

Example:

class Person:

```
class __Address: # Inner class/Member class
```

```
def __init__(self):
```

```
    self.__street=None
```

```
    self.__city=None
```

```
def readAddress(self):
```

```
    self.__street=input("Enter Street ")
```

```
    self.__city=input("Enter City ")
```

```
def printAddress(self):
```

```
    print(f'Street {self.__street}')
```

```
    print(f'City {self.__city}')
```

```
def __init__(self):
```

```
    self.__name=None
```

```
    self.__add1=Person.__Address()
```

```
    self.__add2=Person.__Address()
```

```
def readPerson(self):
```

```
    self.__name=input("Enter Name ")
```

```
    self.__add1.readAddress()
```

```
    self.__add2.readAddress()
```

```
def printPerson(self):
```

```
    print(f'Name : {self.__name}')
```

```
    self.__add1.printAddress()
```

```
    self.__add2.printAddress()
```

```
p1=Person()
```

```
p1.readPerson()
```

```
p1.printPerson()
```

Output

Enter Name naresh

Enter Street abc

Enter City xyz

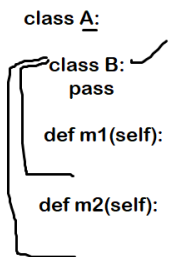
Enter Street pqr

Enter City xyz
Name : naresh
Street abc
City xyz
Street pqr
City xyz

Local class

A class defined inside method or block is called local class.

```
class A:
    class B:
        pass
    def m1(self):
    def m2(self):
```



```
class A:
    def m1(self):
        class B: --> Local class
            pass
```

Local class object is created only inside declared block but cannot accessible outside declared block.

Example:

```
class A: # outer class
    def m1(self):
        class B: # inner class/local class
            def m2(self):
                print("m2 of B class")
        objb=B()
        objb.m2()
        print("m2 of A class")
```

```
obja=A()  
obja.m1()
```

Output

m2 of B class
m2 of A class

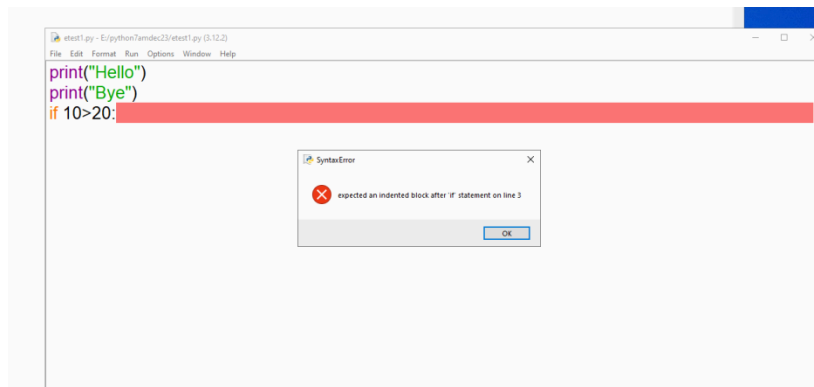
Exception Handling or Error Handling

Types of Errors

1. Compile time errors
2. Logical errors
3. Runtime errors

Compile time errors

All syntax errors are called compile time errors. If there is a syntax error within program, program execution is not done.



Syntax errors must be rectified by programmer in order to execute program.

Logical Errors

A program is having logic to solve given problem. This logic can be condition or formula or anything. If there is mistake in writing logic, program leads to wrong result.

Example

finding maximum of two numbers

```
num1=int(input("Enter first number "))
num2=int(input("Enter second number "))
if num1>num2:
    print(num2)
else:
    print(num1)
```

Output

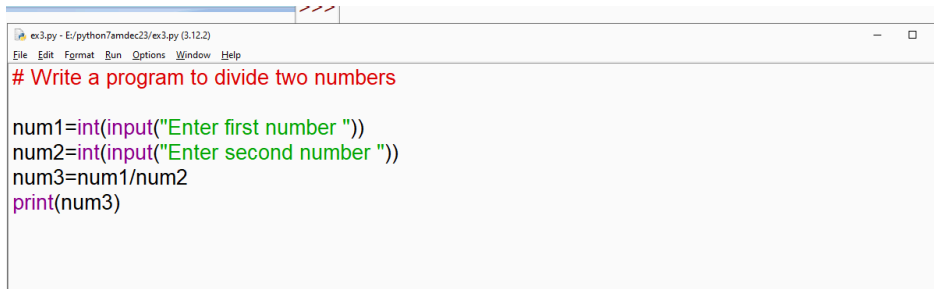
```
Enter first number 20
Enter second number 10
10
```

These logical errors must be rectified by programmer in order to get accurate result.

Runtime errors

The errors which occur during execution of program are called runtime error. The runtime error occurs because of wrong input given by end user. When there is runtime error within program, program execution is terminated.

Example:

A screenshot of a Python IDE window titled 'ex3.py - E:/python7amdec23/ex3.py (3.12.2)'. The window contains the following Python code:

```
# Write a program to divide two numbers

num1=int(input("Enter first number "))
num2=int(input("Enter second number "))
num3=num1/num2
print(num3)
```

Enter first number 6
Enter second number 3
2.0

Enter first number 7
Enter second number 0
Traceback (most recent call last):
 File "E:/python7amdec23/ex3.py", line 5, in <module>
 num3=num1/num2
ZeroDivisionError: division by zero

Enter first number abc
Traceback (most recent call last):
 File "E:/python7amdec23/ex3.py", line 3, in <module>
 num1=int(input("Enter first number "))
ValueError: invalid literal for int() with base 10: 'abc'
>>>

What is exception?

