Computational Thinking and Programming - 2

Review of Dictionaries

XII

Definition

Python dictionary is a mutable data type in Python, in which both the indexes as well as values can be changed. It is an unordered collection of items, changeable and indexed. While other compound data types have only value as an element, a dictionary has a key:value pair. An item in the dictionary has a key and the corresponding value expressed as a pair, key : value.

No two words (keys) in the dictionary are the same, but this restriction does not apply to their values (meanings).

While values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique.

Dictionary keys are case sensitive, same name but different cases of keys will be treated distinctly.

Syntax and Examples of Dictionaries

Syntax

```
Dict_name = {<Index1/Key1>:<Value1>, <Index2/Key2>:<Value2>,.....}

my_dict = {}                            # Empty dictionary

my_dict = {1: 'apple', 2: 'mango'}      # Dictionary with key and value pairs

# Dictionary with details of months

my_dict = {"January":31, "February":28, "March":31}

# Dictionary with details of a student having participation in various events

my_dict = {'Name': 'Hamza', “Events”: ["Quiz","hindi quiz","WarP","English Debate"]}
```

Another way to create a dictionary

- **Specifying Key: Value pairs as arguments to dict() function**
  Keys are passed as arguments and values as the values of the arguments.
  ```
  employee=dict (name = 'John', salary=10000, age=24)
  ```

- **Specifying comma separated key:value pairs**
  Key and value pairs are enclosed in curly brackets
  ```
  employee=dict ({"name" : 'John', "salary":10000, "age":24})
  ```

- **Specify keys separately and corresponding values separately**
  Keys and values are enclosed separately as tuples and are given as arguments to zip() function
  ```
  employee=dict(zip(("name","salary","age"),("John",10000,24)))
  ```

Traversal and Assignment

The contents of a dictionary can be changed. New content can be added in a dictionary and existing content can also be deleted.

```
my_dict= {"January":31, "February":28, "March":31}  # Dictionary assignment

print(my_dict)                              # display all the keys with their values

print(mydict["January"])                    # display value of particular key

# display the number of elements in the dictionary

print("Total elements in the dictionary : ",len(my_dict))
```

OUTPUT

```
{"January":31, "February":28, "March":31}
31
Total elements in the dictionary:3
```

Example

CODE

```
S={'RNo':1,'Name':"Freya Jain"}
print(S["RNo"],S["Name"])
print(S,type(S))
S["RNo"]=2
print(S)
S["Marks"]=90
print(S)
```

OUTPUT

```
1 Freya Jain
{'RNo': 1, 'Name': 'Freya Jain'} <class 'dict'>
{'RNo': 2, 'Name': 'Freya Jain'}
{'RNo': 2, 'Name': 'Freya Jain', 'Marks': 90}
```

Traversal and Assignment

Elements can be accessed using a for loop where the range is specified by keys() method of the dictionary.

```
my_dict= {"January":31, "February":28, "March":31, "April":30}
for k in my_dict.keys():      # each iteration of the for loop refers to a key
     print(k, ' has ',my_dict[k], ' days ')
```

OUTPUT

```
January has 31 days
February has 28 days
March has 31 days
April has 30 days
```

Traversal and Assignment

The method items() in the dictionary iterates over all the keys and helps us to access the key-value pair one after the other in the loop and is also a good method to access dictionary keys with value.

```
for k,v in my_dict.items():
      print(k, ' has ', v, ' days')
# each iteration of the for loop retrieves a tuple
```

