Exception Handeling

In any programming language there are 2 types of errors are possible.

1. Syntax Errors
2. Runtime Errors

Syntax Errors:-

The errors which occurs because of invalid syntax are called syntax errors.

ex:-

```
x = 10
if x == 10
    print("Hello")

  File
"C:\Users\chand\PycharmProjects\testing_17_18_19_python\ExceptionHandeling
\errors.py", line 4
    if x == 10
              ^
SyntaxError: invalid syntax
```

Note:- Programmer is responsible to correct these syntax errors.Once all syntax errors are corrected then only program execution will be started.

Runtime Errors:-

we all are knows as Exceptions

While executing the program if something goes wrong because of end user input or output programming logic or memory problems etc then we will get Runtime Errors.

```
a = int(input("1st"))
b = int(input("2nd"))
print(a/b)
# a= 10
#b = 0

 File
"C:\Users\chand\PycharmProjects\testing_17_18_19_python\ExceptionHandeling
\errors.py", line 10, in <module>
    print(a/b)
ZeroDivisionError: division by zero
```

Note:- Exception Handeling concept applicable for runtime Errors but not for syntax errors.

What is Exception:-

An unwanted and unexcepted event that distrubs normal flow of program is called Exception.
ex:-

ZeroDivisionError
TypeError
ValueError
FileNotFoundError

It is highly recommended to handeling Exceptions.The main objective of exception handeling grecful Termination of the program.

Note:- Exception Handeling does not mean reparing exception.We have to define alternative way to continue rest of the program normally.

Interviews:-(Q)

1. What is Exception?
2. What is the purpose of Exception handeling?
3. What is the meaning of Exception Handeling?

Default Exception Handing In python:-

Every Exception in python is an object,For every exception type the corresponding classes are availaable.

```
a = int(input("1st"))
b = int(input("2nd"))
print(a/b)
# a= 10
#b = 0
```

```
 File
"C:\Users\chand\PycharmProjects\testing_17_18_19_python\ExceptionHandeling
\errors.py", line 10, in <module>
    print(a/b)
ZeroDivisionError: division by zero
```

Python's Eception Hierachy

```
                                        BaseException

Exception               SystenExit                GeneratorExit
KeyboardInterrupt
|
|
|
|
Attribute     Arihmetic       EOF         Name         Lookup      OS
Type         Value
Error         Error           Error       Error        Error       Error
Error        Error

              |                                        |           |
              ZeroDivision                             Index
FileNotFound
              Error                                               Error
Error
              |                                                         |
|
              FloatingPoint                                       Key
Interrputed
              Error                                    Error       Error
              |
               |
              Overflow
Permission
              Error                                                Error

                                                                        |


TimeOut


Error
```

Every Exception in Python is a class.

All exception classes are child classes of BaseException that is every exception class extends BaseExption either directly or indirectly.Hence BaseExption acts as root for Python Exception Hirechy.

Most of the time being a programmer we have to concentrate Exception and its child classes.

26/11/2021

Customized Exception Handeling By using try-except

It is highly recommended to handel exceptions.

The code which may raise exception is called risky code and we have to take risky code inside try block.The corresponding handeling code we have to take inside except block.

```
syntax:-

try:
      Ricky code

except NameOfException:
      Handeliing code/Alternative Code


without try-except
------------------

print("smnt-1")
print(10/0)
print("smnt-2")

Output:-

smnt-1
Traceback (most recent call last):
  File
"C:\Users\chand\PycharmProjects\testing_17_18_19_python\ExceptionHandeling
\errors.py", line 27, in <module>
    print(10/0)
ZeroDivisionError: division by zero


With try-except:
----------------

print("smnt-1")
try:
    print(10/0)
except ZeroDivisionError:
    print(10/2)
print("smnt-2")

output:-
-------

smnt-1
5.0
smnt-2


Control flow in try-except
---------------------------
```

```
try:
      stmnt:1
      stmnt:2
      stmnt:3

except NameOfException:
      stmnt-4

stmnt-5
```

case-1:- If there is no ezxception then 1,2,3,5 Normal Termination.

case-2:- If an exception raised at stmnt-2 and corresponding except block matched then 1,4,5 Normal Termination.

Case-3:- If an exception rasied at stmnt-2 and corresponing except block not matched then 1 Abnormal Termination.

case-4:- If an exception rasied at stmnt-4 or at stmnt-5 then it is always abnormal termination.

How to Print exception information:-

```
a = eval(input("1st:-"))
b = eval(input("2nd:-"))

try:
    print(a/b)
except ZeroDivisionError as msg:
    print("App Zero se divied nehi kar sakte:-",msg)
```

output:-

```
1st:-10
2nd:-0
App Zero se divied nehi kar sakte:- division by zero
```

try-with multi except:-
----

if try with multiple except block available then based on raised exsception the corrresponding except block will be executed.

```
try:
    x = int(input("1st:-"))
    y = int(input("2nd:-"))
    print(x/y)

except ZeroDivisionError:
    print("Can't Divide with zero")

except ValueError:
    print("please provied int value only")


output:-1

1st:-10
2nd:-5
2.0

output:-2

1st:-10
2nd:-0
Can't Divide with zero


#2nd type except:-

1st:-10
2nd:-ten
Traceback (most recent call last):
  File
"C:\Users\chand\PycharmProjects\testing_17_18_19_python\ExceptionHandeling
\errors.py", line 48, in <module>
    y = int(input("2nd:-"))
ValueError: invalid literal for int() with base 10: 'ten'


output-3:-

1st:-10
2nd:-ten
please provied int value only
```

if try with multiple except blocks available then the order of thse except blocks is important .Python interpreter will always consider from top to buttom untill matched except block identified.

Single except block that can handel multiple exception:-
----

we can write a single except block that can handel multiple different types of exceptions.

syntax:-

```
except (Exception_1,Exception_2,Exception_3,.....):
except (Exception_1,Exception_2,Exception_3,.....) as msg:
```

```
Note:- Parenthesis are mandatory and this group of exceptions internally
considered as tuple data.
-------------------------------------------
try:
    x = int(input("1st:-"))
    y = int(input("2nd:-"))
    print(x/y)

except (ZeroDivisionError,ValueError) as msg:
    print("Plz provied valid numbers only and problem is:-",msg)

output:-

1st:-10
2nd:-ten
Plz provied valid numbers only and problem is:- invalid literal for int()
with base 10: 'ten'


output:2:-

1st:-10
2nd:-0
Plz provied valid numbers only and problem is:- division by zero


Default except block:-
--------------------

We can use default except block to handel any type of exceptions.
In default except block generally we can print normal error message.

syntax:-
------

except Exception:-
      stmt

except:
      stmt


try:
    x = int(input("1st:-"))
    y = int(input("2nd:-"))
    print(x/y)

except ZeroDivisionError:
    print("Plz provied valid numbers only and problem is:-")
except:
    print("Handel any type of exception")


output:-
```

```
1st:-10
2nd:-ten
Handel any type of exception
```