|  |  |
| --- | --- |
| **Ex. 10** | **EXPLORING DICTIONARIES** |
| **Date:08/04/24** |  |

**Aim:**

To explore dictionaries in Python by writing programs for the following:

1. Obtain a string from the user and create a dictionary that contains each character in the string mapped to the number of times the character appears in the given string. Write programs with and without the count method.
2. Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes. Assume that the list of codes for each English alphabet is as follows: [".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--","-.","---",".--.","--.- ",".-.","...","-","..-","...-",".--","-..-","-.--","--.."]

Create a dictionary that maps each alphabet to the corresponding Morse code. Obtain a list of words from the user and display the Morse code corresponding to each word. Store the codes in a dictionary. If the user inputs the same word multiple times, fetch the corresponding code from the word dictionary instead of computing the code again. Find out if multiple words have the same Morse code and display them.

1. Obtain the marks secured by N students in 5 subjects namely, Math, English, Physics, Chemistry, and Computer Science from the user. The marks secured by each student may be received in the form of a list from the user and stored in a dictionary. Using the data obtained, create a nested dictionary in the following format, which contains the marks, average mark, and rank of each student:

{Student1: {Marks: [50, 45, 36, 48, 50], Average: 45.8, Rank: 2}, Student 2: {Marks:

[43, 44, 48, 30, 37], Average: 40.4, Rank: 3}, Student3: {Marks: [50, 46, 50, 50, 50],

Average: 49.2, Rank: 1}}

1. Create a dictionary grouping a sequence of key-value pairs into a dictionary of lists.
2. Read N numbers from a user and create a dictionary having two keys, EVEN and ODD which point to a list of even and odd numbers respectively.

**Algorithm:**

**(A)**

**WITHOUT COUNT:**

STEP 1: Prompt the user to enter a string and read the input.

STEP 2: Split the string into individual characters to create a list.

STEP 3: Initialize an empty dictionary to store the frequency of each character.

STEP 4: Iterate through the list of characters. For each character, if it is already in the frequency dictionary, increment its count by 1. If not, add it to the dictionary with an initial count of 1.

STEP 5: Print the frequency dictionary to show how many times each character appears in the string.

**WITH COUNT:**

STEP 1: Prompt the user to enter a string and store this input.

STEP 2: Initialize an empty dictionary `frequency` to store character frequencies and an empty string `string2` to keep track of unique characters encountered.

STEP 3: Iterate through each character in the input string. Add each unique character to `string2` if it hasn't been added already.

STEP 4: For each unique character in `string2`, calculate its frequency in the original string using the `count` method and update the `frequency` dictionary with this data.

STEP 5: Print the `frequency` dictionary, which now contains the frequency of each unique character in the input string.

**(B)**

STEP 1: Define a list of Morse code symbols corresponding to each letter from 'a' to 'z'.

STEP 2: Create a dictionary, `morse\_dict`, mapping each lowercase English letter to its corresponding Morse code using a list comprehension.

STEP 3: Prompt the user to enter a list of words and store this list in the variable `lst`.

STEP 4: Initialize an empty dictionary `word\_dict` to store the Morse code translation for each word.

STEP 5: Iterate over each word in the list. For each word, initialize an empty string `morse` to build its Morse code representation.

STEP 6: For each letter in the word, check if the word is already in `word\_dict`. If it is, set `morse` to the Morse code translation stored in `word\_dict`. If not, concatenate the Morse code for the letter from `morse\_dict` to `morse`.

STEP 7: Update `word\_dict` with the word as the key and its Morse code translation as the value.

STEP 8: Print `word\_dict`, which now contains each word from the list mapped to its Morse code representation**.**

**(C)**

STEP 1: Prompt the user to enter the number of students and initialize necessary data structures: `student\_marks` for storing individual student details, `rank` list for rankings, and `avgl` list for averages.

STEP 2: Loop through each student to input their marks in Maths, English, Chemistry, Computer Science, and Physics. Calculate the average mark and append to the `avgl` list.

STEP 3: Store each student's marks, calculated average, and an initially empty rank in the `student\_marks` dictionary using a unique key for each student.

STEP 4: Sort the `avgl` list in descending order to prepare for ranking.

STEP 5: Assign ranks based on the sorted averages. Update the rank in `student\_marks` for each student corresponding to their average.

STEP 6: Print the `student\_marks` dictionary, showing details including marks, average, and rank for each student.

**(d)**

STEP 1: Define a list of tuples, where each tuple contains a color as a string and a corresponding integer.

