WEEK-9-MCQ-Dictionary

1. Which of the following is feature of Dictionary?

a. Dictionary is mutable.

b. Keys are unique within a dictionary.

c. Keys must be of an immutable data type.

d. All of the mentioned

2. Which of the following is not method of dictionary?

a. update( )

b. del ( )

c. pop( )

d. len( )

3. Which of the following are true of Python dictionaries:

a) All the keys in a dictionary must be of the same type.

b) Items are accessed by their position in a dictionary.

c) A dictionary can contain any object type except another dictionary.

d) Dictionaries can be nested to any depth.

e) Dictionaries are mutable.

f) Dictionaries are accessed by key.

a)d,e,f

b)c,d,e

c)b,c

d)a,b

4. pop( ) function delete and _____ the element of dictionary.

a. return

b. display

c. not return

d. add

5. Only values (without keys) can be printed in dictionary?

a.True

b.False

6. In dictionary Keys and values are separated by _____.

a. Colon (:)

b. Comma(,)

c. Semicolon(;)

d. dot(.)

7. Choose the correct statement, in reference to the following code: D1.update(D2) #D1 and D2 are dictionaries

a. None of the mentioned

b. It will create a new dictionary.

c. It will merge all the elements of dictionary 'D1' in dictionary 'D2'.

d. It will merge all the elements of dictionary 'D2' in dictionary 'D1'.

8. Both keys and values are unique in dictionary.

a.True b.False

9. Dictionaries in python are _____.

a. Mutable data type b. Both Non-Mutable data type and Mapping data type
c. Non-Mutable data type d. Mapping data type

10. _____ function returns the value corresponding to the key passed as the argument.

a. values( ) b. get( ) c. del( ) d. update( )

11. Which of the following are immutable data type? A. String B. Tuple C. List D. Dictionary

a)b and d b) c and d c) a and c d) a and b

12. All elements in dictionary are separated by _____.

a. Comma(,) b. Colon (:) c. Semicolon(;) d. dot(.)

13. Which one of the following is correct?

a. A dictionary can have two same keys with different values.

b. A python, a dictionary can neither have two same keys nor two same values.

c. A dictionary can have two same keys or same values but cannot have two same key-value pair

d. A dictionary can have two same values with different keys.

14. Which of the following is an example of dictionary?

a. C = ( ) b. L = [ ] c. D = { } d. None of the mentioned

15. In Python, Dictionaries are immutable

Select one: True False

WEEK-09-CODING-Dictionary

1. Given a number, convert it into corresponding alphabet.

Input	Output
1	A
26	Z
27	AA
676	YZ

Input Format

Input is an integer

Output Format

Print the alphabets

Constraints

1 <= num <= 4294967295

Sample Input 1

26

Sample Output 1

Z

For example:

Test	Result
print(excelNumber(26))	Z

Coding:

```
def excelNumber(n):
    res = []
    while n > 0:
        n -= 1
        rem = n % 26
        res.append(chr(rem + ord('A')))
        n //= 26

    res.reverse()
    return ''.join(res)
```

Output:

	Test	Expected	Got	
✓	`print(excelNumber(26))`	Z	Z	✓
✓	`print(excelNumber(27))`	AA	AA	✓

Passed all tests! ✓

2. Objective:

Develop a Python program that takes an input string from the user and counts the number of occurrences of each vowel (a, e, i, o, u) in the string. The program should be case-insensitive, meaning it should treat uppercase and lowercase vowels as the same.

Description:

Vowels play a significant role in the English language and other alphabet-based languages. Counting vowels in a given string is a fundamental task that can be applied in various text processing applications, including speech recognition, linguistic research, and text analysis. The objective of this problem is to create a Python script that accurately counts and displays the number of times each vowel appears in a user-provided string.

Program Requirements:

Input:

First line reading String as input, The string can contain any characters, including letters, numbers, and special characters.

Output:

Display the number of occurrences of each vowel in the string.

The output should list each vowel followed by its count.

