4/5/24 UNIT-I OOP, GUI & Database:-

Principles of OOP:

1. Object:
2. Class
3. Inheritance
4. Polymorphism
5. Data Abstraction
6. Data Encapsulation
7. Dynamic Binding
8. Message passing.

1. object: Object is a runtime entity or real world entity & it is a collection of data members and functions.

2. class: class is a collection of objects with with similar features. There are no access modifiers in python.

Syntax: class class_name: (or) class class_name():

```
    instance_variable
    class_variables
    def function_name(self): #instance method
        #statements.  we can access through object
        #statements.
    def function_name(): # class method.
        #statements   we can access through class name.
```

Object creation:

```
obj_name = class_name()
```

Ex:

```
class Student:
    def getData(self):
        self.no = int(input("enter roll number"))
        self.name = input("enter name:")
        self.address = input("enter address:")
```

If we write self.no then this variable is instance variable. without self it is local variable.

→ difference b/w instance method and class method?

access only through object_name and class_name

```
    def putData(self):
        print("Number is:", self.no)
        print("Name is:", self.name)
        print("Address is:", self.address)
s = student()
s.getData()
s.putData()
```

→ A function declared without self parameter then it is called class

```
ex: class Student:
        def display():
            print("I am display")
student.display()
```

It is called static method in java.

s = student
s.display() ✗

Constructors in python:

1) default constructor: A constructor without parameters is called as default constructor.

Rules:

1) A constructor contains __init()__ method.

In python a function begin with '__' double under score is called magic function. or dunder

2) no return type (constructor doesn't allow return type)

3) no calling for constructors. (they are invoked at the time of object creation)

```
ex:
class student:
    def __init__(self):
        self.no = int(input("enter roll number"))
        self.name = input("enter name")
    def putdata(self):
        print("Number is:", self.no)
        print("Name is:", self.name)
s = student()
s.putdata()
```

2)
Parameterized constructor: A constructor which contains parameters is called as parameterized constructor

Ex:

```
class student:
    def __init__(self, no, name, address):
        self.no=no
        self.name=name
        self.address = address
    def putData(self):
        print("Number is:", self.no)
        print("Name is:", self.name)
        print("Address is:", self.address)
a=int(input("enter roll no:"))
b= input("enter name")
c = input("enter address")
s= student(a,b,c)
s.putdata()
```

3) Inheritance in python:

A class is derived from existing class is called as Inheritance. The main advantage of inheritance is code reusability.

Types of inheritance:

1) Single or simple inheritance
2) Multiple inheritance
3) Heirarchical inheritance
4) Multi-level inheritance
5) Multi-path inheritance
6) Hybrid inheritance.

→ only c++ & python supports all inheritances.

```
class A:
    def display1(self):
        print("A")
class B(A):
    def display2(self):
        print("B")
r=B()
r.display1()
r.display2()
```

19/5/24 Single inheritance:

```
Class A:
    def getData(self):
        self.no=input("enter number")
        self.name=input("enter name")
        self.address=input("enter address")
    def getMarks(self):
        self.m=[i for i in map(int,input('enter 6 subject marks')
                                .split())]
class B(A):
    def percentage(self):
        self.p=sum(self.m)/600*100
        print("percentage is:",self.p)
s=B()
s.getData()
s.getMarks()
s.percentage()
```

93 92 99 96 94 93

Polymorphism:

one name many forms

There are two types of polymorphism

i) compile time polymorphism

ii) Runtime polymorphism.

Compiletime polymorphism:

a) method overloading
b) operator overloading } also supported in C++.

Runtime polymorphism:

a) method overriding
b) ABC classes (Abstract base classes)
c) Dynamic method Dispatch.

Compile time polymorphism:

- Python doesn't support directly the method overloading concepts.
- In method overloading same method with different parameters are executed by using default argument concepts.
- Python doesn't allow function name with more than one time.

Ex:

```
class A:
    def display(self, a=5, b=6, c=7):
        print(a,b,c)

a=A()
a.display() # 5,6,7
a.display(1,2) # 1,2,7
a.display(10) # 10,6,7
a.display(2,3,4) # 2,3,4
```

How to configure Jupyter notebook?

cmd
=> pip install Jupyter.

check: open IDLE

file -> save -> location

Run -> paste -> scripts

scripts-paste -> In cmd

```
cd 'Local settings"
cd
cd App Data
cd Local
cd programs
cd python
 cd python 37
  cd scripts
  pip install jupyter.
```

pip update : python -m pip install --upgrade pip.

