lab 7

September 18, 2023

[ ]:
```
#opens the file file.txt in read mode
fileptr = open("file name","r")

if fileptr:
    print("file is opened successfully")
```

[ ]:
```
# opens the file file.txt in read mode
fileptr = open("file name","r")

if fileptr:
    print("file is opened successfully")

#closes the opened file
fileptr.close()
```

[ ]:
```
with open("file name",'r') as f:
    content = f.read();
    print(content)
```

[ ]:
```
# open the file.txt in append mode. Create a new file if no such file exists.
fileptr = open("file name", "w")

# appending the content to the file
fileptr.write('''''Python is the modern day language. It makes things so simple.
 ↪
It is the fastest-growing programing language''')

# closing the opened the file
fileptr.close()
```

[ ]:
```
#open the file.txt in write mode.
fileptr = open("file name","a")

#overwriting the content of the file
fileptr.write(" Python has an easy syntax and user-friendly interaction.")

#closing the opened file
fileptr.close()
```

[ ]:
```
#open the file.txt in read mode. causes error if no such file exists.
fileptr = open("file name","r")
#stores all the data of the file into the variable content
content = fileptr.read(10)
# prints the type of the data stored in the file
print(type(content))
#prints the content of the file
print(content)
#closes the opened file
fileptr.close()
```

[ ]:
```
#open the file.txt in read mode. causes an error if no such file exists.
fileptr = open("file name","r");
#running a for loop
for i in fileptr:
    print(i) # i contains each line of the file
```

[ ]:
```
#open the file.txt in read mode. causes error if no such file exists.
fileptr = open("file name","r");
#stores all the data of the file into the variable content
content = fileptr.readline()
content1 = fileptr.readline()
#prints the content of the file
print(content)
print(content1)
#closes the opened file
fileptr.close()
```

[ ]:
```
#open the file.txt in read mode. causes error if no such file exists.
fileptr = open("file name","r");

#stores all the data of the file into the variable content
content = fileptr.readlines()

#prints the content of the file
print(content)

#closes the opened file
fileptr.close()
```

[ ]:
```
#open the file.txt in read mode. causes error if no such file exists.
fileptr = open("file name","x")
print(fileptr)
if fileptr:
    print("File created successfully")
```

[ ]:
```
# open the file file2.txt in read mode
fileptr = open("file name","r")

#initially the filepointer is at 0
print("The filepointer is at byte :",fileptr.tell())

#reading the content of the file
content = fileptr.read();

#after the read operation file pointer modifies. tell() returns the location of␣
 ↪the fileptr.

print("After reading, the filepointer is at:",fileptr.tell())
```

[ ]:
```
# open the file file2.txt in read mode
fileptr = open("file name","r")

#initially the filepointer is at 0
print("The filepointer is at byte :",fileptr.tell())

#changing the file pointer location to 10.
fileptr.seek(10);

#tell() returns the location of the fileptr.
print("After reading, the filepointer is at:",fileptr.tell())
```

[ ]:
```
import os

#rename file2.txt to file3.txt
os.rename("file name","New file name")
```

[ ]:
```
import os;
#deleting the file named file
os.remove("file name")
```

[ ]:
```
import os

#creating a new directory with the name new
os.mkdir("new")
os.getcwd()
```

[ ]:
```
import os
# Changing current directory with the new directory
os.chdir("new directory path")
#It will display the current working directory
os.getcwd()
os.rmdir("directory_name")
```

[ ]:
```
first file:

temperatures=[10,-20,-289,100]
def c_to_f(c):
    if c< -273.15:
        return "That temperature doesn't make sense!"
    else:
        f=c*9/5+32
        return f
for t in temperatures:
    print(c_to_f(t))

Second file:

import subprocess

with open("output.txt", "wb") as f:
    subprocess.check_call(["python", "file.py"], stdout=f)
```

[ ]:
```
Number = 204
def AddNumber():  # here, we are defining a function with the name Add Number
    # Here, we are accessing the global namespace
    global Number
    Number = Number + 200
print("The number is:", Number)
# here, we are printing the number after performing the addition
AddNumber()    # here, we are calling the function
print("The number is:", Number)
```

[ ]:
```
# Here, we are importing the sys module
import sys
# Here, we are printing the path using sys.path
print("Path of the sys module in the system is:", sys.path)
```

[ ]:
```
# Here, we are importing the complete math module using *
from math import *
# Here, we are accessing functions of math module without using the dot␣
 ↪operator
print( "Calculating square root: ", sqrt(25) )
# here, we are getting the sqrt method and finding the square root of 25
print( "Calculating tangent of an angle: ", tan(pi/6) )
# here pi is also imported from the math module
```

[ ]:
```
# Here, we are creating a simple Python program to show how to import multiple ␣
 ↪
# objects from a module
from math import e, tau
```

```
print( "The value of tau constant is: ", tau )
print( "The value of the euler's number is: ", e )
```

[ ]:
```
# Here, we are creating a simple Python program to show how to import specific
# objects from a module
# Here, we are import euler's number from the math module using the from
keyword
from math import e
# here, the e value represents the euler's number
print( "The value of euler's number is", e )
```

[ ]:
```
# Here, we are creating a simple Python program to show how to import a module
and rename it
# Here, we are import the math module and give a different name to it
import math as mt      # here, we are importing the math module as mt
print( "The value of euler's number is", mt.e )
# here, we are printing the euler's number from the math module
```

[ ]:
```
# Here, we are creating a simple Python program to show how to import a
standard module
# Here, we are import the math module which is a standard module
import math
print( "The value of euler's number is", math.e )
# here, we are printing the euler's number from the math module
```

