humidity, light intensity) and trigger specific actions based on predefined thresholds? Provide a Python code snippet to illustrate this concept.

```
# Example of using bitwise operators to represent sensor readings and trigger actions
# Assume temperature, humidity, and light intensity are represented as 8-bit integers
temperature = 25  # 00110001 in binary
humidity = 60     # 00111100 in binary
light_intensity = 120  # 01111000 in binary

# Define thresholds
temperature_threshold = 30
humidity_threshold = 70
light_intensity_threshold = 100

# Use bitwise AND (&) to check if readings exceed thresholds
if temperature > temperature_threshold and humidity > humidity_threshold and light_intensity > light_intensity_threshold:
    print("Trigger specific actions based on predefined thresholds.")
else:
    print("Conditions not met for triggering actions.")

```

Question 34: You're developing a program to encrypt and decrypt text messages using a custom encryption algorithm. How would you use bitwise XOR (^) operator to perform encryption and decryption operations on the message? Provide a Python code snippet to encrypt and decrypt a sample message.

```
# Example of using bitwise XOR (^) operator for custom encryption and decryption
message = "Hello, World!"
key = 42

# Encryption
encrypted_message = ''.join(chr(ord(char) ^ key) for char in message)
print("Encrypted message:", encrypted_message)

# Decryption (using the same key)
decrypted_message = ''.join(chr(ord(char) ^ key) for char in encrypted_message)
print("Decrypted message:", decrypted_message)

```

Question:35 You're developing a language translation service that supports multiple languages. Each translation request consists of a source language, target language, and the text to be translated. How would you organize this information to handle translation requests efficiently? Recommend a Python data type or structure to manage translation requests and provide a sample code snippet to process a translation request.

```
class TranslationService:
    def __init__(self):
        self.translation_requests = []

    def add_translation_request(self, source_language, target_language, text):
        self.translation_requests.append({
            'source_language': source_language,
            'target_language': target_language,
            'text': text
        })

    def process_translation_requests(self):
        for request in self.translation_requests:
            print("Translating from {} to {}: {}".format(request['source_language'], request['target_language'], request['text']))
            # Your translation logic goes here

# Example usage
translation_service = TranslationService()
translation_service.add_translation_request('english', 'spanish', 'Hello, how are you?')
translation_service.add_translation_request('french', 'german', 'Bonjour, comment ça va?')
translation_service.process_translation_requests()

```

Question 36: You are developing a program to manage a small inventory of office supplies. Write a Python program that allows users to add new items to the inventory, update existing items, and remove items that are no longer in stock.

```
class Inventory:
    def __init__(self):
        self.items = {}

    def add_item(self, item_name, quantity):
        self.items[item_name] = quantity

    def update_item(self, item_name, quantity):
        if item_name in self.items:
            self.items[item_name] += quantity
        else:
            print("Item not found in inventory.")

    def remove_item(self, item_name):
        if item_name in self.items:
            del self.items[item_name]
        else:
            print("Item not found in inventory.")

# Example usage
inventory = Inventory()
inventory.add_item("Pens", 100)
inventory.update_item("Pens", -20)
inventory.remove_item("Pens")

```

Question 37: A sports club wants to keep track of its members' attendance at training sessions. Develop a Python program that allows coaches to record attendance, view attendance records for individual members, and generate attendance reports for the club.

```
class SportsClub:
    def __init__(self):
        self.attendance_records = {}

    def record_attendance(self, member_id):
        if member_id in self.attendance_records:
            self.attendance_records[member_id] += 1
        else:
            self.attendance_records[member_id] = 1

    def view_attendance(self, member_id):
        if member_id in self.attendance_records:
            return self.attendance_records[member_id]
        else:
            return 0

    def generate_attendance_report(self):
        for member_id, attendance in self.attendance_records.items():
            print("Member ID:", member_id, "Attendance:", attendance)

# Example usage
club = SportsClub()
club.record_attendance("001")
club.record_attendance("002")
club.record_attendance("001")
print("Attendance for member 001:", club.view_attendance("001"))
club.generate_attendance_report()

