Sem - 2

CSU

Practical no: 1

Objective: Demonstrate the use of different file accessing modes, different attributes read methods.

Step 1: Create a file object using open method and use the write method and use the write access mode followed by writing 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 readlines() and store the output in variable and finally display the contents of variable.

Step 3: Now use the fileobject for finding the name of the file mode in which its opened whether the file is still open or close and finally the output of the softspace attribute.

```
fileobj = open("abc.txt", "w")  # file open(write mode
fileobj.write("Computer science subjects"+"\n")
fileobj.write("DBMS\nPython\nDS\n")
fileobj.close()

fileobj = open("abc.txt","r") # read mode
# read():
str1 = fileobj.read()
print("The output of read method:", str1)
fileobj.close()
>>>('The outputs of read method:', 'Computer science
    subjects\nDBMS.\nPython\nDs\n')
# readline():
    fileobj = open("abc.txt","r")
    str2 = fileobj.readline()
    print("The output of readline method:", str2)
    fileobj.close()
>>>('The output of readline method:', 'Computer
    science subjects\n')
# readlines():
    fileobj = open("abc.txt","r")
    str3 = fileobj.readlines()
    print("The output of readlines method:", str3)
    fileobj.close()
>>>('The output of readlines method:', ['Computer
    science subjects\n', 'DBMS\n', 'Python\n', 'DS\n']
# 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)
>>> (S%(close) attribute :', 'True')

c = fileobj.mode
print ("file mode", c)
>>> ("file mode', 'r')

d = fileobj.softspace
print ("softspace", d)
>>> ('softspace:', 0)
```

W+ mode:

```
fileobj = open("abc.txt",
                "w+")
fileobj.write ("Loukik
                sir")
fileobj.close()
```

r+ mode:

```
fileobj = open("abc.txt", "r+")
s1 = fileobj.read(6)
print ("output of r+", s1)
fileobj.close()
>>>('output of r+', 'loukik')
```

append mode:

```
fileobj = open("abc.txt", "a")
fileobj.write ("data structure")
fileobj.close()
fileobj = open ("abc.txt", "r")
s3 = fileobj.read()
print ("Output of append mode :", s3)
fileobj.close()
>>> ('output of append mode :', ' loukik sir
        data structure')
```

write mode:

```
fileobj = open("abc.txt","w")
fileobj.write ("DBMS")
fileobj.close()
```

read mode:

```
fileobj = open ("abc.txt","r")
s = fileobj.read()
print("output of read
  mode!", s)
>>> ('output of read
    mode!', 'loukik sir')
```


Step4: Now open the fileobj in write mode write some another content close subsequently Then again open the fileobj in 'W+' mode that is the update mode and write contents.

Step5: Open fileobj in read mode display the update written contents and close open again in (r+) mode with parameter passed and display the output subsequently.

Step6: Now open fileobj in append mode open write method write contents close the fileobj again open the fileobj in read mode and display the appending output.

Step 7: Open the fileobj in read mode declare a variable and perform fileobject 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 file with read mode also use the readlines method and store the output consequently in and print the same for counting the length use the for conditional statement and display the length.


```
# tell():
fileobj = open("abc.txt","r")
pos = fileobj.tell()
print("tell():", pos)
fileobj.close()
>>> ('tell():', 0L)

# seek():
fileobj = open("abc.txt","r")
st = fileobj.seek(0,0)
print("seek(0,0) is:",st)
fileobj.close()
>>>('seek(0,0) is:', None)
fileobj = open("abc.txt","r")
st1 = fileobj.seek(0,1)
print("seek(0,1) is:", st1)
fileobj.close()
>>> ('seek(0,1) is:', None)
fileobj = open("abc.txt")
st2 = fileobj.seek(0,2)
print("seek(0,2)is:", st2)
fileobj.close()
>>> ('seek(0,2) is:', None)
# finding length of different lines exist within lines:
fileobj = open("abc.txt","r")
stat = fileobj.readlines()
print("output:", stat)
for line in stat:
        print(len(line))
fileobj.close()
>>>('output:',['loukik sir data structure'])
26
```

#iter() and next():

```
mytuple1=("banana","Orange","apple")
myiter1=iter(mytuple1)
print(next(myiter1))
myiter2=iter(mytuple1)
print(next(myiter2)
myiter3=iter(mytuple1)
print(next(myiter3))
```

```
>>> banana
    Orange
    apple
```

#for loop:

```
mytuple1=("Akhil","Anil","Anila")
for x in mytuple1:
    print(x)
```

```
>>> Akhil
    Anil
    Anila
```

#square and cube:

```
def square(x):
    y=x*x
    return y
def cube(x):
    z=x*x*x
    return z
sten1=[square,cube]
```


Practical no: 2

Aim: Iterators.

Step 1: Create a tuple with elements that we need to iterate using the iter and next method. The number of time we use the iter and next method we will get the next iterating element in the tuple. Display the same.

Step 2: The similar output can be obtained by using for conditional statement. An iterable variable is to be declared in for loop which will iterate.

Step 3: Define a function name square with a parameter which will obtain output of square value of the given number. In similar fashion declare value to get the value raised 3 and return the same.

Step 4: Call the declared function using function call.

Step 5: using for conditional statement specifying the range use the list typecasting with map method declare a 'lambda' i.e anonymous function and print the same.

Step 6: Declare a listnum variable and declare some elements then use the map method with help of lambda function give two arguments display the same.

Step 7: Define a function even with a parameter then using conditional statements to check whether the number is even and odd and return respectively.

Step 8: Define a class and within that define the iter() method which will initialise the first element within the container object.

Step 9: Now use the next() and define the logic for displaying odd value.


```
for r in range(5):
    value = list(map(lambda x:x(r), funct1))
    print(value)
>>> [0,0]
    [1,1]
    [4,8]
    [9,27]
    [16,64]

# map():
listnum = [0,4,5,7,9,11,13,15,20,19,25]
listnum = list(map(lambda x: x%5, listnum))
print(listnum)
def even(x):
    if (x%2==0):
        return "Even"
    else:
        return "ODD"
list(map(even, listnum))

# odd numbers:
class odd:
    def __iter__(self):
        self.num = 1
        return self
    def __next__(self):
        num.self = num
        num.self += 2
    def __next__(self):
        num = self.num
        self.num += 2
        return num
```

```
myobj = odd()
myiter = iter(myobj)
x = int(input("Enter a number:"))
for i in myiter:
    if(i<x):
        print(i)
```

```
>>> Enter a number : 15
1
3
5
7
9
11
13
```

Code:

```
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 StopIteration
```


Step10: Define an object of a class.

Step11: Accept an number from the user till which we want to display the odd numbers.

Program for finding factorial of a number using iterable concept in range 1 to 10.

Step1: Define aiter() with arguments initialize the value and return the value.

Step2: Define the next() with an argument and compare the upper limit by using an conditional statement.

Step3: Now create an object of the given class & pass this object in the iter method.

Output:

030

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

19/12/11

Code:

```
while True:
    try:
        x = int(input("Enter class:"))
        break
    except ValueError
        print("Enter Numeric value")
```

Output:

Enter class: 425

Code:

```
try:
    f = open("abc.txt", "w")
    f.write("Akhil Pillai")
except IOError
    print("Error writing on the file")
else:
    print("Operation carried out
          successfully")
    f.close()
```

Output:

Operation carried out successfully.

Practical no: 3

Aim: Program to understand exception handling.

1. write a program using the exception method of the native arithmetic error

→ Step 1: Use the try block and except the input using the raw input method and convert it into the integer datatype and subsequently terminate the block.

Step 2: Use the except block with the exception name as Value error and display the appropriate message if the suspecious code is part of the try block.

2. Write a program for accepting the file in a given mode and use the environment error, as an exception of the given input.

→ Step 1: Within the try block open the file using the write mode and write some content on the file.

Step 2: use the except block with IO error and display the message regarding missing of the file or incompability of the mode use the else block to display a message that the operation is carried out successfully.

3. Write a program using the assert() to check if the list elements are empty.

→ Step 1: Define a function which accepts an argument and check using the assert statement whether the given list is empty list and accordingly return the message.

Step 2: Close the function and in the body of program and define certain elements in list and take some appropriate action.

4. Write a program to check the range of the age of the students in given class and if the age do not fall in given range else the value error exception otherwise return the valid no

→ Step 1: Define a function which will accept the age of the student from the standard input.

Step 2: Use the if condition to check whether the input age falls in the range and so return the age else use the 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 and terminate the looping condition.

Code:

```
def assert_(n):
    assert(len(n)==0)
    print("list is empty")
var = []
print(assert_(var))
```

Output:

list is empty

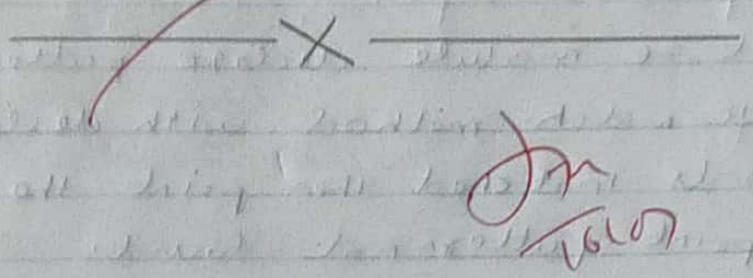
Code:

```
def acceptage():
    age = int(input("Enter age:"))
    if age>30 or age<16:
        raise ValueError
    return age
valid = False
while not valid:
    try:
        age = acceptage()
        valid = True
    except ValueError:
        print("Not a valid age
```

Output:

```
Enter age: 8
Not a valid age
```

Step 4: We except with value Error and print the message not a valid range.



033

Practical no: 4

Topic: Regular expressions.

Step 1: Import re module declare pattern and declare sequence use match method with declare arguments if arguments matched then print the same otherwise print pattern not found.

Step 2: Import re module declare pattern with literal and meta characters declare string value. use the findall() with arguments and print the same.

Step 3: Import the re module declare pattern with meta characters use the spilt() and print the output.

034

```
# match()
import re
pattern = r"FYCS"
sequence = "FYCS represents computer science stream"
if re.match(pattern, sequence):
    print("matched pattern found!")
else:
    print("Not found")
>>> matched pattern found
```

```
#numerical values (segregation)
import re
pattern = r'\d+'
string = 'hello 123, howdy 789, 45 howru'
output = re.findall(pattern, string)
print(output)
>>> ['123', '789', '45']
```

```
# spilt():
import re
pattern = r'\d+'
string = 'hello 123, howdy 789, 45 howru'
output = re.spilt(pattern, string)
print(output)

>>> ['hello', 'howdy', 'howru']
```

no-space:

```
import re
string = 'abc def ghi'
pattern = r'\s+'
replace = ''
v1 = re.sub(pattern, replace, string)
print(v1)
>>> abcdefghi
```

group()

```
import re
sequence = 'python is an interesting language'
v = re.search('^\w Python', sequence)
print (v)
v1 = v.group()
>>> <_sre.SRE_Match object at 0x02810F00>
    Python
```

Verifying the given set of phone no's:

```
import re
list = ['8004567891', '9145673210', '7865432985',
        '9876543208']
for value in list:
  if re.match(r'[8-9]{1} [0,-9]{9}',
      value or len(value) == 10):
          print("Criteria matched for cell
                number")
  else:
      print ("Criteria failed")
>>> Criteria method for cell number
    Criteria matched for cell number
    Criteria failed
    Criteria matched for cell number.
```


Step 4: Import re module, declare string and accordingly declare the pattern replace the blank space with no space. use seek() with 3 arguments and print the string without spaces.

Step 5: Import re module declare a sequence use search method for finding subsequently use the group() with dot operators as search() gives memory location using group() it will show up the matched string.

Step 6: Import re module declare list with numbers. use the conditional statement here we have used up the for condition statement. use if condition for checking first number is either 8 or 9 and next number are in range of 0 to 9 and check whether the entered numbers are equal to 10. If criteria matches print all numbers matches otherwise print failed.

Step 7: Import re module declare a string use the module with findall() for finding the vowels in the string & declare the same.

Step 8: Import re module declare the host and domain name declare pattern for seperating the host & domain name. Use the findall() and print the output respectively.

Step 9: Import re module enter a string use pattern to display only two elements of the particular string use findall() declare two variables with initial value as zero use for condition and subsequently use the if condition check whether condition satisfy add up the or else increment value. And display the values subsequently.

