Experiment 1

AIM

Write a program to read two integers and find the sum, diff, mult and div

Description

- num1 and num2 are the operands
- operators which we are going to use are "+" , "-" , "*" , "/" .
- addition = num1+num2 does addition
- subtraction = num1-num2 does subtraction
- multiplication = num1*num2 does multiplication
- division = num1/num2 does division

*The print() function prints the specified message to the screen, or other standard output device.

*The message can be a string, or any other object, the object will be converted into a string before written to the screen.

Syntax of print function is:

print(value1,value2)

Python Code

```
# Create Variables and Display their values
num1=10
num2=20
print(num1,num2)
```

```
     10 20
```

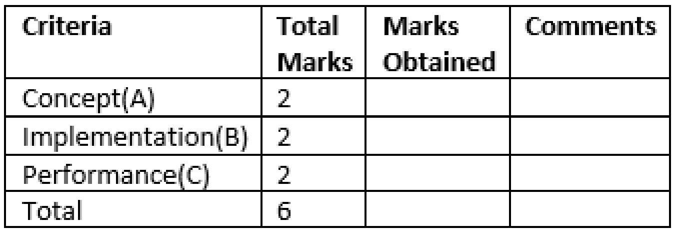
```
# Apply arithmetic operations and display results
add=num1+num2
sub=num1-num2
mul=num1*num2
div=num1/num2
print("RESULTS")
print(add,sub,mul,div)
print(num1,'+',num2, '=',add)
print(num1,'-',num2, '=',sub)
print(num1,'*',num2, '=',mul)
print(num1,'/',num2, '=',div)
```

```
RESULTS
30 -10 200 0.5
10 + 20 = 30
10 - 20 = -10
10 * 20 = 200
10 / 20 = 0.5
```

Conclusion

successfully performed and understood arithmetic operation with the help of python programming language.

Evaluation



Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

✓ 0s completed at 21:05

Experiment 2

AIM

Write a python program to count the number of Characters in a given string and hence to display the characters of string in reverse order.

Description

- **Strings** : String may be defined within single quotes ' ' or double quotes " " or triple quotes ''' ''' (multiline string)
- A string in Python is a sequence of characters. It is a derived data type. Strings are immutable. This means that once defined, they cannot be changed.
- **Counting characters of string** : The **count()** is a built-in function in Python. It will return the total count of a given element in a string.
- The counting begins from the start of the string till the end. It is also possible to specify the start and end index from where you want the search to begin.
- **Slicing** : is a feature that enables accessing parts of sequences like strings, tuples, and lists. You can also use them to modify or delete the items of mutable sequences such as lists.
- Slicing enables writing clean, concise, and readable code

Program

```
# Use input() to enter your full name
name = input(print("enter the name"))
print(name)
```

```
enter the name
Shubham Singh
Shubham Singh
```

```
# Print the string in Reverse Order
name = "Shubham Singh"[::-1]
print(name)
```

```
hgniS mahbuhS
```

```
# Using slicing, print only first name from variable name
name= "Shubham Singh"
firstname_sliced = name[0:7]
```

```
print(firstname_sliced)

# Print only last name from variable name
lastname_sliced = name[8:]
print(lastname_sliced)
```

```
Shubham
Singh
```

```
# Print the total number of characters in name
name= "Shubham singh"
print ("Number of characters: " , len(name))
```

```
Number of characters:  13
```

```
# Print the number of characters in name excluding whitespace
name="Shubham Singh"
count=len(name) - name.count(" ")
print ("Number of characters (excluding whitespace): ",count)
```

```
Number of characters (excluding whitespace):  12
```

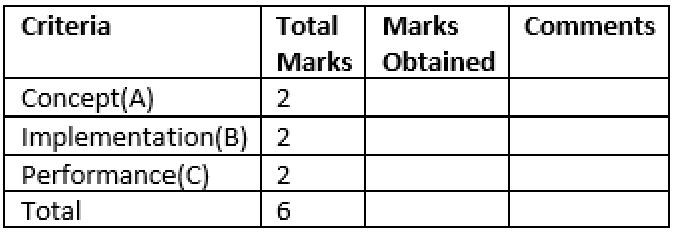
```
# Print the number of vowels (a,e,i,o,u) in name

print("Number of vowels in my name: ")
count = 0
for vowels in name:
  if vowels in "aeiouAEIOU":
    count+=1;
  else:
    continue
print(count)
```

```
Number of vowels are:
3
Number of vowels in my name:  3
Number of vowels in my name:
3
```

Conclusion

Hence i have successfully performed and understood how count the number of Characters in a given string , how to display the characters of string in reverse order and also some concepts of slicing from the performed experiments.

