Practical No:- 1

1.1) Aim:- Write a python program to implement various File operations.

A) Opening the File

CODE:-

```
fw = open ("sample1.txt","x")
if fw:
    print ("File opened successfully)
    fw.close()
```

O/P:-

File opened successfully

B) Reading the File

CODE:-

```
fw = open ("sample2.txt","x")
content = fw.read()
print (content)
```

File → Sample2

Hello World Good Morning

O/P:-

Hello World Good Morning

C) Writing a File

CODE:

```
fw = open ("sample2.txt","w")
fw.write ("python is used to develope machine
                    algorithms")
fw.close()
```

O/P:-

```
=== /Desktop/APP37/prac.1.6.py ===
```

File → Sample2

python is used to develope machine learni
algorithms

D) Append to the File

CODE:-

```
fw = open ("sample2.txt","a")
fw.write ("python has plenty of libraries")
fw.close()
```

O/P :-

=== /Desktop /APP37 / prac.1.d.py ===

File → Sample2

python is used to develope machine learning algorithmspython has plenty of libraries

E) Rename to the File

CODE:-

```
import os
os.rename ("sample2.txt","sample1.txt")
```

O/P:-

=== /Desktop /App37 / Prac.1.e.py ===

The File sample2 are rename in sample1 the

Sample1

python is used to develope machine learning algorithmspython has plenty of libraries

Practical No:- 2

Q.2.) Aim:- Write a python program to demonstrate use of Regular expression for suitable applications.

A) Search the String to see if it starts with 'The' and ends with 'India'.

CODE:-

```
import re
txt = "The rain in India"
x = re.search ("^The.*India$", txt)
if x:
    print ("YES! We have a match!")
else:
    print ("NO Match")
```

O/P:-

YES! We have a match!

6) Search for the first white space character in the string.

CODE:-

```
import re
txt = "python is widely used language"
x = re.search ("\s", txt)
print ("The first white-space character is located in position:",
        x.start())
```

O/P:-

The first white-space character is located in position: 6

c) Split at each wide-space character

code:-

```
import re
txt = "python is widely used language."
x = re.split("\s", txt)
print(x)
```

o/p:-

['python', 'is', 'widely', 'used', 'language.']

D) Replace every white space character with special character #.

CODE:-

```
import re
txt = "python is widely used language."
x = re.sub ("\s", "#", txt)
print (x)
```

O/P:-

python#is#widely#used#language.

E) Program to search that will return a match object.

CODE:-

```
import re
txt = "The rain in spain"
x = re.search ("ai", txt)
print (x)
```

O/P:-

<re.Match object; span=(5, 7), match='ai'>

E) Program to use findall in regular expression

CODE:

```
import re
pattern = "to"
text = "I will go to Mumbai to meet my friend to
        tell something."
for m in re.findall (pattern, text):
    print ("found", m)
for m in re.finditer (pattern, text):
    s = m.start()
    e = m.end()
    print ("Found", m.re.pattern, s, "from", e)
```

O/P:-

```
Found to
Found to
Found to
Found to 10 from 12
Found to 20 from 22
Found to 38 from 40
```

Practical No:- 5

Q.5.) Aim:. Write a program to demonstrate different types of exception handling

A) program to print the reciprocal of even number.

CODE:.

```
try:
    num=int(input("enter a number:"))
    assert num%2==0
except:
    print("Not an even number")
else:
    reciprocal = 1/num
    print(reciprocal)
```

O/P:-

~~try block~~

enter a number: 5

Not an even number

=== C:/Users/Admin/Desktop/APP37/Prac.5.a.py =

enter a number: 4

0.25

B) program to access two numbers from the user perform these division.

CODE:

```
try:
    print ('try block')
    x=int (input ('Enter 1st number:'))
    y=int (input ('Enter 2nd number:'))
    z=x/y

except ZeroDivisionError:
    print ("except ZeroDivisionError Block")
    print ("Divion by 0 not accepted")

else:
    print ("else block")
    print ("division=", z)
Finally:
    print ("Finally block")
    x=0
    y=0
    print ("out of try, except, else, Finally blocks")
```

O/P :-

try block
Enter 1st number : 23
Enter 2nd number : 45
else block
division = 0.511111111111111
Finally block
out of try, except, else, Finally blocks

=== C:/Users/Admin/Desktop/APP37/Prac.5.b.py ==

try block
Enter 1st number : 23
Enter 2nd number : 0
except ZeroDivionError Block
Divion by 0 not accepted
Finally block
out of try, except, else, Finally blocks

c) Program to accept number from the user the tryblock raise value error exception if their no. is outside the allowed range.