Vowels:

```
import re
str = 'plant is life overall'
output = re.findall(r'\b[aeiouAEIOU]\w+',
    str)
print(output)
>>> ['is','overall']
```

host & domain

```
import re
seq = 'abc.tcsc@edu.com,x42@gmail.com'
pattern = r'[\w\.-]+[\w\.-]'
output = re.findall(pattern, str seq)
print(output)
>>> ['abc.tcsc','edu.com','x42',[gmail\.com']
```

counting of first 2 letters:

```
import re
s = 'mr.a, ms.b, ms.c, mr.t'
p = r'[/ms/mr/] +
o = re.findall(p,s)
print(o)
m=0
f=0
for v in o:
    if (v=='ms'):
        f=f+1
    else:
        m=m+1
print("No of males is:"m)
print("No of females is:",f)
```

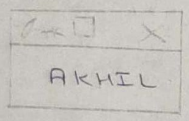
```
>>>['mr','ms','ms',
    'mr']
('No of males is:'2)
('No of females is:',2)
```

• TEXT:

Code:

```
from tkinter import *
root= Tk()
l = Label (root, text ="AKHIL")
l.pack()
root.mainloop()
```

Output:



Practical no: 5

Aim: GUI components.

Step 1: Use the tkinter library for importing the feature of text widget.

Step 2: Create a variable from text method and position it on parent window.

Step 3: Use the pack method along with the object created from text method.

Step 4: Use the mainloop method for triggering of corresponding events.

Step 5: Use the tkinter library for importing the feature of text widget

Step 6: Create a variable from text method and position it onto parent window.

Step 7: Use the pack method along with the variable created from text method and use the parameter.

Step 8: side = LEFT , padx = 30
side = LEFT pady = 40
side = TOP ipadx = 40
side = TOP ipady = 50

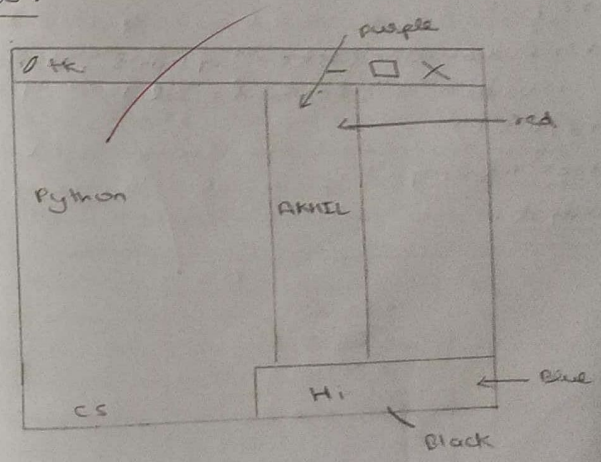
Step 9: Use the mainloop method for triggering of corresponding events.

Step 10: Now repeat the above steps with the label method which takes following arguments
i) Text attribute which defines string
ii) Bg (background) color
iii) Fg (front foreground) color.
iv) Name of parent window
v) Use pack method with the relevant padx attribute.

Code:

```
from tkinter import *
root = Tk()
l = Label(root, text="Python")
l.pack(side = LEFT, pady = 20)
l1 = Label(root, text = "AKHIL", bg="purple
          fg="red")
l1.pack(side = TOP, ipady=40)
m = Label(root, text = "CS")
m.pack(side = LEFT, padx = 20)
m2 = Label(root, text = "Hi", bg="blue",
          fg="black")
m2.pack(side = LEFT, ipadx = 50)
root.mainloop()
```

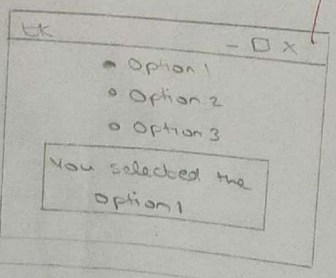
Output:



· Code:

```
from tkinter import *
root = Tk()
def sel():
    selection = "You selected the option" +
                str(var.get())
    l = label.config(root, text = "selection",
                     justify = LEFT)
    l.pack(anchor = S)
    v = IntVar()
r1 = Radiobutton(text = "option1", variable = "v",
     value = '1', command = sel)
r1.pack()
r2 = Radiobutton(text = "option 2", variable = "v",
     value = '2', command = sel)
r2.pack()
r3 = Radiobutton(text = "option 3", variable = "v",
     value = "3", command = sel)
r3.pack()
root.mainloop()
```

Output:



Radiobutton:

write a program making use of control variable and button widget for selection of given Op

step 1: use the tkinter to import relevant method.

step 2: Define a function which tells the user about the given selection need of the multip options available.

step 3: Use the configuration method along with the label object and call the variable as an argument within the method.

step 4: Now define the parent window and define the option using control variable.

step 5: Now create an object from the radio-button method which will take the following arguments.

1) Positioning on parent window
2) Defining the text variable [1, 2, 3, 4]
3) Define the variable argument.
4) Corresponding value and trigger the given function.

Step 6: Pack method for the corresponding radio button objects so created and specify the attribute as an anchor attribute.

Step 7: Now define the label object from the corresponding method and place it on parent window. Subsequently use pack method for this widget and make use of the mainloop method.

Step 8: Import the relevant method from tkinter library.

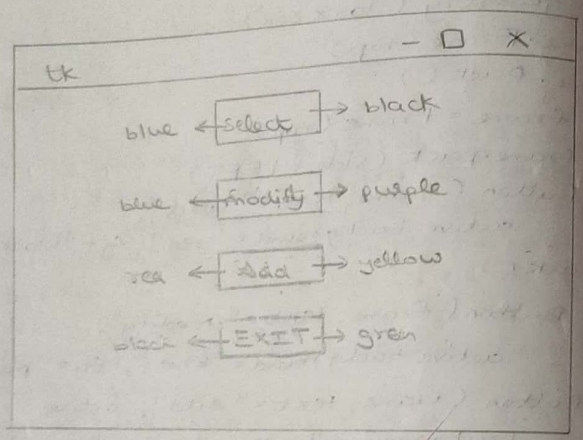
Step 9: Define the object corresponding to the object window and define the size of parent window in terms of no of pixels.

Step 10: Now define the frame object from the method and place it onto the parent window.