OUTPUT

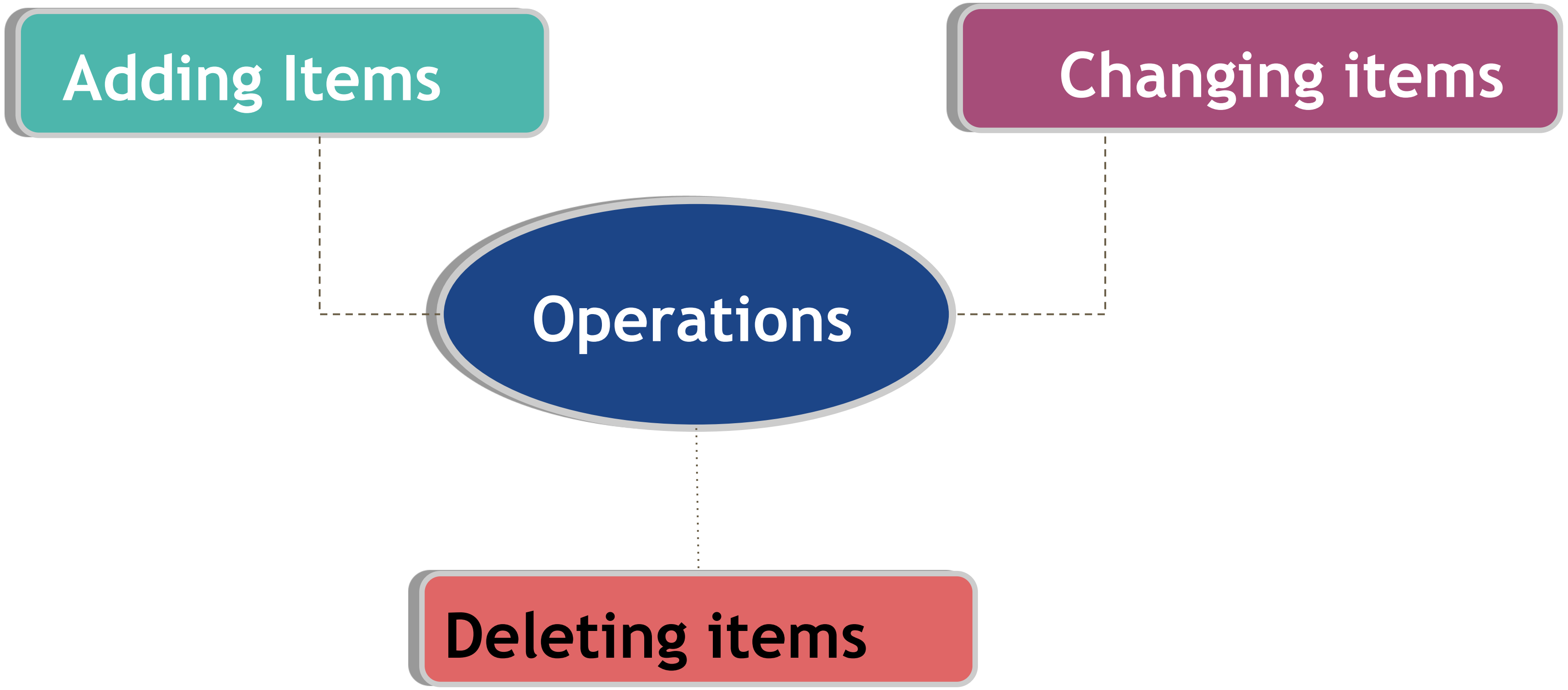
```
January has 31 days
February has 28 days
March has 31 days
```

```
#display the whole dictionary at once using the items() method
print(my_dict.items())
```

OUTPUT

```
dict_items([('January',31),('February',28)('March',31)])
```

Operations on Dictionary



Adding items to dictionary

A new item (key:value pair) can also be added by the simple assignment statement. The only constraint is that the key must be unique. If the key already exists, then the value is simply changed as in the assignment statement above.

Syntax : dictionaryname[new keyname]=new value

Example

CODE

```
d1={1:"Apple",2:"Mango"}
print("Dictionary1 :")
print(d1)
print("After adding :")
d1[3]="Orange"
print(d1)
```

OUTPUT

```
Dictionary1 :
{1:"Apple",2:"Mango"}
After adding :
{1:"Apple",2:"Mango",3:"Orange"}
```

Changing items of existing dictionary

An existing item (key:value pair) can also be changed or modified by the simple assignment statement.

