Assignment:2

#q1 Given a string, print each word in the string as upper case, lower case and its length.

```
words = " Maharaja Sayajirao University offers the best data analysis courses in Gujarat."
print(words.upper())
print(words.lower())
print(len(words))
```

#q2 Sometimes when the user enters a string in my application, they enter special characters which might harm the internal working of my application. I want to ensure that any function that takes user input as string, any where in my application, must not allow any special character from these (=,!,:,@), unless otherwise mentioned. What is a good way to work this out?

```
# s=input("enter the string")
s="hii"
notAllowed=['@','!','=',':']
flag=True
for i in s:
    if s in notAllowed:
        flag=False
        break
if flag:
    print("string is allowed")
else:
    print("string is not allowed")
```

#q3 Consider two dictionaries given below. Create a third dictionary with keys 1 and dictionary values 2.

```
Dict1 = {'a':1, 'b':2}
Dict2 = {'c':4, 'd':5}

keys=list(Dict1.keys())
value=list(Dict2.values())

mix=dict(zip(keys,value))
#mix= dict(map(lambda k, v: (k, v), keys, value))
print(mix)
```

#q4 Use filter to eliminate all words that are shorter than 4 letters from a list of words.

```
list1=["hello","hiiiiiiiii","hey","oyyee","hi","ok","k","okayyyyy","okay"]
print(list(filter(lambda list1:len(list1)>=4,list1)))
```

#q5 Write a list comprehension statement to generate a list of all pairs of odd positive integer values less than 10 where the first value is less than the second value

```
print([(i,j) for i in range(1,10,2) for j in range(i+2,10,2)])
```

#q6 Use map and lambda function to find the maximum x coordinate in the a list of points

```
points = [(1, 2), (3, 4), (5, 6), (0, 7)]
print( max(map(lambda point: point[0], points)))
```

#q7 7. Find all the files in a project folder by scanning recursively from the project root folder. Restrict them to files with a particular suffix, (.py). Open each file and read it line by line break each line into words

```
import os
def find_and_process_files(root_folder, suffix):
    # Traverse the directory tree
    for root, dirs, files in os.walk(root_folder):
        for file in files:
            # Check if the file ends with the specified suffix
            if file.endswith(suffix):
                file_path = os.path.join(root, file)
                print(f"Processing file: {file_path}")

                # Open and read the file line by line
                with open(file_path, 'r') as f:
                    for line in f:
                        # Break each line into words
                        words = line.split()
                        print(words)  # Do something with the words (e.g., print or process)

#usage
project_root = 'C:/Users/BHADRIKA RAVAL/OneDrive/Bachelor_Of_Engineering/BE-IV/python'  #
Replace with your project root directory
file_suffix = '.py'
find_and_process_files(project_root, file_suffix)
```

# q8 Consider a list having elements [[1,3], [3,6]]. Write a function that takes such a list, and returns a list with as elements the elements of the sublists e.g. [1,3,3,6].

```
def flatten_list(l):
 return [item for sublist in l for item in sublist]
l = [[1,3] , [3,6]]
print("The list is: " , l)
print("The flattened list is: " , flatten_list(l))
```

#q9 . Write a function that returns the longest word in a variable text that contains a sentence. While text may contain puctuation, these should not be taken into account. What happens with the ties? "Hello, how was the football match earlier today??"

```
import re
def longest_word(text):
 words = re.findall(r'\w+' , text)
 longest = max(words , key = len)
 return [word for word in words if len(word) == len(longest)]
text = "Hello, how was the football match earlier today??"
```

```
print("The text is: " , text)
print("The longest word in the text is: " , longest_word(text))
```

#q13 class exercise

```
D1 ={'1': ('anant' , 50 , 70 , 90) , '2': ('isha' , 60 , 70 , 80) ,
 '3': ('aakash' , 70 , 80 , 90) , '4': ('jaya' , 35 , 65 , 45) ,
 '5': ('shweta' , 50 , 70 , 90)
 }
#calculate the total marks:
total_marks = {}
for id , marks in D1.items():
 total_marks[id] = tuple((marks[0] , sum(marks[1:])))
print("Total marks of each student: " , total_marks)

#highest of marks in students
max_marks = max(total_marks.values() , key = lambda x: x[1])
print("The student with highest marks:" , max_marks[0] , " Their score: " ,
max_marks[1])
min_first_sub = min(D1.values() , key = lambda x: x[1])
print("The student needing help in the first subject: " , min_first_sub[0])
total_third_sub = sum(marks[3] for marks in D1.values())
print("The total of marks for the third subject: " , total_third_sub)
max_second_sub = max(marks[2] for marks in D1. values())
print("Maximum marks for the second subject: " , max_second_sub)
```