STEP 2: Initialize an empty dictionary `dict1` to store the aggregated sums for each color.

STEP 3: Iterate over each tuple in the list. For each tuple, check if the color is already a key in the dictionary `dict1`.

STEP 4: If the color exists in the dictionary, increment its associated value by the integer from the tuple. If it does not exist, add the color to the dictionary with its associated value set to the integer from the tuple.

STEP 5: Print the dictionary `dict1`, which now contains the sum of numbers associated with each color**.**

**(E)**

STEP 1: Prompt the user to enter a list of numbers and store this list in the variable `n`.

STEP 2: Initialize a dictionary `dict1` with keys "ODD" and "EVEN", both set to an initial count of 0.

STEP 3: Iterate through each number in the list `n`.

STEP 4: For each number, check if it is even by using the modulus operation `num % 2 == 0`. If true, increment the count for "EVEN" in `dict1`. If false, increment the count for "ODD".

STEP 5: Print the dictionary `dict1`, which now contains the counts of odd and even numbers from the list.

**Program:**

(A)#without using count frequency={} string=input("Enter a string") lst1=[word for word in string] for i in lst1:

if i in frequency: frequency[i]+=1

else:

frequency[i]=1

print(frequency)

**USING COUNT**

string=input("Enter a string:") lst=[word for word in string] frequency={}

string2=""

for i in string:

if i not in string2: string2+=i

for i in string2: frequency.update({i:string.count(i)})

print(frequency)

**(b)**

morse\_code=[".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-

",".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-

..-","-.--","--.."]

morse\_dict={chr(i+97):code for (i,code)in enumerate(morse\_code)}

lst=eval(input("Enter the list of words")) word\_dict={}

for i in lst:

morse="" for j in i:

if j in word\_dict.keys(): morse=word\_dict[j]

else:

morse+=morse\_dict[j] word\_dict.update({i:morse})

print(word\_dict)

**(c)**

n=int(input("Enter the number of students")) student\_marks={}

rank=[] avgl=[] dict1={} avg=0

for i in range(n): mark=[]

print("Enter the details of student",i+1) math=int(input("Maths"))

english=int(input("English:"))

physics=int(input("Physics")) chemistry=int(input("Chemistry")) cs=int(input("Computer")) mark.append([math,english,physics,chemistry,cs]) avg=(math+english+physics+chemistry+cs)/5 avgl.append(avg)

student\_marks[f"student{i+1}"]={"Marks":mark,"Average":avg,"Rank":''}

avgl.sort(reverse=True) for avg in avgl:

rank=n

for students,info in student\_marks.items(): rank-=1

if info["Average"]==avg: info["Rank"]=rank+1

print(student\_marks)

**(d)**

lst=[('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)] dict1={}

for colour,number in lst:

if colour in dict1.keys(): dict1[colour]+=number

else:

dict1[colour]=number print(dict1)

**(e)**

n=eval(input("Enter the elements as list:")) dict1={"ODD":0,"EVEN":0}

for num in n:

if num%2==0:

dict1["EVEN"]+=1

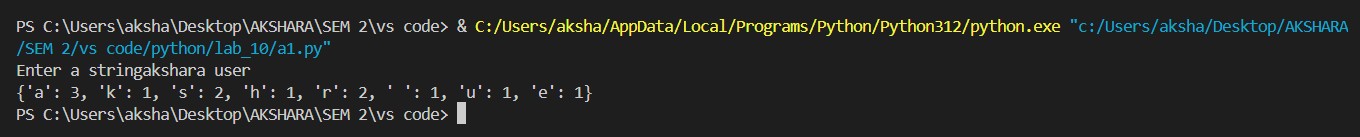
else:

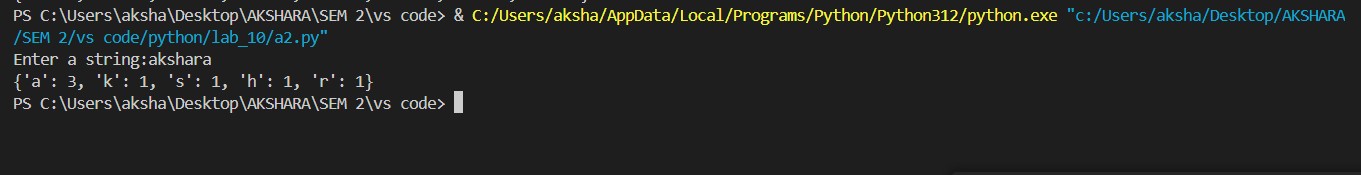
dict1["ODD"]+=1

print(dict1)