Example:

Consider the following example for better understanding:

- **Input:** "Python Programming"
- **Output**

  a = 1
  e = 0
  i = 1
  o = 2
  u = 0

For example:

Input	Result
Hello World	a = 0 e = 1 i = 0 o = 2 u = 0
Python	a = 0 e = 0 i = 0 o = 1 u = 0

Coding:

```
def main(s):
    s = s.lower()
    v = {'a':0,'e':0,'i':0,'o':0,'u':0}

    for char in s:
        if char in v:
            v[char] += 1

    for vowel in ['a','e','i','o','u']:
        print(f"{vowel} = {v[vowel]}")

s = input()
main(s)
```

Output:

	Input	Expected	Got	
✓	Hello World	a = 0 e = 1 i = 0 o = 2 u = 0	a = 0 e = 1 i = 0 o = 2 u = 0	✓
✓	AEIOU aeio u	a = 2 e = 2 i = 2 o = 2 u = 2	a = 2 e = 2 i = 2 o = 2 u = 2	✓
✓	Python	a = 0 e = 0 i = 0 o = 1 u = 0	a = 0 e = 0 i = 0 o = 1 u = 0	✓
✓	abcdefghijklmnopqrstuvwxyz	a = 1 e = 1 i = 1 o = 1 u = 1	a = 1 e = 1 i = 1 o = 1 u = 1	✓
✓	12345!@#$%AEIOU	a = 1 e = 1 i = 1 o = 1 u = 1	a = 1 e = 1 i = 1 o = 1 u = 1	✓

Passed all tests! ✓

3. In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points. The points associated with each letter are shown below:

Points Letters

1 A, E, I, L, N, O, R, S, T and U

2 D and G

3 B, C, M and P

4 F, H, V, W and Y

5 K

8 J and X

10 Q and Z

Write a program that computes and displays the Scrabble™ score for a word. Create a dictionary that maps from letters to point values. Then use the dictionary to compute the score.

A Scrabble™ board includes some squares that multiply the value of a letter or the value of an entire word. We will ignore these squares in this exercise.

Sample Input

REC

Sample Output

REC is worth 5 points.

For example:

Input	Result
REC	REC is worth 5 points.

Coding:

```
def main(w):
    letter = {
        'A':1,'E':1,'I':1,'L':1,'N':1,'O':1,'R':1,'S':1,'T':1,'U':1,
        'D':2,'G':2,
        'B':3,'C':3,'M':3,'P':3,
        'F':4,'H':4,'V':4,'W':4,'Y':4,
        'K':5,
        'J':8,'X':8,
        'Q':10,'Z':10
    }
    t = 0
    w = w.upper()

    for l in w:
        if l in letter:
            t +=letter[l]
    return t

n = input()
s = main(n)
print(f"{n} is worth {s} points.")
```

Output:

	Input	Expected	Got	
✓	GOD	GOD is worth 5 points.	GOD is worth 5 points.	✓
✓	REC	REC is worth 5 points.	REC is worth 5 points.	✓

Passed all tests! ✓

4. Create a student dictionary for n students with the student name as key and their test mark assignment mark and lab mark as values. Do the following computations and display the result.

1.Identify the student with the highest average score

2.Identify the student who as the highest Assignment marks

3.Identify the student with the Lowest lab marks

4.Identify the student with the lowest average score

Note:

If more than one student has the same score display all the student names

Sample input:

4

James 67 89 56

Lalith 89 45 45

Ram 89 89 89

Sita 70 70 70

Sample Output:

Ram

James Ram

Lalith

Lalith

For example:

Input	**Result**
4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith

Coding:

```
def main(m):
    return sum(m) / len(m)
n = int(input())
s ={}
for _ in range(n):
    d = input().split()
    n = d[0]
    m = list(map(float,d[1:]))
    s[n] = m

ma = max(main(m) for m in s.values())
```

```
ha = [n for n,m in s.items() if main(m) == ma]

mas = max(m[1] for m in s.values())
has = [n for n,m in s.items() if m[1] == mas]

ml = min(m[2] for m in s.values())
ll = [n for n, m in s.items() if m[2] == ml]

mi = min(main(m) for m in s.values())
la = [n for n, m in s.items() if main(m) == mi]

print(" ".join(ha))
print(" ".join(has))
print(" ".join(sorted(ll)))
print(" ".join(la))
```

Output:

	Input	Expected	Got	
✓	4 James 67 89 56 Lalith 89 45 45 Ram 89 89 89 Sita 70 70 70	Ram James Ram Lalith Lalith	Ram James Ram Lalith Lalith	✓
✓	3 Raja 95 67 90 Aarav 89 90 90 Shadhana 95 95 91	Shadhana Shadhana Aarav Raja Raja	Shadhana Shadhana Aarav Raja Raja	✓

Passed all tests! ✓

5. A sentence is a string of single-space separated words where each word consists only of lowercase letters.A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.

Given two sentences s1 and s2, return a list of all the uncommon words. You may return the answer in any order.

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

Constraints:

1 <= s1.length, s2.length <= 200

s1 and s2 consist of lowercase English letters and spaces.

s1 and s2 do not have leading or trailing spaces.

All the words in s1 and s2 are separated by a single space.

Note:

Use dictionary to solve the problem

For example:

Input	Result
this apple is sweet this apple is sour	sweet sour

Coding:

```
def main(s1,s2):
    from collections import Counter

    c1 = Counter(s1.split())
    c2 = Counter(s2.split())

    u1 = {word for word in c1  if c1[word] == 1 and word not in c2}
    u2 = {word for word in c2 if c2[word] == 1 and word not in c1}

    u = list(u1.union(u2))
    return ' '.join(u)

s1 = input()
s2 = input()
print(main(s1,s2))
```

Output:

	Input	Expected	Got	
✓	this apple is sweet this apple is sour	sweet sour	sweet sour	✓
✓	apple apple banana	banana	banana	✓

Passed all tests! ✓

6. You are given a string word. A letter is called **special** if it appears both in lowercase and uppercase in word.

Your task is to return the number of **special** letters in word.

Constraints

- The input string word will contain only alphabetic characters (both lowercase and uppercase).
- The solution must utilize a dictionary to determine the number of special letters.
- The function should handle various edge cases, such as strings without any special letters, strings with only lowercase or uppercase letters, and mixed strings.

Examples
Example 1:

Input: word = "aaAbcBC"
Output: 3

Explanation:
The special characters in `word` are 'a', 'b', and 'c'.

Example 2:

Input: word = "abc"
Output: 0

Explanation:
No character in `word` appears in uppercase.

For example:

Test	Result
print(count_special_letters("AaBbCcDdEe"))	5

Coding:

```
def count_special_letters(word: str) -> int:
    # Your implementation here
    l = {}
    u = {}

    for char in word:
        if char.islower():
            l[char] = True
        elif char.isupper():
            u[char.lower()] = True
    c = 0
    for char in l:
        if char in u:
            c += 1
    return c
```

Output:

	Test	Expected	Got	
✓	print(count_special_letters("AaBbCcDdEe"))	5	5	✓
✓	print(count_special_letters("ABCDE"))	0	0	✓
✓	print(count_special_letters("abcde"))	0	0	✓

Passed all tests! ✓

7. A company wants to send its quotation secretly to its client. The company decided to encrypt the amount they are sending to their client with some special symbols so that the equation amount will not be revealed to any external person. They used the special symbols !,@,#,$,%,^,&,*,>,< for 0,1,2,3,4,5,6,7,8,9 respectively. Write a python code to help the company to convert the amount to special symbols.

(Value rounded off to 2 decimal points)

Input

n: Float data type which reads amount to send

Output

s: : String data type which displays symbols

Sample Testcase 1

Input

10000

Output

@!!!!.!!

Sample Testcase2

1234.56

Output

@#$%.^&

For example:

Input	Result
1345.23	@$%^.#$
15000.59	@^!!!.^<
156789	@^&*><.!!

Coding:

```
def main(a):
    s={'0':'!','1':'@','2':'#','3':'$','4':'%','5':'^','6':'&','7':'*','8':'>','9':'<','.':'}
    am = f"{a:.2f}"
    res =''.join(s[char] for char in am if char in s)
    return res

n = float(input())
print(main(n))
```

Output:

	Input	Expected	Got	
✓	1345.23	@$%^.#$	@$%^.#$	✓
✓	15000.59	@^!!!.^<	@^!!!.^<	✓
✓	1234	@#$%.!!	@#$%.!!	✓
✓	156789	@^&*><.!!	@^&*><.!!	✓

Passed all tests! ✓

8. Given an array of names of candidates in an election. A candidate name in the array represents a vote cast to the candidate. Print the name of candidates received Max vote. If there is tie, print a lexicographically smaller name.