```

Question:38 You are building a simple calculator application. Write a Python program that prompts users to enter two numbers and choose an operation (addition, subtraction, multiplication, division). Perform the selected operation and display the result.

```
def calculator():
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    operation = input("Choose operation (+, -, *, /): ")
    if operation == '+':
        print("Result:", num1 + num2)
    elif operation == '-':
        print("Result:", num1 - num2)
    elif operation == '*':
        print("Result:", num1 * num2)
    elif operation == '/':
        if num2 != 0:
            print("Result:", num1 / num2)
        else:
            print("Error: Cannot divide by zero.")

# Example usage
calculator()

```

Question 39:A school is organizing a quiz competition for its students. Develop a Python program that generates random quiz questions from different subjects (e.g., math, science, history) and allows students to answer them. Provide feedback on the correctness of each answer and calculate the final score.

```
import random

class Quiz:
    def __init__(self):
        self.questions = {
            'Math': ['What is 2 + 2?', 'What is the square root of 16?'],
            'Science': ['What is the chemical symbol for water?', 'What is the capital of France?'],
            # Add more subjects and questions as needed
        }

    def generate_question(self):
        subject = random.choice(list(self.questions.keys()))
        question = random.choice(self.questions[subject])
        return subject, question

# Example usage
quiz = Quiz()
subject, question = quiz.generate_question()
print("Subject:", subject)
print("Question:", question)

```

Question 40: You are designing a program to track daily expenses. Write a Python program that allows users to input their expenses for various categories (e.g., groceries, transportation, entertainment) and calculate the total expenditure for each category and overall.

```
class ExpenseTracker:
    def __init__(self):
        self.expenses = {}

    def add_expense(self, category, amount):
        if category in self.expenses:
            self.expenses[category] += amount
        else:
            self.expenses[category] = amount

    def calculate_total_expenditure(self):
        total = sum(self.expenses.values())
        print("Total expenditure:", total)

# Example usage
tracker = ExpenseTracker()
tracker.add_expense("Groceries", 50)
tracker.add_expense("Transportation", 30)
tracker.calculate_total_expenditure()

```

Question 41:A company wants to analyze employee performance based on their monthly sales data. Develop a Python program that allows managers to input sales data for each employee, calculate their total sales for the month, and generate performance reports.

```
class EmployeePerformance:
    def __init__(self):
        self.sales_data = {}

    def input_sales_data(self, employee_id, sales_amount):
        if employee_id in self.sales_data:
            self.sales_data[employee_id] += sales_amount
        else:
            self.sales_data[employee_id] = sales_amount

    def generate_performance_report(self):
        for employee_id, total_sales in self.sales_data.items():
            print("Employee ID:", employee_id, "Total Sales:", total_sales)

# Example usage
performance = EmployeePerformance()
performance.input_sales_data("E001", 5000)
performance.input_sales_data("E002", 7000)
performance.generate_performance_report()

```

Question 42: You are developing a program to manage student grades for a school. Write a Python program that allows teachers to input grades for multiple students in different subjects, calculate their average grades, and generate grade reports.

```
class GradeManager:
    def __init__(self):
        self.student_grades = {}

    def input_grades(self, student_id, subject, grade):
        if student_id in self.student_grades:
            if subject in self.student_grades[student_id]:
                self.student_grades[student_id][subject].append(grade)
            else:
                self.student_grades[student_id][subject] = [grade]
        else:
            self.student_grades[student_id] = {subject: [grade]}

    def calculate_average_grades(self, student_id):
        if student_id in self.student_grades:
            average_grades = {subject: sum(grades) / len(grades) for subject, grades in self.student_grades[student_id].items()}
            return average_grades
        else:
            return {}

# Example usage
grade_manager = GradeManager()
grade_manager.input_grades("S001", "Math", 90)
grade_manager.input_grades("S001", "Science", 85)
average_grades = grade_manager.calculate_average_grades("S001")
print("Average Grades:", average_grades)

