Agenda

Exception Handling in Python3(try,except,else,finally,raise)

- Syntax Error vs Runtime Error
- Exception Hierarchy
- Exception handling using try-except
- Control flow in try-except
- How to print Exception information to the console
- try with multiple except blocks
- single except block that can handle multiple different exceptions
- Default except block
- finally block and its usage
- finally block vs Destructor
- Guess the Output?

In [1]:

```
# Syntax error
x=100
if x==100
print(x)
```

```
  File "<ipython-input-1-a89562612b27>", line 2
    if x==100
             ^
SyntaxError: invalid syntax
```

In [ ]:

```
stmt1
stmt2
stm3
.
.
.
.
stmt n
```

In [2]:

```
print("Welcome")
print("To")
print(10/0) # Run time Error
print("Python")
print("programming")
print("Stop")
```

```
Welcome
To

---------------------------------------------------------------------------
-
ZeroDivisionError                         Traceback (most recent call las
t)
<ipython-input-2-2632d0b73180> in <module>
      1 print("Welcome")
      2 print("To")
----> 3 print(10/0)
      4 print("Python")
      5 print("programming")

ZeroDivisionError: division by zero
```

try-except

- Syntax

In [ ]:

```
try:
    Risky Code
except:
    Handling code/Alternative code
```

In [37]:

```
print("Welcome to Exception in Python")
print("This is example program of try-except")
try:
    print("Risky Code")
    print(10/0)
except ZeroDivisionError:
    print("Alternative Code")
    print(10/2)
print("Stop Excecution")
```

```
Welcome to Exception in Python
This is example program of try-except
Risky Code
Alternative Code
5.0
Stop Excecution
```

In [43]:

```
print("Welcome to Exception in Python")
try:
    x=int(input("Enter any value"))
    print(x/2)
except ValueError:
    print("Alternative Code")
    print(10/2)
print("Stop Excecution")
```

```
Welcome to Exception in Python
Enter any valueten
Alternative Code
5.0
Stop Excecution
```

Control flow try-except

In [50]:

```
try:
    print("Welcome")
    print("Python3")
    x=int(input("Enter any value"))
    print(x/2)
    print("Division")
    print("Zero")
except ZeroDivisionError:
    print("Alternative Operation-----:\t",10/2)
except ValueError:
    print("The Value Error Code is..\t",50/2)

print("END")
```

```
Welcome
Python3
Enter any valuefifty
The Value Error Code is..        25.0
END
```

printing Exception Information

In [52]:

```
try:
    print(10/0)
except ZeroDivisionError as msg:
    print("The Exception is...:",msg)
```

```
The Exception is...: division by zero
```

In [53]:

```
try:
    t=int(input("Enter any value"))
    print(t/2)
except ValueError as msg:
    print("The Exceptionis...:",msg)
```

```
Enter any valuefifty
The Exceptionis...: invalid literal for int() with base 10: 'fifty'
```

In [56]:

```
try:
    print(10/0)
except ZeroDivisionError as message:
    print("The Exceptionis...:",type(message))
```

```
The Exceptionis...: <class 'ZeroDivisionError'>
```

In [32]:

```
try:
    print(10/0)
except ZeroDivisionError as msg:
    print("The Exceptionis...:",msg.__class__)
```

```
The Exceptionis...: <class 'ZeroDivisionError'>
```

In [33]:

```
try:
    print(10/0)
except ZeroDivisionError as msg:
    print("The Exceptionis...:",msg.__class__.__name__)
```

```
The Exceptionis...: ZeroDivisionError
```

try with Multiple except blocks

In [61]:

```
try:
    print("Welcome")
    print("Python3")
    x=int(input("Enter x value"))
    y=int(input("Enter y value"))
    print(x/y)
except ZeroDivisionError:
    print("Alternative Operation-----:\t",10/2)
except ValueError:
    print("The Value Error Code is..\t",50/2)

print("Stop Execution")
```

```
Welcome
Python3
Enter x valueten
The Value Error Code is..        25.0
Stop Execution
```

In [67]:

```
try:
    print("Welcome")
    print("Python3")
    x=int(input("Enter x value"))
    y=int(input("Enter y value"))
    print(x/y)

except ZeroDivisionError:
    print("Alternative Operation-----:\t",10/2)
except ValueError:
    print("The Value Error Code is..\t",50/2)
except ArithmeticError:
    print("Arithmetic Excdeption....")


print("Stop Execution")
```

```
Welcome
Python3
Enter x value10
Enter y value0
Alternative Operation-----:      5.0
Stop Execution
```