Evaluation



Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

✓ 0s completed at 20:13

Experiment 3

Aim

Write a program to find the simple interest for a given value P, T and R. The program must take the input from the user.

Description

$$SI = \frac{P \times R \times T}{100}$$

P = Principal Amount

R = Rate of Interest

T = Time Period

SI = Simple Interest

Total Amount = P + SI

FOR LOOP

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

example:

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

Code

```
# Accept Pricipal, Rate, Time as inputs from user

P = int(input("enter the principal"))
R = int(input("enter the rate"))
T = int(input("enter the rate"))
SI = (P*R*T)/100
print("Simple interest is " , SI)
```

```
enter the principal4000
enter the rate4
enter the rate2
Simple interest is  320.0
```

```
# Calculate simple interest
SI =
print("Simple Interest:", SI)
```

```
Simple Interest: 160.0
```

```
# Evaluate Total Amount at end of each year till the year entered by the user
# Use for loop
for year in range(1,T+1):
  interest = (P*R*(year))/100
  print("Total Amount at end of year ",year," =",P+interest)
```

```
Total Amount at end of year  1  = 4160.0
Total Amount at end of year  2  = 4320.0
```

```
# Evaluate Simple Interest for 4 different Interest rates
# Use for loop
rates = [2.5, 4, 5, 6.5]
for r in  rates:
  si = (P*r*T)/100
  print(f"SI at rate {r}% = {si}")
```

```
SI at rate 2.5% = 200.0
SI at rate 4% = 320.0
SI at rate 5% = 400.0
SI at rate 6.5% = 520.0
```

```
# Use Single line for-loop to create a list containing SI for each rate in list rates
rates = [2.5, 4, 5, 6.5]
print('For rates: ',rates)
SI = [(P*r*T)/100 for r in rates]
print('SI: ',SI)
```

```
For rates:  [2.5, 4, 5, 6.5]
SI:  [200.0, 320.0, 400.0, 520.0]
```

Conclusion

Hence we succesfully found the simple interest with the provided values by the user.

Evaluation

Criteria	**Total Marks**	**Marks Obtained**	**Comments**
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

Experiment 4

AIM

Write a Python program

- Create two new files f1 and f2
- Read and Display the contents of files f1 and f2
- Create and display the file f3 which is a combination of f1 and f2.

Description

Functions used :

- open() : this function opens a file, and returns it as file object
-
    - Syntax: open("file_name", mode)
- write() : writes a string to a file.
-
    - Syntax FileObject.write(str)
- read() : reads at most size bytes from the file. If the read hits EOF before obtaining size bytes, then it reads only available bytes.
-
    - Syntax fileObject.read()

Program

```
lst1 = ['Aug','Sep','Oct','Nov','Sep','Jul','Aug','Sep']
lst2 = [1,2,3,4, 2,4,4,1]
```

```
# Write list lst1 into file file1.txt and
with open("file1.txt", mode = "w") as f1:
  for item in lst1:
    f1.write(item + '\n')
```

```
# Write list lst2 into file file2.txt
with open('file2.txt', mode = "w") as f2:
  for item in lst2:
     f2.write(f'{item}\n')
```

```
# deleting lst1 and lst2
del lst1, lst2
```

```python
# Read contents of file1.txt and display them
print('Contents of file1.txt:')
with open("file1.txt", mode = 'r') as f1:
  print(f1.read())
```

```
Contents of file1.txt:
Aug
Sep
Oct
Nov
Sep
Jul
Aug
Sep
```

```python
# Read contents of file2.txt and display them
print('Contents of file2.txt:')
with open("file2.txt", mode = 'r') as f2:
  print(f2.read())
```

```
Contents of file2.txt:
1
2
3
4
2
4
4
1
```

```python
# Create a file3.txt, which is a combination of file1.txt and file2.txt
with open("file1.txt", mode = "r") as f1:
  with open("file2.txt", mode = "r") as f2:
    with open("file3.txt", mode  = "w") as f3:
      for l1,l2 in zip(f1,f2):
                l1 = l1.strip()
                l2 = l2.strip()
                print(f"{l2} {l1} 2020", file=f3)

# Display contents of file3.txt
with open("file3.txt", mode = "r") as f3:
  print(f3.read())
```

```
1 Aug 2020
2 Sep 2020
3 Oct 2020
4 Nov 2020
2 Sep 2020
4 Jul 2020
4 Aug 2020
```

```
    1 Sep 2020
```

