Exception handling

Wednesday, August 6, 2025 7:53 PM

- Also known as error handling.
- If the error is already known to us i.e, syntax error then it is not considered as an exception.
- Apart from syntax error, whatever the error we get that is considered as an exception.
- Exception is an unauthorized event that comes while execution of program and it stops the flow of execution.
- To run the program without any intruption of exception, exception handling is used.

There are 2 keywords to handle the exception:

i. Try: here we write the whole program.
ii. Except: here we write the error to handle.

Exception handling is done in 3ways:

i. Specific exception handling
ii. Generic exception handling
iii. Default exception handling

1. Specific exception handling: When we know which type of error is going to be occurred in the program.

   Syntax:

```
Try:
    Program
Except error_name1:
    Solution
Except error_name2:
    Solution
```

   #error_name should be standard i.e, what the system throws that only write.

   Eg:

```
try:
  a=int(input('Enter the 1st number: '))
```

```
    b=int(input('Enter the 2nd number: '))
    c=a/b
    print(c)
except ZeroDivisionError:
    print('Value of b should not be 0')
except ValueError:
    print('a and b should be of int type only')
```

2. Generic Exception handling: When we don't know what type of error we are going to get, we use generic exception handling.

   Syntax:
   Try:
       Program
   Except Exception as var: #if want to know which type of error it is
       Solution

           Or

   Try:
       Program
   Except Exception: #if don't want to know which type of error it is
       Solution

- Exception is a class that will handle all the errors.

  Eg:

```
'''try:
    a=int(input('Enter the 1st number: '))
    b=int(input('Enter the 2nd number: '))
    c=a/b
    print(c)
except Exception as e:
    print(e)
    print('Exception is handled')'''
```

```
try:
    a=int(input('Enter the 1st number: '))
    b=int(input('Enter the 2nd number: '))
    c=a/b
    print(c)
except Exception:
    print('Exception is handled')
```

- KeyboardIntrupt is the error that generic exception handling can't handle.

3. Default Exception handling: To overcome with generic, we go for default exception handling.
   It can handle all the errors including keyboard intrupt.

   Syntax:
   Try:
   Program
   Except:
   Solution

   Eg:

```
try:
   for i in range(1,501):
      print(i)
except:
   print('Exception is handled')
```

- If we are knowing the type of error we use specific and if we don't know the type of error then we use default.

- If we want to write all the exception handlings then follow the sequence of: Specific -> Generic -> Default

- To handle the exception, try and except keyword is enough but at some places 2 more keywords are used: else and finally

  Syntax:
  Try:
  Program
  Except:
  Solution
  Else:
  S B
  Finally:
  S B

  If Error found:

Try -> except -> finally

If no error found:

Try -> else -> finally

Eg:

```
try:
   import random
   target=100
   score=random.randint(0,1)
   amo=target/score
except ZeroDivisionError:
   print('Player scored 0')
else:
   print(f'Player scored {score} runs')
finally:
   print('Total amount: ',score*1000)
```

- Custom Exceptions:
  When the developer wants to create exception by their own.
- Raise is the keyword used to throw error.
  Syntax:
  raise error_name('msg')
  # error_name should be standard
  #passing message is optional

Eg:

```
n=int(input('Enter the number of product: '))
if n<=0:
   raise ValueError('Out of stock')
else:
   print('You can buy')
```

- User-defined Exception:
  Here we can create our own exception name.

  Syntax:
  Class Error_name(Exception):
  Pass

Eg:

```
class NoProduct(Exception):
   pass
```

```
n=int(input('Enter the number of product: '))
if n<=0:
   raise NoProduct('Out of stock')
else:
   print('You can buy')
```

- Assertion: If the developer wants to pass error without error name then they use assertion error.

  Syntax:
  Assert condition, 'message'
  S B

  If true: S B
  If false: msg

  Eg:

```
s1=input('Enter the string: ')
s2=input('Enter the string: ')
assert len(s1)==len(s2),'Length is different'
print(s1+s2)
```