TDI Python for Data Analytics Week 2 Assignment

Welcome to Python Week 2!

This week, you will deepen your understanding of control structures (like loops and conditionals) and explore data structures such as lists, dictionaries, and sets. Mastering these concepts is essential for writing efficient, logical Python code that handles data effectively.

Your Mission:

In this assignment, you'll work with:

- Control Structures: Using "if", "else", "elif" statements to control the flow of your program, and "for" and "while" loops to iterate through data.

- Data Structures: Working with lists, dictionaries, and sets to store, manipulate, and retrieve data.

- Dictionaries and Sets: Understanding how to use these powerful data structures to manage data in Python.

Resources:

These resources will help you understand control structures, dictionaries, and sets. Please review them before attempting the tasks below:

- LINK 1: Control structures

- LINK 2: Python Lists Tutorial

- LINK 3: Python Dictionaries and set

- LINK 5: Python Tuple

Part A: Practice Problems

1. If-Else and Elif Statements

Write a Python function called “check_grade” that takes a score as input and prints "Pass" if the score is 50 or higher, and "Fail" otherwise. Extend the function to print "Excellent" if the score is 90 or above.

2. For Loop with List

Create a list called “numbers” containing integers from 1 to 10. Write a “for” loop to iterate through the list and print the square of each number.

3. While Loop Example

Write a Python script that uses a “while” loop to sum numbers from 1 to 100. Print the result.

4. Using Dictionaries

Create a dictionary “student_scores” with the following key-value pairs:

- “'Alice': 85”,

- “'Bob': 75”,

- “'Charlie': 95”.

Write a function called “get_top_student” that returns the student with the highest score.

5. Updating Dictionary

Add a new student “'David': 88” to the “student_scores” dictionary and then print the updated dictionary.

6. Set Operations

Create two sets:

- “set_a = {1, 2, 3, 4, 5}”

- “set_b = {4, 5, 6, 7, 8}”

Write Python statements to find and print:

- The union of the two sets.

- The intersection of the two sets.

- The difference between “set_a” and “set_b”.

7. List Comprehension

Create a list of numbers from 1 to 20, then use a list comprehension to create a new list containing only the even numbers from this range. Print the new list.

8. Nested Dictionaries

Create a nested dictionary “class_data” to store information about two classes of students. Each class should have three students, and each student should have a name and a score. Example:

“““python

```
class_data = {
    'Class A': {'John': 85, 'Jane': 92, 'Tom': 78},
    'Class B': {'Alex': 88, 'Chris': 79, 'Emma': 91}
  }
```

“““

Write a function to print the average score of students in each class.

9. Conditional and Loop Combination

Write a Python function “find_divisible_numbers” that takes two arguments: a list of numbers and a divisor. The function should return a list of numbers from the input list that are divisible by the divisor. Test the function with a list of numbers from 1 to 20 and a divisor of 3.

10. Nested Loops and Lists

Create a list of lists called “matrix”, representing a 3x3 matrix of integers. Write a Python script that uses nested loops to print each element of the matrix. Example matrix:

“““python

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

“““

11. Dictionary Comprehension

Write a Python dictionary comprehension that generates a dictionary where the keys are numbers from 1 to 5, and the values are the squares of these numbers. Print the resulting dictionary.

12. Modifying Sets

Given the set “fruits = {'apple', 'banana', 'orange'}”, write a Python script to:

- Add “'mango'“ to the set.

- Remove “'banana'“ from the set.

- Check if “'apple'“ is in the set, and print an appropriate message.

- Print the final set.

13. For Loop with Range and Break

Write a Python function called “find_first_multiple” that takes an integer “n” as input and uses a “for” loop to find the first number divisible by both 3 and 5 between 1 and “n”. If no such number is found, print “"No multiple found"“.

14. Using Zip with Lists

Create two lists:

- “names = ['Alice', 'Bob', 'Charlie']”

- “scores = [85, 90, 78]”

Write a Python function that uses the “zip()” function to combine these lists into a dictionary where the names are the keys and the scores are the values.

15. Counting Frequency with Dictionary

Write a Python function called “count_frequencies” that takes a list of words and returns a dictionary where the keys are the words and the values are the number of times each word appears in the list. Test the function with the list: “['apple', 'banana', 'apple', 'orange', 'banana', 'apple']”.

16. Set Operations with Multiple Sets

Given the sets:

““““python

set_1 = {1, 2, 3, 4}

set_2 = {3, 4, 5, 6}

set_3 = {5, 6, 7, 8}

““““

Write Python code to:

- Find the union of “set_1”, “set_2”, and “set_3”.

- Find the intersection of all three sets.

- Find the symmetric difference between “set_1” and “set_2”.

17. Handling a List of Dictionaries

Create a list of dictionaries where each dictionary represents a student, with keys for “'name'“ and “'grade'“. Example:

““““python

students = [{'name': 'Alice', 'grade': 85}, {'name': 'Bob', 'grade': 90}, {'name': 'Charlie', 'grade': 78}]

““““

Write a Python function to calculate and return the average grade of all students.

18. Dictionary Lookup with Default Values

Create a dictionary “product_prices” with product names as keys and prices as values. Write a Python function that takes a product name as input and returns its price. If the product is not found in the dictionary, the function should return “'Product not found'“.

19. List Manipulation with Control Structures

Write a Python function “filter_and_sort” that takes a list of numbers, filters out all numbers less than 5, and returns the remaining numbers sorted in descending order. Test it with the list “[3, 8, 1, 10, 4, 7]”.

20. Set Operations with Conditional Logic

Write a Python script that creates two sets of numbers:

- “evens = {2, 4, 6, 8, 10}”
- “odds = {1, 3, 5, 7, 9}”

Use conditional logic to check if any number in “evens” also exists in “odds”. If so, print a message indicating which number exists in both sets; otherwise, print “"No common elements"“.

Part B: Submission Instructions

- Complete the assignment in a Python script or Jupyter notebook.
- Ensure that your code is clean, well-commented, and organized.
- Submit your assignment through our designated platform.

By completing this assignment, you’ll strengthen your Python programming skills in control structures and data structures—key areas that are essential for effective data analysis!

Good luck!