

# WEEK 11 & 12

**1- Assume that the variable data refers to the dictionary {"b":20, "a":35}. Write the values of the following expressions:**

```
data = {"b": 20, "a": 35}
print("The following expressions with the dictionary: ")
print("data['a']:", data["a"])
print("data.get('c', None):", data.get("c", None))
print("len(data):", len(data))
print("data.keys():", data.keys())
print("data.values():", data.values())
print("data.pop('b'):", data.pop("b", None))
print("Updated data:", data)
```



The screenshot shows a Python IDE interface with a terminal window. The terminal output matches the code provided in the previous block, showing the results of each print statement. The IDE window title is 'Python' and the file name is 'w11q1.py'.

```
PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q1.py
The following expressions with the dictionary:
data['a']: 35
data.get('c', None): None
len(data): 2
data.keys(): dict_keys(['b', 'a'])
data.values(): dict_values([20, 35])
data.pop('b'): 20
Updated data: {'a': 35}
```

## 2- Write the expressions that perform the following tasks:

**Replace the value at the key “b” in data with that value’s negation. Add the key/value pair “c”:40 to data. Remove the value at key “b” in data, safely. Print the keys in data in alphabetical order.**

```
data = {"b": 20, "a": 35}
print("The following expressions with the dictionary: ")
data["b"] = -data["b"]
data["c"] = 40
data.pop("b", None)
print(sorted(data.keys()))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + - [ ] ... ^ x

```
PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q2.py
The following expressions with the dictionary:
['a', 'c']
PS C:\Users\CSD\Documents\AAKIFPythonLAB> █
```

### 3- Write a python program to read a string and count how many times each letter appears. (Histogram)

```
string = input("Enter a string to count how many times each letter appears and to represent it via histogram- ")
histogram = {}
for char in string:
    if char.isalpha():
        histogram[char] = histogram.get(char, 0) + 1
for char, count in histogram.items():
    print(f"{char}: {count} {'#' * count}")
```



The screenshot shows a terminal window with the following content:

```
PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q3.py
Enter a string to count how many times each letter appears and to represent it via histogram- HANZAH AHMAD YASH
H: 4 #####
A: 5 #####
M: 2 ##
Z: 1 #
D: 1 #
Y: 1 #
S: 1 #
```

#### 4- Write a python program to create a dictionary, read a value from the user and search the key element (Reverse lookup).

```
dict1 = {"a": 1, "b": 2, "c": 3, "d": 4}
value = int(input("Enter a value to search
for its key in the dictionary: "))
keys = [key for key, val in dict1.items() if
val == value]
if keys:
    print(f"Key(s) found: {'',
'.join(keys)}")
else:
    print("No key found for the given
value.")
```

A screenshot of a Python IDE's terminal window. The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the command prompt path: PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q4.py. Below the command, the program's output is displayed: Enter a value to search for its key in the dictionary: 3, followed by Key(s) found: c on a new line.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [ ] ... ^ x
PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q4.py
Enter a value to search for its key in the dictionary: 3
Key(s) found: c
```

## 5- Write a python program to create two dictionaries and merge them.

```
import random
dict1 = {}
for i in range(5):
    dict1[f'key{i}'] = random.randint(1,
100)
print("Dictionary 1:", dict1)

dict2 = {}
for i in range(5):
    dict2[f'key{i}'] = random.randint(1,
100)
print("Dictionary 2:", dict2)

merged_dict = {}
for key, value in dict1.items():
    merged_dict[key] = [value]

for key, value in dict2.items():
    if key in merged_dict:
        merged_dict[key].append(value)
    else:
        merged_dict[key] = [value]

print("Merged Dictionary:", merged_dict)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + - □ □ ... ^ ×

```
PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q5.py
Dictionary 1: {'key0': 2, 'key1': 63, 'key2': 56, 'key3': 95, 'key4': 17}
Dictionary 2: {'key0': 21, 'key1': 57, 'key2': 28, 'key3': 30, 'key4': 17}
Merged Dictionary: {'key0': [2, 21], 'key1': [63, 57], 'key2': [56, 28], 'key3': [95, 30], 'key4': [17, 17]}
```

**6- Write a Python script to create a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys.**

```
print("To find the square of consecutive  
integers upto a range-")  
ran=int(input("Enter range:"))  
result = {x: x**2 for x in range(1, ran)}  
print(result)
```



The screenshot shows a terminal window with the following content:

```
PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q6.py  
To find the square of consecutive integers upto a range-  
Enter range:14  
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169}
```

The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The title bar indicates it is a Python window.

## 7- Implement the Matrix Chain Multiplication problem using dynamic programming in Python.

```
def matrix_chain_order(p):
    n = len(p) - 1
    dp = [[0] * n for _ in range(n)]
    for l in range(2, n + 1):
        for i in range(n - l + 1):
            j = i + l - 1
            dp[i][j] = float('inf')
            for k in range(i, j):
                dp[i][j] = min(dp[i][k] + dp[k + 1][j] + p[i] * p[k + 1] *
                                p[j + 1])
    return dp[0][n - 1]

p = [40, 20, 30, 10, 30]
print(matrix_chain_order(p))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + - [ ] ... ^ x

PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q7.py  
26000  
PS C:\Users\CSD\Documents\AAKIFPythonLAB>

## 8- Develop a Python program to find the shortest path in a graph using Dijkstra's algorithm.

```
import heapq
def dijkstra(graph, start):
    distances = {node: float('inf') for node
in graph}
    distances[start] = 0
    priority_queue = [(0, start)]

    while priority_queue:
        current_distance, current_node =
heapq.heappop(priority_queue)

        if current_distance >
distances[current_node]:
            continue

        for neighbor, weight in
graph[current_node].items():
            distance = current_distance +
weight
            if distance <
distances[neighbor]:
                distances[neighbor] =
distance

    heapq.heappush(priority_queue, (distance,
neighbor))

    return distances
```



```
graph = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 6},
    'C': {'A': 4, 'B': 2, 'D': 3},
    'D': {'B': 6, 'C': 3}
}
start_node = 'A'
print(dijkstra(graph, start_node))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python + - [ ] ... ^ x

PS C:\Users\CSD\Documents\AAKIFPythonLAB> & C:/Users/CSD/AppData/Local/Programs/Python/Python313/python.exe c:/Users/CSD/Documents/AAKIFPythonLAB/w11q8.py  
{'A': 0, 'B': 1, 'C': 3, 'D': 6}