Syntax : dictionaryname[keyname]=new value

Example

CODE

```
d1={1:"Apple",2:"Mango"}
print("Dictionary1 :")
print(d1)
print("After changing :")
d1[2]="Orange"
print(d1)
```

OUTPUT

```
Dictionary1 :
{1:"Apple",2:"Mango"}
After changing :
{1:"Apple",2:"Orange"}
```

Creating a dictionary with input from the user

CODE 1

```
d = { }
for i in range(5):
    name =input("Enter name : ")
     marks=int(input("Enter marks :"))
    d[name]=marks
print(d)
```

CODE 2

```
d = { }
for i in range(5):
    name =input("Enter name : ")
     marks=int(input("Enter marks :"))
    d[name]=marks
for i in d:
    print(i,"has scored" ,d[i])
```

OUTPUT

```
Enter name:Aditya
Enter marks:77
.
.
.
{"Aditya":77,"Ravi":85}
```

```
Enter name:Aditya
Enter marks:77
.
.
Aditya has scored 77
Ravi has scored 85
```

Exercise :

Find the output of the following statements :

```
weight={'A1':8,'A2':7,'B1':6,'B2':5,'C1':4}
```

```
1.  for k in weight:
        print(k,"-",weight[k])
2.  for k in weight.keys():
        print(k,"-",weight[k])
3.  for k in weight.values():
        print(k, end=' ')
4.  for k,v in weight.items():
        print(k,"-",v)
```

1.
```
A1 - 8
A2 - 7
B1 - 6
B2 - 5
C1 - 4
```

2.
```
A1 - 8
A2 - 7
B1 - 6
B2 - 5
C1 - 4
```

3.
```
8 7 6 5 4
```

4.
```
A1 - 8
A2 - 7
B1 - 6
B2 - 5
C1 - 4
```

Dictionary keys() and values()

