Coding Tasks - Week 6

Welcome to the Python Programming Exercise Sheet!
In this exercise sheet, we will cover some of the fundamental concepts in Python programming.
Topics covered are: Recall of previous concepts.

DEADLINE: 12th June until 12:15
Your name here: Shanza Amber
Your university mail: samber@uni-osnabrueck.de

Important information:
In order to pass this sheet you need to achieve / points.
For the best possible grade you require / points, however, since some harder tasks may take a lot of time you don't have to pressure yourself.
If you complete any three tasks on a sheet you will definitely get a good grade for that sheet.
Hand in your sheet in studip in the respective folder until the deadline.
If you receive no email until a few days after submission you will have passed, the sample solution will also be uploaded around then.
If you receive an email you don't need to worry, you can fail one sheet and also your total points will also be taken into account for the final pass or fail.

Tuples

Task 1 - City Population Tracker (3 points)

Write a program that takes a list of cities and their corresponding populations as input. The program should calculate the total population of all the cities and determine the cities with the highest and lowest populations.

Your program should perform the following steps:

1. Define lists, where each list contains the name of a city and its population.
2. Use a for loop to iterate over the lists and calculate the total population by adding up the populations of all the cities.
3. Use if statements to determine the city with the highest population and the city with the lowest population.
4. Print the total population of all the cities, as well as the names and populations of the cities with the highest and lowest populations.

In this task, you'll need to use lists to store the cities and their populations. You'll iterate over the list using a for loop, perform calculations to determine the total population, and find the cities with the highest and lowest populations using if statements. Finally, you'll print the population summary and the details of the cities with the highest and lowest populations.

Hint: Using lambda function might be useful when finding the cities with the highest and lowest populations.

```python
# Define an empty list to store city population tuples
cities = []

# Get the number of cities from the user
num_cities = int(input("Enter the number of cities: "))

# Iterate over the range of the number of cities
for i in range(num_cities):
    city_name = input("Enter the name of the city: ")
    population = int(input("Enter the population of the city: "))
    cities.append((city_name, population))

# Calculate the total population
total_population = sum(population for city, population in cities)

# Find the city with the highest population
highest_city = max(cities, key=lambda x: x[1])

# Find the city with the lowest population
lowest_city = min(cities, key=lambda x: x[1])

# Print the population summary
print("\nPopulation Summary:")
print(f"Total population of all cities: {total_population}\n")

# Print the city with the highest population
print("City with the highest population:")
print(f"Name: {highest_city[0]}")
print(f"Population: {highest_city[1]}\n")
```

```
# Print the city with the lowest population
print("City with the lowest population:")
print(f"Name: {lowest_city[0]}")
print(f"Population: {lowest_city[1]}")
```

```
Enter the number of cities: 2
Enter the name of the city: Faisalabad
Enter the population of the city: 15
Enter the name of the city: Karachi
Enter the population of the city: 50

Population Summary:
Total population of all cities: 65

City with the highest population:
Name: Karachi
Population: 50

City with the lowest population:
Name: Faisalabad
Population: 15
```

Task 2 - Student Grades (2 points)

You are a teacher and you need to calculate the average grade for each student in your class. Each student has four subject grades: Math, Science, English, and History. You have a list of tuples, where each tuple contains the student's name followed by their subject grades. Write a program that calculates and prints the average grade for each student in the following format:

Alice: Average Grade = 88.75

Bob: Average Grade = 79.0

Claire: Average Grade = 91.0

Instructions:

1. Define a function named `calculate_average` that takes a tuple of grades as input and returns the average grade.
2. Iterate over each tuple in the grades list.
3. Extract the student's name and grades from each tuple.
4. Call the `calculate_average` function with the grades tuple to calculate the average grade.
5. Print the student's name and average grade using the provided format.

Note: You can assume that each tuple in the list will have the correct format and number of grades.

