```
file obj = open ("abc.txt","r")
file obj.write ("computer science subject" + "\n")
file obj.write ("DSA in Python\n DS\n")
file obj.close()
file obj = open ("abc.txt","r")
str1 = file obj.read()
print ("The output of read method :",str1)
file obj.close()
>>> The output of read method : computer science
                                subject
                                DBMS
                                Python
                                DS
# readlines
file obj = open ("abc.txt","r")
str2 = file obj.readlines()
print ("The output of readline method :",str2)
file obj.close()
>>> The output of readline method : ['computer science subject\n', 'DBMS\n', 'Python\n', 'DS\n']
# readline
file obj = open ("abc.txt","r")
str3 = file obj.readline()
print ("The output of readline method",str3)
file obj.close()
>>> The output of readline method : computer science
                                    subject
                                    DBMS
                                    Python
                                    DS
```


PRACTICAL NO:- 01

* AIM:- Demonstrate the use of different file accessing mode, attributes & read method.

STEP 1:- Create a file object using open method and use file usable accessing mode followed up by writting some contents onto the file and then closing the file

STEP 2:- Now open the file in read mode and then use read() , readline() and finally display the output with variable and finally display the contents of variable

STEP 3:- Now use the file object for finding the name of file, file mode and whether it is closed whether file is still open or close and finally the output of the seekpose attribute

Step 4: Now open the fileobject in write mode write some another content close subsequently then again open the fileobject in 'wt' mode that is the update mode and write content

Step 5: Open fileobj in read mode, display the update written content and close open again in 'rt' mode with parameter passed & display the output subsequently

Step 6: Now open fileobj in append mode open write method write content close the fileobj again. Open the fileobj in read mode and display the append output


```
# file attributes
a = fileobj.name
Print ("Name of file (name attribute : ", a)
>>> (Name of file (name attribute, abc.txt)
b = fileobj.closed
  Print ("close) attribute :" b)
>>> (close) attribute = True

c = fileobj.mode
Print ("file mode,"c)
>>> ('file mode, 'rs')
d = file obj.mode.softspace
Print ("softspace,"d)
>>> (" softspace :", 0)

# wt mode
Fileobj = open ("abc.txt", "wt")
Fileobj.write ("saurabh")
fileobj.close()

# rt mode
fileobj = open ("abc.txt", "rt")
str1 = Fileobj.read (r)
Print ("output of rt", str1)
File object close ()
>>> ('output of rt', 'saurabh')

# write mode
Fileobj = open (abc.txt," "w")
File obj.write ("pbms")
Fileobj.close()

# read mode
FileObj = open("abc.txt ","r")
str2 = FileObj.read()
Print ("output of read mode")str2)
>>> (" output of read mode", 'saurabh')
```

ASO:

```
#Opened mode
File obj : open ("abc.txt", "a")
File obj .write ("Data structure")
File .close()
File object = open ("abc.txt", "r")
Str 3 = File obj. read()
Print ("output of append mode", str3)
File obj. close()
>>> ("output of append mode:" Saurabh", "Data structure")

# tell ()
File obj =open ("abc.txt", "r")
POS: File obj . tell()
Print ("tell() :", pos)
File obj . closed
>>> ! tell (): ', pos)

# seek ()
File obj =open ("abc.txt", "r")
Str 4 = File obj. seek (0,0)
Str 8 = File obj. read (10)
Print ("the beginning of the line is =", str8)
```

Step 7: Open the fileobj in read mode, declare a variable & perform fileobj dot tell method and store the output consequently in variable

Step 8: Use the seek method with the arguments with opening the file obj in read mode and closing subsequently.

Step 9: Open fileobj with read mode also use the readlines method & store the output consequently in & print the same for counting the length use the for condition statement & display the length.

No.	
12	Develop ~~progra~~

TSO

PRACTICAL No: 2.

AIM: Demonstrate the use of iterators iterables

THEORY: In python iterator is an object which implement iterator class which has 2 method namely __iter__() and __next__().
List, tuple, dictionary, and the set all represents a iterable object

Q1] Write a program using iterable objects for displaying the odd numbers in range 1 to 10

ALGORITHMS:

STEP 1: Define a iter() with argument & initialise the value and return that value

STEP 2: Define the next() with an argument and compare the upper limit by using a condition statement

STEP 3: Now create an object of the given class and pass this object in the iter method

CODING:

028

```
Class odd :
    def __iter__(self):
        self.num = 1
        return self
    def next(self):
        if self.num <= 10:
            num = self.num
            self.num += 2
            return num
        else:
            raise stop iteration
```