CODE:-

```
try:
    num = int(input('Enter a number upto 50:'))
    if num>50:
        raise ValueError(num)
except ValueError:
        print(num,"It is out of range")
else:
        print(num,"It is within the range")
```

O/P:-

```
Enter a number upto 50: 45
45 It is within the range

=== C:/Users/Admin/Desktop/APP37/prac.5.c.py ===
Enter a number upto 50: 56
56 It is out of range
```

Practical :- 6

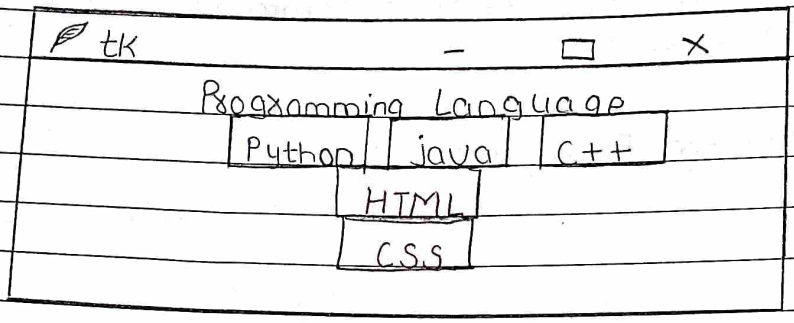
Q.6. Aim:- Write a GUI program in python to design a application.

A) Different Fonds and colours

code:-

```
from thintex import *
root = Tk()
root.geometry("350x100")
w = Label (root, text = 'programming language', font="40")
w.pack ()
Frame = Frame (root)
Frame.pack()
bottomFrame = Frame (root)
bottomFrame.pack (side = BOTTOM)
b1_button = Button (Frame, text = "python", fg = "blue")
b1_button.pack (side = LEFT)
b2_button = Button (Frame, text = "java", fg = "blue")
b2_button.pack (side = LEFT)
b3_button = Button (Frame, text = "c++", fg = "blue")
b3_button.pack (side = LEFT)
b4_button = Button (bottomFrame, text = "HTML", fg = "Red")
b4_button.pack (side = BOTTOM)
b5_button = Button (bottomFrame, text = "CSS", fg = "Red")
b5_button.pack (side = BOTTOM)
root.mainloop ()
```

O/P :-

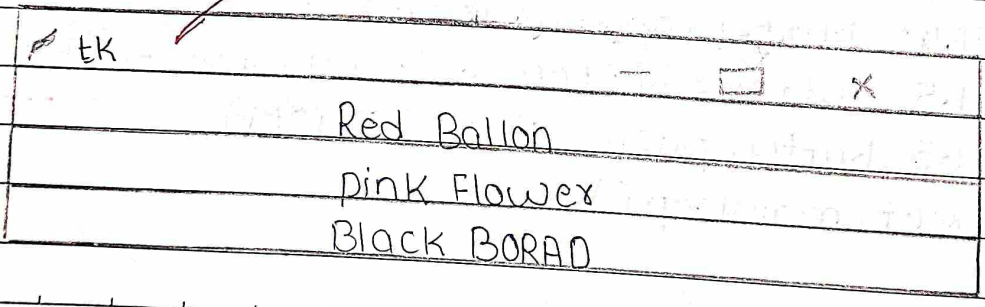


B) Different layout manager

CODE :-

```
import tkinter as tk
root=tk.Tk()
w=tk.Label(root, text="Red Ballon", bg="red", fg="white")
w.pack(fill=tk.X)
w=tk.Label(root, text="pink Flower", bg="pink", fg="black")
w.pack(fill=tk.X)
w=tk.Label(root, text="Black BORAD", bg="black", fg="white")
w.pack(fill=tk.X)
tk.mainloop()
```

O/P :-



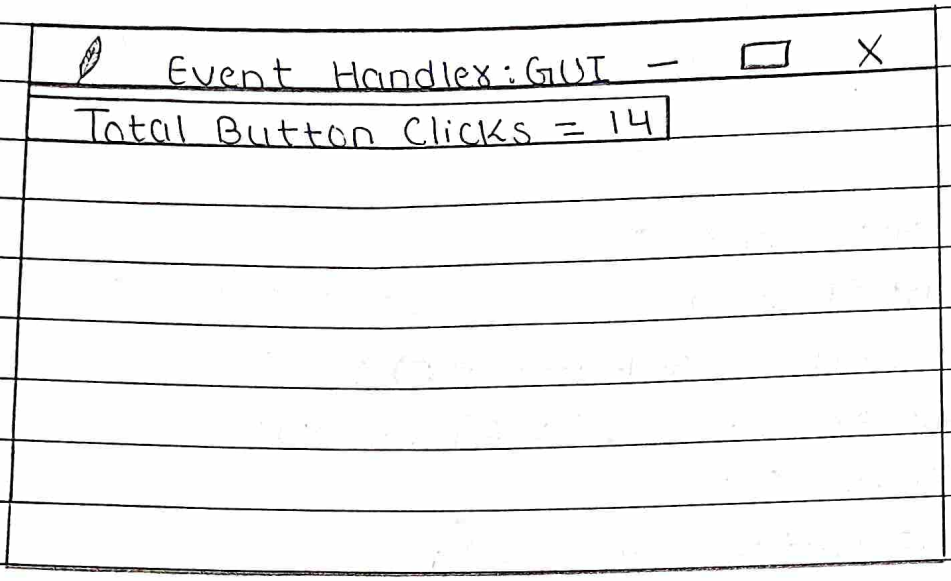
c) event handling

CODE:

```
From tkintex import *;
class Event (Fxame):
    def __init__ (self, mastex):
        Fxame.__init__ (self, mastex);
        self.gxid ();
        Self.button_clicks = 0;
        Self.cxeate_widget ();
    def cxeate_widget (self):
        Self.buttn = Button (self, text = "total button clicks
                                          =0");
        Self.buttn ["command"] = self.update_click_count;
        self.buttn.gxid ();
    def update_click_count (self):
        self.button_clicks+=1;
        Self.buttn ["text"] = "Total Button Clicks=" + Stx
                                  (self.button_clicks);

mainWindow = TK()
mainWindow.title ("Event Handlex : GUI");
mainWindow.geometxy ("300x200");
app = Event (mainWindow);
mainWindow.mainloop();
```

O/P :-



Practical no :- 9

Q.9. Aim :- Write a python program to implement Concept of Oops such as types of Methods, Inhexitance and polymorphism.

A.) Program to use types of methods such as class, instance and static.

Code :-

```
class student :
    # Constructor
    def __init__ (self, n, r):
        self.name = n
        self.roll = r
    # instance method
    def disp (self):
        print ("student name :", self.name)
        print ("Student Roll Number :", self.roll)
class User :
    # static method
    def show (s):
        print ("User Name :", s.name)
        print ("User Roll :", s.roll)
        s.disp ()
st = student ('Amit', 1001)
User.show (st)
```

O/P :-

```
Usex Name : Amit
Usex Roll : 1001
Student name: Amit
Student Roll Numbex : 1001
```

B) Inhexitance

CODE:

```
Class Animal:
    def speak(self):
        pxint ("Animal Speaking")
Class Dog(Animal):
    def baxk(self):
        pxint ("Dog Baxking")
Class DogChild(Dog):
    def eat(self):
        pxint ("Eating Bxead")
d = DogChild()
d.baxk()
d.speak()
d.eat()
```

O/P :- Dog Baxking
Animal Speaking
Eating Bxead

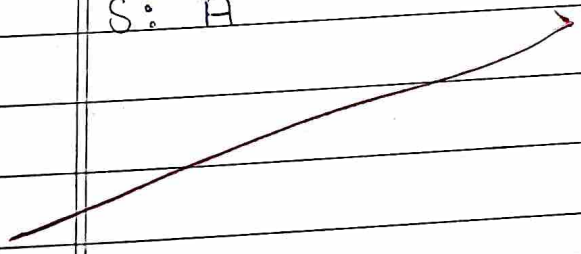
c) Polymorphism

i) CODE:-

```
def show(s):
    print("s:",s)
show(12)
show(12.34)
show("python")
show('A')
```

O/P:-

```
S: 12
S: 12.34
S: python
S: A
```



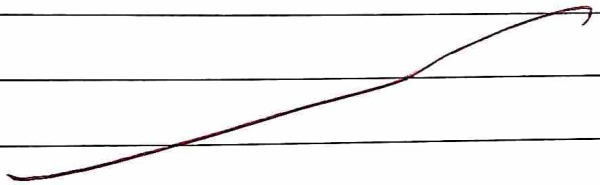
Q)

