26/12/2020

* Exception handling :- Exception are the blockage in a flow of Executing the program

* Exception are the upstrackles which will create a blockage during a the Executing of any programm

* When ever an Exception has been arrised the Controller Stops the Execution and in python apart from Syntax Error, Every type of error is exception.

The occured Exc The Exception will get occured because of invalid input by the user usually.

(i) The occured Exp Exception is needed to be Solved or handeled to Executive the file with no blockage, the occured Exception can be Solved or handled with an alternate Solution

(ii) The process to handel any Exception or error is known as Exception handling phenomeanon.

In python we can handel any Exception with the help of these two statements.

(i) ~~Jav~~ try:

(ii) except:

* try block :- it is a block which will store all those statements or program or code which is having the possiblity to raise an Exception.

Envoke

* Except block :- It is a block which acts as a solution for those and all Exception's which was been occured in a try block.

We can have the Except block of three types

* Specific..Expect block
* generic except block
* default except block.

=> Specific . Expect block :- It is a type of block which can act as a solution for the Specified Exception . which is occured in the try block

Syntax :-

try:

Except Name of Exception:

```
print ('start')
try:
    a = int (input ('Enter a val : '))
    b = int (input ('Enter b val : '))
    c = a/b
    print ('Div is : ', c)
```

```
Except ZeroDivisonError:
    print('val of b is Zero.....')

Except ValueError:
    print('Improper input Value')

print('End')
```

o/p...

Eg① Start
Enter a Val : hello
Improper inputs Vals
End

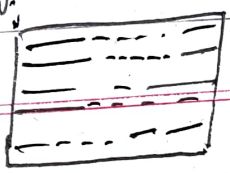
Eg② Start
Enter a Val : 10
Enter b Val : 0
Val of b is Zer ....
End

* Generic Except block :- it is a type of except block which works as a solution for any type of Exception which is occured in a try block.

* Generic Except block will not work as a solution for any hardware base Exception
Eg :- Keyboard interupt, io error, os error :

Syntax :-

try:



Except Exception :

```
print ('start')
try:
    a = int (input) ('Enter a val :'))
    b = int (input) ('Enter b val :'))
    c = a/b
    print ('Div is :', c)
except Exception:
        print ('exception was been handeled')
print ('End')
```

o/p.

Eg① Start
Enter a Val : 30
Enter b val 0
exception was been handeled.
End

Eg② Start
Enter a val : hello.
exception Was been handeled
End.

We can give a combination of generic and specific Except block. but first specify ~~should be the~~ occurance block is needed then comes the generic block.

```
try:
    a = int(input('Enter a val: '))
    b = int(input('Enter b val:'))
    c = a/b
    print('Div is:', c)
except ZeroDivisonError: # Specific Except block
    print('Val b is zero')
except Exception # generic except block.
    print("Exception was been handled')
print('End')
```

O/P.

Eg① Start
Enter a val : 10
Enter b val : 0
Exception was been handled.
End

Eg② Start
Enter a val : 10
Enter b val : 0
Val b is zero
End

Default Exception :- It is a type of Except block which works as a Solution for any type of Exception which is getting arrised or envoked.

Except block can work as a Solution for the

hardware base solution also, system based Exception, Custom Exception or user defined Exception.

Syntax:

```
try:
    ---- Statement ----
except:
    ---- Solution ----
```

Eg:

```
print ('Start')
try:
    for i in range (1, 200):
        print (i)
except: default.
    print ('Exception is Handled succesfully')
print ('End')
```

Eg①

o/p:

```
1
:
:
6
Exception is Handled Succesfully.
End.
```

We can give the Combination of types of Expection.

When ever we are giving the combination of all the types of Except block then do follow this below approch / process.

→ try block
→ specific except block.
→ generic except block
→ Deafult Except block.

⇒
```
def get_vals ()
    try:
        a = int (input ("Enter val for a": '))
        b = int (input ("Enter val for b: '))
        return a, b
    Except :
        print ('_' * 30)
        print ('values are improper')
        return get_values ()
def div ()
    try:
        print ('_' * 30)
        x, y = get_values ()
        z = x/y
        print ('_' * 30)
        print ('Res : ' z)
    Except ZeroDivisionError :
        print ('Val b is Zero)
        div ()
    Except Expection :
        print (' Improper vals.)
        div ()
    Except :
        div ()
```

27/12/2020.

* Finally block:- It is a type of block which will get executed manditorly.
* We can use this finally block below of either try block or except block.
* Finally block is a type of block which can consist of any task and will surely get executed even if thier is any return statement in try block or except block.