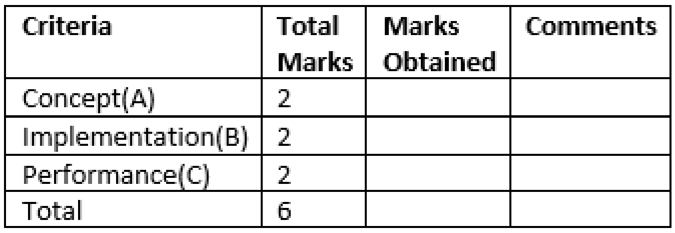
```
# Create a file4.txt, which is a combination of file1.txt and file2.txt
with open("file1.txt", mode = "r") as f1:
  with open("file2.txt", mode = "r") as f2:
    with open("file3.txt", mode = "w") as f3:
       for l1,l2 in zip(f1,f2):
          l1 = l1.strip()
          l2 = l2.strip()
          print(f"{l2} {l1} 2020", file=f3)

# Display contents of file3.txt
with open("file3.txt", mode = "r") as f3:
  print(f3.read())
```

```
    1 Aug 2020
    2 Sep 2020
    3 Oct 2020
    4 Nov 2020
    2 Sep 2020
    4 Jul 2020
    4 Aug 2020
    1 Sep 2020
```

Conclusion

We have successfully performed reading and writing operations and also got brief idea about different modes of file operations.

Evaluation



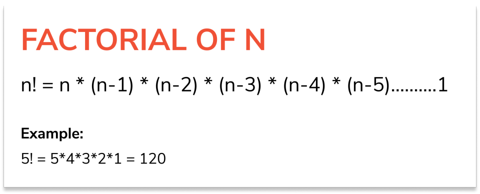
Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

Experiment 5

AIM

Write a Python program to generate first n Fibonacci number and factorial of n using functions.

Description


FACTORIAL OF N

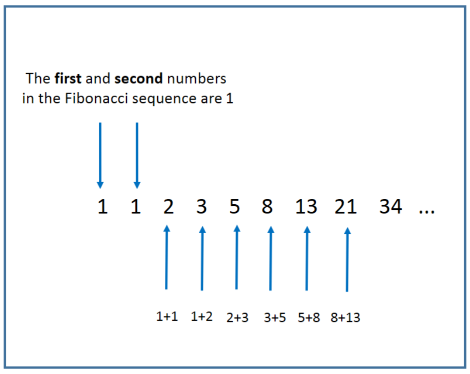
n! = n * (n-1) * (n-2) * (n-3) * (n-4) * (n-5)..........1

Example:
5! = 5*4*3*2*1 = 120

Binomial Coefficient

$$_nC_k = \frac{n!}{k!(n-k)!}$$

The Fibonacci Sequence


The **first** and **second** numbers in the Fibonacci sequence are 1
1 1 2 3 5 8 13 21 34 ...
1+1 1+2 2+3 3+5 5+8 8+13

- Functions

Function blocks begin with the keyword def followed by the function name and parentheses ()

def function_name(parameter_list):

