14-10-2024

Training Day – 20

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):
#     try:
#         no1 = int(input("Enter First No:"))
#        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))
```