Single except block that can Handle multiple different exceptions

In [68]:

```
try:
    print("*****Python Exceptions*****")
    x=int(input("Enter x value"))
    y=int(input("Enter y value"))
    print(x/y)
except (ZeroDivisionError,ValueError,ArithmeticError) as msg:
    print("The Exception is.....",msg)
print("Stop Execution")
```

```
*****Python Exceptions*****
Enter x value10
Enter y value0
The Exception is..... division by zero
Stop Execution
```

Default except block

- If no except block is matched, then default except block will be executed.
- Default block contains only information,but not handling code
- Default except block must be last statement

In [3]:

```
try:
    a=int(input("Enter a Value:"))
    b=int(input("Enter b Value:"))
    print(a/b)
except ZeroDivisionError:
    print("Divide By Zero Error")
except: # Default Exception block
    print("This is Default Except block")
```

```
Enter a Value:fifty
This is Default Except block
```

In [ ]:

```
# output 1:
Enter a Value:10
Enter b Value:0
Divide By Zero Error
# output 2:
Enter a Value:fifty
This is Default Except block
```

finally

- This is optional block
- This block will be executed wether the exception is occured or not(except os.exit(0))
- finally block is meant for cleanup code activities

In [ ]:

```
# Syntax
try:
    Risky Code
except:
    Handling Code
finally:
    Cleanup Code
```

In [4]:

```
try:
    print("This is ---try--- block")
    print(50/0)
except ZeroDivisionError:
    print("Divide By Zero Error")
finally:
    print("This is ***finally*** block")
```

```
This is ---try--- block
Divide By Zero Error
This is ***finally*** block
```

In [5]:

```
try:
    print("This is ---try--- block")
    print(50/0)
except ValueError:
    print("Divide By Zero Error")
finally:
    print("This is ***finally*** block")
```

```
This is ---try--- block
This is ***finally*** block

---------------------------------------------------------------------------
-
ZeroDivisionError                         Traceback (most recent call las
t)
<ipython-input-5-0e889b774511> in <module>
      1 try:
      2     print("This is ---try--- block")
----> 3     print(50/0)
      4 except ValueError:
      5     print("Divide By Zero Error")

ZeroDivisionError: division by zero
```

finally block is not executed when we use a function os._exit(0)

In [ ]:

```
import os
try:
    print("TRY-BLOCK")
    os._exit(0)
except ValueError:
    print("Exception Handling Code")
finally:
    print("Finally Block")
```

finally vs destructor

- finally block is meant for maintaining clean up activities
- destructor is meant for maintaining clean up activities
- try block related cleanup activities can be released in finally block, where as object related cleanup activities can released by destructor.

In [ ]:

```

```

Guess The Output of the following snippets

In [10]:

```
x=()
if x:
    print(x,"True")
else:
    print(x,"False")
```

```
() False
```

In [12]:

```
p=(1,10)
q=0
r=[]
if p or q and r: # 0 and []--->False and False==>False------p or False-->(1,10) or Fals
e===>True
    print("True")
else:
    print("False")
```

```
True
```

In [14]:

```
l=["ABC","DEFG","IJK","pqr"]
l2=[]
for i in l:
    x=i.lower()
    l2.append(x)
print(l)
print(l2)
```

```
['ABC', 'DEFG', 'IJK', 'pqr']
['abc', 'defg', 'ijk', 'pqr']
```

In [ ]:

```
l1=["abc","123"]
for i in l1:  # l1=["abc","123","abc"]--->["abc","123","abc","123","abc","12
3",.............]
    l1.append(i)
print(l1)
```

In [6]:

```
for i in range(1,20):
    if i == 5:
        break
    else:
        print(i)
else:
    print("------Success----")
```

```
1
2
3
4
```

In [7]:

```
for i in range(1,20):
    if i == 5:
        continue
    else:
        print(i)
else:
    print("------Success----")
```

```
1
2
3
4
6
7
8
9
10
11
12
13
14
15
16
17
18
19
------Success----
```

In [8]:

```
lst=["These","are","a",["few","words"],"that","we","use"]
print(lst[3:4])
print(lst[3:4][0])
print(lst[3:4][0][1][2])
```

```
[['few', 'words']]
['few', 'words']
r
```

THANK YOU*

In [ ]:

In [ ]: