**Dictionaries**

This section will serve as a brief introduction to dictionaries and consist of:

1.) Constructing a Dictionary

2.) Accessing objects from a dictionary

3.) Nesting Dictionaries

4.) Basic Dictionary Methods

So what are mappings? Mappings are a collection of objects that are stored by a *key*, unlike a sequence that store objects by their relative position. This is an important distinction, since mappings won't retain order since they have objects defined by a key.

A Python dictionary consists of a key and then an associated value. That value can be almost any Python object.

Let's see how we can construct dictionaries to get a better understanding of how they work!

#Code

*# Make a dictionary with {} and : to signify a key and a value*

my\_dict = {'key1':'value1','key2':'value2'}

*# Call values by their key*

my\_dict['key2']

Output

'value2'

It's important to note that dictionaries are very flexible in the data types they can hold. For example:

#Code

my\_dict = {'key1':123,'key2':[12,23,33],'key3':['item0','item1','item2']}

*# Let's call items from the dictionary*

my\_dict['key3']

*# Can call an index on that value*

my\_dict['key3'][0]

Output

['item0', 'item1', 'item2']

'item0'

We can affect the values of a key as well. For instance:

my\_dict['key1']

*# Subtract 123 from the value*

my\_dict['key1'] = my\_dict['key1'] - 123

*#Check*

my\_dict['key1']

Output

123

0

We can also create keys by assignment. For instance if we started off with an empty dictionary, we could continually add to it:

#Code

*# Create a new dictionary*

d = {}

*# Create a new key through assignment*

d['animal'] = 'Dog'

*# Can do this with any object*

d['answer'] = 42

*#Show*

D

Output

{'animal': 'Dog', 'answer': 42}

**Nesting with Dictionaries**

Hopefully you're starting to see how powerful Python is with its flexibility of nesting objects and calling methods on them. Let's see a dictionary nested inside a dictionary:

In:

*# Dictionary nested inside a dictionary nested inside a dictionary*

d = {'key1':{'nestkey':{'subnestkey':'value'}}}

Let's see how we can grab that value:

In:

*# Keep calling the keys*

d['key1']['nestkey']['subnestkey']

Out:

'Value'

**Built-in Dictionary methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| **clear()** | Removes all the elements from the dictionary |
| **copy()** | Returns a copy of the dictionary |
| **fromkeys()** | Returns a dictionary with the specified keys and value |
| **get()** | Returns the value of the specified key |
| **items()** | Returns a list containing a tuple for each key value pair |
| **keys()** | Returns a list containing the dictionary's keys |
| **pop()** | Removes the element with the specified key |
| **popitem()** | Removes the last inserted key-value pair |
| **setdefault()** | Returns the value of the specified key. If the key does not exist: insert the key, with the specified value |
| **update()** | Updates the dictionary with the specified key-value pairs |
| **values()** | Returns a list of all the values in the dictionary |
| **has\_key()** | Returns *true* if key in dictionary *dict*, *false* otherwise |

**PROBLEM STATEMENTS(Prepared by our group)**

1.Take a statement as an input.Traverse through the statement and count the occurrences of distinct characters.

Eg. In “ This is great” , the output should be [{a:1},{e:1},{g:1},{h:1},{i:,2},{r:1},{s:2},{t:1},{T:1}]

 2.Write a function called acceptlogin(users, username, password)with three parameters:

●     users, showing no. of users

●     username,a string for a login name and

●     password,a string for a password.

The function should return True if the user exists and the password is correct and False otherwise.

Hint: Initially setup a few users and then verify whether any input user exists.

**PROBLEM STATEMENTS(Solved by our group)**

Q1.Write a function to invert a dictionary. It should accept a dictionary as a parameter and return a   dictionary where the keys are values from the input dictionary and the values are lists of keys from the input dictionary. For example, this input:

CODE:

n=int(input("Enter the number of dictionary values you want to input "))

new={}

rev={}

for \_ in range(n):

    i=input("enter the key and value separated by a :").split(":")

    new[i[0]]=i[1]

for key, value in new.items():

    rev.setdefault(value, list()).append(key)

print("The given dictionary is",new)

print("The required reversed dictionary is ",rev)

OUTPUT:

Enter the number of dictionary values you want to input 3

enter the key and value separated by a :key1:value1

enter the key and value separated by a :key2:value2

enter the key and value separated by a :key3:value1

The given dictionary is {'key1': 'value1', 'key2': 'value2', 'key3': 'value1'}

The required reversed dictionary is  {'value1': ['key1', 'key3'], 'value2': ['key2']}

Q2. Write a python program to remove duplicates from Dictionary.