Assignment:3

#data is given , create suitable structure and answer the question

```
students = [(1,"Meghna"), (2,"Dhyani"), (3,"Jaya"), (4,"Shankar"),
 (5,"Amita"), (6,"Dhrumil"), (7,"Akshara"), (8,"Ram"),
 (9,"Sanket"), (10,"Aabha"), (11,"Daxa"), (12,"Divyesh"),
 (13,"Pritam"), (14,"Jatin"), (15,"Ananya"), (16,"Akshat"),
 (17,"Vaibhav"), (18,"Tejas"), (19,"Krunal"), (20,"Krishna")
 ]
scholarship = [(1,"Meghna",2000), (2,"Dhyani",3000), (3,"Jaya",3000),
 (4,"Shankar",2500), (11,"Daxa",3500), (3,"Jaya",2000),
 (4,"Shankar",2500), (9,"Sanket",2000), (2,"Dhyani",1500),
 (2,"Dhyani",3000), (15,"Ananya",2500), (16,"Akshat",1000)
 ]

#scholarship_info = { name : (count , total) for name in students}
scholarship_info = {}
for sid , sname , amount in scholarship:
 if sname in scholarship_info:
   scholarship_info[sname] = (scholarship_info[sname][0] + 1,
scholarship_info[sname][1] + amount)
 else:
   scholarship_info[sname] = (1, amount)
print(scholarship_info)
```

#Find the students who have received scholarship only once

```
scholarship_once = [name for name , tup in scholarship_info.items() if tup[0] == 1]
print("Students who received scholarship only once: ", scholarship_once)
```

#Find the students who have received multiple scholarships

```
scholarship_multiple = [name for name , tup in scholarship_info.items() if tup[0] > 1]
print("Students who received scholarship multiple times: ", scholarship_multiple)
```

#Find the total amount of scholarship received by each student

```
total_scholarship_received = {name: total for name, (count, total) in scholarship_info.items()}
print("Total scholarship received by each student:", total_scholarship_received)
```

#Find the students who have received scholarship only once

```
no_scholarship = [name for id , name in students if name not in scholarship_info]
print("Students who have not received scholarship even once: ", no_scholarship)
```

Assignment:4

#give the output of the snippet and justify your answers

Q1:

```
➢ sum = 0
  for i in range (10, 0, -2):
      sum = sum + i
      print(i)
      if i == 4:
          continue
  print(sum)
```

sum = 0 initializes the variable sum to 0. The loop iterates over the range range(10, 0, -2) So the values of i will be: 10, 8, 6, 4, and 2. f i == 4, the continue statement is encountered.
Hence Output ould be :
10
8
6
4
2
26 # final sum

Q2:

```
➢ def func(b):
      global x
      print('Global x=', x)
      y = x + b
      x = 7
      z = x - b
      print('Local x = ',x)
      print('y = ',y)
      print('z = ',z)
  x=5
  func(10)
```

The global variable x is initialized to 5. y = x + b calculates y = 5 + 10 = 15. Here, y is a local variable inside the function. The global x is reassigned to 7.then the value of z being calculated and in the last the values are printed. The output is :
Global x= 5
Local x = 7
y = 15
z = -3

Q3:

```
➢ List=[1,6,8,4,5]
  print(List[-4:])

➢ print("My" *3 + "Blog" +'7')

➢ dict1 = {"name": "Mike", "salary": 8000}
  temp = dict1.pop("age")
  print(temp)
```

#1 here in first snippet the list is declared and it reads the character from the 4 postion from the right , and goes to the end of list. This is slicing concept.

Output: [6, 8, 4, 5]

#2 in the second snippet string operation can be done , where the multiplication make the string 3 times , and the in addition sign the strings are concataneted: the output is :
MyMyMyBlog7

#3 in third snippet the dict is declared and we trying to pop the element by the key value of age , which is not declared there , so it will give the error.

Q4: Write a Python function, with appropriate comments, to reverse each word of a sentence passed to it as a string parameter. For example,
Input :‘My Name is Jessa’
Output :‘yM emaN si asseJ’

```
def reverse_word(sentence):
        words = sentence.split()
        reversed_words = [word[::-1] for word in words]
        reversed_sentence = ' '.join(reversed_words)
        return reversed_sentence

sentence=‘My Name is Jessa’
print(reverse_word(sentence))
```

Q5: Write a Python function, with appropriate comments, that takes name and age of a person as input and displays an appropriate message whether the person is eligible to vote or not. Minimum age for voting being 18 years.

```
def is_eligible(name , age):
   if (age >= 18):
      return f"{name} is eligible to vote."
   else:
      return f"{name} is not eligible to vote."
name = input("Enter your name: ")
age = int(input("Enter your age: "))
print(f"Name: {name} \nAge: {age}")
print(is_eligible(name , age))
```

Q6: • Write a Python function with appropriate comments that takes an integer n as input to find the sum of the following series:1*3/1! + 2*5/2! + 3*7/3! + … + n*(2n+1)/n! Use appropriate assertions where needed.

```
import math
def series_sum(n):
   sum = 0
   for i in range(1 , n+1):
      sum += (i * ((2*i) + 1)) / math.factorial(i)
   return sum
number = int(input("Enter the number: "))
sum = series_sum(number)
```

```
print("The series: (1*3/1!) + (2*5/2!) + (3*7/3!) + ... + (n*(2n+1)/n!)")
print(f"Number: {number}")
print(f"The series sum: {sum}")
```

Q7:

```
➢ 'hello' * (5 - 2)
➢ 5 % 10 + 10 - 25 * 8 / 5
➢ -15 & 22
➢ 'bye' < 'Bye'
➢ 8<<3
```

Output:

```
Hellohellohello #string multiplication
-25.0             # mathes operatopr priorities
16
False
64
```

Q8:

s = set (['a', 'b', 'c', 'd', 'e']) v = set (['x', 'y', 'z'])

```
➢ print('w' in v)
➢ u=s.union(v)
  print(u)
➢ print (s.intersection(v))
➢ print (s.difference(v))
```

```
False
{'y', 'z', 'b', 'a', 'x', 'e', 'c', 'd'}
set()
{'a', 'e', 'c', 'd', 'b'}
```

Q9:

- **Write a Python function with appropriate comments, that takes a list of values as input parameter and returns a new list after removing the duplicate values in the original list. The function should also return the number of elements in the new list. For example,**

Input to the function: [1,4, 6, 1, 3, 1, 6] Values returned form the function: [1,4,6,3], 4

```
def remove_duplicates(input_list):
unique_list = list(set(input_list)
num_elements = len(unique_list)
return unique_list, num_elements
original_list = [1, 4, 6, 1, 3, 1, 6]
new_list, count = remove_duplicates(original_list)
print(f"The new list after removing duplicates: {new_list}, {count}")
```

Output:-

```
The new list after removing duplicates: [1, 3, 4, 6],
```

Assignment:5

#CSV queries to be solved by csv module and pandas library

```
import pandas as pd
import csv
import json
# Load the CSV file into a pandas DataFrame
df = pd.read_csv('demo.csv')
```

#1. Longest Distance Journey in miles

```
longest_distance = df['Miles Travelled'].max()
print(f"Longest Distance Journey: {longest_distance} miles \n")
```

#2. Longest return journey in miles

```
longest_return_journey = df[df['Ticket Single or Return'] == 'Return']['Miles Travelled'].max()
print(f"Longest Return Journey: {longest_return_journey} miles \n")
```

#3. Longest single journey in miles

```
longest_single_journey = df[df['Ticket Single or Return'] == 'Single']['Miles Travelled'].max()
print(f"Longest Single Journey: {longest_single_journey} miles \n")
```

#4. Carrier whom we paid the most

```
most_paid_carrier = df.groupby('Air Carrier')['Total Cost ex VAT'].sum().idxmax()
print(f"Carrier we paid the most: {most_paid_carrier} \n")
```

#5. Carrier whom we flew the most number of times

```
most_frequent_carrier = df['Air Carrier'].value_counts().idxmax()
print(f"Carrier we flew the most number of times: {most_frequent_carrier} \n")
```

#6. List of all distinct carriers

```
distinct_carriers = df['Air Carrier'].unique()
print(f"Distinct Carriers: {distinct_carriers} \n")
```

#7. List of all distinct destinations

```
distinct_destinations = df['Journey Finish Point'].unique()
print(f"Distinct Destinations: {distinct_destinations} \n")
```

#8. Mean price of all single route ticket prices

```
mean_single_route_price = df[df['Ticket Single or Return'] == 'Single']['Ticket Price ex VAT'].mean()
print(f"Mean price of all single route tickets: {mean_single_route_price} \n")
```

#9. Create a new CSV file for destination Amsterdam

```
df_amsterdam = df[df['Journey Finish Point'] == 'Amsterdam']
df_amsterdam.to_csv('amsterdam_trips.csv', index=False)
print("CSV file for Amsterdam trips has been created. \n")
```

#Create a JSON file from the CSV

```
df.to_json('demo.json', orient='records')
#Read JSON file and display first 5 rows
with open('demo.json', 'r') as file:
    data = json.load(file)
print("First 5 rows from the JSON file:" )
print(data[:5])
print()
#Read and display all the distinct destinations from JSON
distinct_destinations_from_json = set([row['Journey Finish Point'] for row in data])
print(f"Distinct destinations from JSON: {distinct_destinations_from_json} \n")
#Write one new row to the JSON file (Example row)
```

```
new_row = {
"Customer": "New Customer",
"Ticket Price ex VAT": 250,
"Number of Travellers": 1,
"Total Cost ex VAT": 250,
"Ticket Type": "Economy",
"Ticket Single or Return": "Single",
"Travel Class": "Business",
"Travel Date": "2024-09-11",
"Miles Travelled": 1500,
"Journey Start Point": "Paris",
"Journey Finish Point": "Berlin",
"Air Carrier": "Lufthansa"
}
data.append(new_row)
#Save the updated JSON file
with open('demo.json', 'w') as file:
   json.dump(data, file, indent=4)
print("New row added to the JSON file. \n")
```

#Total number of miles travelled

```
total_miles = df['Miles Travelled'].sum()
print(f"Total number of miles travelled: {total_miles} \n")
```