. Code:

```
from tkinter import *
top Tk()
top.geometry('100x200')
frame.Frame(top)
frame.pack()
leftframe = Frame(top)
leftframe.pack(side = LEFT)
b1= Button(frame, text = "select",
        activebackground = "red", fg = "black")
b1.pack()
b2.= Button(frame, text = "modify",
        activebackground = "blue", fg = "purple")
b2 = Button(frame, text = "add", active
        background = "yellow", fg = "red")
b3.pack()
b4 = Button(frame, text = "EXIT",
     activebackground = "green", fg = "black")
b4.pack()
top.mainloop()
```

Output:



Step 11: Create another frame object the left frame and put it onto parent window on its left side.

Step 12: Similarly define the right frame and subsequently define the button object placed onto the given frame with the attributes as text active bg & fg.

Step 13: Now we the pack method along with the side attribute.

Step 14: Similarly create the button object corresponding to modify option and put it onto the frame object with side equal to right attribute set.

Step 15: Add another button and put it on right frame object and turn it as EXIT.

Step 16: use the pack method for all the objects and finally use the mainloop method.

Practical no: 5B

Aim: GUI components.

#1:

Step1: Import the relevant methods from the tkinter library create an object with the parent window.

Step2: Use the parent window object along with the geometry() declaring specific pixel size of the parent window.

Step3: Now define a function which tells the user about the given selection made from multiple option available.

Step4: Now define the parent window and define the option with control variable.

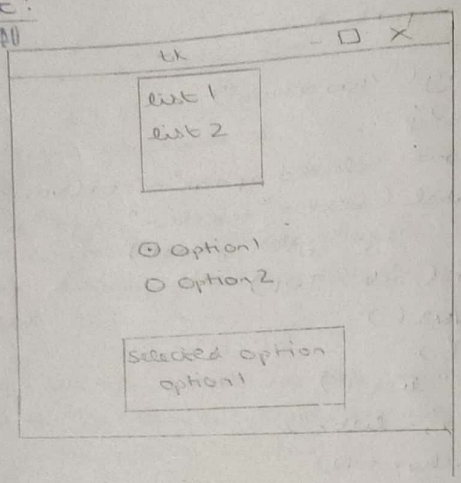
Step5: Use the listbox() and insert options on the parent window along with the pack() with specifying anchor attribute.

Step6: Create an object from radiobutton which will take following arguments (parent window object, text variable which will take the values option no 1, 2, 3... variable arguments, corresponding value & trigger the function) declared.

Radiobutton:


```
from tkinter import *
root=Tk()
root.geometry("400x400")
def select():
    selection="selected option"+str(var.get())
    l=Label(text="selection", bg="white",
            fg="green", root)
    l.pack(side=TOP)
var = stringVar()
l1=listbox()
l1.insert(1,"list1")
l2.insert(2,"list2")
l1.pack(anchor=N)
l2.pack()
r1=Radiobutton(root, text="option1", value=
    "option1", command=select)
r1.pack()
r2=Radiobutton(root, text="option2",
    value="option2", variable=var,
    command=select)
r2.pack()
root.mainloop()
```

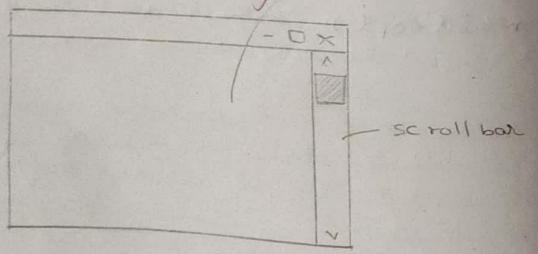
Output:



#2:

```
from tkinter import *
root= Tk()
root.geometry("500x500")
s=scrollbar()
s.pack(side="right", fill="Y")
root.mainloop()
```

Output:



Step 7: Now call the pack() for radio object so created and specify the argument using anchor attribute.

Step 8: Finally make use of mainloop() along with parent window.

#2:

Step 1: Import relevant methods from the tkinter library.

Step 2: Create a parent ~~window~~ object corresponding to parent window.

Step 3: Use the geometry() for laying of the window.

Step 4: Create an object and use the scrollbar

Step 5: Use the pack() along with the scrollbar object with side & fill attribute.

Step 6: Use the mainloop method with the parent window.

#3:

Step 1: Import the relevant libraries from the tkinter method.

Step 2: Create an corresponding object of the parent window.

Step 3: Use the geometry manager with pixel size (680 x 500) or any other suitable pixel value.

Step 4: Use the label widget along with the parent ~~window~~ object created and subsequently use the pack method.

Step 5: Use the frame widget along with the parent object created and use the pack method.

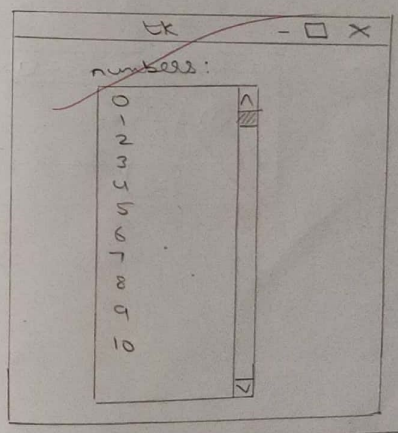
Step 6: Use the listbox method along with the attribute like width, height, font, do create a listbox methods object use pack() for same.

Step 7: Use the Scrollbar() with an object with the attribute of vertical then configure the same with object created from the scrollbar() and use pack().

#3:

```
from tkinter import *
root = Tk()
root.geometry("680x500")
Label(root, text="numbers").pack()
frame = Frame(root)
frame.pack()
listnodes = Listbox(frame, width=20, height=20,
        font=("Times New Roman", 10))
listnodes.pack(side="LEFT", fill="Y")
scrollbar = Scrollbar(frame, orient="vertical")
scrollbar.config(command=listnodes.yview)
scrollbar.pack(side="right", fill="Y")
for x in range(100):
    listnodes.insert(END, str(x))
root.mainloop()
```

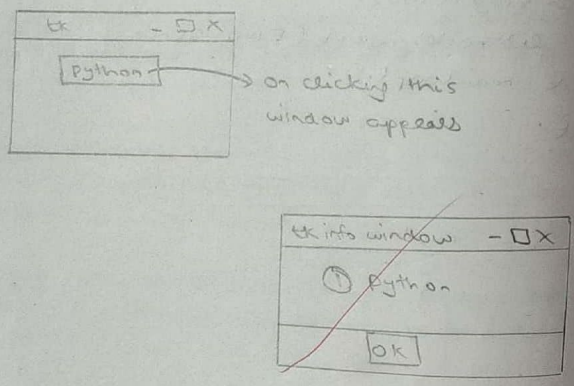
output:



#4:

```
from tkinter import *
import tkmessageBox
root = Tk()
def function():
    tkMessageBox.showinfo("info window",
        "python")
    b1 = Button(root, text = "python",
        command = function)
    b1.pack()
    root.mainloop()
```

Output:



Step 8: Trigger the events using mainloop.

