Error (or) Exception Handling

Errors are the problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which changes the normal flow of the program.
Two types of Errors may occur in python.

1. Syntax errors
2. Logical errors (Exceptions)

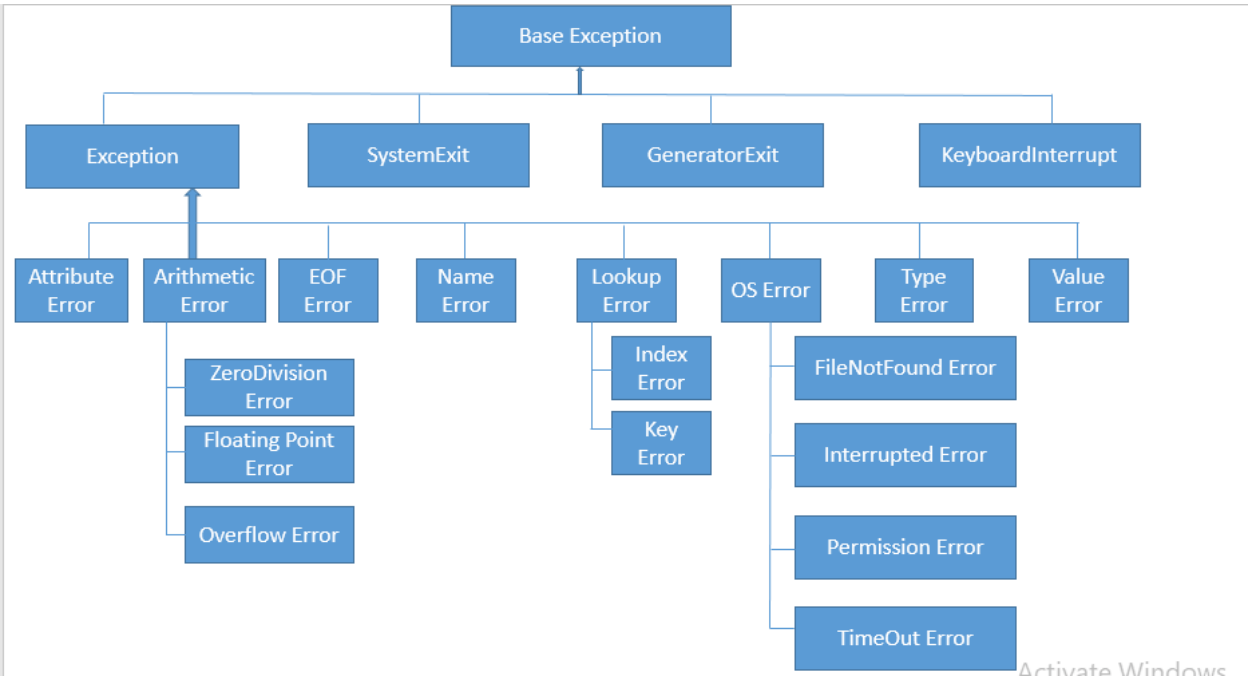
Error handling is a concept which is used to provide alternate solution for any error occurred and continue the flow of execution smoothly till the end of the program.

Common Exceptions

Python provides the number of built-in exceptions, but here we are describing the common standard exceptions. A list of common exceptions that can be thrown from a standard Python program is given below.

ZeroDivisionError: Occurs when a number is divided by zero.
NameError: It occurs when a name is not found. It may be local or global.
IndentationError: If incorrect indentation is given.
IOError: It occurs when Input Output operation fails.
EOFError: It occurs when the end of the file is reached, and yet operations are being performed.

The following diagram shows list of errors hierarchy:



To achieve error handling in python we use the following keywords:

try,

except

else

raise

finally

try: Place the statements which are prone to error in try block.

except: Prove the alternate solution by catching the error in the except block.

```
# Error Handling
# try with except block
a,b = input("Enter two numbers: ").split(' ')
a = int(a)
b = int(b)

try:
    c = a/b
```

```
    print("Division result is: ", c)
except ZeroDivisionError:
    print("Pls do not enter zeros")

print("End of operation")
print("End of program...")
print("Exiting from program...")


# A single try block can have any no of except blocks
# each except block should handle an individual error

a,b = input("Enter two numbers: ").split(' ')

try:
    a = int(a)
    b = int(b)
    c = a/b
    print("Division result is: ", c)
except ZeroDivisionError:
    print("Pls do not enter zeros")
except ValueError:
    print("Enter only numbers...")

print("End of operation")
print("End of program...")
print("Exiting from program...")
```

else:

Else block is optional.