Program

```
# Factorial of a number n>=0
# Function definition
def fact(num):
  val = 1
  while num!=0:
    val = val*num
    num = num - 1
  return val
# Ask user to input the number
num = int(input("Enter a number: "))
# Function invocation
factorial = fact(num)
print(factorial)
```

```
Enter a number: 4
24
```

```
# Determine Binomial Coefficient nCk
def bcoeff(n,k):
  nCk = fact(n)/(fact(k)*fact(n-k))
  return nCk

n, k = 5, 3
nCk = bcoeff(n,k)
print('C (',n,',',k,') = ',nCk)
```

```
C ( 5 , 3 ) =  10.0
```

```
# Function for nth Fibonacci number
def Fibo(number):
           if(number == 0):
                      return 0
           elif(number == 1):
                      return 1
           else:
                      return (Fibo(number - 2)+ Fibo(number - 1))
number = int(input("Enter the Range Number: "))
for n in range(0, number):
           print(Fibo(n))
```

```
Enter the Range Number: 10
0
1
1
2
3
5
8
13
21
34
```

Conclusion

Hence we have succesfully understood the conecpt of function definition and how to define user defined function.

Evaluation

Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

Experiment 6

AIM

Write a Python program to convert a list of characters into a string.

Description

Functions used

Python String join() method is a string method and returns a string in which the elements of the sequence have been joined by the str separator.

Syntax

string.join(iterable)

EXAMPLE

Lst1 = ['delhi',6]

Sep = "_"

x = Sep.join(Lst1)

print(x)

OUTPUT

delhi_6

Program

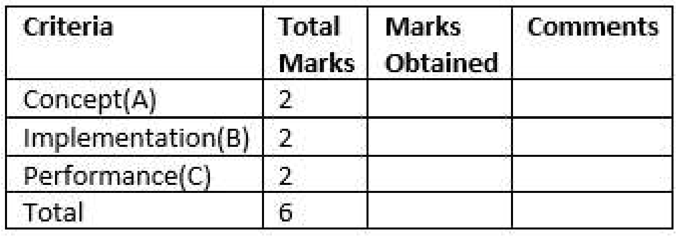
```
# Define a list of characters
lst1=['H', 'e', 'l', 'l', 'o', ' ', 'P', 'y', 't', 'h', 'o', 'n']
print(lst1,type(lst1))
# Convert the list to string
str = "".join(lst1)
print("After Join method()")
print(str,type(str))
```

```
['H', 'e', 'l', 'l', 'o', ' ', 'P', 'y', 't', 'h', 'o', 'n'] <class 'list'>
After Join method()
Hello Python <class 'str'>
```

Conclusion

Hence we have successfully understood and performed the join() method

Evaluation



Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

Experiment 7

AIM

Write a Python program to find the index of an item in a specified list.

Description

List Comprehension

List comprehensions are used for creating new lists from other iterables like tuples, strings, arrays, lists, etc. A list comprehension consists of brackets containing the expression, which is executed for each element along with the for loop to iterate over each element.

Syntax

newList = [expression(element) for element in oldList if condition]

List using index number

print("Accessing a element from the list")

print(List[0])

print(List[2])

Output:

10

14

Program

```
# Consider this string
str1 = 'Will I go to college again'
print(str1,type(str1))

# Form a list from the string
list1 = [x for x in str1]
print(list1)
print("\n")

# Find index of 'o' in the list
print("Index of 'o' in the list: ")
print(list1.index('o'))
```

```
Will I go to college again <class 'str'>
```

```
['W', 'i', 'l', 'l', ' ', 'I', ' ', 'g', 'o', ' ', 't', 'o', ' ', 'c', 'o', 'l', 'l',


Index of 'o' in the list:
8
```

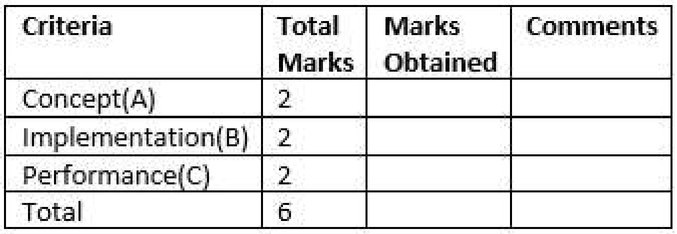
```
## Use list comprehension to form

## a list with index of all occurances of 'o' in the list
lst2 = [x for x in range(len(list1)) if list1[x]=='o']
print(f"List with indices of 'o': {lst2}")
print('\n')
```

```
List with indices of 'o': [8, 11, 14]
```

Conclusion

Hence we have successfully written a python code and found the index of an item in the list.