```

Question 43: An online store wants to offer discounts to customers based on their total purchase amount. Develop a Python program that prompts users to input their total purchase amount and applies different discount rates based on predefined criteria (e.g., $100 - 10200$ - 20% discount).

```
def apply_discount(purchase_amount):
    if purchase_amount >= 200:
        return purchase_amount * 0.8  # 20% discount for purchases >= $200
    elif purchase_amount >= 100:
        return purchase_amount * 0.9  # 10% discount for purchases >= $100
    else:
        return purchase_amount

# Example usage
total_purchase_amount = float(input("Enter total purchase amount: "))
discounted_amount = apply_discount(total_purchase_amount)
print("Discounted amount:", discounted_amount)

```

Question 44: You are organizing a charity event and need to track donations from various sponsors. Write a Python program that allows users to input donation amounts from sponsors, calculate the total amount raised, and generate thank-you letters for each sponsor.

```
class DonationTracker:
    def __init__(self):
        self.total_amount = 0
        self.sponsors = {}

    def record_donation(self, sponsor, amount):
        if sponsor in self.sponsors:
            self.sponsors[sponsor] += amount
        else:
            self.sponsors[sponsor] = amount
        self.total_amount += amount

    def generate_thank_you_letters(self):
        for sponsor, amount in self.sponsors.items():
            print("Thank you, {}! Your donation of ${} is greatly appreciated.".format(sponsor, amount))
        print("Total amount raised:", self.total_amount)

# Example usage
donations = DonationTracker()
donations.record_donation("Sponsor1", 100)
donations.record_donation("Sponsor2", 200)
donations.generate_thank_you_letters()

```

Question 45:A company wants to automate the process of generating invoices for its clients. Develop a Python program that prompts users to input client details (e.g., name, address, services rendered), calculates the total amount due, and generates invoices in PDF format.

```
class InvoiceGenerator:
    def __init__(self):
        self.invoices = {}

    def generate_invoice(self, client_name, address, services_rendered, total_amount):
        self.invoices[client_name] = {
            'Address': address,
            'Services Rendered': services_rendered,
            'Total Amount': total_amount
        }
        print("Invoice generated for:", client_name)

# Example usage
generator = InvoiceGenerator()
generator.generate_invoice("Client1", "123 Main St", "Consulting", 500)

```

Question 46: You are building a program to manage a small restaurant's menu and orders. Write a Python program that allows users to view the menu, place orders, calculate the total bill, and generate order receipts.

```
class Restaurant:
    def __init__(self):
```

Question 47:A fitness trainer wants to track their clients' progress over time. Develop a Python program that allows trainers to input client measurements (e.g., weight, body fat percentage) at different time points, calculate changes over time, and generate progress reports.

```
    def add_to_menu(self, item, price):
```

```
class FitnessTracker:
    def __init__(self):
        self.client_measurements = {}

    def input_measurements(self, client_id, weight, body_fat_percentage):
        if client_id in self.client_measurements:
            self.client_measurements[client_id].append((weight, body_fat_percentage))
        else:
            self.client_measurements[client_id] = [(weight, body_fat_percentage)]

    def calculate_progress(self, client_id):
        if client_id in self.client_measurements:
            measurements = self.client_measurements[client_id]
            initial_weight, initial_body_fat = measurements[0]
            latest_weight, latest_body_fat = measurements[-1]
            weight_loss = initial_weight - latest_weight
            fat_loss = initial_body_fat - latest_body_fat
            return weight_loss, fat_loss
        else:
            return None

# Example usage
tracker = FitnessTracker()
tracker.input_measurements("C001", 70, 20)
tracker.input_measurements("C001", 68, 18)
progress = tracker.calculate_progress("C001")
print("Weight loss:", progress[0], "kg")
print("Fat loss:", progress[1], "%")

```