[ ]:
```
file 1:

# Here, we are creating a simple Python program to show how to create a module.
# defining a function in the module to reuse it
def square( number ):
   # here, the above function will square the number passed as the input
    result = number ** 2
    return result      # here, we are returning the result of the function

file 2:
import example_module
# here, we are calling the module square method and passing the value 4
result = example_module.square(  4  )
print("By using the module square of number is: ", result )
```

[ ]:
```
# Python code to catch an exception and handle it using try and except code
blocks

a = ["Python", "Exceptions", "try and except"]
try:
```

```
    #looping through the elements of the array a, choosing a range that goes␣
 ↪beyond the length of the array
     for i in range( 4 ):
        print( "The index and element from the array is", i, a[i] )
#if an error occurs in the try block, then except block will be executed by the␣
 ↪Python interpreter
except:
    print ("Index out of range")
```

[ ]:
```
#Python code to show how to raise an exception in Python
num = [3, 4, 5, 7]
if len(num) > 3:
    raise Exception( f"Length of the given list must be less than or equal to 3␣
 ↪but is {len(num)}" )
```

[ ]:
```
#Python program to show how to use assert keyword
# defining a function
def square_root( Number ):
    assert ( Number < 0), "Give a positive integer"
    return Number**(1/2)

#Calling function and passing the values
print( square_root( 36 ) )
print( square_root( -36 ) )
```

[ ]:
```
# Python program to show how to use else clause with try and except clauses

# Defining a function which returns reciprocal of a number
def reciprocal( num1 ):
    try:
        reci = 1 / num1
    except ZeroDivisionError:
        print( "We cannot divide by zero" )
    else:
        print ( reci )
# Calling the function and passing values
reciprocal( 4 )
reciprocal( 0 )
```

[ ]:
```
# Python code to show the use of finally clause

# Raising an exception in try block
try:
    div = 4 // 0
    print( div )
# this block will handle the exception raised
except ZeroDivisionError:
```

```
    print( "Atepting to divide by zero" )
# this will always be executed no matter exception is raised or not
finally:
    print( 'This is code of finally clause' )
```

[ ]:
```
class EmptyError( RuntimeError ):
   def __init__(self, argument):
      self.arguments = argument
Once the preceding class has been created, the following is how to raise an␣
 ↪exception:
Code
var = " "
try:
    raise EmptyError( "The variable is empty" )
except (EmptyError, var):
    print( var.arguments )
```

[ ]:
```
# importing csv module
import csv

# csv file name
filename = "aapl.csv"

# initializing the titles and rows list
fields = []
rows = []

# reading csv file
with open(filename, 'r') as csvfile:
    # creating a csv reader object
    csvreader = csv.reader(csvfile)

    # extracting field names through first row
    fields = next(csvreader)

    # extracting each data row one by one
    for row in csvreader:
        rows.append(row)

    # get total number of rows
    print("Total no. of rows: %d"%(csvreader.line_num))

# printing the field names
print('Field names are:' + ', '.join(field for field in fields))

# printing first 5 rows
print('\nFirst 5 rows are:\n')
```

```
for row in rows[:5]:
    # parsing each column of a row
    for col in row:
        print("%10s"%col,end=" "),
    print('\n')
```

[ ]:
```
# importing the csv module
import csv

# field names
fields = ['Name', 'Branch', 'Year', 'CGPA']

# data rows of csv file
rows = [ ['Nikhil', 'COE', '2', '9.0'],
         ['Sanchit', 'COE', '2', '9.1'],
         ['Aditya', 'IT', '2', '9.3'],
         ['Sagar', 'SE', '1', '9.5'],
         ['Prateek', 'MCE', '3', '7.8'],
         ['Sahil', 'EP', '2', '9.1']]

# name of csv file
filename = "university_records.csv"

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(fields)

    # writing the data rows
    csvwriter.writerows(rows)
```

[ ]:
```
# importing the csv module
import csv

# my data rows as dictionary objects
mydict =[{'branch': 'COE', 'cgpa': '9.0',
          'name': 'Nikhil', 'year': '2'},
         {'branch': 'COE', 'cgpa': '9.1',
          'name': 'Sanchit', 'year': '2'},
         {'branch': 'IT', 'cgpa': '9.3',
          'name': 'Aditya', 'year': '2'},
         {'branch': 'SE', 'cgpa': '9.5',
          'name': 'Sagar', 'year': '1'},
         {'branch': 'MCE', 'cgpa': '7.8',
```

```
         'name': 'Prateek', 'year': '3'},
        {'branch': 'EP', 'cgpa': '9.1',
         'name': 'Sahil', 'year': '2'}]

# field names
fields = ['name', 'branch', 'year', 'cgpa']

# name of csv file
filename = "university_records.csv"

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv dict writer object
    writer = csv.DictWriter(csvfile, fieldnames = fields)

    # writing headers (field names)
    writer.writeheader()

    # writing data rows
    writer.writerows(mydict)
```

[ ]:
```
# importing the csv module
import csv

# field names
fields = ['Name', 'Email']

# data rows of csv file
rows = [ ['Nikhil', 'nikhil.gfg@gmail.com'],
         ['Sanchit', 'sanchit.gfg@gmail.com'],
         ['Aditya', 'aditya.gfg@gmail.com'],
         ['Sagar', 'sagar.gfg@gmail.com'],
         ['Prateek', 'prateek.gfg@gmail.com'],
         ['Sahil', 'sahil.gfg@gmail.com']]

# name of csv file
filename = "email_records.csv"

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(fields)

    # writing the data rows
```

```
    csvwriter.writerows(rows)
```

[ ]: