**Exercise 5**

#1. Write a Python program to read an entire text file.

k  = open("C:\Cdac\Python\excercise\_4.py","r")

print(k.read())

k.close()

#2.  Write a program that counts lines and characters in a file. With your favorite text editor, code

a Python module called mymod.py, which exports three top-level names:

a) A countLines(name) function that reads an input file and counts the number of lines

in it

b) A countChars(name) function that reads an input file and counts the number of

characters in it

c) A test(name) function that calls both counting functions with a given input filename.

All three mymod functions should expect a filename string to be passed in.

Now, test your module interactively, using import and name qualification to fetch your

exports.

file\_path = r"C:\Cdac\Python\excercise\_5.py"

def countlines(file\_path):

    k = open(file\_path,'r')

    m = k.readlines()

    k.close()

    lines = len(m)

    print(lines)

def countchar(file\_path):

    c = open(file\_path,'r')

    d = len(c.read())

    char = d

    c.close()

    count\_char = print(char)

def count\_both(file\_path):

    count\_lines = countlines(file\_path)

    count\_char = countchar(file\_path)

    return count\_lines, count\_char

#import function

import mymod

file\_path = input("enter file")

result1 = mymod.countchar(file\_path)

result2 = mymod.countlines(file\_path)

result3 = mymod.count\_both(file\_path)

#3. Test your mymod module from Exercise 2 interactively, by using from to load the exports

directly, first by name, then using the from\* variant to fetch everything.

#mymod.py functions

def countlines(file\_path):

    k = open(file\_path,'r')

    m = k.readlines()

    k.close()

    lines = len(m)

    print(lines)

def countchar(file\_path):

    c = open(file\_path,'r')

    d = len(c.read())

    char = d

    c.close()

    count\_char = print(char)

def count\_both(file\_path):

    count\_lines = countlines(file\_path)

    count\_char = countchar(file\_path)

    return count\_lines, count\_char

#importing by all mmethod

#first method

import mymod

file\_path = input("enter file")

result1 = mymod.countchar(file\_path)

#from second method

from mymod import countlines

result2 = mymod.countlines(file\_path)

#from method 3

from mymod import \*

result3 = count\_both(file\_path)

#4.    Now, add a line in your mymod module that calls the test function automatically only when the module is run as a script, not when it is imported The line you add will probably test the value of \_\_name\_\_ for the string &quot;\_\_main\_\_&quot;,. Import the module and test its functions interactively.

def countlines(file\_path):

    k = open(file\_path,'r')

    m = k.readlines()

    k.close()

    lines = len(m)

    print(lines)

def countchar(file\_path):

    c = open(file\_path,'r')

    d = len(c.read())

    char = d

    c.close()

    count\_char = print(char)

def count\_both(file\_path):

    count\_lines = countlines(file\_path)

    count\_char = countchar(file\_path)

    return count\_lines, count\_char

if \_\_name\_\_ == "\_\_main\_\_":

    file\_path = input("enter file")

    count\_both(file\_path)

#5.    Write a second module, myclient.py, which imports mymod and tests its functions;

run myclient from the system command line. If myclient uses from to fetch from mymod,

will mymod’s functions be accessible from the top level of myclient? What if it imports

with import instead? Try coding both variations in myclient and test interactively, by

importing myclient .

import mymodule

file\_path = input("enter file")

mymodule.count\_both(file\_path)

import mymodule

from mymodule import countchar

file\_path = input("enter file")

mymodule.countchar(file\_path)

#6.      Package imports. Finally, import your file from a package. Create a subdirectory

called mypkg nested in a directory on your module import search path, move

the mymod.py module file you created in exercises 2 or 4 into the new directory, and try to

import it with a package import of the form: import mypkg.mymod.

import mypkg.mymod

file\_path = input("enter file")

mypkg.mymod.count\_both(file\_path)

#7.  Write a Python program to read first n lines of a file.

with open("name.txt") as f:

    k = f.readlines()

n = int(input("desired number of lines"))

for i in range(len(k)):

    if i < n:

        print(k[i])

#8. Write a Python program to append text to a file and display the text.

with open("name.txt", 'a') as f:

    f.write("\nI am adding new line here")

k = open("name.txt")

print(k.read())

#9. Write a Python program to read a file line by line and store it into a list.

with open("name.txt") as f:

    k = f.readlines()

print(k)

#10. Write a program to print each line of a file in reverse order.

with open("name.txt") as f:

    k = f.readlines()

print(k[::-1])

#11. Write a Python program to write a list content to a file.

n = ['my', 'name', 'is', 'Navanish']

s = str(n)

with open("name.txt",'w') as f:

    f.write(s)

p = open("name.txt")

q = p.read()

print(q)

#12. Write a program to compute the number of characters, words and lines in a file.

def number\_lines(file\_path):

    k = open(file\_path,'r')

    m = k.readlines()

    k.close()

    lines = len(m)

    print(lines)

def number\_char(file\_path):

    c = open(file\_path,'r')

    d = len(c.read())

    char = d

    c.close()

    count\_char = print(char)

def number\_words(file\_path):

    c = open(file\_path)

    d = c.read()

    words = len(d.split())

    c.close()

    count\_words = print(words)

number\_words("name.txt")

#13. Write a program to accept string/sentences from the user till the user enters “END” to. Save the data in a text file and then display only those sentences which begin with an uppercase alphabet.

def txt\_file():

    while True:

        text = input("enter your text here")

        if text !="END":

            with open("name.txt",'a') as f:

                f.write(text + "\n")

        else:

            break

    with open("name.txt") as f:

        k = f.read()

        q = k.split()

    found = False

    for i in q:

        if i[0].isupper():

            print(i)

            found = True

    if not found:

            print("No uppercase letter found")

txt\_file()

#14. Write a program to enter the following records in a binary file:

import pickle

print("working woth binary file")

k= open("name.txt","ab")

n = int(input("Enter the number of items"))

for i in range(n):

    item\_no = int(input("\tEnter item no"))

    item\_name = input("\tEnter item name")

    qty = int(input("\tenter the quantity"))

    price = float(input("\tenter the price"))

    amount = qty\*price

    items = [item\_no, item\_name, qty, price, amount]

    print()

    pickle.dump(items, k)

k.close()

k = open("name.txt", "rb")

while True:

    try:

        items = pickle.load(k)

        print(f"\tItem No: {items[0]}")

        print(f"\tItem Name: {items[1]}")

        print(f"\tQuantity: {items[2]}")

        print(f"\tPrice: {items[3]}")

        print(f"\tAmount: {items[4]}")

    except EOFError:

        break

k.close()