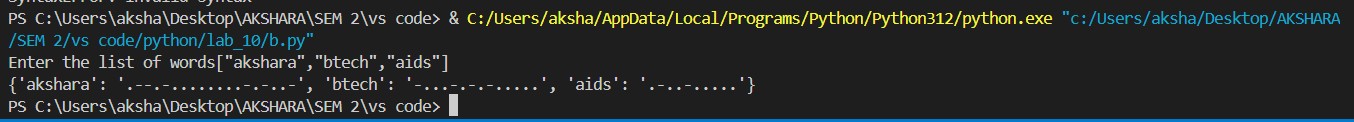
**Screenshot of Output:**

(a)

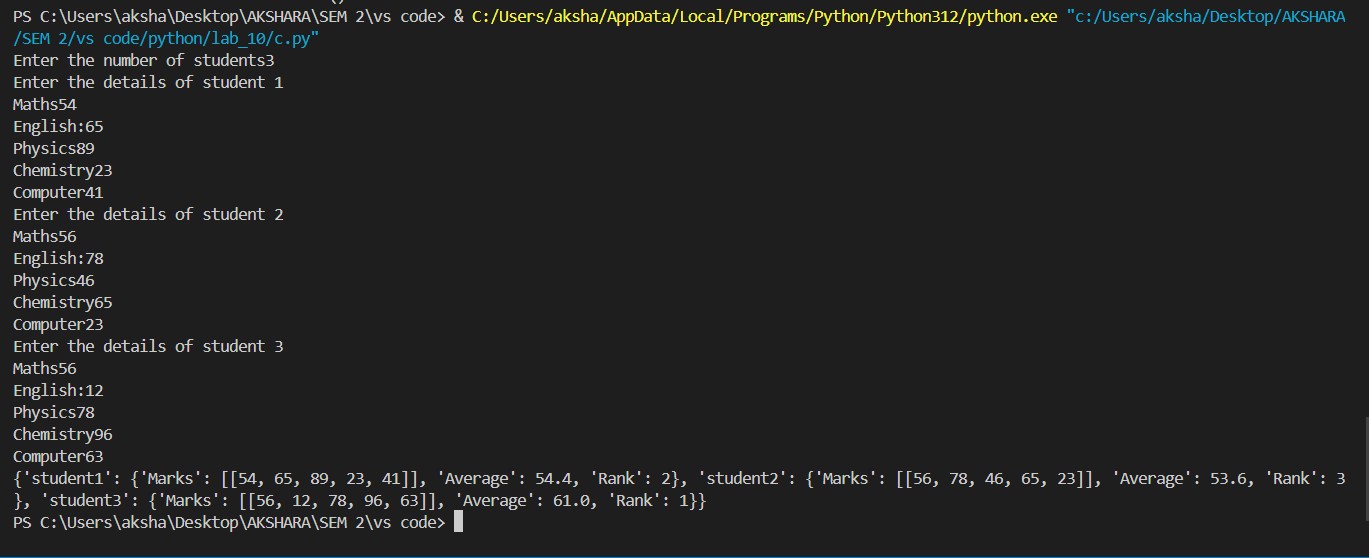




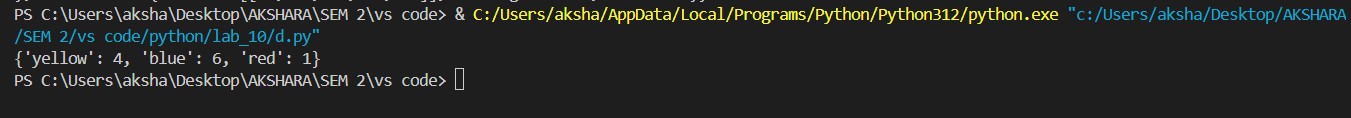
(b)



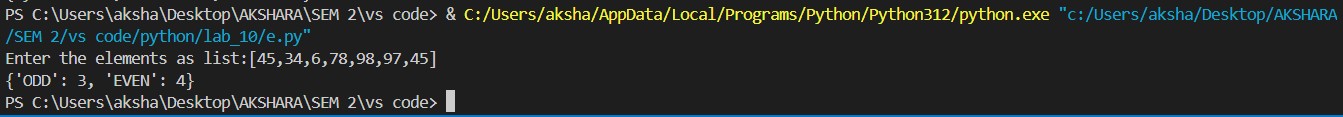
©



(d)



(e)



**Result:**

Thus, programs have been written and executed to explore dictionaries in Python.