

Practical No:- 1

Q.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 develop machine  
algorithms")
```

```
fw.close()
```

O/P:-

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

File →

Sample2

python is used to develop machine learning
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 / prac1.d.py ===

File :-

Sample2

python is used to develop machine learning  
algorithms python has plenty of libraries

E) Rename to the File

CODE :-

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

O/P:-

~~sample1~~

==== Desktop / App37 / Prac1.e.py ===

The file sample2 are renamed in sample1 the

sample1

python is used to develop machine learning  
algorithms python 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 train 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!

B) 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 characters

CODE:-

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

O/P:-

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

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~~

~~Output~~

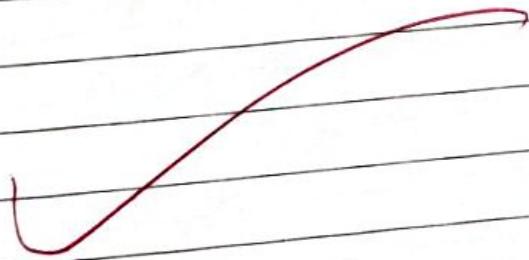
E) Program to search that will return a match object.

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~~

~~Output~~

## Practical No.: 5

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

A) program to print the reciprocal of even numbers.

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 and 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("Division 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

Enters 1st number : 23

Enters 2nd number : 45

else block

division = 0.5555555555555556

Finally block

out of try, except, else, Finally blocks

==> C:/Users/Admin/Desktop/APP37/Proc-5.b.py ==>

try block

Enters 1st number : 23

Enters 2nd number : 0

except ZeroDivisionError Block

Divion by 0 not accepted

Finally block

out of try, except, else, Finally blocks

c) Program to accept number from the user the try block raise value error exception if the 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 / pycnc.5.c.py =====

Enter a number upto 50: 56

56 It is out of range

✓ M

## Practical :- 6

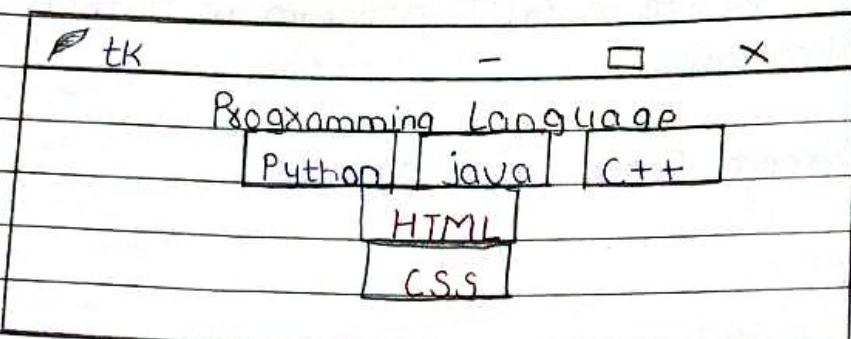
Q.6. Aim:- Write a GUI program in python to design a application.

A) Different Fonts and colours

CODE:-

```
from tkinter import *
root = Tk()
root.geometry("350x100")
w = Label(root, text='programming language', font="40")
w.pack()
Fframe = Frame(root)
Fframe.pack()
bottomFrame = Frame(root)
bottomFrame.pack(side=BOTTOM)
b1_button = Button(Fframe, text="python", fg="blue")
b1_button.pack(side=LEFT)
b2_button = Button(Fframe, text="java", fg="blue")
b2_button.pack(side=LEFT)
b3_button = Button(Fframe, 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 managers

CODE:-

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

O/P:-

The diagram shows a rectangular window frame. Inside, the title bar has a leaf icon and the text "tk" followed by a dash and a close button. Below the title bar, there are three lines of text: "Red Ballon", "pink Flower", and "Black BOARD".

### c) event handling

CODE:-

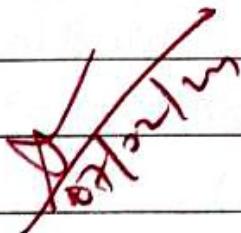
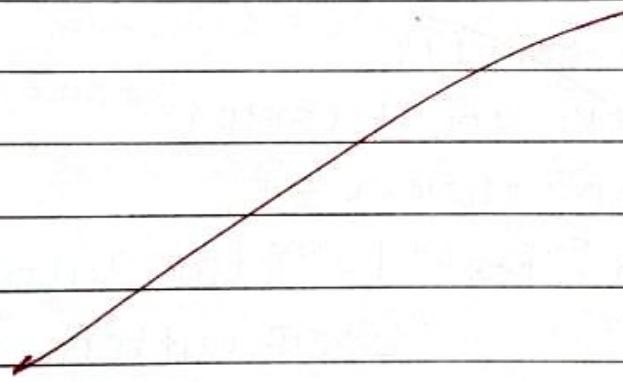
```
from tkinter import *
class Event(Frame):
 def __init__(self, master):
 Frame.__init__(self, master)
 self.grid()
 self.button_clicks = 0
 self.create_widget()
 def create_widget(self):
 self.button = Button(self, text="total button clicks = 0")
 self.button["command"] = self.update_click_count
 self.button.grid()
 def update_click_count(self):
 self.button_clicks += 1
 self.button["text"] = "Total Button Clicks = " + str(self.button_clicks)
mainWindow = Tk()
mainWindow.title("Event Handler GUI")
mainWindow.geometry("300x200")
app = Event(mainWindow)
mainWindow.mainloop()
```

OIP :-



Event Handler: GUI -  X

Total Button Clicks = 14



Practical no :- 9

Q.9. Aim: Write a python program to implement concept of OOPS such as types of Methods, Inheritance and Polymorphism.

