# Python

# replace None values with the previous non-None value

def ReplaceNoneValue (ls):

preVal = None

for i in range(len(ls)):

if ls[i] == None:

ls[i] = preVal

preVal = ls[i]

else:

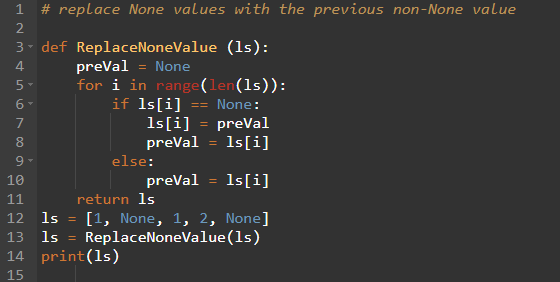
preVal = ls[i]

return ls

ls = [1, None, 1, 2, None]

ls = ReplaceNoneValue(ls)

print(ls)



# Write a function that returns a list of words present in two strings that don't match in case.

def MismatchedWordsFinder (str1, str2):

ls1 = str1.split()

ls2 = str2.split()

resLs = []

minLenght = min(len(ls1), len(ls2))

for i in range(minLenght):

if ls1[i].lower() == ls2[i].lower() and ls1[i] != ls2[i]:

resLs.append(ls1[i])

resLs.append(ls2[i])

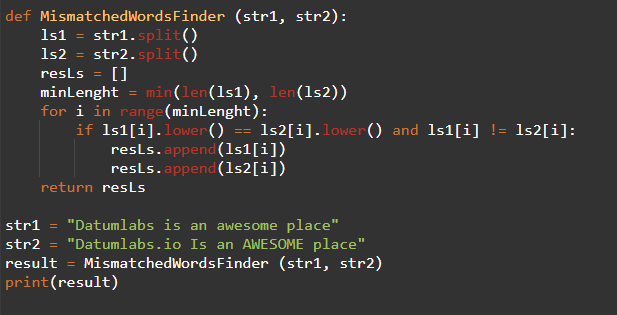
return resLs

str1 = "Datumlabs is an awesome place"

str2 = "Datumlabs.io Is an AWESOME place"

result = MismatchedWordsFinder (str1, str2)

print(result)



# Create a function to count the occurrences of a specific character in a string.

def CharFrequencyCounter (inputStr, countChar):

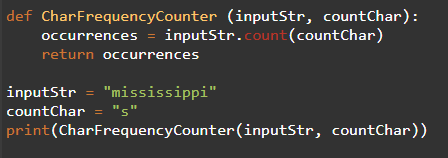
occurrences = inputStr.count(countChar)

return occurrences

inputStr = "mississippi"

countChar = "s"

print(CharFrequencyCounter(inputStr, countChar))



# Write a function to find the key of the nth largest value in a dictionary.

def NthLargestValueKeyFinder (inputDic, n):

sortDic = sorted(inputDic.items(), key = lambda item: item[1], reverse = True)

if n <= len(inputDic):

return sortDic[n-1][0]

else:

return None

inputDic = {

"a": 1,

"b": 2,

"c": 100,

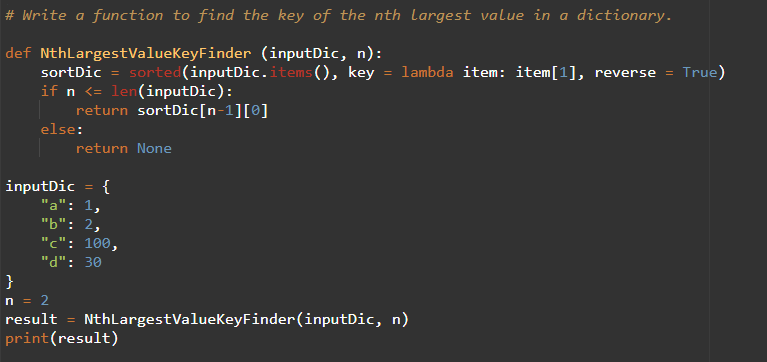
"d": 30

}

n = 2

result = NthLargestValueKeyFinder(inputDic, n)

print(result)



**Explanation:**

1. **Input**:
   * We have a dictionary d = {'a': 1, 'b': 2, 'c': 100, 'd': 30} where the keys are 'a', 'b', 'c', and 'd', and their corresponding values are 1, 2, 100, and 30.
   * We want to find the key of the nth largest value in the dictionary. For example, if n=2, we want the second-largest value.
2. **Sorting the dictionary by values**:
   * d.items() converts the dictionary into a list of key-value pairs, like: [('a', 1), ('b', 2), ('c', 100), ('d', 30)].
   * We then use the sorted() function to sort these items based on their values. The key=lambda item: item[1] part tells Python to sort by the second item in each key-value pair (i.e., the value).
   * The reverse=True argument ensures that the sorting is done in **descending** order, meaning the largest values come first.
   * After sorting, we get: [('c', 100), ('d', 30), ('b', 2), ('a', 1)].
3. **Accessing the nth largest value**:
   * Since the list is sorted in descending order by value, the nth largest value will be the n-1 element in the sorted list (n-1 because lists are zero-indexed in Python).
   * If n = 2, we want the 1st element in the list, which is ('d', 30).
   * The key is extracted using sorted\_items[n-1][0], which gives 'd'.
4. **Handling out-of-bounds n**:
   * If n is greater than the number of elements in the dictionary, the function returns None. This is done by checking if n <= len(sorted\_items) to ensure n is valid.

**Example Walkthrough:**

For d = {'a': 1, 'b': 2, 'c': 100, 'd': 30} and n = 2:

* After sorting by value in descending order, we get sorted\_items = [('c', 100), ('d', 30), ('b', 2), ('a', 1)].
* The second-largest value corresponds to the 1st element in this sorted list, which is ('d', 30).
* The function returns 'd', which is the key for the second-largest value.

# SQL

WITH CustomerBoughtAB AS (

SELECT COUNT(C.customer\_id) AS TotalCustomerWithAB

FROM (

SELECT customer\_id

FROM CustomerPurchases

WHERE payment\_status = 'Paid' AND

product\_id IN ('A', 'B')

GROUP BY customer\_id

HAVING COUNT(DISTINCT product\_id) = 2

) C

),

TotalPaid AS (

SELECT COUNT(DISTINCT customer\_id) AS TotalPaidCustomer

FROM CustomerPurchases

WHERE payment\_status = 'Paid'

)

SELECT (c.TotalCustomerWithAB \* 100) / t.TotalPaidCustomer AS [Percentage Paid Customers AB]

FROM CustomerBoughtAB c

CROSS JOIN TotalPaid t

SELECT (SP.SalesAmount \* 100) / TS.TotalAmount AS [Percentage Sales On Promo Days]

FROM (

SELECT SUM(s.amount) AS SalesAmount

FROM dbo.Sales s

JOIN dbo.Promotions p

ON s.promotion\_id = p.promotion\_id

WHERE s.sale\_date = p.start\_date

OR s.sale\_date = p.end\_date

) AS SP

CROSS JOIN (

SELECT SUM(amount) AS TotalAmount

FROM dbo.Sales

) TS

SELECT TOP 5 C.[Complementary Products]

FROM (

SELECT DISTINCT product\_id AS [Complementary Products]

from dbo.CustomerPurchases

WHERE customer\_id in (

SELECT customer\_id

FROM [dbo].[CustomerPurchases]

WHERE product\_id = 'A')

AND product\_id != 'A') C

# PySpark Metrics Calculation