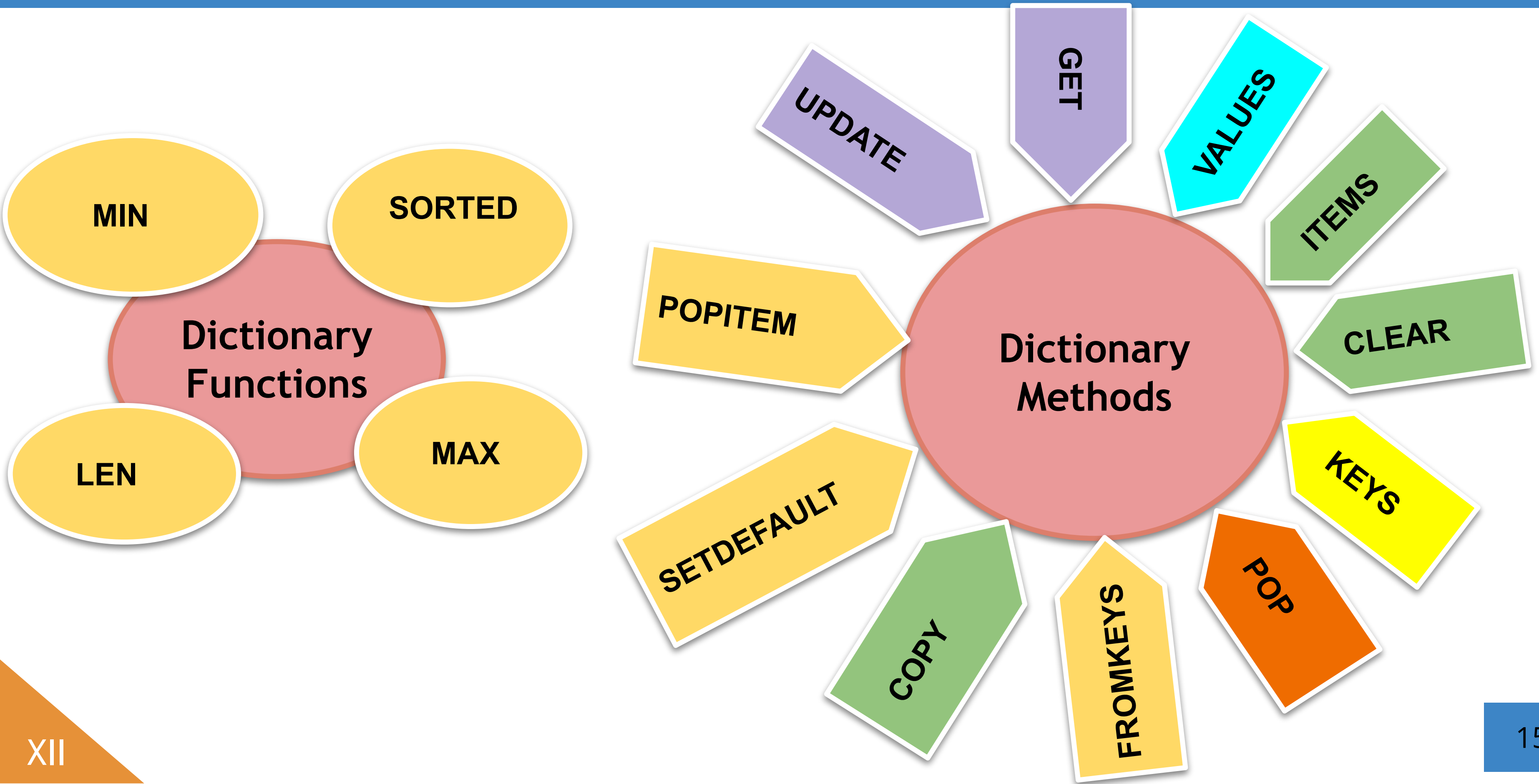
Consider the following code:

```
S={'RNo':1,'Name':"Freya Jain"}
print("Keys : ", S.keys())            # Displays the list of keys
print("Values : ",S.values())         # Displays the list of values
print("Dictionary S : ",S)
print("Items : ",S.items())           # Displays the list of items
```

OUTPUT

```
Keys :  dict_keys(['RNo', 'Name'])
Values :  dict_values([1, 'Freya Jain'])
Dictionary S :  {'RNo': 1, 'Name': 'Freya Jain'}
Items : dict_items([('RNo', 1), ('Name', 'Freya Jain')])
```

Dictionary Functions and Methods



Dictionary - len()

1. **len()**

Returns the length of the dictionary or the number of elements in the dictionary.

Syntax : **len(dictionaryname)**

Example

```
D1={1:"Apple",2:"Mango"}
print("Length:",len(d1))
```

OUTPUT

```
Length: 2
```

Dictionary - max()

2. max()

Returns the largest key from the dictionary. Keys must be comparable.

Syntax : max(dictionaryname)

Example

```
D1={1:"Apple",2:"Mango"}
print("Maximum:",max(d1))
```

OUTPUT

```
Maximum: 2
```

Dictionary - min()

3. min()

Returns the smallest key from the dictionary. Keys must be comparable.

Syntax : min(dictionaryname)

Example

```
D1={1:"Apple",2:"Mango"}
print("Minimum:",min(d1))
```

OUTPUT

```
Minimum: 1
```

Dictionary - sorted()

4. **sorted()**

It returns the list in sorted order (ascending by default). The dictionary passed as parameter remains unaffected.