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

Code:

class student:

# Constructor

def \_\_init\_\_(self, n, x):

self.name = n

self.roll = x

# 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 :-

User Name : Amit

User Roll : 1001

Student name : Amit

Student Roll Number : 1001

## B) Inheritance

CODE :-

Class Animal :

```
def speak(self):
```

```
 print("Animal Speaking")
```

Class Dog(Animal) :

```
def bark(self):
```

```
 print("Dog Barking")
```

Class DogChild(Dog) :

```
def eat(self):
```

~~```
    print("Eating Bread")
```~~

d = DogChild()

d.bark()

~~d.speak()~~~~d.eat()~~

O/P :- Dog Barking

Animal Speaking

Eating Bread

c) Polymorphism

i) CODE:-

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

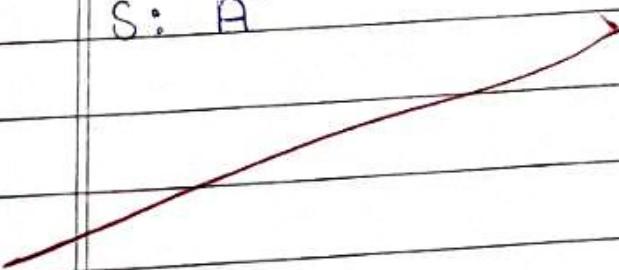
O/P:-

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_spx = Sparrow()

Obj_ost = Sparrow()

Obj_bird.intro()

Obj_bird.Flight()

Obj_spx.intro()

Obj_spx.Flight()

Obj_ost.intro()

Obj_ost.Flight()

OIP:-

These are many types of birds.

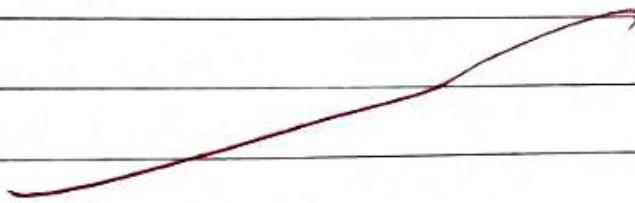
Most of the birds can fly but some cannot.

These are many types of birds.

Sparrows can fly.

These are many types of birds.

Sparrows can fly.



~~of them~~

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 Cax(ABC):
    def mileage(self):
        pass
class Tesla(Cax):
    def mileage(self):
        print("The mileage is 30Kmph")
class Suzuki(Cax):
    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, item):
```

```
        return item 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

QUESTION

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: {date: %A, %B %d, %Y}")
    label.config(font="calibri, 20")
```

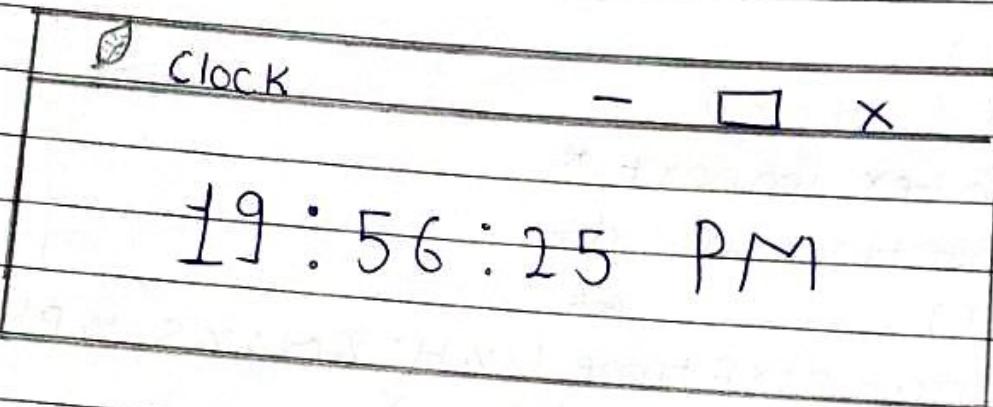
date()

mainloop()

label.pack (pady=20)

O/P:-

time :



Date:

Display Current date -

Friday, March 15, 2024

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 server:', + data)
        message = input("->")
    client_socket.close()
if __name__ == '__main__':
    client_program()
```

O/P:

| \client.py | \Server.py |
|-------------------------------|--|
| == | == |
| → hii | Connection from: ('192.168.75.1', 59447) |
| received from server: hii | From connected user: hii |
| → hello b80 | → hii |
| received from server: its b80 | From connected user: hello b80 |
| → Bye | → its b80 |