#4:

Step 1: Import relevant methods from tkinter library.

Step 2: Import tkmessagebox.

Step 3: Define a parent window object along with a parent window.

Step 4: Define a function which will use tkmessage box with showinfo method along with info window attribute.

Step 5: Declare a button object with parent window along with the command attribute.

Step 6: Place the button widget onto the parent window and finally call mainloop() for triggering of the event called above.

- Relief style:

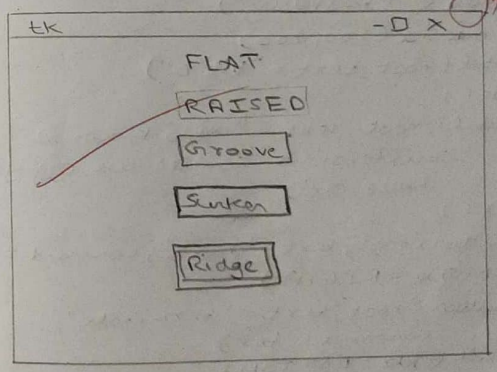
Step 1: Use the button object with the following attributes.
1. The parent window.
2. Text attribute.
3. Relief.

Step 2: Use the corresponding pack method for the respective button objects and trigger the corresponding event.

Step 3: Finally use the mainloop method.


```
from tkinter import *
root = Tk()
b1 = Button(root, text = "FLAT", relief = FLAT)
b1.pack()
b2 = Button(root, text = "RAISED, relief = RAISED)
b2.pack()
b3 = Button(root, text = 'Groove', relief = GROOVE)
b3.pack()
b4 = Button(root, text = 'Sunken', relief = SUNKEN)
b4.pack()
b5 = Button(root, text = 'Ridge', relief = RIDGE)
b5.pack()
root.mainloop()
```

Output:



```
from tkinter import *
root=TK()
root.title("Transversing")
root.maxsize(1000,900)
root.config(bg="red")
leftframe=Frame(root, bg="white", height="400",
    width="200")
leftframe.grid(row=0, column=0)
rightframe=Frame(root, bg="green", height="400",
    width="250")
rightframe.grid(row=0, column=0)
Label(leftframe, text="Image", height=2, width=20
    grid(row=0, column=0)
```

image

```
def main():
    root=TK()
    root.config(bg="red")
    root.title("Transvelling")
    root.minsize(200,200)
    l=Label(root,text="Vit D")
    l.pack()
    l1=Label(root, text="also known as
        calciferol in -sources are egg yolk,
        cheese, etc")
    l1.pack()
    b1=Button(root, text="next", command=SEL)
    b1.pack(side=RIGHT)
    b2=Button(root, text="terminate",
        command=ter)
    b2.pack(side=BOTTOM)
    root.mainloop()
def sel():
    root=TK()
    root.config(bg="blue")
```


Practical no: 5C

Aim: GUI components.

a) Transversing and making use of geometry layout manager method.

Step 1: Define a function and create a object of the given window by using the three methods namely config, title and minsize.

Step 2: Create a button object and use the text and command attribute for triggering the given event and use grid method along with internal and external padding specified similarly create another button object which will allow application to terminate.

Step 3: Define second function corresponding to second window with attributes config, title, minsize for the window object and define one button object which will shift the focus onto the third window.

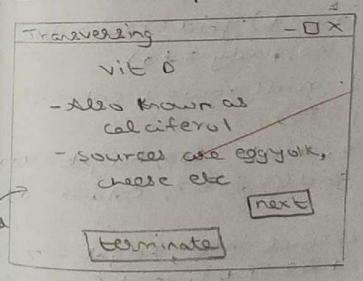
Step 4: Create third window object and in this create two button object for moving on to first window for selecting the process and second button for terminating.

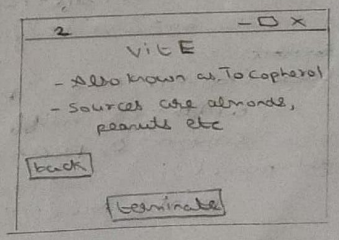
Step 5: Define a function for termination and call the quit method and finally call the first function created and trigger mainloop method.

```
root.title("2")
root.minsize(200,200)
l2 = Label(root, text="vit-E")
l2.pack()
l3 = Label(root, text="- Also known as
      Tocopherol in -sources are almonds,
      peanuts etc)
l3.pack()
b3 = Button(root, text="back", command=
      main)
b3.pack(side=LEFT)
b4 = Button(root, text="terminate",
      command = ter)
b4.pack(side=BOTTOM)
root.mainloop()

def ter():
    quit()

b5 = Button(root, text="know about your
      vitamins", command=main)
b5.pack()
root.mainloop()
```



```
from tkinter import *
root = TK()
root.title("Python")
root.maxsize(1000, 900)
root.config(bg="black")
leftframe = Frame(root, bg="red", height="400",
        width="200")
leftframe.grid(row=0, column=0)
rf = Frame(root, bg="green", height="400",
        width="200")
rf.grid(row=0, column=0)
Label(leftframe, text="Photo", height=2,
   width=20).
   grid(row=0, column=0)
image1 = PhotoImage(file="11.gif")
image1.subsample(1,2)
image2 = PhotoImage(file="2.gif")
image2.subsample(3,4)
Label(leftframe, image=image1).grid(row=0,
        column=0, padx=20, pady=10)
Label(rf, image=image2).grid(row=0, column=
        1, padx=10, pady=10)
toolbar = Frame(leftframe, width=200, height=400
        bg="white").grid(row=2, column=0)
toolbar
Label(toolbar, text="Personal info", height=2,
        width=20, relief=RAISED).grid(row=0
        column=0, padx=20, pady=10)
def none():
    print("None:")
```


5) Displaying the image:

Step 1: Create an object corresponding to the parent window and use the following three methods title, config & maxsize.

Step 2: Create a leftframe object from the frame method and place it onto the parent window with the height, width and bg specified subsequently use the grid method with the row, column, padx, pady specified.

