24-02-2025 Training Day – 18

Multiple Inheritance: One Child Multiple Parents # class C1:

# def init (self):

# self.a=1

# self.b=2

# # def showData(self):

# # print("I am in Class C1") # class C2:

# def init (self):

# self.c=3

# self.d=4

# # def showData(self):

# # print("I am in Class C2") # class C3:

# def init (self):

# self.e=5

# self.f=6

# def showData(self):

# print("I am in Class C3") # class C4(C1,C2,C3):

# def init (self):

# self.g=7

# self.h=8

# super(). init ()

# C2. init (self)

# C3. init (self) # # def showData(self):

# # print("I am in Class C4")

# obj=C4() #object of C4. obj. init () # obj.showData()

# print(obj.a,obj.b,obj.c,obj.d,obj.e,obj.f,obj.g,obj.h)

"""

In above program, if we have showData method in all 3 parents but not in child. Now if we call showData method using child object then it may create an ambiguity or in Java it creates ambiguity that which parent showData method will be called? So this ambiguity in Java is called Diamond Problem and there is no direct solution made for diamond problem in Java ie Java doesn't support Multiple Inheritance. But in Python this a solution is made for this ambiguity using Priority setting and making MRO (Method Resolution Order) If there are multiple parents of one child ie C1, C2, C3 then direct priority will be given from left to right as we mention parents name in child class.

Now in above program as we have multiple parents for one child so to create variables of all parents, we need to call constructor of all the parents in child class.

# # Hybrid Inheritance:

# Case 1 as shared in painting:

# """

# class C1:

# def init (self):

# self.a=1

# self.b=2

# def showData(self):

# print("I am in Class C1") # class C2(C1):

# def init (self):

# self.c=3

# self.d=4

# super(). init () # # def showData(self):

# # print("I am in Class C2") # class C3(C1):

# def init (self):

# self.e=5

# self.f=6

# super(). init () # # def showData(self):

# # print("I am in Class C3") # class C4(C2,C3):

# def init (self):

# self.g=7

# self.h=8

# super(). init ()

# C3. init (self) # # def showData(self):

# # print("I am in Class C4") # obj=C4()

# print(obj.a,obj.b,obj.c,obj.d,obj.e,obj.f,obj.g,obj.h) # obj.showData()

# Polymorphism:

**Overloading:** is also called compile time polymorphism

**Overriding:** is also called run time polymorphism

Real definition of access specifiers as per OOPS concepts used in C++ or Java or other:

***Public:*** Can be accessed anywhere in itself or child or outside class

***Protected***: Can be accessed only in itself or in child class

***Private:*** Can be accessed only in itself.

.

25-02-2025 Training Day – 19

**Exception Handling** It is possible to write programs that handle selected exceptions. Look at the following example, which asks the user for input until a valid integer has been entered, but allows the user to interrupt the program (using Control-C or whatever the operating system supports); note that a user-generated interruption is signalled by raising

the KeyboardInterrupt exception

# class Count:

# object\_count = 0 # def init (self):

# Count.object\_count+= 1 # # classmethod

# def my\_object\_count(cls):

# return cls.object\_count # #PL

# obj1 =Count() # obj2 =Count() # obj3 =Count() # obj4 =Count()

# print(Count.object\_count)

# #New Program

# x=5 #x address 1000 # class x: #x address 2000 # pass

# print(x)

**Exception Handling:** How to handle exceptions (errors) In real life projects, if we get errors while program execution then program doesn't stop working or doesn't throw the error at the output rather some error message is given and projects keep on running.

***The target of exception handling is:*** Catching/Managing the errors while program execution and try not to stop the program.

"""

# #New Program

# id\_list=[10,20,30,40]

# id=int(input("Enter the ID:")) #id=50 # i=id\_list.index(id)

# print(i)

## # import Module12 #ModuleNotFoundError: No module named 'Module12'

To handle exceptions in the program there are 4 keywords designed for exception handling.

