EXCEPTIONAL HANDLING

The purpose of exceptional handling is that to build roburst app.

definition of exception

runtime error of every program is called exception.

definition of exceptional handling

the process of convering technical error messages into user-friendly error messages is called exceptional handling.

types of errors

during compile and runtime process we get various type of errors.They are classified into 3 types

1.compile time error

2.logical errors

3.runtime errors

Compile Time Errors

compile time error occur due to programmer not following syntaxes.

.py------>pyc(convertig source code into byte code)

Logical Errors

occure during execution time,logical error occur due to wrong representation/invalid logic of problem.

always give wrong result.

ex:a*b/2 or (*ab)/2 ,here 1 logic is wrong 2 logic is correct.

Runtime Errors

occur during execution time,runtime error occur due to wrong or invalid inputs entered by end-user or app user.

by default runtime error generates technical error messages which are understand by programmer not by end users.

types of exceptions in python

1.Pre-definedor or build-in exception

availabe in python software ,used by pythonlang programmer for dealing with universal problem.

ex: a)division by zero(ZeroDivisionError)

b)invalid attribute names(AttributeError)

c)invalid key(KeyError)

d)module does not exist(ModuleNotFoundError)

e)wrong index(IndexError)

2.programmer or user defined exception

these exception developed by python lang programmers and avalilabe in python project and used by other team members of same project for dealing with common problems in project.

ex: enter invalid pin in atm

withdraw more amount than existing

enter invalid user or password

Handling the exception keywords in python

1.try usd to write block of statements generating exceptions.

every try block must be contain atleast one except block ,otherwise we get syntax error.

2.except used to write block of statements generates user-friendly error messages.

except block will execute when there is an exception occurs in try block.

3.else it a block in which we write block of statemets will display results of the program . optional block,write else block after exception block and before finally block.

4.finally finally block will execute compulsary.is optional to write.

5.raise raise keyword is used for hittng / raising / generating the exception provided some condition must be satisfied.

raise always used inside of fun definition only.

In [ ]:
```
#exception_handling.py

# program 1
print('program execution started')
try:
    a=int(input('enter first value'))
    b=int(input('enter second values'))
    c=a/b
except ZeroDivisionError:
    print('do not enter zero for den...')
except ValueError:
    print('do not enter str,alumns and symbols')
else:
    print('value of a ={}'.format(a))
    print('vallue of b ={}'.format(b))
    print('Dividion of a and b ={} '.format(c))
finally:
    print('excetion completed')


#program 2
# program 1
print('program execution started')
try:
    a=int(input('enter first value'))
    b=int(input('enter second values'))
    c=a/b
except ZeroDivisionError:
    print('do not enter zero for den...')
except ValueError:
    print('do not enter str,alumns and symbols')
except: #default exception
    print('ops something went wrong')
else:
    print('value of a ={}'.format(a))
    print('vallue of b ={}'.format(b))
    print('Dividion of a and b ={} '.format(c))
finally:
    print('excetion completed')
```

In [ ]:
```
#raise 1.0
#hyd.py -->file name act as modlue name
class HydDivisionError(Exception):pass
```

In [ ]:
```
#raise 1.2
#from hyd import HydDivisionError
def divop(a,b):
    if (b==0):
        raise HydDivisionError
    else:
        return (a/b)
```

In [ ]:

```
#raise 1.3
from division import divop
from hyd import HydDivisionError
a=int(input('enter a first values'))
b=int(input("enter a second value"))
try:
    res=divop(a,b)  #fun call
except HydDivisionError:
    print('do not enter zero for den..')
else:
    print('division ={}'.format(res))
finally:
    print('i am from final block')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js