This task will test your understanding of tuples, iteration, and function usage. Good luck!

```
 def calculate_average(grades):
  # Add implementation here
  # Remove pass after the implementation
    return sum(grades) / len(grades)

grades = [("Alice", 85, 90, 92, 88),
          ("Bob", 76, 82, 80, 78),
          ("Claire", 92, 88, 90, 94)]

for student_grades in grades:
    name = student_grades[0]
    average_grade = calculate_average(student_grades[1:])
    print(f"{name}: Average Grade = {average_grade}")
```

```
Alice: Average Grade = 88.75
Bob: Average Grade = 79.0
Claire: Average Grade = 91.0
```

Sets

Task 3 - Set Operations Calculators (2 points)

Write a Python program that defines the following functions to perform set operations:

1. union(set1, set2): This function takes two sets as input and returns their union.
2. intersection(set1, set2): This function takes two sets as input and returns their intersection.
3. difference(set1, set2): This function takes two sets as input and returns the difference between the first set and the second set (i.e., elements in set1 that are not present in set2).
4. symmetric_difference(set1, set2): This function takes two sets as input and returns their symmetric difference (i.e., elements that are present in either set1 or set2, but not both).

Your program should then prompt the user to enter the elements of two sets and call the respective functions to perform the set operations. The results should be displayed to the user.

```
set1={1, 2, 3, 4}
set2={3, 4, 5, 6}

def union(set1, set2):
  # Add implementation here
  # Remove pass after the implementation
    return set1.union(set2)

def intersection(set1, set2):
  # Add implementation here
  # Remove pass after the implementation
    return set1.intersection(set2)

def difference(set1, set2):
  # Add implementation here
  # Remove pass after the implementation
    return set1.difference(set2)

def symmetric_difference(set1, set2):
  # Add implementation here
  # Remove pass after the implementation
    return set1.symmetric_difference(set2)

# Perform the set operations using the defined functions
set_union = union(set1, set2)
set_intersection = intersection(set1, set2)
set_difference = difference(set1, set2)
set_symmetric_difference = symmetric_difference(set1, set2)

# Display the results
print("Union:", set_union)
print("Intersection:", set_intersection)
print("Difference (Set1 - Set2):", set_difference)
print("Symmetric Difference:", set_symmetric_difference)
```

```
Union: {1, 2, 3, 4, 5, 6}
Intersection: {3, 4}
Difference (Set1 - Set2): {1, 2}
Symmetric Difference: {1, 2, 5, 6}
```

Task 4 - Unique Elements Counter (3 points)

Write a Python program that defines a function called `count_unique_elements` which takes a list as input and returns the count of unique elements in the list and performs the following steps:

1. Define the function called `count_unique_elements`.
2. Prompt the user to enter a list of numbers.
3. Convert the numbers from strings to integers.
4. Call the `count_unique_elements` function and display the result.

Hint: List comprehension might help you convert the numbers from strings to integers.

```
def count_unique_elements(lst):
  # Add implementation here
  # Remove pass after the implementation
    return len(set(lst))


# Prompt the user to enter a list of numbers
user_input = input("Enter a list of numbers (separated by spaces): ")

# Split the user input into individual numbers
numbers = user_input.split()

# Convert the numbers from strings to integers
numbers = [int(num) for num in numbers]

# Call the count_unique_elements function and display the result
unique_count = count_unique_elements(numbers)
print("Number of unique elements:", unique_count)
```

```
Enter a list of numbers (separated by spaces): 1 2 3 7 9 2 1
Number of unique elements: 5
```

Dictionaries

Task 5 - Book Collection Manager (5 points)

You are working on a project that requires managing a book collection. You need to create a program that uses a dictionary to store information about the books. Each book will have a unique ID as the key and a dictionary of details as the value. The details of each book include the title, author, and publication year.