-> My computer> advanced system settings > environment variables> paste path.

Run
Jupyter notebook

Jupyter > new > python3

How to install pandas:

command prompt: pip install pandas
pip install openpyxl.

Run: excel.

create an excel file with following information.

Marks

save as data.xls

Program:

```
import pandas as p
d= p.read_excel("e:/data.xlsx")
print("Mean is", d[marks].mean())
print("standard deviation is:", d['Marks'].std())
```

LAB - 20:

```
class WeCare:
    __id=None
    __type=None
    __cost=None
    __premium=None
    def set__id(self,a):
        self.__id=a
    def set__type(self,b):
        self.__type=b
    def set__cost(self,c):
        self.__cost=c
    def set__premium(self):
        if self.__type="Two Wheeler":
            self.__premium=self.__cost*2/100
        elif self.__type="four Wheeler":
            self.__premium=self.__cost*6/100
        else:
            self.__premium="Invalid"
    def get__id(self):
        return self.__id
    def get__cost(self):
        return self.__cost
    def get__type(self):
        return self.__type
    def get__premium(self):
        return self.__premium
r=WeCare()
int=int(input("enter id of the vehicle"))
t=input("enter type of vehicle")
c=int(input("enter cost"))
```

```
r.set_id(i)
r.set_type(t)
r.set_cost(c)
r.set_premium()
print("vehicle id is:",r.get_id())
print("vehicle type is:",r.get_type())
print("vehicle cost is:",r.get_cost())
print("vehicle premium is:",r.get_premium())
```

6/8/24

Tkinter Widgets:

Tkinter Module is used to design Graphical User Applications in python.

Tkinter contains set of controls to maintains GUI applications in the form of widget (it is a form control)

List of Widgets:

1. Label
2. Entry Text field
3. Button
4. Checkbutton
5. Radiobutton
6. Text
7. Menu
8. Spinbox
9. Combobox

All these widgets are resides in 'Master'. pack() method packs this widget into the master.

mainloop() – writes in end of program – until user close the applet window and runs in loop continuously

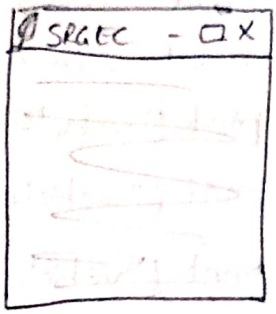
bg
fg
font

geometry() method used to specify width and height of tkinter.

title()
config()

1. Design a GUI to display red colour tkinter window.

```
from tkinter import
t= Tkl() # master → predefined method (master)
t.geometry('500x500')
t.title('SRGEC')
t.config(bg='red')
t.mainloop()
```



→ A tkinter program contains any no. of masters.

2. Design GUI application to display four labels CSE, IT, AIDS, AIML

```
from tkinter import *
t=Tkl()
t.geometry('500x500')
t.title("SRGEC")
t.config(bg='red')
m1=Label(t,text="CSE",bg='green')
m2=Label(t,text="IT")
m3=Label(t,text="AIDS")
m4=Label(t,text="AI&ML")
m1.packl()
m2.packl()
m3.packl()
m4.packl()
t.mainloop()
```

by default center.

Label is a passive control

(or)

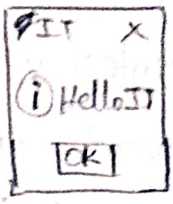
```
m1.pack(side=LEFT)
m2.pack(side=RIGHT)
m3.pack(side=TOP)
m4.pack(side=BOTTOM)
```

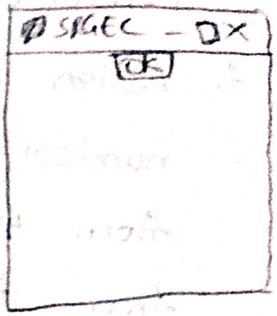
Button Control/widget:

Button is a active widget & active control used to perform user click events.

Ex: tkinter 1) How to display a message box in tkinter window?

```
from import *
from tkinter import message-box
t=Tk() #master
t.geometry('500x500')
t.title('SRGEC')
t.config(bg='red')
def display():
    messagebox.showinfo("IT","HelloIT")
    print("hello IT")
    t.config(bg='yellow')
b=Button(t,text="ok", command=display)
b.pack()
t.mainloop().
```





Entry widget: Entry is nothing but textfield in Java.