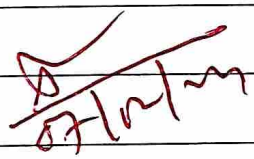
ii) CODE :-

```
Class Bird :
    def intro (self):
        print ("There are many types of birds
    def flight (self):
        print ("Most of the birds can fly but
                                    cannot.")
class sparrow (Bird):
    def flight (self):
        print ("Sparrows can fly.")
class ostrich (Bird):
    def flight (self):
        print ("Ostriches cannot fly.")
obj_bird = Bird ()
obj_spr = sparrow ()
obj_ost = sparrow ()
obj_bird.intro ()
obj_bird.flight ()
obj_spr.intro ()
obj_spr.flight ()
obj_ost.intro ()
obj_ost.flight ()
```

O/P :-

There are many types of birds.
Most of the birds can fly but some cannot.
There are many types of birds.
Sparrows can fly.
There are many types of birds.
Sparrows can fly.





Practical no:-10

Q.10.) Aim:- Write a python program to implement Concept of oops such as abstract methods and class, interface.

A.) Abstract Method

CODE:-

```
From abc import ABC, abstractmethod
Class Car (ABC):
    def mileage (self):
        pass
Class Tesla (car):
    def mileage (self):
        print ("The mileage is 30Kmph")
Class Suzuki (car):
    def mileage (self):
        print ("The mileage is 25Kmph")

t = Tesla()
t.mileage()
s = Suzuki()
s.mileage()
```

O/P:-

The mileage is 30Kmph
The mileage is 25Kmph

B) Program to use informal interface

CODE:

```
Class Chocolate :
    def __init__ (self, items):
        self.__items = items
    def __len__ (self):
        return len (self.__items)
    def __contains__ (self, items):
        return items in self.__items
fav_ch = Chocolate (["KitKat", "Silk", "Munch", "5star"])
print (len (fav_ch))
print ("KitKat" in fav_ch)
print ("Silk" in fav_ch)
print ("Munch" in fav_ch)
print ("5star" in fav_ch)
```

O/P: -

```
4
True
True
True
True
```

c) Program to use Formal interface

CODE:-

```
import abc
class Food (abc.ABC):
    def taste (self):
        Pass
class north_indian(Food):
    def taste (self):
        print ("Cooking!")
s = north_indian()
print (isinstance (s, Food))
```

O/P :-

True

Q.3.) Aim: Write a program to demonstrate the concept of threading

A:) Program to demonstrate the concept of threading

code:

```
import logging
import threading
import time
def thread_function(name):
    logging.info("Thread %s: starting", name)
    time.sleep(2)
    logging.info("Thread %s: finishing", name)
if __name__ == "__main__":
    Format = "%(asctime)s: %(message)s"
    logging.basicConfig(Format=Format, level=logging.
                        INFO, datefmt="%H:%M:%S")

    logging.info("Main    : before creating thread")
    x = threading.Thread(target=thread_function, args=(1,))
    logging.info("Main    : before running thread")
    x.start()
    logging.info("Main    : wait for the thread to
                        finish")

    x.join()
    logging.info("Main    : all done)
```

O/P:

21:27:03: Main: before creating thread
21:27:03: Main: before running thread
21:27:03: thread 1: starting
21:27:03: Main: wait for the thread to finish
21:27:05: thread 1: Finishing
21:27:05: main: all done

B.) Program to demonstrate the concept of multitasking

code:

```
import thread as thread, time
def counter (myID, count):
    for i in range (count):
        time.sleep (1)
        mutex.acquire()
        print ("Thread [%s] => %s" % (myID, i))
        mutex.release()

mutex = thread.allocate_lock()
for i in range (3):
        thread.start_new_thread (counter, (i,3))
        time.sleep(6)
print ('Main thread exiting')
```

O/P:

```
Thread [0] => 0
Thread [0] => 1
Thread [0] => 2
Thread [1] => 0
Thread [1] => 1
Thread [1] => 2
Thread [2] => 0
Thread [2] => 1
Thread [2] => 2
```

Q.7.) Aim: Write python program to create application which use date and time in python.