Syntax : sorted(<dictionary>,<reverse = True/False (default)>)

Example

```
d1={5:"May",2:"Feb",7:"Aug"}
print("Sorted values:")
print(sorted(d1))
```

OUTPUT

```
Sorted values:
[2,5,7]
```

Dictionary - sorted()

```
D1={1:"Apple",2:"Mango",4:"Banana",3:"Kiwi",5:"Pear"}
L1 = sorted(D1)
print(L1)
L2 = sorted(D1.items())
print(L2)
L3 = sorted(D1.values())
print(L3)
```

OUTPUT

```
[1, 2, 3, 4, 5]
[(1, 'Apple'), (2, 'Mango'), (3, 'Kiwi'), (4, 'Banana'), (5, 'Pear')]
['Apple', 'Banana', 'Kiwi', 'Mango', 'Pear']
```

Example - min(), max(), sorted()

```
d1 = {40:"A",30:"B",10:"C"}
print("Smallest value : ",min(d1))
print("Largest value : ",max(d1))
list1 = sorted(d1,reverse=False)
print("Type of Dictionary ", type(d1))
for i in d1:
    print(i,end = " * ")
print()
print("Type of list1", type(list1))
for i in list1:
    print(i,end = " * ")
print()
```

OUTPUT:

```
Smallest value :  10
Largest value :  40
Type of Dictionary  <class 'dict'>
40 * 30 * 10 *
Type of list1 <class 'list'>
10 * 30 * 40 *
```

Dictionary - fromkeys()

6. fromkeys()

It creates a dictionary from a collection of keys (tuple/list/string) with a default value for all (if given) or a value None assigned to all of them.

Syntax : dict.fromkeys(seq)

```
a=["rno","name","marks"]
d1=dict.fromkeys(a)
print(d1)
d2=dict.fromkeys(a,"*")
print(d2)
num = 100
d3=dict.fromkeys(a,num)
print(d3)
```

OUTPUT

```
{"rno":None,"name":None,"marks":None}
{"rno":"*","name":"*","marks":"*"}
{"rno":100,"name":100,"marks":100}
```

Dictionary - get()

7. **get()**

It returns the value for the given key. If key is not present , it returns None or Default value (if specified).

Syntax : dictionaryname.get(keyname)

Example

```
d1={5:"May",2:"Feb",7:"Aug"}
print("Value:")
print(d1.get(7))
print(d1.get(10))
print(d1.get(12,"Not found"))
```

OUTPUT

```
Value :
Aug
None
Not found
```

Dictionary - keys()

8. keys()

It returns a list of the keys of the given dictionary

Syntax : dictionaryname.keys()

Example

```
d1={5:"May",2:"Feb",7:"Aug"}
print("Keys of dictionary:")
print(d1.keys())
```

OUTPUT

```
Keys of dictionary:
dict_keys([5,2,7])
```

Dictionary - values()

8. values()

It returns a list of all the values of the given dictionary.

Syntax : dictionaryname.values()

```
d1={5:"May",2:"Feb",7:"Aug"}
print("Values:")
print(d1.values())
```

OUTPUT

```
Values :
dict_values(["May","Feb",
"Aug"])
```

Dictionary - clear()

9. clear()

It removes all the items(key and value pairs) of the given dictionary

Syntax : dictionaryname.clear()

Example

```
d1={5:"May",2:"Feb",7:"Aug"}
d1.clear()
print("After clearing")
print(d1)
```

OUTPUT

```
After clearing :
{}
```

Dictionary - pop()

10. pop()

It removes item with provided key of the given dictionary and returns the value.

Syntax : dictionaryname.pop(keyname)

Example

```
d1={5:"May",2:"Feb",7:"Aug"}
print(d1.pop(2))
print("After popping")
print(d1)
```

OUTPUT

```
"Feb"
After popping :
{5:"May",7:"Aug"}
```

Dictionary - popitem()

11. popitem()

It removes the last item from the given dictionary and returns the key:value pair as tuple. It returns a KeyError in case the dictionary is empty.

Syntax : **dictionaryname.popitem()**

Example

```
d1={5:"May",2:"Feb",7:"Aug"}
print(d1.popitem())
print("After popping ")
print(d1)
d1.clear()
d1.popitem()
```

OUTPUT

```
(7:"Aug")
After popping :
{5:"May",2:"Feb"}
KeyError
```

Dictionary - setdefault()

12. setdefault()

It returns the value of the key if it is in dictionary, and None if it is not dictionary.If key does not exist in the dictionary, default value is specified then it becomes key’s value.