Design tkinter window to find out factorial of a given number, read the input through textbox.

```
from tkinter import *
from tkinter import messagebox
t=Tk() #master
t.geometry('500x500')
t.title("factorial")
t.config(bg='red')
m=Label(t,text="Enter an Integer")
x=StringVar()
t1=Entry(t, textVariable=x)

def factorial():
    f=1
    for i in range(1,int(x.get())+1):
        f=f*i
    messagebox.showinfo("factorial is", str(f))
```

```
b=Button(t,text="ok", command=factorial)
m.pack()
t1.pack()
b.pack()
t.mainloop()
```

2. Design tkinter window to findout reverse of a given number, read input through textbox.

```
from tkinter import *
from tkinter import messagebox
 t=Tk()
t.geometry('500x500')
t.title('Reverse')
t.config(bg='red')
m=Label(t, text='enter an integer')
x=StringVar()
t1=Entry(t,textvariable=x)
 def reverse():
n=int(x.get())
        s=0
        while n>0:
            s=s*10+n%10
            n=n//10
        messagebox.showinfo("Reverse is",str(s))
b=Button(t,text="ok", command=reverse)
m.pack()
t1.pack()
b.pack()
t.mainloop().
```

messagebox.showinfo('reverse','reverse is {}'.format(str(s)))

3. Design GUI to validate given username and password.
(default username is admin, and password is 1234).
user user@123.

```
from tkinter import *
from tkinter import messagebox
t=Tk()
t.geometry('500X500')
t.title("User Information")
t.config(bg='red')
x=StringVar()  → data type in tkinter.
y=StringVar()
m1=Label(t, text="enter user name")
m2=Label(t, text="Enter password")
t1=Entry(t, textvariable=x)
t2=Entry(t, textVariable=y, show="*")
def validate():
    u=x.get() #read text from t1
    p=y.get()
    if(u=='admin' and p=="1234"):
        messagebox.showinfo("Given user and id",
                            u+'\n'+p)
    else:
        messagebox.showinfo("Invalid user/password")
b=Button(t,text="OK", command=validate)
m1.pack()
t1.pack()
m2.pack()
t2.pack()
b.pack()
t.mainloop()
```

7/6/24

CRUD- Create, Retrieval, Update, Delete
Read

LAB PROGRAM -22

? place holder

Create:

```
from sqlite3 import *
from tkinter import *
from tkinter import messagebox
t=Tk()
t.geometry('500X500')
t.title("Student Information")
t.config(bg='red')
m1=Label(t, text="Enter Roll No")
m2=Label(t, text="Enter name")
m3=Label(t, text="enter address")
x=StringVar()
y=StringVar()
z=StringVar()
t1=Entry(t, textvariable=x)
t2=Entry(t, textvariable=y)
t3=Entry(t, textvariable=z)
def upload():
    con=connect("gec.db")
    c=con.cursor()
    c.execute('insert into student values(?,?,?)',
              (x.get(), y.get(), z.get()))
    con.commit()
    con.close()
    messagebox.showinfo("record inserted")
b=Button(t, text="ok", command=upload)
m1.pack()
t1.pack()
m2.pack()
t2.pack()
m3.pack()
t3.pack()
b.pack()
t.mainloop()
```

create:

```
from sqlite3 import *
con=connect("gec.db")
c=con.cursor()
c.execute("create table student(no text, name text,
                                address text)")
con.commit()
con.close()
print('table created')
```

select:

fetchall()
fetchmany()
fetchone()

```
from sqlite3 import *
con=connect("gec.db")
c=con.cursor()
#c.execute("create table student(no te
z=c.execute("select * from student")
for i in z.fetchall():
        print("No is:", i[0])
        print("Name is:", i[1])
        print("address is:", i[2])
con.commit()
con.close()
```