Evaluation



Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

✓ 0s completed at 10:58 ● ×

Experiment 8

AIM

Write an Object-Oriented Python program to

1. Create two Time objects
   - Current Time, which contains the current time, and
   - Bread Time, which contains the amount of time it takes for a bread maker to make bread
2. Then use add_time to determine when the bread will be done.
3. Write the print_time function to display the time when the bread will be done by the bread maker.

Description

OOP: Object-oriented programming has some advantages over other design patterns. Development is faster and cheaper, with better software maintainability. This, in turn, leads to higher-quality software, which is also extensible with new methods and attributes.

How to create a class

To define a class in Python, you can use the **class keyword**, followed by the class name and a colon.

Inside the class, an **init** method has to be defined with def. This is the initializer that you can later use to instantiate objects. It's similar to a constructor in Java.

init must always be present! **It takes one argument: self,** which refers to the object itself. Inside the method, the pass keyword is used as of now, because Python expects you to type something there. Remember to use correct indentation! , class definition, object instantiation, methods

Instantiating objects

To instantiate an object, type the class name, followed by two brackets. You can assign this to a variable to keep track of the object.

Program

```
'''
Define class Time containing
method  print_time()  to print time as hrs:mins:sec
```

```
  a method  correct_time()  to maintain correct time format so that mins<60 and sec<60
'''
class Time:
  def __init__(self,hrs = 00,mins = 00,sec = 00):
    self.hrs = hrs
    self.mins = mins
    self.sec = sec

  def __str__(self):
     return f"{self.hrs}:{self.mins}:{self.sec}"

  def correct_time(self):
    while True:
      if self.mins>60:
        self.mins-=60
        self.hrs+=1
      elif self.sec>60:
        self.sec -= 60
        self.mins += 1
      else:
        break
```

```
# Create object for Current Time and print the currect time
current_time= Time(4,55,54)
current_time.correct_time()
print(current_time)
```

```
     4:55:54
```

```
# Create object for Bread Time and print the bread time
bread_time = Time(7,5,45)
bread_time.correct_time()
print(bread_time)
```

```
     7:5:45
```

```
'''
Define function add_time() to add two time objects and
return a time object containing total time after correcting its format
'''
def add_time(t1: Time,t2: Time):
  hours = t1.hrs+t2.hrs
  minutes = t1.mins + t2.mins
  seconds = t1.sec + t2.sec
  combined_time = Time(hours,minutes,seconds)
  combined_time.correct_time()
  return combined_time
```

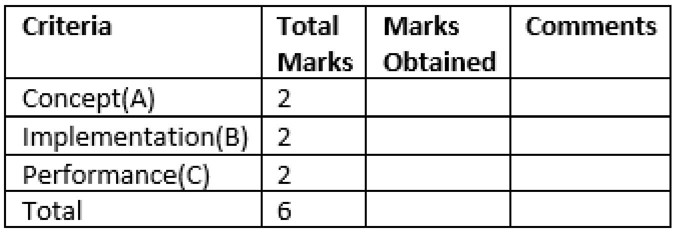
```
# Use add_time function to add current time and bread time. Print the Total Time
totaltime = add_time(current_time, bread_time)
print(f"Total Time taken : \n{total_time}")
```

```
Total Time taken :
8:31:34
```

Conclusion

Hence we have successfully understood class and objects concepts and some parts of OOP with the help of compiled program.

Evaluation



Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

Experiment 9

AIM

Write a Python program to convert a list of tuples into a dictionary

Description

zip(): The zip() function returns a zip object, which is an iterator of tuples where the first item in each passed iterator is paired together, and then the second item in each passed iterator are paired together etc.