Examples:

Input : votes[] = {"john", "johnny", "jackie", "johnny", "john", "jackie","jamie", "jamie", "john","johnny", "jamie", "johnny","john"};

Output : John

We have four Candidates with name as 'John', 'Johnny', 'jamie', 'jackie'. The candidates John and Johny get maximum votes. Since John is alphabetically smaller, we print it. Use dictionary to solve the above problem

Sample Input:

10

John

John

Johny

Jamie

Jamie

Johny

Jack

Johny

Johny

Jackie

Sample Output:

Johny

Coding:

```
from collections import defaultdict

def main(vs):
    vc = defaultdict(int)

    for v in vs:
        vc[v.lower()] += 1

    mv = max(vc.values())
    mc = [c for c in vc if vc[c] == mv]

    return min(mc).capitalize()

n = int(input())
vs = []
for _ in range(n):
    v = input()
    vs.append(v)

print(main(vs))
```

Output:

	Input	Expected	Got	
✓	10 John John Johny Jamie Jamie Johny Jack Johny Johny Jackie	Johny	Johny	✓
✓	6 Ida Ida Ida Kiruba Kiruba Kiruba	Ida	Ida	✓

Passed all tests! ✓

9. Give a dictionary with value lists, sort the keys by summation of values in value list.

Input : test_dict = {'Gfg' : [6, 7, 4], 'best' : [7, 6, 5]}

Output : {'Gfg': 17, 'best': 18}

Explanation : Sorted by sum, and replaced.

Input : test_dict = {'Gfg' : [8,8], 'best' : [5,5]}

Output : {'best': 10, 'Gfg': 16}

Explanation : Sorted by sum, and replaced.

Sample Input:

2

Gfg 6 7 4

Best 7 6 5

Sample Output

Gfg 17

Best 18

For example:

Input	Result
2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18

Coding:

```
def main():
    import sys
    input = sys.stdin.read

    d = input().strip().split('\n')
    e = int(d[0])
    dd ={}

    for i in range(1 , e+1):
        p = d[i].split()
        key = p[0]
        v = list(map(int,p[1:]))
        s = sum(v)
        dd[key] = s

    sd = dict(sorted(dd.items(),key=lambda item:item[1]))

    for key, res in sd.items():
        print(f"{key} {res}")
main()
```

Output:

	Input	Expected	Got	
✓	2 Gfg 6 7 4 Best 7 6 5	Gfg 17 Best 18	Gfg 17 Best 18	✓
✓	2 Gfg 6 6 Best 5 5	Best 10 Gfg 12	Best 10 Gfg 12	✓

Passed all tests! ✓

10. A sentence is a list of words that are separated by a single space with no leading or trailing spaces. Each word consists of lowercase and uppercase English letters.

A sentence can be shuffled by appending the 1-indexed word position to each word then rearranging the words in the sentence.

For example, the sentence "This is a sentence" can be shuffled as "sentence4 a3 is2 This1" or "is2 sentence4 This1 a3".

Given a shuffled sentence s containing no more than 9 words, reconstruct and return the original sentence.

Example 1:

Input:

is2 sentence4 This1 a3

Output:

This is a sentence

Explanation: Sort the words in s to their original positions "This1 is2 a3 sentence4", then remove the numbers.

Example 2:

Input:

Myself2 Me1 I4 and3

Output:

Me Myself and I

Explanation: Sort the words in s to their original positions "Me1 Myself2 and3 I4", then remove the numbers.

Constraints:

2 <= s.length <= 200

s consists of lowercase and uppercase English letters, spaces, and digits from 1 to 9.

The number of words in s is between 1 and 9.

The words in s are separated by a single space.

s contains no leading or trailing spaces.

Coding:

```
def show(s):
    ws = s.split()
    p = []
    for w in ws:
        if w[-1].isdigit():
            pos = int(w[-1])
            o = w[:-1]
            p.append((pos,o))
            p.sort()
    os = ' '.join(w for _,w in p)
    return os

n = input()
print(show(n))
```

Output:

	Input	Expected	Got	
✓	is2 sentence4 This1 a3	This is a sentence	This is a sentence	✓

Passed all tests! ✓