OUTPUT:

```
>>> y = count()
>>> z = iter(y)
>>> z.next()
1
>>> z.next()
3
>>> z.next()
5
>>> z.next()
7
>>> z.next()
9
>>> z.next()
11
```

No. 12 Develop purge

CODING:

```
def __iter__(self):
    self.p=0
    return self
def next(self):
    if self.p<20
    num=self.p
    self.p+=1
    po = 2**num
    Print("2**",self.p-1," :",po)
    return po
else:
    raise stopiteration
```

OUTPUT:

```
>>> P=powers()
>>> x=iter(P)
>>> x.next()
* 0 =2
>>> x.next()
   2**1 = 2
>>> x.next()
   2**2 = 4
>>> x.next()
   2**3 = 8
```


02] Write a program using an iterator for calculating the powers of a given number for instance numbers entered as 2 then value calculated should be $1, 2^1, 2^2, 2^3, 2^4$

ALGORITHM:

STEP 1: Define iter() with argument and initialize value and return the value

STEP 2: Now define next() with an argument and compare the upper limit by using conditional stat

STEP 3: Now create an object of the given class and pass this object in the iter method.

No.

12 Develo
Prog

Q3] Write program using iterable concept to find factorial of numbers in range 1 to 10

ALGORITHM:

STEP 1: Define a iter() with argument & initize the value & return the value

STEP 2: Define the next() with an argument & compare the upper limit by using a conditional stat

STEP 3: Now create an object of the given class & pass this obj in the iter method

Q4] Write a program using iterable concept to display multiple of 2 in range 1 to 10.

ALGORITHM:

STEP 1: Define a iter() with argument and initialise the value and returns the value

CODING:

```
class fact:
    def __iter__(self):
        self.f = 1
        return self
    def next(self):
        if self.f <=10
            num =self.f
            self.f+=1
            fac=1
            for i in range(1,num+1):
                fac=fac*i
            print(self.f-1,"!=",fac)
        else:
            raise stop iteration
```

OUTPUT:

```
>>> f = fact()
>>> x= iter(f)
>>> x.next()
    1! =1
>>> x.next()
    2! =2
>>> x.next()
    3! =6
```

No.

12 Develop

purge

CODING:

```
class mult:
    def __iter__(self):
        self.n = 1
        return self
    def next(self):
        if self.n <= 10:
            num = self.n
            self.n += 1
            table = 2 * num
            print("2 *", num, "=", table)
        else:
            raise stop iteration.
```

OUTPUT:

```
>>> m = mult()
>>> x = iter(m)
>>> x.next()
2 * 1 = 2
>>> x.next()
2 * 2 = 4
>>> x.next()
2 * 3 = 6
>>> x.next()
2 * 4 = 8
```


STEP 2: Define the next() with an argument & compare the upper limit by using a conditional stat

STEP 3: Now create an object of the given class and pass the object in the iter method

No.	
12	Develop program

ISE LAB

PRACTICAL NO: 5

AIM: Demonstrate the use of exception Handling.

THEORY: An exception is an event which occurs during execution of program which disrupts the normal flow of program. When a exception represent object which represents an error. This is derived from given class & when the python script raised an exception it must be handled immediately otherwise it will be terminated and close the program

Q1] Write the program to check the range of the age of the students in given class and if age does not fall in given range use value error exception otherwise return the valid

ALGORITHM:

STEP 1: Define a function which will accept the age of the student from standard input

STEP 2: Use if conditional to check whether the input age falls in range and so.

CODING:

```
def accept_age()
    age = int (input ("enter your age:")
    if age >30  or age <16
        raise value error
    else:
        print ("your age is ",age)
valid = False
while not valid:
    try:
        age = accept_age()
        valid = True
    except value error
        print ("your age is not in range")
```

OUTPUT:

```
Enter your age: 15
Your age is not in range
Enter your age: 32
  Your age is not in range
Enter your age: 17
  Your age is 17
```

032

No. 12 Develope program

CODING:

```
while True:
    try:
    a= int (input ("Enter a number"))
    print ("valid number")
    break
except value error :
    print ("Not a valid number ! Try again )
```

OUTPUT:

Enter a number : 17.2
Not a valid number ! Try again
Enter a number : 17
Valid number

return the age else we value error exception

STEP 3: Define the while loop to check whether the boolean expression holds true. Use the try block to accept the age of student & terminate the looping condition

STEP 4: Use except with value error and print the message not a valid range

Q27 Write a program to check whether the number is given class & if the no. is a floating point use value error as exception for given input

ALGORITHM

STEP 1: Use try block and accept the input () and convert it into integer datatype & subsequently terminate the block

STEP 2: Use the except block with exception as value error and display appropriate message a suspicious code is part of try block

Q3] Write a program to demonstrate use of zero division error.