Step 3: Now create a rightframe object from the frame method with the width, height specified and the row and the column value should be specified.

Step 4: Create a label object from the label method and place it onto the leftframe, with text attribute denoting the original image and with relief attribute used as RAISED value and subsequently use grid method with row, column, value specified as (0,0) with some external padding values.

Step 5: Now use the photo image method with the file attribute specified.

Step 6: Use the subsample method with the object of the image and give the x, y co-ordinate values.

Step 7: Use the label method and position it onto the leftframe and placing the image after the sampling and use the grid method for the positioning in the first row.

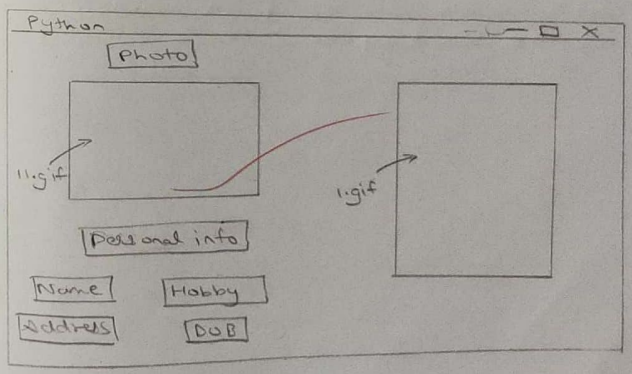
Step 8: Create another label object positioning it onto the rightframe and specifying the image and background attribute with row and column attribute specify it as (0, 0).

Step 9: Now create a toolbox object from the frame method and position it onto the leftframe with the height and width specified and position it onto the second row.

Step 10: Now define the various function for different toolbox options provided in the leftframe.

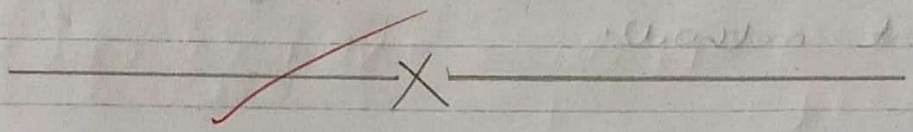
Step 11: From the label method position the text on the toolbar use the relief attribute and corresponding grid value and incorporate the internal padding as well.

```
def hob():
    print("Hobby: ")
def add():
    print("Address: Mumbai")
def bob():
    print("DOB: 29/01/1966")
Button(toolbar, text="Name", height=1,
    width=16, command=name).grid
    (row=1, column=0)
Button(toolbar, text="Hobby", height=1,
    width=16, command=hob).grid(row=1,
    column=1)
Button(toolbar, text="Address", height=1,
    width=16, command=add).grid(row=2,
    column=0)
Button(toolbar, text="DOB", height=1,
    width=16, command=dob).grid(row=2,
    column=1)
root.mainloop()
```



Step 12 : Create the label method position it on to the toolbar with the next title as personal information and position it into same row but next column.

Step 13 : Now make use of mainloop method.



Practical no: 5D

Aim: GUI components

a) Write a program to make use of spinbox widget.

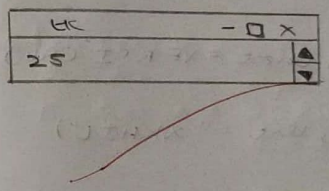
→ Step 1: We the tkinter library to import the relevant methods.

Step 2: Create the parent window object.

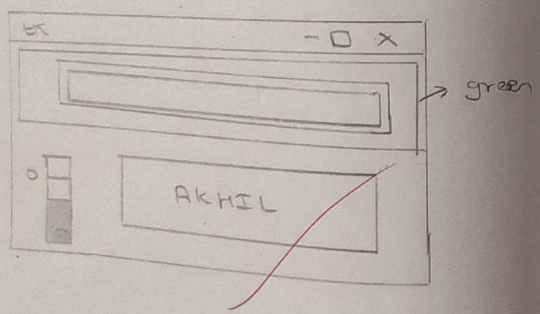
Step 3: Create an object from the .spinbox method and place it onto the parent window with the option specified.

Step 4: Now we the pack method to make the object visible onto the parent window & call the mainloop.

```
from tkinter import *
master = TK()
s = spinbox(master, from_=0, to=25)
s.pack()
master.mainloop()
```

```
from tkinter import *
root = Tk()
m = PanedWindow(bg="green", orient=VERTICAL
        height=100, width=200)
m.pack(fill=BOTH, expand=10)
e = Entry(m, bd=10)
m.add(e)
p = PanedWindow(m, orient=HORIZONTAL)
m.add(p)
t = Scale(p, orient=VERTICAL)
p.add(t)
b = Button(p, text="AKHIL")
p.add(b)
root.mainloop()
```



Paned window:

→ Step 1: Create an object from the pane window method & use the pack method to make this object visible.

Step 2: Now create an object from the entry widget & place it onto the paned window & use the add method.

Step 3: Similarly create an object of the paned window & add it onto the existing window.

Step 4: Create an object from the scale method & place it onto the preceding paned window and use the add method accordingly.

Step 5: Create a button widget and place it onto the paned window & define a functionality along with the button event.

Step 6: Use the pack method & mainloop method for the corresponding events to be triggered.

Canvas:

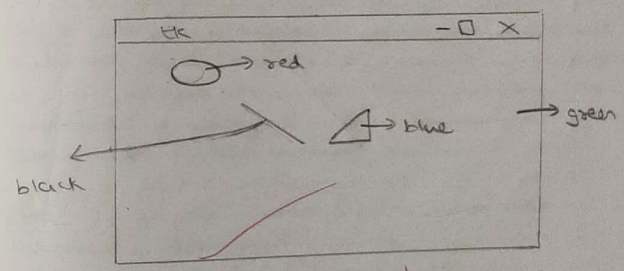
→ Step 1: Create an object from the Canvas widget by using the attribute height, width, background colour and the parent window object.

Step 2: Use the corresponding method for drawing the simple geometrical shapes like arc, oval & line and specify the co-ordinate values.

Step 3: Similarly use the create line and create oval method along with the co-ordinate values and the fill attribute for specifying the colour.

