TASK FIVE FILE HANDLING AND EXCEPTION HANDLING

1. Write a program in Python to allow the error of syntax to be handled using exception handling.

HINT: Use SyntaxError

while True print('Hello world')

  File "<stdin>", line 1

    while True print('Hello world')

                   ^

2. Write a program in Python to allow the user to open a file by using the argv module. If the entered name is incorrect throw an exception and ask them to enter the name again. Make sure to use read only mode.

3. Write a program to handle an error if the user entered a number more than four digits it should return “The length is too short/long !!! Please provide only four digits”

def raw\_input():

    print('Please enter your 10 digit number')

    a = raw\_input("Please enter the Number: ")

    if len(a) > 4:

        print ("length is too short/long !!! Please provide only four digits")

        a = int(a)

    aa = (a\*10)

4. Create a login page backend to ask users to enter the username and password. Make sure to ask for a Re-Type Password and if the password is incorrect give chance to enter it again but it should not be more than 3 times.

print('Enter correct username and password to continue')

count=0

while count < 3:

    username = input('Enter username: ')

    password = input('Enter password: ')

    if password=='Admin123' and username=='admin':

        print('Access granted')

        break

    else:

        print('Access denied. Try again.')

        count += 1

5. Go through the link provided below to understand finally and raise concept: https://www.programiz.com/python-programming/exception-handling

6. Read doc.txt file using Python File handling concept and return only the even length string from the file. Consider the content of doc.txt as given below: Hello I am a file Where you need to return the data string Which is of even length Make sure you return the content in The same link as it is present.

L = ["Hello I am a file Where you need to return the data string Which is of even length Make sure you return the content in The same link as it is present."]

file1 = open('doc.txt', 'w')

file1.writelines(L)

file1.close()

file1 = open('doc.txt', 'r')

Lines = file1.readlines()

words = list(Lines.split(' '))

print("Value: ", Lines)

for word in Lines:

    if len(word)%2==0:

        print(word)

TASK SIX GENERATORS, LIST COMPREHENSION AND DECORATORS

1. Write a program in Python to find out the character in a string which is uppercase using list comprehension.

c = ['India','United States', 'Mexico','Canada','China']

converted\_list = [x.upper() for x in c]

print(converted\_list)

2. Write a program to construct a dictionary from the two lists containing the names of students and their corresponding subjects. The dictionary should map the students with their respective subjects. Let’s see how to do this using for loops and dictionary comprehension. HINT - Use Zip function also Sample input: students = ['Smit', 'Jaya', 'Rayyan'] subjects = ['CSE', 'Networking', 'Operating System'] Expected output: {‘Smit’ : ’CSE’ , ’Jaya’ : ’Networking’ , ’Rayyan’ : ’Operating System’}

students = ['Smit', 'Jaya', 'Rayyan']

subjects = ['CSE', 'Networking', 'Operating System']

dictionary = dict(zip(students, subjects))

print(dictionary)

3. Learn More about Yield, next and Generators

It is fairly simple to create a generator in Python. It is as easy as defining a normal function, but with a yield statement instead of a return statement. If a function contains at least one yield statement (it may contain other yield or return statements), it becomes a generator function. Both yield and return will return some value from a function. The difference is that while a return statement terminates a function entirely, yield statement pauses the function saving all its states and later continues from there on successive calls.

4. Write a program in Python using generators to reverse the string. Input String = “Consultadd Training”

def reverse\_string(str):

    str1 = ""

    for i in str:

        str1 = i + str1

    return str1

str = "Consultadd Training"

print("The original string is: ",str)

print("The reverse string is: ",reverse\_string(str))

5. Write an example on decorators.

Decorator is used to add functionality to an existing code.

# example of decorator

def sampleDecorator(func):

    def addingFunction():

        # some new statments or flow control

        print("This is the added text to the actual function.")

        # calling the function

        func()

    return addingFunction

@sampleDecorator

def actualFunction():

    print("This is the actual function.")

actualFunction()