ALGORITHM:

STEP 1: Use the try block & accept the input using input() & then convert it into integer datatype

STEP 2: Define a function with 2 parameter to divide the nos given by user

STEP 3: Define while loop to check whether the boolean expression hold true

STEP 4: Use except with zero division error and print the message

CODING:

```
def divide(a,b)
    ans = a/b
    return ans
while True:
    try:
        a= int(input("Enter first number:"))
        b= int(input("Enter second number:"))
        ans = divide(a,b)
            print("division of" a,"and "b" is" ans)
        break
    except zero division error:
        print("Error!")
```

OUTPUT:

```
>>> Enter first no:1
>>> Enter second number:1
 Division of 1 and 1 is 1
>>> Enter first number:1
    Error!
```

No.	
12	Develop programs

CODE:

```
import re
String = "hello1234 abc 4567"
result = re.findall ("\d+", String)
result = re.findall ("\D+", String)
print (result)
print (result)
```

OUTPUT:

```
>>> ['1234', '4567']
>>> ['hello', 'abc']
```


PRACTICAL NO: 4.

AIM: To demonstrate the use of regular expression.

THEORY: Regular expression operates the sequence of characters which is mainly used for finding & replacing the given pattern in a common usage of regular expression involves four functionalities:-

- Searching a given string
- Finding a string
- Breaking a string into smaller substring.
- Replacing part of string.

Q1) Write a regular expression segregating numeric & alphabetical value from a given string

ALGORITHMS:

STEP 1: Now apply string & pattern in findall() and display the output

STEP 2: \d is used for matching all decimal digits whereas \D is used to match non-decimal digits

Q2] Write a regular expression for finding the match string at beginning of the sequence

ALGORITHM:

STEP 1: Import remodule and apply a string

STEP 2: Use search() with "\A Python" and string as two parameters

STEP 3: Now display the output.

STEP 4: Now use if conditional statement for user to know whether the match is found or not

CODE 2:

```
Import re
String = "Python is an important language";
result = re.search("\A Python", string)
print (result)
if result:
    print ("match found")
else:
    Print ("match not found")
```

OUTPUT:

```
>>> re match object : span=(0,6),
>>> match found.
```

#CODE:

```
li = ["9876543210", "876543210",
      "7654321098", "6543210987"]

for element in li:
    result = re.match("[8-9][1]
                       [0-9][9]", element)
    if result:
        print("correct mobile no")
        print((result.group(0)))
    else:
        print("Incorrect mobile no")
```

#OUTPUT:

```
>>> correct mobile no.
    6543210
    correct mobile no
    8765432109
    Incorrect mobile no.
    Incorrect mobile no.
```


Q3] Write a regular expression to check whether the given mobile number starts with 8 or 9 the total length of digit should be atleast 10

ALGORITHM:

STEP 1: Import re module and apply a string of numbers

STEP 2: Now use for conditional stat to find if the number starts with 8 or 9 and the total no should be length of 10. Use match() inside for statement to find the match in given string.

STEP 3: Use if conditional statement to know whether we have a match or not. If we have use group() to display the output and if we dont display incorrect mobile no.

No. 12 Develop

Q6] Write a regular expression for extracting a word from a given string along with space characters in b/w the & subsequently extract the word without space char

ALGORITHM:

STEP 1: Import re module & apply a string.

STEP 2: Use findall() to extract a word from given string

STEP 3: Use "\w" to extract word along with space and use "\w+" to extract word without space

STEP 4: Now display the output.

CODE :

```
import re
string = " Python is important"
result1 = re.findall ("\w+ ", string)
result2 = re.findall ("\w+", string)
print (result1)
print (result2)
```

OUTPUT

```
>>> ['Python ', 'is ', 'important']
['Python', 'is', 'important']
```

BE0

```
# message box
from Tkinter import
import tkmessage Box
root = Tk()
def function():
    tkmessageBox.showinfo ("Info window", "Python")
b1 = Button (root, text = "Python", command = function)
b1.pack()
root.mainloop()
```


PRACTICAL NO : 5 (c)

AIM: GUI COMPONENTS

STEP 1: Import the relevant method from tkinter message box

STEP 2: Import the tk message box

STEP 3: Define a parent window object along with the parent window

STEP 4: Define a function which will use tk.messagebox with show info method along with info window attribute

STEP 5: Declare a button with parent window object along with command attribute

STEP 6: Place the button widget onto the parent window on a finally call mainloop() for gui update of the called above

No.	
12	Develo ping

STEP 1: Import the relevant method from the tkinter library along with parent window object

STEP 2: Use parent window object along with minsize function for window size.