Steps:

1.Input dictionary A from user

2. Create a new empty dictionary, let us call it as B

3. Go to the items of our main dictionary (A) one by one

4. Search if the current item from A exists in B

5. If it exists, move to next item, if it does not exist then add it in B

6. Repeat these steps until all items of A are processed. In the end, you will get a dictionary B which has only unique items.

Code:

A = dict()

B = dict()

f = 1

while(f==1):

    key = input("Enter the key : ")

    value = input("Enter the value : ")

    A[key] = value

    f = int(input("Press 1 to enter more values and 0 to stop : "))

for key,value in A.items():

  if value not in B.values():

    B[key] = value

print("Dictionary before ", A)

print("Dictionary with unique values", B)

Output:

Enter the key : a

Enter the value : 100

Press 1 to enter more values and 0 to stop : 1

Enter the key : b

Enter the value : 500

Press 1 to enter more values and 0 to stop : 1

Enter the key : b

Enter the value : 500

Press 1 to enter more values and 0 to stop : 1

Enter the key : c

Enter the value : 20

Press 1 to enter more values and 0 to stop : 1

Enter the key : d

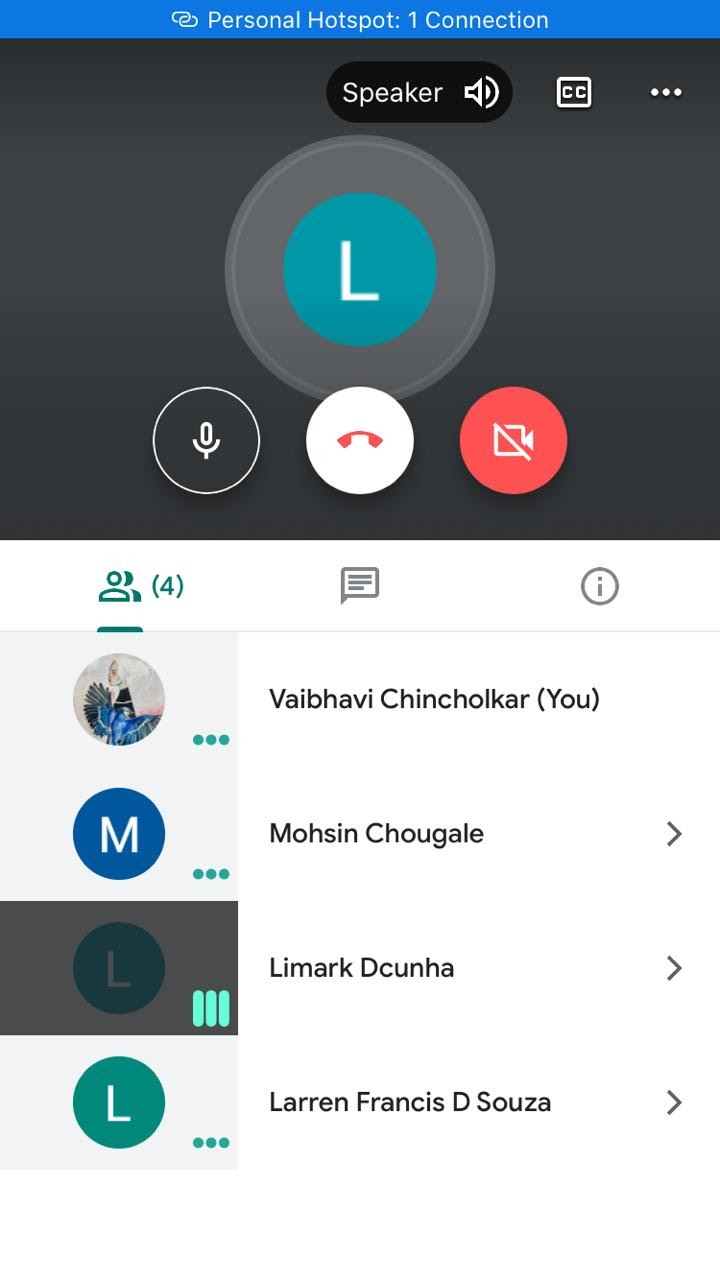
