**DAY8 –**

Assignment 52:

Consider sample data as follows: sample\_data=range(1,11)

Create two functions: **odd() and even()**  
The function even() returns a list of all the even numbers from sample\_data  
The function odd() returns a list of all the odd numbers from sample\_data

Create a function sum\_of\_numbers() which will accept the sample data and/or a function.  
If a function is not passed, the sum\_of\_numbers() should return the sum of all the numbers from sample\_data  
If a function is passed, the sum\_of\_numbers() should return the sum of numbers returned from the function passed.

#PF-Assgn-52

#This verification is based on string match.

def sum\_of\_numbers(list\_of\_num,filter\_func=None):

#Remove pass and write the logic here

if filter\_func =="odd":

list\_of\_num = odd(list\_of\_num)

elif filter\_func == "even":

list\_of\_num = even(list\_of\_num)

else:

list\_of\_num = list\_of\_num

return sum(list\_of\_num)

def even(data):

#Remove pass and write the logic here

lst1=[]

for i in data:

if i%2==0:

lst1+=[i]

return lst1

def odd(data):

#Remove pass and write the logic here

lst1=[]

for i in data:

if i%2==0:

lst1+=[i]

return lst1

sample\_data = range(1,11)

print(sum\_of\_numbers(sample\_data,”even”))

Assignment 53:

Assume that a poem is given. Write the regular expressions for the following:

1. Print how many times the letter 'v' appears in the poem.
2. Remove all the newlines from the poem and print the poem in a single line.
3. If a word has 'ch' or 'co', replace it with 'Ch' or 'Co'.
4. If the pattern has characters 'ai' or 'hi', replace the next three characters with \*\\*.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| If I can stop one heart from breaking, I shall not live in vain; If I can ease one life the aching, Or cool one pain, Or help one fainting robin Unto his nest again, I shall not live in vain. | 4  If I can stop one heart from breaking, I shall not live in vain; If I can ease one life the aching, Or cool one pain, Or help one fainting robin Unto his nest again, I shall not live in vain.  If I can stop one heart from breaking, I shall not live in vain; If I can ease one life the aChing, Or Cool one pain, Or help one fainting robin Unto his nest again, I shall not live in vain.  If I can stop one heart from breaking, I shall not live in vain; If I can ease one life the achi\*\\* Or cool one pain, Or help one fai\*\\*ng robin Unto hi\*\\*est again, I shall not live in vain. |

Test your code by using the given sample inputs.  
Verify your code by using the **2nd** sample input(highlighted) given below:

#PF-Assgn-53

#This verification is based on string match.

import re

poem='''

It takes strength for being certain,

It takes courage to have doubt.

It takes strength for challenging alone,

It takes courage to lean on another.

It takes strength for loving other souls,

It takes courage to be loved.

It takes strength for hiding our own pain,

It takes courage to help if it is paining for someone.

'''

#Note: Triple quotes can be used to enclose Strings which has lines of text.

#Write your logic here for question 1

print(len(re.findall('v',poem)))

res=[]

for i in poem:

res.append(re.sub("\n","",i))

print("".join(res))

poem= re.sub('ch','Ch',poem)

poem= re.sub('co','Co',poem)

print(poem)

print(re.sub(r'(ai|hi)(...)',r'\1\*\\*',poem))

Assignment 54:

Write a python function, **check\_anagram()** which accepts two strings and returns True, if one string is an anagram of another string. Otherwise returns False.  
The two strings are considered to be an anagram if they contain repeating characters but none of the characters repeat at the same position. The length of the strings should be the same.  
Also write the pytest test cases to test the program.  
**Note:**Perform case insensitive comparison wherever applicable.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| eat, tea | True |
| backward,drawback | False (Reason: character 'a' repeats at position 6, not an anagram) |
| Reductions,discounter | True |
| About, table | False |

#PF-Assgn-54

def check\_anagram(data1,data2):

#start writing your code here

res=True

a=set(data1)

for i in a:

if data2.count(i)>=2:

res= False

break

if i in data2:

res=True

else:

res=False

break

return res

print(check\_anagram("About","table"))

Assignment 55:

Write a python program to help an airport manager to generate few statistics based on the ticket details available for a day.