**try: except**:

**finally:**

**raise:**

"""

# #New Program

# x=int(input("Enter First No:")) #ValueError: invalid literal for int() with base 10: 'cetpa' # y=int(input("Enter Second No:"))

# res=x/y #ZeroDivisionError: division by zero # print("Result:",res)

# #New Program # while(1):

# try:

# x=int(input("Enter First No:")) #ValueError: invalid literal for int() with base 10: 'cetpa'

# y=int(input("Enter Second No:"))

# res=x/y #ZeroDivisionError: division by zero # print("Result:",res)

# break

# except:

# print("Error!")

RAISE KEYWORD:

To raise ie to throw the errors/exceptions intentionally in the program

Syntax:

raise ErrorClass\_Name(Message for user) """

# raise ValueError("Error!")

# #New Program # #BLL

# import math # def add(a,b):

# return a+b # def sub(a,b):

# return a-b # def mul(a,b):

# return a\*b # def div(a,b):

# return a/b # def pow(a,b):

# return a\*\*b #

# #PL

# while (1):

no2 = int(input("Enter Second No:"))

# choice = input("Enter Any operation +,-,\*,/,pow,log:") # if (choice == "+"):

# res = add(no1, no2)

# print("Result:", res) # elif (choice == "-"):

# res = sub(no1, no2)

# print("Result:", res)

# elif (choice == "\*"):

# res = mul(no1, no2)

# print("Result:", res) # elif (choice == "/"):

# res = div(no1, no2)

# print("Result:", res) # elif (choice == "pow"):

# res = pow(no1, no2)

# print("Result:", res) # elif (choice == "log"):

# res = math.log(no1, no2)

# print("Result:", res) # else:

# raise NotImplementedError("Incorrect Choice") # except Exception as err:

# print("Error!",err)

# print(type(err))

27-02-2025 Training Day – 20

Libraries In Python where the name of library given at installation time is different and while importing, the library name is different. Sometimes new versions of libraries comes with new names.

import speech\_recognition

**For installation library name is:** SpeechRecognition import speech\_recognition

### DATABASE:

Types of databases: multiple types but two are majorly used:

1. Relational Database
2. Non-relational Database
3. **Relational Database:** data is stored in the form of tables and tables are connected through some keys.

Examples: MySQL, Oracle, MS-SQL Server, MS-Access, PostGre-SQL

**2 .Non-relational** database: No-sql databases. Data is available in multiple structures

Examples: MongoDB, Cassandra Server...

**Relational databases:** MySQL Database For installation: CETPA Video

There are two options to works on database:

1. Directly install the database and start writing the queries.
2. We can install LAMP, WAMP or XAMP Server on our machine and then use database: Here you can work in

**Database Connectivity**: ie we run the query in python and it should be executed in database

**For Database Connectivity**: We need to have

1. IP Address: If database is installed in our machine:

Local IP Address of each machine: 127.0.0.1 and this address is called 'localhost'

1. User Id: ie user name user=root
2. Password:

pwd=root123

1. Database name: db1 (Can be given through queries)
2. Port: 3306 (Optional) """

28-02-2025 Training Day – 21

GUI PROGRAMMING

* + Pack
  + Place
  + Grid geometry won't work together. Grid is the best geometry to use

……New Program……. # import tkinter as tk

# def leftClick():

# print("I am Clicked") # root=tk.Tk()

# root.geometry("300x400")

# btn1=tk.Button(root,text="ABCD",font=1,bg="Red",fg="Yellow",command=leftClick) # btn1.grid(row=0,column=0)

# root.mainloop()

print function in python is used to print the data on console window. Label widget in tkinter is used to display the data on GUI screen

……New Program……. # import tkinter as tk

# root=tk.Tk()

# root.geometry("300x400") lbl\_id=tk.Label(root,text="Enter Cust ID:",font=1) lbl\_id.grid(row=0,column=0) lbl\_name=tk.Label(root,text="Enter Cust Name:",font=1) lbl\_name.grid(row=1,column=0)

root.mainloop()

**Entry widget:** To take input from GUI screen in single line.

Like input function, we take the data in some variable, similarly in Entry widget we will input the data in some variable. Till now in Python we have discussed, that variables are created in python by assigning the value. Variables created through input functions will always be of type

string. But in GUI tkinter programming, the variables are not automatically created by assigning the value. Here also like in C Lang or Java, we need to firstly define the variable type.

# L=list() #Empty List

……New Program……. import tkinter as tk

def data\_capture(): id=var\_id.get() print(id) var\_id.set("")

root=tk.Tk()

root.geometry("400x500") lbl\_id=tk.Label(root,text="Enter Cust Id:",font=1) lbl\_id.grid(row=0,column=0) var\_id=tk.StringVar() entry\_id=tk.Entry(root,textvariable=var\_id,font=1) entry\_id.grid(row=0,column=1)

btn\_submit=tk.Button(root,text="Submit",font=1,command=data\_capture) btn\_submit.grid(row=1,column=1)

root.mainloop()