Chap.- 8 || Exception Handling

In [2]:
```
a = input("Enter :"

# SyntaxError: unexpected EOF while parsing
# class name : SyntaxError
```

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

In [4]:
```
print(b)

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

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<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]:
```
a = 2
b = '34'
print(a + b)

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

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<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]:
```
int('2')
# initialized
```

Out[8]:
```
2
```

In [9]:
```python
int('a')

# 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]:
```python
if True:
print(True)

# IndentationError: expected an indented block
```

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

In [12]:
```python
l = [1,2,3]
print(l[10])

# 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]:
```python
l = ['a', 'b', 'c']
l.upper()

# 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]:
```python
d = {'a': 1}
print(d[10])
print(d[1])

# KeyError: 10
# 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]:
```python
import mypy

# 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]:
```python
a = 5/0

# 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]:
```python
import math
math.exp(1000)
math.exp(10) # 22026.465794806718

# 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]:
```python
try:
#     a = int(input())
#     b = int(input())
    a = 1
    b = 0
    c = a/b
except:
    print("Error")
finally:
    print("Finally Block")
```

```
Error
Finally Block
```

In [38]:
```python
try:
    a = 1
    b = 0
    c = a/b
except Exception as e:
    print("Error :", e)
finally:
    print("Finally Block")

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

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

In [42]:
```python
# Specific class
try:
    a = 1
    b = 0
    c = a/b
except ZeroDivisionError as e:
    print("Error :", e)
finally:
    print("Finally Block")

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

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

In [43]:
```python
# Specific class
try:
    a = 1
    b = 'a'
    c = a/b
except ZeroDivisionError as e:
    print("Error :", e)
finally:
    print("Finally Block")

# Error not handled cuz specific class is defined
# TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

```
Finally Block

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
<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]:
```python
# Specific class
try:
    a = 'x'
    b = 'y'
    c = a/b
except (ZeroDivisionError, TypeError) as e:
    print("Error :", e)
finally:
    print("Finally Block")

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

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

In [51]:
```python
try:
    a = 1
    b = 0
    c = a/b
except ZeroDivisionError as e:
    b = 5
    c = a/b
    print(b) # 5
    print(c) # 0.2
finally:
    print("Finally Block")

# 5
# 0.2
# Finally Block
```

```
5
0.2
Finally Block
```

In [56]:
```python
try:
    a = int(input())
    b = int(input())
    c = a/b
except ZeroDivisionError as e:
    print("Error :", e)
except ValueError as v:
    print('Value Error : ', v)
finally:
    print("Finally Block")

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

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

In [58]:
```python
try:
    a = int(input())
    b = int(input()
    c = a/b
except ZeroDivisionError as e:
    print("Error :", e)
except ValueError as v:
    print('Value Error : ', v)
finally:
    print("Finally Block")

# File "<ipython-input-57-844a742cdd88>", line 4
#     c = a/b
#     ^
# SyntaxError: invalid syntax
```

```
  File "<ipython-input-58-adde3000e7ef>", line 4
    c = a/b
    ^
SyntaxError: invalid syntax
```

In [60]:
```python
try:
    if (condition):
        raise Exception("msg.")
except:
else:
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]:
```python
d = {'user' : 1234}

ch = input("Have you Login(l) or Sign-Up(s)? : ")
if(ch == 'l'):
    try:
        u = input("Enter user Name : ")
        pwd = int(input("Enter Password : "))
        if(d[u] == pwd):
            print('Login Successful!')
        else:
            raise Exception("Doesn't Find user Name!")
    except Exception as e:
        print(e)

elif(ch == 's'):
    try:
        u = input("Enter user Name : ")
        p = int(input("Enter Password : "))
        cp = int(input("Enter Confirm Password : "))

        if(cp == p):
            d[u] = p
        else:
            raise Exception("Enter Password again!")
    except Exception as e:
        print(e)
    else:
        print("Sign-Up Successfull.")
else:
    print("Invalid Choice!")

print(d)

# Have you Login(l) or Sign-Up(s)? : s
# Enter user Name : rk
# Enter Password : 12358
# Enter Confirm Password : 12358
# Sign-Up Successfull.
# {'user': 1234, 'rk': 12358}

# Have you Login(l) or Sign-Up(s)? : l
# Enter user Name : user
# Enter Password : 1234
# Login Successful!
# {'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]:
```python
d = {'user':{'pwd':'1234', 'bal' : 5000}}

try:
    ch = input('1.Creadit\n2.Debit\n')
    amt = int(input("Enter Amt:"))
    bal = d[uname]['bal']

    if ch==1:
        bal += amt
        print('Available Balance :', bal)
    elif ch==2:
        bal -= amt
        if bal<1000:
            raise Exception("Insfficient Balance!")
except Exception as e:
    print(e)
    bal += amt
else:
    print("Transactoin Successfull :)")
```

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 compulsary**

In [15]:

```python
# class A:
class A():
    def demo(self, a,b):
        print(a) # 10
        print(b) # 20
        print(self) # <__main__.A object at 0x0000024B033C63D0>

x = A()
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]:
```python
class A():
    def demo(self, a,b):
        x = a
        y = b

    def display(self):
        print(x,y)
m = A()
m.demo(10,20)
m.display()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
<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]:
```python
class A():
    def demo(self, a,b):
        self.x = a
        self.y = b

    def display(self):
        print(self.x,self.y) # 10 20
m = A()
m.demo(10,20)
m.display()
print(m.x) # 10
```

```
10 20
10
```

- **Constructor - When class object is create then automatically called.**

In [21]:
```python
class A():
    def __init__(self, v):
        self.a = v

    def display(self):
        print(self.a) # 10

o = A(10)
o2 = A(99)
o.display()
```

```
10
```

- **Distructor - When class Object is deleted then automatically run.**

In [2]:
```python
class Atm:
    def __init__(self):
        print('Object Created.')
    def __del__(self):
        print('Object Deleted.')

ob = Atm()
del ob
```

```
Object Created.
Object Deleted.
```

In [4]:
```python
class A:
    def demo(self):
        a = 10
        b = 20
        c = a + b
        return a
        return b
        return c
ob = A()
print(ob.demo()) # 10
```

```
10
```

- **when one or more return keywords = only consider first return**

- **yield - to return a multiple values**

In [7]:
```python
class A:
    def demo(self):
        a = 10
        b = 20
        c = a + b
        yield a
        yield b
        yield c
ob = A()
for i in ob.demo():
    print(i)
# 10
# 20
# 30
```

```
10
20
30
```

- **U can return multiple value using return and it will return as a tuple.**

In [9]:
```python
class A:
    def demo(self):
        a = 10
        b = 20
        c = a + b
        return a,b,c
ob = A()
print(ob.demo()) # (10, 20, 30)
```

```
(10, 20, 30)
```

In [ ]:
```python

```