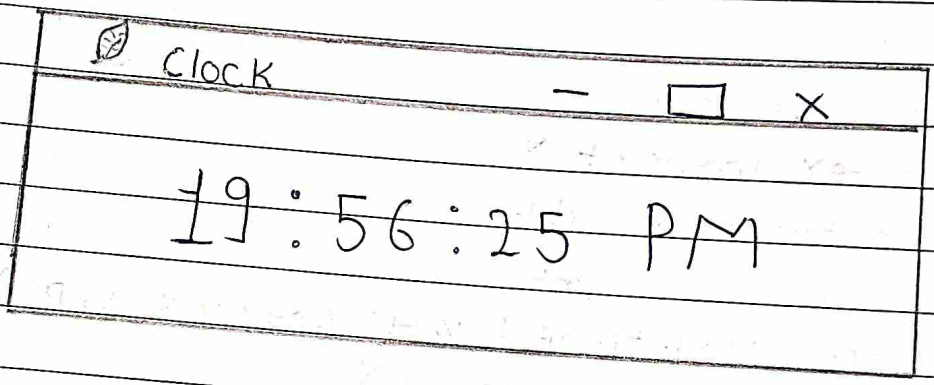
Code:

```
from tkinter import *
from tkinter.ttk import *
from time import strftime
root = Tk()
root.title('Clock')
from tkinter import *
import datetime as dt
def time():
    string = strftime('%H:%M:%S %p')
    lbl.config(text=string)
    lbl.after(1000, time)
lbl = Label(root, font=('calibri', 40, 'bold'), background
                = 'purple', foreground='white')
lbl.pack(anchor='center')
time()
mainloop()
def date():
    win=Tk()
    win.title("Display Current Date")
    win.geometry("700x350")
    date=dt.datetime.now()
    label =Label(win, text=f"{date:%A, %B %d, %Y}
                Font="calibri,20")
```

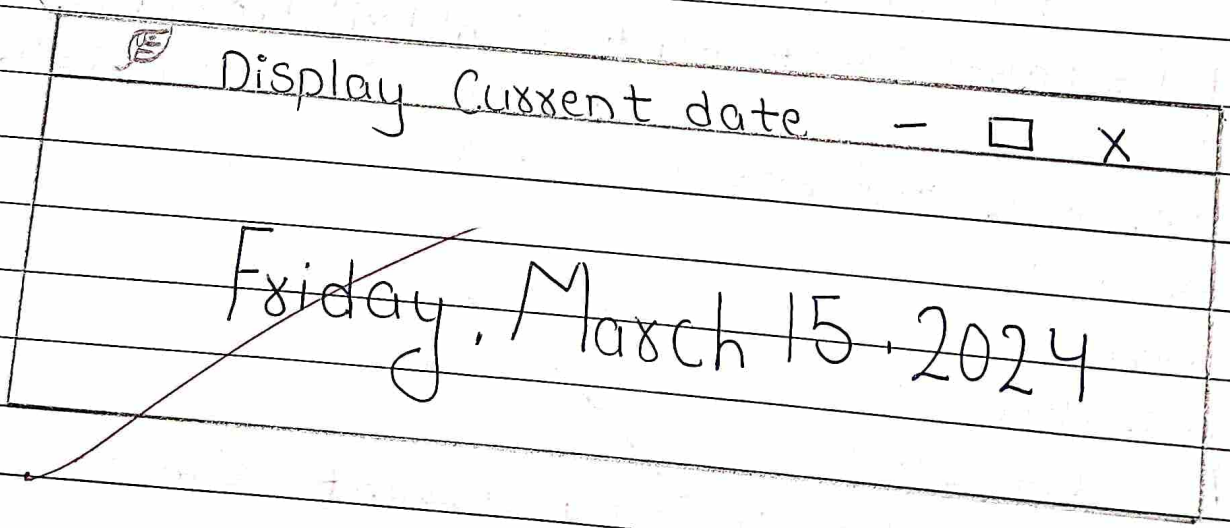
```
        label.pack (pady = 20)
date()
mainloop()
```

O/P:-

time :



Date :



Q.8.) Aim: Write a python program to create server, client and exchange basic information

code:

```
# Server program

import socket
def server_program():
    host = socket.gethostname()
    port = 5000
    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(2)
    conn, address = server_socket.accept()
    print("Connection From:" + str(address))
    while True:
        data = conn.recv(1024).decode()
        if not data:
            break
        print("from connected user:" + str(data))
        data = input('->')
        conn.send(data.encode())
    conn.close()
if __name__ == '__main__':
    server_program()
```

Client program

Code:

```
import socket
def client_program():
    host = socket.gethostname()
    port = 5000
    client_socket = socket.socket()
    client_socket.connect((host,port))
    message = input("->")
    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())
        data = client_socket.recv(1024).decode()
        print('received from sexue:' +data)
        message = input("->")
    client_socket.close()
if __name__ == '__main__':
    client_program()
```

O/P:

\client.py	\Sexuex.py
==	==
-> hii	Connection from: ('192.168.75.1', 59447)
xeceived from sexue: hii	From connected usex: hii
-> hello bxo	->hii
xeceived from sexue: itsbxo	from connected usex: hellobxo
-> Bye	->its bxo