

Chap.- 8 || Exception Handling

```
In [2]: 1 a = input("Enter :")
        2
        3 # SyntaxError: unexpected EOF while parsing
        4 # class name : SyntaxError

File "<ipython-input-2-7e365740942f>", line 4
      # class name : SyntaxError
                        ^
SyntaxError: unexpected EOF while parsing
```

```
In [4]: 1 print(b)
        2
        3 # NameError: name 'b' is not defined

-----
-
NameError                                Traceback (most recent call last)
<ipython-input-4-363b803a2142> in <module>
----> 1 print(b)
      2
      3 # NameError: name 'b' is not defined

NameError: name 'b' is not defined
```

```
In [5]: 1 a = 2
        2 b = '34'
        3 print(a + b)
        4
        5 # TypeError: unsupported operand type(s) for +: 'int' and 'str'

-----
-
TypeError                                Traceback (most recent call last)
<ipython-input-5-f526107410a0> in <module>
      1 a = 2
      2 b = '34'
----> 3 print(a + b)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
In [8]: 1 int('2')
        2 # initialized
```

Out[8]: 2

In [9]:

```
1 int('a')
2
3 # ValueError: invalid literal for int() with base 10: 'a'
```

```
-----
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-9-0e680a129950> in <module>
----> 1 int('a')
      2
      3 # ValueError: invalid literal for int() with base 10: 'a'

ValueError: invalid literal for int() with base 10: 'a'
```

In [11]:

```
1 if True:
2     print(True)
3
4 # IndentationError: expected an indented block
```

```
File "<ipython-input-11-7d20c3e1839a>", line 2
    print(True)
    ^
IndentationError: expected an indented block
```

In [12]:

```
1 l = [1,2,3]
2 print(l[10])
3
4 # IndexError: list index out of range
```

```
-----
-
IndexError                                Traceback (most recent call las
t)
<ipython-input-12-06e5cfc3082f> in <module>
      1 l = [1,2,3]
----> 2 print(l[10])

IndexError: list index out of range
```

In [13]:

```
1 l = ['a', 'b', 'c']
2 l.upper()
3
4 # AttributeError: 'list' object has no attribute 'upper'
```

```
-----
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-13-a1bb18f2df1e> in <module>
      1 l = ['a', 'b', 'c']
----> 2 l.upper()

AttributeError: 'list' object has no attribute 'upper'
```

```
In [21]: 1 d = {'a': 1}
          2 print(d[10])
          3 print(d[1])
          4
          5 # KeyError: 10
          6 # KeyError: 1
```

```
-----
-
KeyError                                Traceback (most recent call las
t)
<ipython-input-21-8ef5d0297023> in <module>
      1 d = {'a': 1}
----> 2 print(d[10])
      3 print(d[1])
      4
      5 # KeyError: 10

KeyError: 10
```

```
In [22]: 1 import mypy
          2
          3 # ModuleNotFoundError: No module named 'mypy'
```

```
-----
-
ModuleNotFoundError                    Traceback (most recent call las
t)
<ipython-input-22-dddae9e5207e> in <module>
----> 1 import mypy

ModuleNotFoundError: No module named 'mypy'
```

```
In [23]: 1 a = 5/0
          2
          3 # ZeroDivisionError: division by zero
```

```
-----
-
ZeroDivisionError                      Traceback (most recent call las
t)
<ipython-input-23-2ba49e50984d> in <module>
----> 1 a = 5/0

ZeroDivisionError: division by zero
```

```
In [32]: 1 import math
2 math.exp(1000)
3 math.exp(10) # 22026.465794806718
4
5 # OverflowError: math range error
```

```
-----
-
OverflowError                                Traceback (most recent call las
t)
<ipython-input-32-03b4fef87082> in <module>
      1 import math
----> 2 math.exp(1000)
      3 math.exp(10) # 22026.465794806718
      4
      5 # OverflowError: math range error

OverflowError: math range error
```

How to Handle this error (Error Handling)

```
In [34]: 1 try:
2 #     a = int(input())
3 #     b = int(input())
4     a = 1
5     b = 0
6     c = a/b
7 except:
8     print("Error")
9 finally:
10    print("Finally Block")
```

```
Error
Finally Block
```

```
In [38]: 1 try:
2     a = 1
3     b = 0
4     c = a/b
5 except Exception as e:
6     print("Error :", e)
7 finally:
8     print("Finally Block")
9
10 # Error : division by zero
11 # Finally Block
```

```
Error : division by zero
Finally Block
```

```
In [42]: 1 # Specific class
2 try:
3     a = 1
4     b = 0
5     c = a/b
6 except ZeroDivisionError as e:
7     print("Error :", e)
8 finally:
9     print("Finally Block")
10
11 # Error : division by zero
12 # Finally Block
```

Error : division by zero
Finally Block

```
In [43]: 1 # Specific class
2 try:
3     a = 1
4     b = 'a'
5     c = a/b
6 except ZeroDivisionError as e:
7     print("Error :", e)
8 finally:
9     print("Finally Block")
10
11 # Error not handled cuz specific class is defined
12 # TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

Finally Block

```
-----
-
TypeError                                Traceback (most recent call last)
<ipython-input-43-98c595340a93> in <module>
      3     a = 1
      4     b = 'a'
----> 5     c = a/b
      6 except ZeroDivisionError as e:
      7     print("Error :", e)
```

TypeError: unsupported operand type(s) for /: 'int' and 'str'

```
In [49]: 1 # Specific class
2 try:
3     a = 'x'
4     b = 'y'
5     c = a/b
6 except (ZeroDivisionError, TypeError) as e:
7     print("Error :", e)
8 finally:
9     print("Finally Block")
10
11 # Error : unsupported operand type(s) for /: 'str' and 'str'
12 # Finally Block
```

Error : unsupported operand type(s) for /: 'str' and 'str'
Finally Block

```
In [51]: 1 try:
2     a = 1
3     b = 0
4     c = a/b
5 except ZeroDivisionError as e:
6     b = 5
7     c = a/b
8     print(b) # 5
9     print(c) # 0.2
10 finally:
11     print("Finally Block")
12
13 # 5
14 # 0.2
15 # Finally Block
```

5
0.2
Finally Block

```
In [56]: 1 try:
2     a = int(input())
3     b = int(input())
4     c = a/b
5 except ZeroDivisionError as e:
6     print("Error :", e)
7 except ValueError as v:
8     print('Value Error : ', v)
9 finally:
10     print("Finally Block")
11
12 # a
13 # Value Error : invalid literal for int() with base 10: 'a'
14 # Finally Block
```

a
Value Error : invalid literal for int() with base 10: 'a'
Finally Block

In [58]:

```
1 try:
2     a = int(input())
3     b = int(input())
4     c = a/b
5 except ZeroDivisionError as e:
6     print("Error :", e)
7 except ValueError as v:
8     print('Value Error : ', v)
9 finally:
10    print("Finally Block")
11
12 # File "<ipython-input-57-844a742cdd88>", line 4
13 #     c = a/b
14 #     ^
15 # SyntaxError: invalid syntax
```

File "<ipython-input-58-adde3000e7ef>", line 4

```
c = a/b
^
```

SyntaxError: invalid syntax

In [60]:

```
1 try:
2     if (condition):
3         raise Exception("msg.")
4 except:
5 else:
6 finally:
```

- **if error in try then go to except**
- **if not error in try then go to else**
- **indentation & syntex erro will not be handle by error exception**
- **it's call compiletime error.**

Password Varification from Dict.

In [89]:

```

1  d = {'user' : 1234}
2
3  ch = input("Have you Login(l) or Sign-Up(s)? : ")
4  if(ch == 'l'):
5      try:
6          u = input("Enter user Name : ")
7          pwd = int(input("Enter Password : "))
8          if(d[u] == pwd):
9              print('Login Successful!')
10         else:
11             raise Exception("Doesn't Find user Name!")
12     except Exception as e:
13         print(e)
14
15 elif(ch == 's'):
16     try:
17         u = input("Enter user Name : ")
18         p = int(input("Enter Password : "))
19         cp = int(input("Enter Confirm Password : "))
20
21         if(cp == p):
22             d[u] = p
23         else:
24             raise Exception("Enter Password again!")
25     except Exception as e:
26         print(e)
27     else:
28         print("Sign-Up Successfull.")
29 else:
30     print("Invalid Choice!")
31
32 print(d)
33
34 # Have you Login(l) or Sign-Up(s)? : s
35 # Enter user Name : rk
36 # Enter Password : 12358
37 # Enter Confirm Password : 12358
38 # Sign-Up Successfull.
39 # {'user': 1234, 'rk': 12358}
40
41 # Have you Login(l) or Sign-Up(s)? : l
42 # Enter user Name : user
43 # Enter Password : 1234
44 # Login Successful!
45 # {'user': 1234}

```

```

Have you Login(l) or Sign-Up(s)? : l
Enter user Name : user
Enter Password : 1234
Login Successful!
{'user': 1234}

```

Creadited & Debited

```
In [93]: 1 d = {'user':{'pwd':'1234', 'bal' : 5000}}
2
3 try:
4     ch = input('1.Credit\n2.Debit\n')
5     amt = int(input("Enter Amt:"))
6     bal = d[uname]['bal']
7
8     if ch==1:
9         bal += amt
10        print('Available Balance :', bal)
11    elif ch==2:
12        bal -= amt
13        if bal<1000:
14            raise Exception("Insufficient Balance!")
15 except Exception as e:
16     print(e)
17     bal += amt
18 else:
19     print("Transaction Successful :")
```

OOP

- 1) Encapsulation
- 2) Inheritance
- 3) Abstraction
- 4) Polymorphism = same name different functuinallty

Var

- Instance
- class

if we are make a method in class then, def demo(self):

- self is compulsory

```
In [15]: 1 # class A:
2 class A():
3     def demo(self, a,b):
4         print(a) # 10
5         print(b) # 20
6         print(self) # <__main__.A object at 0x0000024B033C63D0>
7
8 x = A()
9 x.demo(10,20)

10
20
<__main__.A object at 0x0000024B03544370>
```

- In this case x, y is local variable and it's only accessible for demo method

In [10]:

```

1 class A():
2     def demo(self, a,b):
3         x = a
4         y = b
5
6     def display(self):
7         print(x,y)
8 m = A()
9 m.demo(10,20)
10 m.display()

```

NameError

Traceback (most recent call last)

<ipython-input-10-b6ad73848f2d> in <module>

```

7 m = A()
8 m.demo(10,20)
----> 9 m.display()

```

<ipython-input-10-b6ad73848f2d> in display(self)

```

4         y = b
5     def display(self):
----> 6         print(x,y)
7 m = A()
8 m.demo(10,20)

```

NameError: name 'y' is not defined

In [16]:

```

1 class A():
2     def demo(self, a,b):
3         self.x = a
4         self.y = b
5
6     def display(self):
7         print(self.x,self.y) # 10 20
8 m = A()
9 m.demo(10,20)
10 m.display()
11 print(m.x) # 10

```

10 20

10

- Constructor

```
In [21]: 1 class A():
2         def __init__(self, v):
3             self.a = v
4
5         def display(self):
6             print(self.a) # 10
7
8 o = A(10)
9 o2 = A(99)
10 o.display()
```

10

```
In [ ]: 1
```