Go through the below program and complete it based on the comments mentioned in it.

**Note:**Perform case sensitive string comparisons wherever necessary

#PF-Assgn-55

#Sample ticket list - ticket format: "flight\_no:source:destination:ticket\_no"

#Note: flight\_no has the following format - "airline\_name followed by three digit number

#Global variable

ticket\_list=['AI101:MUM:LON:001', 'AI101:MUM:LON:002', 'SI456:MUM:SIN:145', 'EM456:MUM:DUB:098', 'SI456:MUM:SIN:050', 'SI456:MUM:SIN:051']

def find\_passengers\_flight(airline\_name="AI"):

#This function finds and returns the number of passengers travelling in the specified airline.

count=0

for i in ticket\_list:

string\_list=i.split(":")

if(string\_list[0].startswith(airline\_name)):

count+=1

return count

def find\_passengers\_destination(destination):

#Write the logic to find and return the number of passengers traveling to the specified destination

#Remove pass and write your logic here

count=0

for i in ticket\_list:

string\_list=i.split(":")

if string\_list[2]==destination:

count+=1

return count

def find\_passengers\_per\_flight():

'''Write the logic to find and return a list having number of passengers traveling per flight based on the details in the ticket\_list

In the list, details should be provided in the format:

[flight\_no:no\_of\_passengers, flight\_no:no\_of\_passengers, etc.].'''

#Remove pass and write your logic here

flight\_number=[]

for i in ticket\_list:

flight\_number.append(i.split(":")[0])

num\_set=set(flight\_number)

dictionary = dict.fromkeys(num\_set,0)

for i in ticket\_list:

string\_list=i.split(":")

if dictionary.get(string\_list[0]) != None:

val = dictionary.get(string\_list[0])+int( string\_list[3])

dictionary[string\_list[0]]=val

return dictionary

def sort\_passenger\_list():

#Write the logic to sort the list returned from find\_passengers\_per\_flight() function in the descending order of number of passengers

#Remove pass and write your logic here

d=find\_passengers\_per\_flight()

l=list(d.items())

l.sort(reverse=True)

dict1=dict(l)

return dict1

#Provide different values for airline\_name and destination and test your program.

print(find\_passengers\_flight("AI"))

print(find\_passengers\_destination("LON"))

print(sort\_passenger\_list())

Assignment 56:

Write a python program that accepts a text and displays a string which contains the word with the largest frequency in the text and the frequency itself separated by a space.  
  
**Rules:**

1. The word should have the largest frequency.
2. In case multiple words have the same frequency, then choose the word that has the maximum length.

**Assumptions:**

1. The text has no special characters other than space.
2. The text would begin with a word and there will be only a single space between the words.

Perform case insensitive string comparisons wherever necessary.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| "Work like you do not need money love like you have never been hurt and dance like no one is watching" | like 3 |
| "Courage is not the absence of fear but rather the judgement that something else is more important than fear" | fear 2 |

#PF-Assgn-56

def max\_frequency\_word\_counter(data):

word=""

frequency=0

w1= data.split(" ")

for i in w1:

curr\_frequency = w1.count(i)

if(curr\_frequency> frequency):

frequency = curr\_frequency

word = i

if(curr\_frequency == frequency):

if len(i) > len(word):

word=i

if frequency ==1:

word=""

frequency=0

print(word,frequency)

#start writing your code here

#Populate the variables: word and frequency

# Use the below given print statements to display the output

# Also, do not modify them for verification to work

#print(word,frequency)

#Provide different values for data and test your program.

data="Listen to the big clock Tick tock tick"

max\_frequency\_word\_counter(data)