Step 4: Finally use the pack & mainloop method.


```
from tkinter import *
root = TK()
c = Canvas(root, bg="green")
oval = c.create_oval(10, 20, 30, 40, fill=
                     "red")
arc = c.create_arc(70, 30, 90, 100, start=0,
                   extent= 50, fill="blue")
line = c.create_line(50, 60, 70, 90, fill=
                     "black")
c.pack()
root.mainloop()
```



```
>>> import dbm
>>> database = dbm.open("database", flag='c',
                        mode = 438)
>>> database["name"] = "name"
>>> if database["name"] != None:
        print("database not empty//match!")
    else:
        print("database empty !")
>>> match
>>> database.close()
```


Practical no: 6

Aim: Database connectivity.

1. Step 1: Import the (DBM) dbm library & use the open() for creating the database by specifying the name of the database along with the corresponding flag.

Step 2: Use the object so created for accessing the given website & corresponding regular name for the website.

Step 3: Check whether the given web address matches with the regular name of the page is not equal to none than display the message that particular found / match or else not found / unmatched.

Step 4: use the close() to terminate database library

2. Step 1: Import corresponding library to make database connection, OS & SQLite-3.

Step 2: Now create the connection object using SQLite-3 library & the connect() for creating new database.

Step 3: Now create cursor object using the cursor() & from the connection object created.

Step 4: Now use the execute() for creating the table with the column name & respective datatypes.

Step 5: Now with cursor-object use the insert statement for entering the values corresponding to different fields, corresponding the datatype.

Step 6: use the commit() to complete the transaction using the connection object.

Step 7: use execute statement along with cursor-object for accessing the values from the database & using the select from where clause.


```
import os, sqlite3
conn = sqlite3.connect("student.db")
cur = conn.cursor()
cur.execute('create table student(
        Name char, rollno int')
cur.execute("insert into student values
        ("Akhil", 1794)")
cur.execute("insert into student values
        ("Vivek", 1806)")
conn.commit()
cur.execute('select * from student')
cur.fetchall()
cur.close()
```

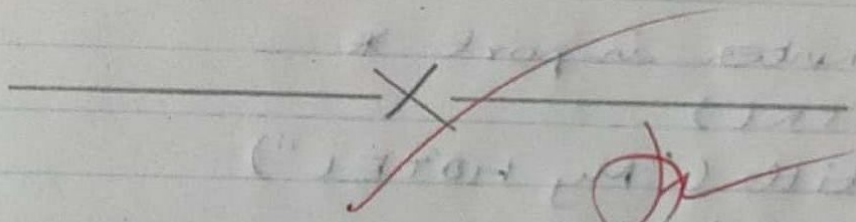
Output:

[("Akhil", 1794), ("Vivek", 1806)]

27/2

Step 8: Finally use the fetch() or fetchall() for displaying the values from the table using the cursor-object.

Step 9: Execute() & drop table syntax for terminating the database & finally use the close().



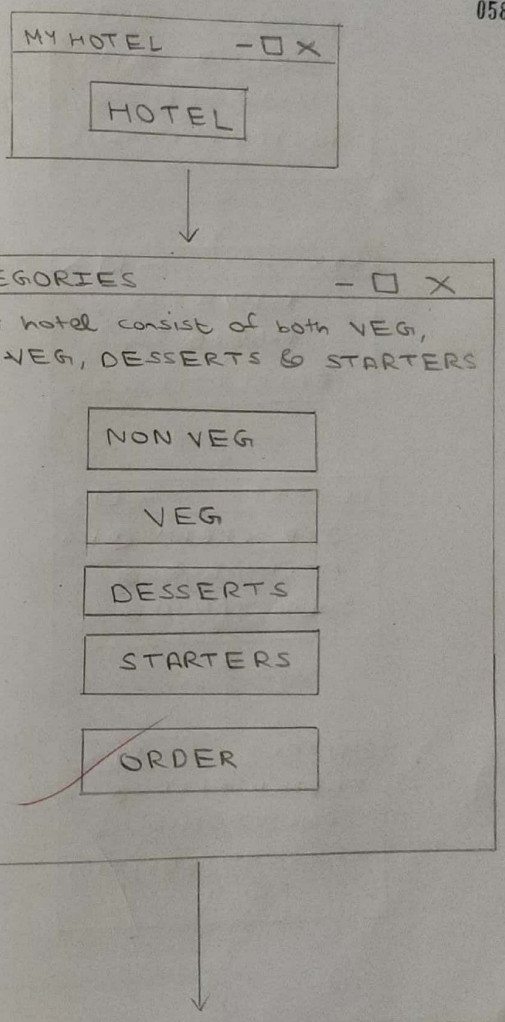
Practical no: 7.

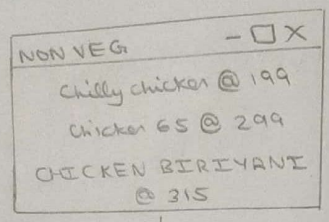
Aim: Project
a) GUI components
b) Database operations

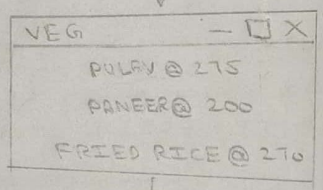
a) GUI components:

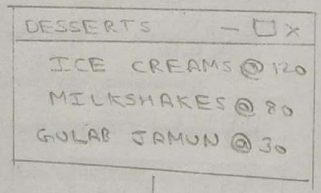
```
from Tkinter import*
root=Tk()
root.title("MY HOTEL")
def categories():
    root1=Tk()
    root1.title("CATEGORIES")
    l=Label(root1,text="Our hotel consist of both VEG,NON VEG,DESSERTS &
STARTERS")
    l.pack(side=TOP)
    b=Button(root1,text="NON VEG",command=nonveg)
    b.pack()
    b1=Button(root1,text="VEG",command=veg)
    b1.pack()
    b2=Button(root1,text="DESSERTS",command=desserts)
    b2.pack()
    b3=Button(root1,text="STARTERS",command=starters)
    b3.pack()
    b4=Button(root1,text="ORDER",command=order)
    b4.pack()
    root1.mainloop()
def nonveg():
    root2=Tk()
    root2.title("NON VEG")
    l=Label(root2,text="Chilly chicken @199")
    l.pack()
    l1=Label(root2,text="Chicken 65 @299")
    l1.pack()
    l2=Label(root2,text="CHICKEN BIRIYANI @315")
    l2.pack()
    root2.mainloop()
```

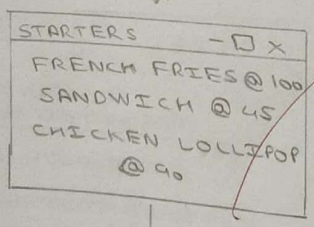
Output:





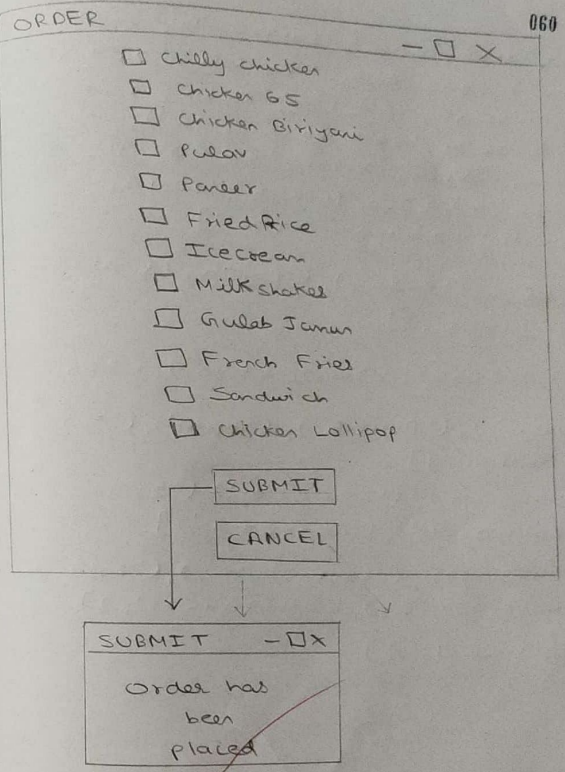







```
def veg():
    root3=Tk()
    root3.title("VEG")
    l3=Label(root3,text="PULAV @275")
    l3.pack()
    l4=Label(root3,text="PANEER @200")
    l4.pack()
    l5=Label(root3,text="FRIED RICE @270")
    l5.pack()
    root3.mainloop()
def desserts():
    root4=Tk()
    root4.title("DESSERTS")
    l6=Label(root4,text="ICE CREAMS @120")
    l6.pack()
    l7=Label(root4,text="MILKSHAKES @80")
    l7.pack()
    l8=Label(root4,text="GULAB JAMUN @30")
    l8.pack()
    root4.mainloop()
def starters():
    root5=Tk()
    root5.title("STARTERS")
    l9=Label(root5,text="FRENCH FRIES @100")
    l9.pack()
    l10=Label(root5,text="SANDWICH @45")
    l10.pack()
    l11=Label(root5,text="CHICKEN LOLLIPOP @90")
    l11.pack()
    root5.mainloop()
def order():
    root6=Tk()
    root6.title("ORDER")
    c=Checkbutton(root6,text="Chilly Chicken")
    c.pack()
    c1=Checkbutton(root6,text="Chicken 65")
    c1.pack()
    c2=Checkbutton(root6,text="Chicken Biriyani")
    c2.pack()
    c3=Checkbutton(root6,text="Pulav")
    c3.pack()
```

```
    c4=Checkbutton(root6,text="Paneer")
    c4.pack()
    c5=Checkbutton(root6,text="Fried Rice")
    c5.pack()
    c6=Checkbutton(root6,text="Icecream")
    c6.pack()
    c7=Checkbutton(root6,text="Milkshakes")
    c7.pack()
    c8=Checkbutton(root6,text="Gulab Jamun")
    c8.pack()
    c9=Checkbutton(root6,text="French Fries")
    c9.pack()
    c10=Checkbutton(root6,text="Sandwich")
    c10.pack()
    c11=Checkbutton(root6,text="Chicken Lollipop")
    c11.pack()
    b5=Button(root6,text="SUBMIT",command=submit)
    b5.pack()
    b6=Button(root6,text="CANCEL",command=terminate)
    b6.pack()
    root6.mainloop()
def submit():
    root7=Tk()
    root7.title("SUBMIT")
    m=Message(root7,text="Order has been placed")
    m.pack()
    root7.mainloop()
def terminate():
    quit()
b=Button(root,text="HOTEL",command=categories)
b.pack()
root.mainloop()
```

b) Database operations

```
import os,sqlite3
connect=sqlite3.connect("Hotel1.db")
cur=connect.cursor()
cur.execute('create table customer1(room_no int(3),Name char(15),address
varchar(10),Intime date,Outtime date,gender char(1))')
cur.execute('insert into customer1 values(101,"Akhil","Vasai","2019-08-
10","2019-08-13","m")')
cur.execute('insert into customer1 values(205,"Anila","Borivali","2019-07-
01","2019-07-08","f")')
cur.execute('insert into customer1 values(310,"Rajesh","Bandra","2019-06-
19","2019-06-25","m")')
cur.execute('insert into customer1 values(440,"Akhila","Virar","2019-05-
11","2019-05-25","f")')
cur.execute('insert into customer1 values(320,"Nishant","Andheri","2019-04-
18","2019-04-28","m")')
cur.execute('select * from customer1')
cur.fetchall()
cur.execute('select Name from customer1')
cur.fetchall()
cur.execute('select room_no from customer1')
cur.fetchall()
cur.execute('select address from customer1')
cur.fetchall()
cur.execute('select Intime from customer1')
cur.fetchall()
cur.execute('select Outtime from customer1')
cur.fetchall()
cur.execute('select Outtime from customer1 where room_no=101')
cur.fetchall()
cur.execute('update customer1 set room_no=103')
cur.fetchall()
cur.close()
```

Output

```
[(101,'Akhil','Vasai','2019-08-10','2019-08-13',
'm'),(205,'Anila','Borivali','2019-07-01',
'2019-07-08','f'),(310,'Rajesh','Bandra',
'2019-06-19','2019-06-25','m'),
(440,'Akhila','Virar','2019-05-11','2019-05
-25',f),(320,'Nishant','Andheri','2019-
04-18','2019-04-28','m')]

[('Akhil',),('Anila',),('Rajesh',)('Akhila',),
('Nishant',)]

[(101,),(205,),(310,),(440,),(320,)]

[('Vasai',),('Borivali'),('Bandra'),('Virar'),
('Andheri',)]

[('2019-08-10'),('2019-07-01'),('2019-06-19'),
('2019-05-11'),('2019-04-18',)]

[('2019-08-13'),('2019-07-08',),('2019-06-25'),
('2019-05-25',),('2019-04-28',)]

[('2019-08-13',)]
```