STEP 3: Define a function main, declare parent window object and use config (), title () minimize (), Label () as well as button () and we packe & mainloop ()

STEP 4: Similarly define the function second and use the attribute accordingly

STEP 5: Declare another function button along with parent object and declare button with attribute like FLAT, RIDGE, GROOVE

STEP 6: Finally called the mainLoop () for event driven programming


```
# Multiple window
# Different button (Relief())
from Tkinter import *
root = Tk()
root.minsize (300,300)
def main ();
        top = TK()
        top.config (bg = "black")
        top.title ("Home")
        top.minsize (300,300)
        L = Label (top, text = " SAN FRANCISCO  places of Inter
              \n Golden gate Bridge \n Lombard street \n
              Chinatown \n coit Tower")
        L.pack()
      b1 = Button (top, text = "next", command = second)
      b1.pack (side = RIGHT)
      b2 = Button (top, text = "exit", command = terminate)
      b2.pack (side = LEFT)
      top.mainloop()
```

```
def second():
    top2 = Tk()
    top2.config(bg="orang")
    top2.title("About Us")
    top2.minsize(300,300)
    l = Label(top2, text="Created by : Tanvi Mane \n for main details contact to our official account")
    l.pack()
    b2 = Button(top2, text="Next", command=main)
    b3.pack(side=LEFT)
    b2 = Button(top2, text="Exit", command=terminate)
    b2.pack(side=RIGHT)
    top2.mainloop()
button():
    top3=Tk()
    top2.geometry("300x300")
    b1=Button(top3, text="flat button", relief=FLAT)
    b1.pack()
    b2=Button(top3, text="groove button", relief=GROOVE RAISED)
    b2.pack()
    b3=Button(top3, text="raised button", relief=RAISED)
    b3.pack()
    b4=Button(top3, text="SUNKEN Button", relief=SUNKEN)
    top3.mainloop()
    def terminate():
```


[A]

TOPIC: GUI COMPONENTS

STEP 1: Use the tkinter library for importing the functions of the text widget

STEP 2: Create an object using the Tk()

STEP 3: Create an variable saving the widget label and use the pack method

STEP 4: Use the mainloop() for triggering of the corresponding above main commands

#2:

STEP 1: Use the tkinter library for importing the text. features of the text widget

STEP 2: Create a variable from the text method and position it on the parent window

STEP 3: Use the pack along with the object created from the text() and use the parameters

1) side: LEFT, padx = 20
2) side = LEFT, pady = 30
3) side = TOP, ipadx = 40
4) side = Top, ipady = 50

STEP 4: Use the mainloop fun the triggering of the corresponding exits.

STEP 5: Now repeat above steps with the label w which takes the foll. garuments

1) Name of the parent window
2) Text attribute which defines the string
3) The background colour (bg)
4) The foreground fg & then use the pack w with a relief padding attribute


```
from TK inter import
root=TK()
w= label (root, text="python")
l= pack()
root. mainloop()
```

#2: label ; attributes

```
from Tkinter import
root TK()
l= label (root, text="python")
l. pack()
l1 = label (root, text="[3]", bg="gaey", fg="black", font="20")
l1. pack( side= LEFT, padx= 20)
l2 = Label (root, text = "eg", bg = "lightblue", fg="black", R
l2. pack( side= LEFT, padx = "30")
l3 = Label Croot. text = "csl", bg="yellow", fg" black" font"
l3. pack (side =TOP, padx="30")
l4 = label (root, text= "CS", bg="orange", fg = "black")
        found = "110")
l4. pack (side = top, pady=50)
roo. mainloop()
```

[E]

#1 SPINBOX

STEP1 : Import relevant CLs from the tkinter library

STEP2 : Create parent window object along with TK()

STEP3 : create an object from spinbox method and use attributes parent window object, from_, and to attributes

STEP3 : Subsequent call the pack method along with spinbox object & called the mainloop()

#2 PANED WINDOW

STEP1 : Import relevant methods from the tkinter library also create a parent window object

STEP2 : Create an object along with paned window & subsequently use the pack method with the panned window object along with attributes like fill & expand.

STEP3 : Create an label object along with label() use panel window object, orient & background color

No.

12 Tuesday

#3 CANVAS :

STEP 1: Import relevant Ui's from the intel library & create a parent window obj.

STEP 2: Create a canvas object along with the canvas with background colour

STEP 3 : Use create arc() along with canvas object along with start & extent & coordinates

STEP 4: Similarly for oval & line use