It executes when try is successfully executed.

There should be only one else block per try.

Else block should be written only after except block.

```
# try with else and except blocks
a,b = input("Enter two numbers: ").split(' ')
a = int(a)
b = int(b)

try:
    c = a/b
    print("Division result is: ", c)
except ZeroDivisionError:
    print("Pls do not enter zeros")
except ValueError:
    print("Enter only numbers...")
else:
    print("Else block will execute only when try has no
error")

print("End of operation")
print("End of program...")
print("Exiting from program...")
```

raise:

raising an error explicitly is used to check the error handling nature of a program.

Raise can be used in any block.

```
# 'raise' is used to raise an error
explicitly(intentionally)
a,b = input("Enter two numbers: ").split(' ')
a = int(a)
b = int(b)

try:
    raise TypeError("Just for fun")
except ZeroDivisionError:
```

```
    print("Pls do not enter zeros")
except ValueError:
    print("Enter only numbers...")
else:
    print("Else block will execute only when try has no
error")

print("End of operation")
print("End of program...")
print("Exiting from program...")
```

finally:

This is block is used to execute those statements which are mandatory for execution.

Despite to error handled or not, finally block will get executed.

Generally, closure statements are written in finally block.

There should be only one finally block per try.

Finally, block should be the last block among the error handling blocks.

```
# try with else, except and finally blocks
a,b = input("Enter two numbers: ").split(' ')
a = int(a)
b = int(b)

try:
    raise TypeError("Just for fun")
    c=a/b
    print("Result is: ", c)
except ZeroDivisionError:
    print("Pls do not enter zeros")
except ValueError:
```

```
    print("Enter only numbers...")
else:
    print("Else block will execute only when try has no
error")
finally:
    print("no matter what happens, finally gets exected at
all times...")

print("End of operation")
print("End of program...")
print("Exiting from program...")


# try with default Exception class
a,b = input("Enter two numbers: ").split(' ')
a = int(a)
b = int(b)

try:
    raise AttributeError("Just for fun")
except ZeroDivisionError:
    print("Pls do not enter zeros")
except ValueError:
    print("Enter only numbers...")
except Exception as e:
    print(e, 'has been handled here')
else:
    print("Else block will execute only when try has no
error")

print("End of operation")
print("End of program...")
print("Exiting from program...")
```

```python
# try with finally block

try:
    print("test case...")
    raise ZeroDivisionError()
finally:
    print("This is finally....")



# try with mulitiple raised errors.
# only the first error can be handled.
try:
    raise ZeroDivisionError()
    raise AttributeError()
except ZeroDivisionError:
    print("ZDE handled...")
except AttributeError:
    print("AE handled...")



# nested try block
try:
    print("first line")
    print("second line")
    try:
        print("third line")
    except:
        print("This is except for innner try block")
except:
    print("This is except for outer try...")
```

```python
# try with function
def calc():
    x,y = input("Enter two values: ").split()
    x,y = int(x),int(y)
    try:
        z = x/y
        print("Result is:", z)
    except ZeroDivisionError:
        print("ZDE handled...")
    pass

calc()


# error propagation
def calc():
    x,y = input("Enter two values: ").split()
    x,y = int(x),int(y)
    z = x/y
    print("Result is:", z)
    pass

try:
    calc()
except ZeroDivisionError:
    print("ZDE handled...")
#error propagation

def a():
    print(10/0)

def b():
    a()

def c():
    try:
        b()
    except ZeroDivisionError:
        print("ZDE handled in fun c")
```

```
c()



# rethrowing an error

def one():
    try:
        print(10/0)
    except ZeroDivisionError as zde:
        raise zde

def two():
    try:
        one()
    except ZeroDivisionError:
        print("ZDE handled...")

two()
```