Eg:-
```
def Sample():
    try:
        a=[1,2,3]
        return a[0]
    except Exception:
        print('exception Handled')
    Finally:
        print('Finally beig Exe
print(Sample())
```

o/p ⇒ Finally being Executed.
1

Eg②
```
def Sample():
    try:
        a=10
        return a[0]
    Except Exception:
        print('Exception Handled Successfully')
        return 420
    finally:
        print('Finally being Executed')
Print(Sample())
```

27/12/2020.

* Finally block :- It is a type of block which will get executed manditorly
* We can use this finally block below of either try block or Except block.
* Finally block is a type of block which can consist of any task and will surely get executed Even if their is any return statement in try block or Except block.

Eg :-
```
def Sample ():
    try :
        a = [1,2,3]
        return a[0]
    except Exception:
        print ('exception Handled s
    Finally :
        print ('Finally beig Exec
print (Sample ())
```

o/p ⇒ Finally being Executed.
1

Eg ②
```
def Sample ():
    try :
        a = 10
        return a[0]
    Except Exception:
        print (' Exception Handled Succesfu
        return 420
    finally :
        print ('Finally being Executed')
print (Sample ())
```

o/p :- Exception Handled Successfully
Finally being Executed
420

Eg ③ def Sample ():
try :
a = 10
return a[0]
Except Exception :
print ('Exception Handled Successfully')
~~Finally~~ return 420
~~Pr~~
Finally :
print ('Finally being Executed')
return 88

print (Sample())

o/p. Exception Handled Successfully
Finally being Executed
88.

Note :- When ever their Exist a finally block in a program at any cost Control must listen the finally block or must Execute the finally block. even though their Exist a return Statment in the try or Except block.

```
def Div()
    try
        a,b = get_elements()
        c = a/b
        return c
    except:
        print("Change the denominator to anon
        return div()

def get_Elemnts()
    try:
        a = int(input("Enter the val"))
        b = int("Enter the val"))
        return a, b
    Except Value Error:
        return get_elements()
                (a,b)

def Sheela():
    try:
        print("ctrl is in try block")
        raise IndexError
        return 10
    except:
        print("ctrl is in except block")
        return 20
    finally:
        print("Control is in finally block
        return 5
```

* raise : raise is a keyword which is used to invoke the ~~Exception~~ any type of Exception on users demand.

We can raise any type of exception in any part of a program.

* The raised expection can be handeled with the help of try and Except block.

Syntax:

raise Name_of_Exception.

raise Name_of_Exception ("msg to display")

Eg①. for i in range (1,200):
    if i == 100:
        raise StopIteration ('Hey you ..Sto
    Print (i)

o/p:
1
2
3
4
5
.
.
.
99
raise StopIteration ('Hey you .... stop it')
StopIteration : Hey you! Stop it.

Eg② for i in range (1,200):
    try:
        if i == 100
            raise StopIteration ('Hey you.. Sto
        Print (i)
    Except Exception :
        print ('Exception Handled....')

o/p

o/p 1
2
3
4
5
:
:
99
Exception Handled

Eg ②:

```
try:
    for i in range(1,200):
        if i==100
            raise StopIteration
        print(i)
except:
    print('Exception Handled.....')
```

o/p 1
2
3
4
5
:
:
99
Exception Handled.

* User defined Exception or Custom Exception:-

Usually we finist the Exceptions which are already redymade or Standard Exception But when it is a time to use that type of Exception which is not present in the standard Exception in that case we gona create our own type of Exception.

* In python we can create our own types of Exception with some simple steps.

* To create our own itype of Exception, we use the help of a class and we inherit from Exception Super class

* Exception is a type of base class / Super class which will provide all the properties to create the Subclass to work like an Exception.

Syntax:-

```
class classname (Exception)
    Pass.
```

```
class low_val (Exception):
    pass
class higher_val (Exception):
    pass
a = int(input('Enter the val : '))
if a<10 :
    raise low_val ('given val is duser than 10')
elif a >20 :
    raise higher_val ('given val is grt than 20')
else :
    print ('perfect input')
```

o/p :-

Enter the number : 6
give value / number is less than 10 :
Traceback /.....

Enter the number : 33

given value / number is more than 20

Enter the Enumber : 12
perfect input :

Own type of Error.

```
class syed (Exception):
    pass
try:
    for i in range (1,20):
        if i==10:
            raise syed ('stop it....!')
        print(i)
except Exception as e:
    print ('Exception handled')
    print (e)
```

o/p:

```
1
2
3
4
5
6
.
.
.
9
Exception handled
stop it!
```