Syntax:

```
 zip(iterator1, iterator2, iterator3 ...)
```

Program

```
details = ['Name','Age','Education','Address']
person1 = ['Ravi',14,'Xth','Delhi']
# Create list of tuples
l1 = list(zip(details,person1))
print(l1)
# Create dictionary
dictionary = dict(l1)
print(dictionary)
```

```
[('Name', 'Ravi'), ('Age', 14), ('Education', 'Xth'), ('Address', 'Delhi')]
{'Name': 'Ravi', 'Age': 14, 'Education': 'Xth', 'Address': 'Delhi'}
```

```
person2 = ['Anuj',16,'XIIth','Noida']
person3 = ['Ram',14,'Xth','Delhi']
# Create list of tuples
l2 = list(zip(person1,person2,person3))
print(l2)
l3 = list(zip(details,l2))
print(l3)
# Create dictionary
dictionary = dict(l3)
print(dictionary)
```

```
[('Ravi', 'Anuj', 'Ram'), (14, 16, 14), ('Xth', 'XIIth', 'Xth'), ('Delhi', 'No
[('Name', ('Ravi', 'Anuj', 'Ram')), ('Age', (14, 16, 14)), ('Education', ('Xth
{'Name': ('Ravi', 'Anuj', 'Ram'), 'Age': (14, 16, 14), 'Education': ('Xth', 'X
```

Conclusion

Hence, we have zipped two lists details and person1 using **zip()** function then created a dictionary using the new list l1. After that, we created a new list l2 by zipping 3 lists: person1, person2, person3. Then zipped that list with another list details. And finally, created a dictionary from that list.

Evaluation

Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

Experiment 10

AIM

Write a Python program to get the frequency of the elements in a list.

Description

Sets: Sets are used to store multiple items in a single variable. A set is a collection which is unordered, unchangeable, and unindexed. Sets are written with curly brackets.

Syntax:

```
my_set = {1, 2, 3}
```

List: A list can be defined as a collection of values or items of different types. The items in the list are separated with the comma (,) and enclosed with the square brackets [].

Syntax:

```
my_list = [1,2,3,3,4,4,5.45,'hello world']
```

Program

```
lst1 = list('Everyday is a Good day to learn Python')
print("List:",lst1)
# Determine unique elements in lst1 by converting to Set
unique_elements = set(lst1)
print(unique_elements)
# Determine Count of unique elements in lst1
count = {}
for i in lst1:
  if i in count:
    count[i]+=1
  else:
    count[i]=1

print("Items\tFrequency")
for i,j in count.items():
  print(f"{i}\t{j}")
```

```
List: ['E', 'v', 'e', 'r', 'y', 'd', 'a', 'y', ' ', 'i', 's', ' ', 'a', ' ', '
{'a', 'o', 't', 'P', 'E', 'i', 'G', 'h', 'r', 's', 'e', 'v', 'y', 'l', 'd', 'r
Items   Frequency
E       1
v       1
```

```
e        2
r        2
y        4
d        3
a        4
         7
i        1
s        1
G        1
o        4
t        2
l        1
n        2
P        1
h        1
```

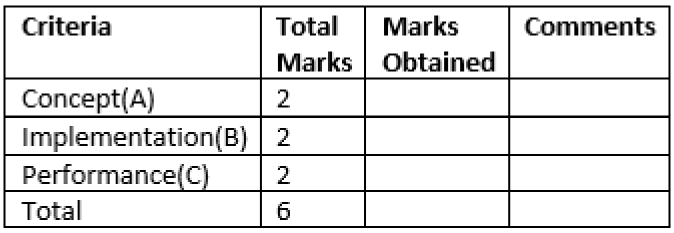
```
# Use dict comprehension to do the same
dictionary = {i:lst1.count(i) for i in unique_elements}
print(dictionary)
print('\n')

keys = []
values = []
for i,j in dictionary.items():
  keys.append(i)
  values.append(j)

import pandas as pd
df = pd.DataFrame({'Items':keys,'Values':values})
print(df)
```

```
{'a': 4, 'o': 4, 't': 2, 'P': 1, 'E': 1, 'i': 1, 'G': 1, 'h': 1, 'r': 2, 's':


   Items  Values
0      a       4
1      o       4
2      t       2
3      P       1
4      E       1
5      i       1
6      G       1
7      h       1
8      r       2
9      s       1
10     e       2
11     v       1
12     y       4
13     l       1
14     d       3
15     n       2
16             7
```

Conclusion

Hence, we have created a set of unique elements from list lst1, calculated the frequency of each element and stored them in a dictionary. In second method, we used Dictionary comprehension to create a dictionary key:value pairs of characters and their freqeuency in the list.

Evaluation



Criteria	Total Marks	Marks Obtained	Comments
Concept(A)	2		
Implementation(B)	2		
Performance(C)	2		
Total	6		

✓ 0s completed at 12:49 PM