**ADITYA AMIN**

**ASSIGN : 12**

1. Write a Python program to Extract Unique values dictionary values?

def extract\_unique\_values\_from\_dict\_values(dictionary):

# Create a set to store unique values

unique\_values = set()

# Iterate through dictionary values

for values in dictionary.values():

# If value is a list or tuple, iterate through its elements

if isinstance(values, list) or isinstance(values, tuple):

for value in values:

unique\_values.add(value)

else:

# If value is not a list or tuple, add it directly to set

unique\_values.add(values)

return unique\_values

# Input dictionary

dictionary = {"A": [1, 2, 3], "B": (3, 4, 5), "C": 6}

# Call the function to extract unique values from dictionary values

unique\_values = extract\_unique\_values\_from\_dict\_values(dictionary)

# Print the unique values

print("Unique values: ", unique\_values)

1. Write a Python program to find the sum of all items in a dictionary?

def sum\_of\_all\_items\_in\_dict(dictionary):

# Initialize sum to 0

total\_sum = 0

# Iterate through the values of the dictionary

for value in dictionary.values():

# If value is a number (int or float), add it to the sum

if isinstance(value, (int, float)):

total\_sum += value

return total\_sum

# Input dictionary

dictionary = {"A": 10, "B": 20, "C": 30, "D": 40.5}

# Call the function to find the sum of all items in the dictionary

total\_sum = sum\_of\_all\_items\_in\_dict(dictionary)

# Print the total sum

print("Total sum of all items in the dictionary: ", total\_sum)

1. Write a Python program to Merging two Dictionaries?

def merge\_two\_dicts(dict1, dict2):

# Create a new dictionary to store the merged result

merged\_dict = dict()

# Merge dict1 into the merged\_dict

merged\_dict.update(dict1)

# Merge dict2 into the merged\_dict

merged\_dict.update(dict2)

return merged\_dict

# Input dictionaries

dict1 = {"A": 1, "B": 2, "C": 3}

dict2 = {"C": 4, "D": 5, "E": 6}

# Call the function to merge the two dictionaries

merged\_dict = merge\_two\_dicts(dict1, dict2)

# Print the merged dictionary

print("Merged dictionary: ", merged\_dict)

1. Write a Python program to convert key-values list to flat dictionary?

def list\_to\_dict(lst):

# Create an empty dictionary to store the flat result

flat\_dict = dict()

# Iterate through the list of key-value pairs

for pair in lst:

# Unpack the pair into key and value

key, value = pair

# Add key-value pair to the flat\_dict

flat\_dict[key] = value

return flat\_dict

# Input list of key-value pairs

lst = [("A", 1), ("B", 2), ("C", 3), ("D", 4)]

# Call the function to convert the list to a flat dictionary

flat\_dict = list\_to\_dict(lst)

# Print the flat dictionary

print("Flat dictionary: ", flat\_dict)

1. Write a Python program to insertion at the beginning in OrderedDict?

from collections import OrderedDict

def insert\_at\_beginning(od, key, value):

# Create a new OrderedDict to store the updated result

updated\_od = OrderedDict()

# Add the new key-value pair to the updated\_od

updated\_od[key] = value

# Iterate through the original OrderedDict and add its items to updated\_od

for k, v in od.items():

updated\_od[k] = v

return updated\_od

# Input OrderedDict

od = OrderedDict([('A', 1), ('B', 2), ('C', 3)])

# Key-value pair to be inserted at the beginning

key = 'X'

value = 10

# Call the function to insert the key-value pair at the beginning

updated\_od = insert\_at\_beginning(od, key, value)

# Print the updated OrderedDict

print("Updated OrderedDict: ", updated\_od)

1. Write a Python program to check order of character in string using OrderedDict()?

from collections import OrderedDict

def check\_char\_order(string, pattern):

# Create an OrderedDict to store the characters and their indices in string

char\_indices = OrderedDict()

# Iterate through the characters in string

for i, char in enumerate(string):

# Store the character and its index in char\_indices

char\_indices[char] = i

# Initialize the previous index to a very small value

prev\_index = float('-inf')

# Iterate through the characters in pattern

for char in pattern:

# If the character is not present in char\_indices or its index is less than the previous index,

# then the order is not maintained

if char not in char\_indices or char\_indices[char] < prev\_index:

return False

# Update the previous index with the current character's index

prev\_index = char\_indices[char]

# If all characters in pattern are present in char\_indices and their indices maintain the order, return True

return True

# Input string and pattern

string = "hello world"

pattern = "eoo"

# Call the function to check the order of characters in string using OrderedDict

result = check\_char\_order(string, pattern)

# Print the result

print("Order maintained: ", result)

1. Write a Python program to sort Python Dictionaries by Key or Value?

def sort\_dict\_by\_key(d):

"""

Sorts a dictionary by key in ascending order.

Args:

d (dict): The input dictionary to sort.

Returns:

dict: The sorted dictionary by key.

"""

return dict(sorted(d.items()))

def sort\_dict\_by\_value(d, reverse=False):

return dict(sorted(d.items(), key=lambda x: x[1], reverse=reverse))

# Input dictionary

my\_dict = {"one": 1, "three": 3, "four": 4, "two": 2, "five": 5}

# Sort dictionary by key

sorted\_dict\_by\_key = sort\_dict\_by\_key(my\_dict)

print("Sorted dictionary by key: ", sorted\_dict\_by\_key)

# Sort dictionary by value in ascending order

sorted\_dict\_by\_value\_asc = sort\_dict\_by\_value(my\_dict)

print("Sorted dictionary by value (ascending): ", sorted\_dict\_by\_value\_asc)

# Sort dictionary by value in descending order

sorted\_dict\_by\_value\_desc = sort\_dict\_by\_value(my\_dict, reverse=True)

print("Sorted dictionary by value (descending): ", sorted\_dict\_by\_value\_desc)