Syntax : **dictionaryname.setdefault(keyname,default value)**

Dictionary - copy()

13. copy()

This method returns a copy of the dictionary. It doesn't modify the original dictionary.

Syntax : dictionaryname.copy()

```
d1={5:"May",2:"Feb",7:"Aug"}
d2=d1.copy()
print("After using copy()")
print(d2)
```

OUTPUT

```
{5:"May",2:"Feb",7:"Aug"}
After using copy()
{5:"May",2:"Feb",7:"Aug"}
```

setdefault() - example

```
D = {'EName':'Sunita','Department':'CS'}
print(D)
D_Name = D.setdefault('EName',"Name not available")
D_Department = D.setdefault('Department')
D_Gender = D.setdefault('Gender',"Not available")
D_Mobile = D.setdefault('Mobile',9910199101)
print("Employee Name : ", D_Name)
print("Employee Department : ",D_Department)
print("Gender : ",D_Gender)
print("Mobile : ",D_Mobile)
print(D)
```

OUTPUT

```
{'EName': 'Sunita', 'Department': 'CS'}
Employee Name :  Sunita
Employee Department :  CS
Gender :  Not available
Mobile  :  9910199101
{'EName': 'Sunita', 'Department': 'CS', 'Gender': 'Not
available', 'Mobile': 9910199101}
```

Dictionary - update()

14. update()

This method adds element(s) to the dictionary if the key is not in the dictionary. If the key is in the dictionary, it updates the key with the new value.The update() method takes either a dictionary or an iterable object of key/value pairs (generally tuples).

Syntax : dictionaryname.update(dictname/key:value pair)

```
d1={5:"May",2:"Feb",7:"Aug"}
print("After updating")
d1.update({5:"June"})
print(d1)
d1.update({10:"Oct"})
print(d1)
```

OUTPUT

```
print("After Updating")
{5:"June",2:"Feb",7:"Aug"}
{5:"June",2:"Feb",7:"Aug",10:
"Oct"}
```

Dictionary - del statement

del statement

This statement deletes the specified key:value pair or entire dictionary.

Syntax : del dictionaryname
del dictionaryname[keyvalue]

```
d1={5:"May",2:"Feb",7:"Aug"}
del d1[2]
print("After deleting")
print(d1)
del d1
print(d1)
```

OUTPUT

```
After deleting"
{5:"June",7:"Aug"}
NameError
```

Dictionary - Add new item/Modify existing

CODE

```
D = {'EName':'Sunita','Department':'Computer Science'}
if 'Salary' not in D:
    D['Salary']=10000
else:
    D['Salary']+=5000
print(D)
```

OUTPUT

```
{'EName': 'Sunita', 'Department': 'Computer Science'}
{'EName': 'Sunita', 'Department': 'Computer Science', 'Salary': 10000}
```

Programs on Dictionaries:

1. Write a Python program to generate a dictionary that contains numbers (x)(between 1 and n) as the keys and x*x as their corresponding value. (the dictionary should be of the form {x :x*x ,.....} where x ranges from 1 to n.

2. Write a Python Program to create a Dictionary with Key as First Character and Value as Words Starting with that Character (the words have to be input by the user) as long as the user wants.

3. Write a program to input a string and then count and store the frequency of each character of the string in a dictionary.

Programs on Dictionaries:

4. **To create a dictionary PRODUCT with the keys:**

    a) productno: Product Number
    b) prodname: name of product
    c) quantity: quantity of product
    d) price: price of the product

Then add the following keys to the dictionary:

    e)amount : multiplication of quantity and price

Now enter details of n products.

Write a menu based program:

1) To display all details of various products.
2) To search for particular product details for given productname/productno
3) To remove entire detail of one particular product
4) To add a new entry of the product in existing list of details

THANK YOU!

DEPARTMENT OF COMPUTER SCIENCE