Enter the value : 75

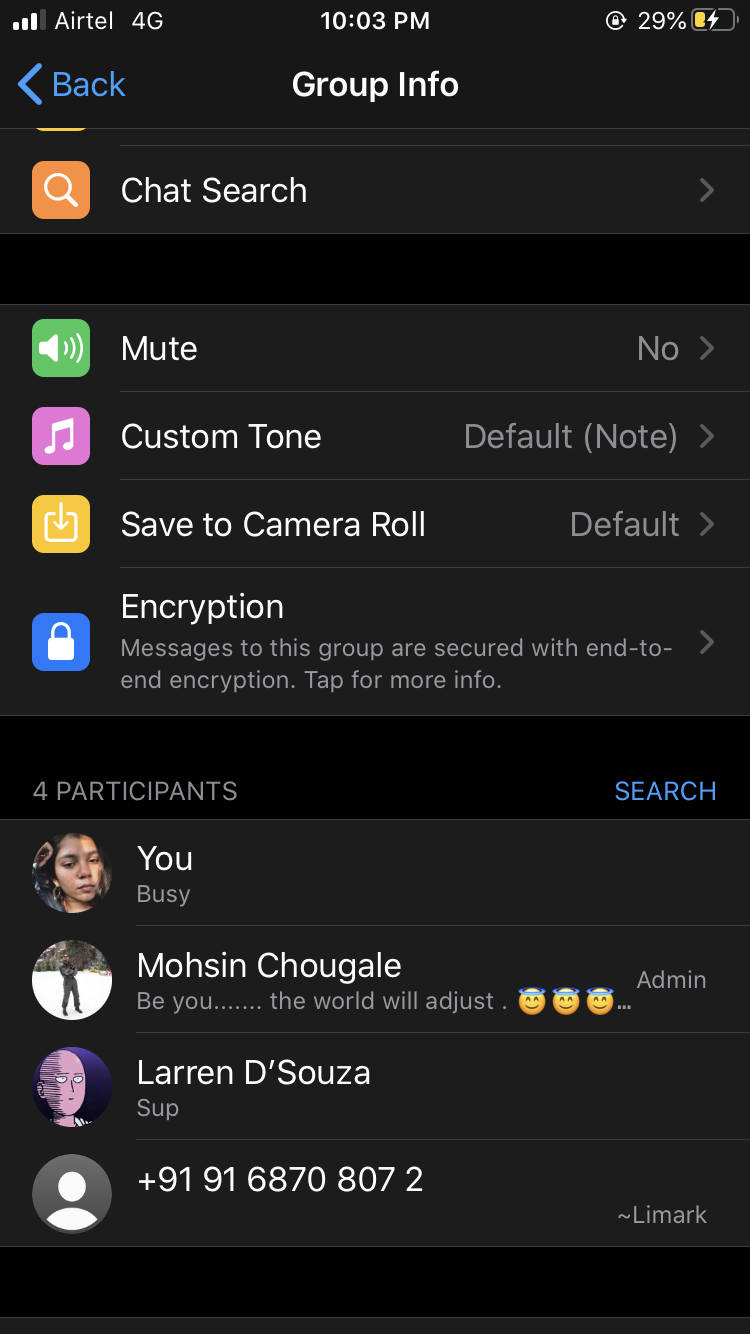
Press 1 to enter more values and 0 to stop : 0

Dictionary before {‘a’: ‘100’, ‘b’: ‘500’, ‘b’: ‘500’, ‘c’: ‘20’, ‘d’: ‘75’}

Dictionary with unique values {‘a’: ‘100’, ‘b’: ‘500’, ‘c’: ‘20’, ‘d’: ‘75’}

**Proof of communication:**

****

****

**Member’s contribution:**

**Vaibhavi - 2019140012 - solution for problem no. 2**

**Mohsin - 2019140013 - Basics of dictionaries, 2nd problem statement(For other group)**

**Larren - 2019140014- solution for problem no.1**

**Limark - 2019140015- Built-in Dictionary methods,1st problem statement (For other group)**

**Reference:**

[**www.tutorialspoint.com**](http://www.tutorialspoint.com/)

[**www.w3schools.com**](http://www.w3schools.com/)

[**https://github.com/Pierian-Data/Complete-Python-3-Bootcamp/tree/master/00-Python%20Object%20and%20Data%20Structure%20Basics**](https://github.com/Pierian-Data/Complete-Python-3-Bootcamp/tree/master/00-Python%20Object%20and%20Data%20Structure%20Basics)

https://docs.google.com/document/d/1uWCpmu0OBjt2bnccRI8O5Bt22ySbU8HaHm1Oj022yuw/edit?usp=sharing

Group Roll nos.

2019140012

2019140013

2019140014

2019140015