10-14-2024 Python || Exception Handling

Types of Errors

1. Syntax Error (Compile Time Error)

```
printf('teshello world') ## Will throw an error since printf is not a valid function
```

2. Logical Error

These are the types of exceptions which occur during the run time of the program based on the values that are being processed by the program. These Exceptions can be handled or managed by a process called as Exception Handling.

```
i = int(input("Enter a Number: "))
j = int(input("Enter a Another Number: "))
print("i/j = ",i/j) ## will throw an error if we use a string instead of number
print("Hello Exception Handling")
```

Exception Handling

We can use *try:* & *except exceptionName:* to handle these errors. We use exception handling to make sure that the program continues to run the lines after the one that caused the exception.

```
try:
    i = int(input("Enter a Number: "))
    j = int(input("Enter a Another Number: "))
    print("i/j = ",i/j) ## will throw an error if we use a string instead of number

except ZeroDivisionError:
    print("Trying to Divide by Zero")
    print("Numbers cant be divided by 0")

except ValueError:
    print("Please enter valid numbers")

finally:
    print("Program Executed Successfully")
```

The Finally Keyword Can be used to execute lines of code after try: and except: finish executing

```
try:
    i = int(input("Enter a Number: "))
    j = int(input("Enter a Another Number: "))
    print("i/j = ",i/j) ## will throw an error if we use a string instead of number
except:
    print("Ran into an Exception")
finally:
    print("finally")
```

User Defined Exceptions:

We can create Custom Exception by creating a class that inherits from the Exception base class.

```
class MyError(Exception):
    pass

class ZeroNotDivisible(MyError):
    pass

try:
    i = int(input("Enter a Number: "))

    if(i==0):
        raise ZeroNotDivisible

    else:
        print(i)

except ZeroNotDivisible:
        print("Enter a valid value")
```