1. Design GUI to perform arithmetic operations addition(+) subtraction(-), multiplication(*) and division(/).. read two values through textbox and perform these operations by using 4 button controls.

```
from tkinter import *
from tkinter import messagebox
t=Tk()
t.geometry('500x500')
t.title('Arithmetic operations')
t.config(bg='green')
m1=Label(t, text='enter 1st integer')
m2=Label(t, text='enter 2nd integer')
```

```
x = StringVar()
y = StringVar()
t1 = Entry(t, textvariable=x)
t2 = Entry(t, textvariable=y)
def addition():
    messagebox.showinfo("Addition", "Addition is"+
        str(int(x.get())+int(y.get())))
def subtraction():
    messagebox.showinfo("Subtraction", "Subtraction is"
        +str(int(x.get())-int(y.get())))
def multiplication():
    messagebox.showinfo("Multiplication", "Multiplication
        is"+str(int(x.get())*int(y.get())))
def division():
    messagebox.showinfo("Division", "Division is"
        +str(int(x.get())/int(y.get())))
b1 = Button(t, text="ADD", command=addition)
b2 = Button(t, text="SUBTRACT", command=subtraction
b3 = Button(t, text="MULTIPLY", command=multiplicatio
b4 = Button(t, text="DIVISION", command=division)
m1.pack()
t1.pack()
m2.pack()
t2.pack()
b1.pack()
b2.pack()
b3.pack()
```

by.pack()

t.mainloop()

Check Button:

```
from tkinter import *
from tkinter import messagebox
t=Tk()
t.geometry('300x400')
t.title('checkbutton demo...')
t.config(bg='cyan')
x=IntVar()
y=IntVar()
z=IntVar()
c1=Checkbutton(t,text="CSE",variable=x,command=display1)
c2=Checkbutton(t,text="IT",variable=y,command=display2)
c3=Checkbutton(t,text="ECE",variable=x,command=display3)
c1.pack()
c2.pack()
c3.pack()
t.mainloop()
```

```
def display1():
    if x.get()==1:
        messagebox.showinfo("","CSE")
def display2():
    if y.get()==1:
        messagebox.showinfo("","IT")
def display3():
    if z.get()==1:
        messagebox.showinfo("","ECE")
```

Radio Button: Radio button is used to select only one option.

```
from tkinter import *
from tkinter import messagebox
t=Tk()
t.geometry('500x500')
t.title('RadioButton Demo')
t.config(bg='black')
x=IntVar()
r1=RadioButton(t,text="RED",variable=x,value=1,command=display)
r2=RadioButton(t,text="GREEN",variable=x,value=2,command=display)
r3=RadioButton(t,text="YELLOW",variable=x,value=3,command=display)
```

```
def display():
    if x.get()==1:
        t.config(bg='red')
        r1.config(bg='magenta')
    elif x.get()==2:
        t.config(bg='green')
    elif x.get()==3:
        t.config(bg='yellow')
```

```
r1.pack()
r2.pack()
r3.pack()
t.mainloop()
```

Combobox widget in tkinter:

Combobox is a combination of textbox and dropdownlist.

Eg:

```
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
t=Tk()
t.geometry('500x500')
t.title('combobox demo ....')
t.config(bg='red')
v=['22481A1200', '22481A1300', '22481A1400', '22481A1500']
y=StringVar()
c=ttk.Combobox(t,values=v, textvariable=y)
b=Button(t,text='ok', command=display)
def display():
    messagebox.showinfo("Roll Number", y.get())
c.pack()
b.pack()
mainloop()
```

2.
```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
t=Tk()
t.geometry('500x500')
t.title("combobox demo...")
t.config(bg='yellow')
v=['IT','CSE','EEE','ECE','IOT'] t1=StringVar()
c=ttk.combobox(t,values=v,textvariable=t1)
def disp(event):
    messagebox.showinfo("selected value","""+t1.get())
c.bind("<<combobox selected>>",disp)
c.pack()
t.mainloop().
```

Spinbox: It is an alternative of textbox control.

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox.
t=Tk()
t.geometry('300x600')
t.title("combobox demo..")
t.config(bg='yellow')
s=spinbox(t,from=500,to=10000)
s.pack()
t.mainloop()
```

Text:

It is also a entry control. The information in the form of multiple rows.

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
t=Tk()
t.geometry('300X600')
t.title('combobox Demo...')
t.config(bg='yellow')
t=Text(t,width=20,height=10)
s='''This is
     python program'''
t.insert(END,s)
t.pack()
t.mainloop().
```

Menus:

```
from tkinter import *
from tkinter import messagebox
t=Tk()
t.geometry('500X500')
t.title("Menu demo....")
t.config(bg='red')
m=Menu(t)
f=Menu(m)
f.add_command(label="New")
f.add_command(label="Save",accelerator="ctrl+s")
f.add_command(label="Save As")
f.add_separator()
f.add_command(label="print")
```

```
m.add_cascade(label="File", menu=f)
t.config(menu=m)
t.mainloop()
```