Implement the following functions to interact with the book collection:

1. add_book(collection, book_id, title, author, year): Adds a new book to the collection with the given book ID, title, author, and publication year.
2. remove_book(collection, book_id): Removes a book from the collection with the given book ID.
3. search_book(collection, book_id): Searches for a book in the collection with the given book ID and displays its details (title, author, year) if found.
4. display_all_books(collection): Displays the details of all books in the collection.

```
def add_book(collection, book_id, title, author, year):
    # Add implementation here
    # Remove pass after the implementation
    collection[book_id] = {"title": title, "author": author, "year": year}


def remove_book(collection, book_id):
    # Add implementation here
    # Remove pass after the implementation
    if book_id in collection:
        del collection[book_id]

def search_book(collection, book_id):
    # Add implementation here
    # Remove pass after the implementation
    if book_id in collection:
        book = collection[book_id]
        print(f"Book ID: {book_id}")
        print(f"Title: {book['title']}")
        print(f"Author: {book['author']}")
        print(f"Year: {book['year']}")

def display_all_books(collection):
    # Add implementation here
    # Remove pass after the implementation
     for book_id, book in collection.items():
        print(f"Book ID: {book_id}")
        print(f"Title: {book['title']}")
        print(f"Author: {book['author']}")
        print(f"Year: {book['year']}")
        print()

# Create an empty book collection
book_collection = {}

# Perform operations on the book collection
add_book(book_collection, 1, "Python Programming", "John Smith", 2022)
add_book(book_collection, 2, "Data Science Basics", "Alice Johnson", 2021)
add_book(book_collection, 3, "Web Development Guide", "Bob Williams", 2020)

remove_book(book_collection, 2)

search_book(book_collection, 1)
search_book(book_collection, 2)

display_all_books(book_collection)
```

```
Book ID: 1
Title: Python Programming
Author: John Smith
Year: 2022
Book ID: 1
Title: Python Programming
Author: John Smith
Year: 2022

Book ID: 3
Title: Web Development Guide
Author: Bob Williams
```

```
Year: 2020
```

Task 6 - Student Grade Tracker (5 points)

Create a class called `StudentGradeTracker` that represents a grade tracker for students. The class should have the following methods:

1. init(self): Initializes an empty dictionary to store student grades.
2. add_grade(self, student_id, grade): Adds a grade to the grade tracker for the given student. The student_id should be the key, and the grade should be the value in the dictionary. If the student_id already exists in the grade tracker, print a message indicating that the grade has been updated.
3. remove_grade(self, student_id): Removes the grade for the given student from the grade tracker. If the student_id is found and removed, print a message indicating that the grade has been removed. If the student_id is not found in the grade tracker, print a message indicating that the student does not have a grade recorded.
4. display_grades(self): Displays all the grades in the grade tracker. If the grade tracker is not empty, print each student's grade on a new line in the format: "Student ID: Grade". If the grade tracker is empty, print a message indicating that no grades have been recorded.

Instructions:

1. Implement the StudentGradeTracker class with the methods described above.
2. Create an instance of the StudentGradeTracker class.
3. Add at least three grades to the grade tracker using the add_grade() method.
4. Display all the grades in the grade tracker using the display_grades() method.
5. Remove one grade from the grade tracker using the remove_grade() method.
6. Display the updated grades in the grade tracker using the display_grades() method.

```python
class StudentGradeTracker:
    def __init__(self):
        # Add implementation here
        # Remove pass after the implementation
        self.grades = {}

    def add_grade(self, student_id, grade):
        # Add implementation here
        # Remove pass after the implementation
        if student_id in self.grades:
            print(f"Grade for student {student_id} has been updated.")

    def remove_grade(self, student_id):
         # Add implementation here
         # Remove pass after the implementation
        if student_id in self.grades:
            del self.grades[student_id]
            print(f"Grade for student {student_id} has been removed.")
        else:
            print(f"Student {student_id} does not have a grade recorded.")

    def display_grades(self):
        # Add implementation here
        # Remove pass after the implementation
        if self.grades:
            for student_id, grade in self.grades.items():
                print(f"Student ID: {student_id}, Grade: {grade}")
        else:
            print("No grades have been recorded.")

# Create an instance of the StudentGradeTracker class
grade_tracker = StudentGradeTracker()

# Add grades to the tracker
grade_tracker.add_grade("001", 85)
grade_tracker.add_grade("002", 92)
grade_tracker.add_grade("003", 78)

# Display the grades
grade_tracker.display_grades()

# Remove a grade
grade_tracker.remove_grade("002")

# Display the updated grades
grade_tracker.display_grades()
```

```
No grades have been recorded.
Student 002 does not have a grade recorded.
No grades have been recorded.
```