Exception is a runtime error. When there is an exception within program, program execution is terminated. To avoid abnormal termination of program we use exception handlers or exception handling.

Advantage of exception handling

1. To avoid abnormal termination of program

2. To convert predefined error messages to user defined error messages.
3. To separate business logic and error logic

Keywords used in exception handling

1. try
2. except
3. finally
4. raise

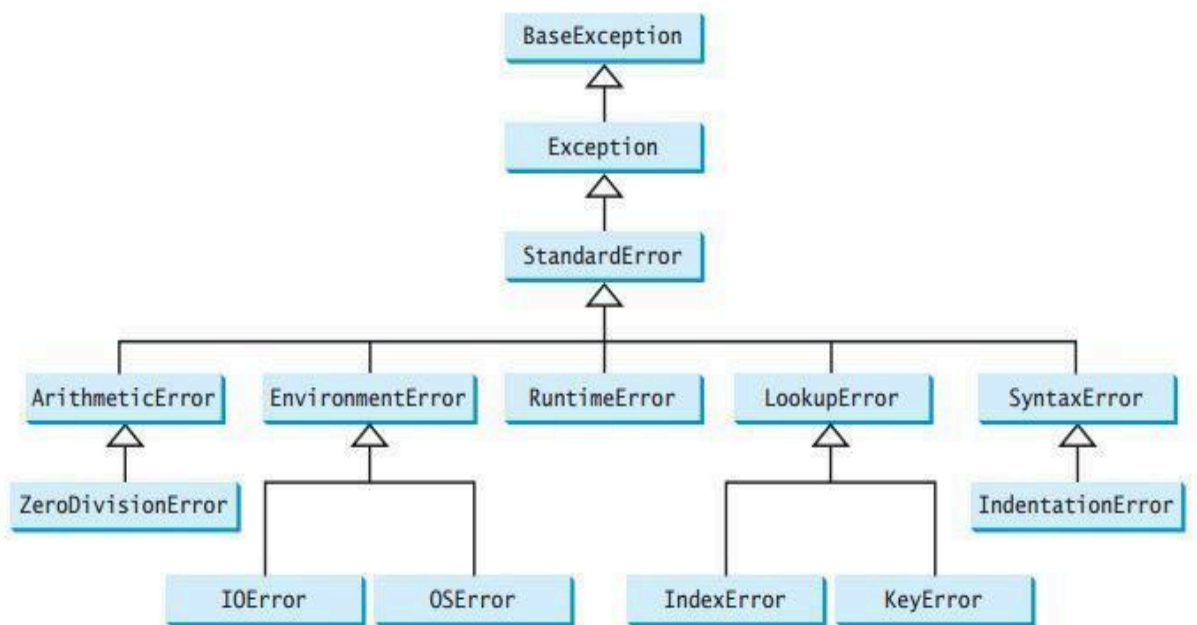
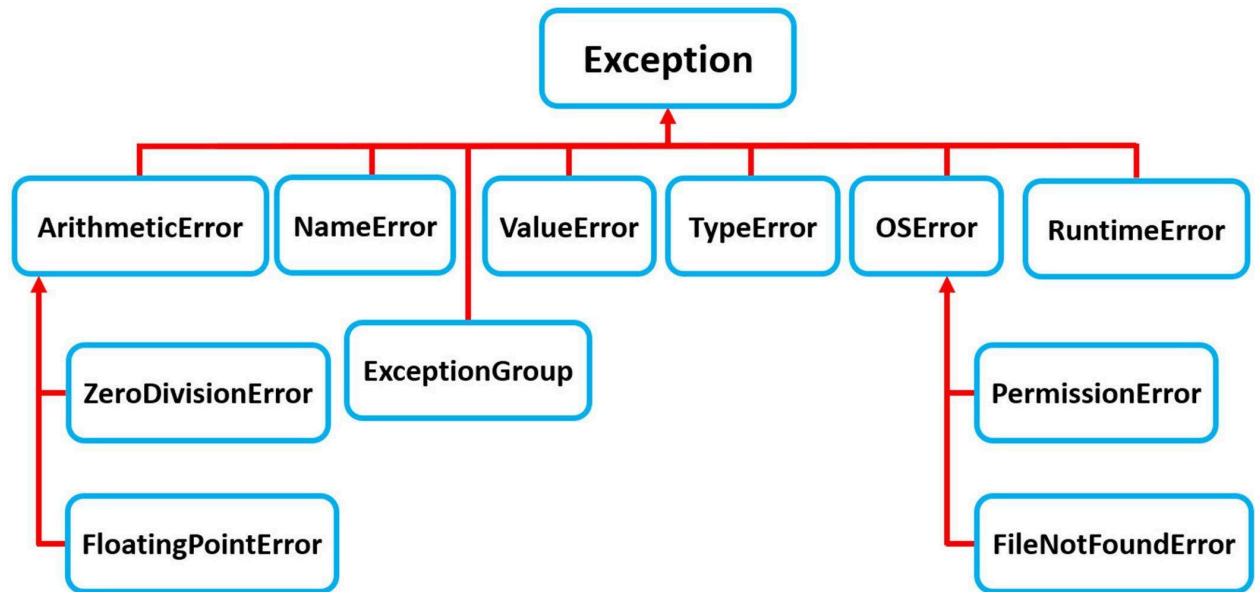
Every exception or error is one type (class). Generate error or exception is nothing but creating object of error class and giving to PVM.

All error classes are inherited from Exception class.

Exception class is a root class or parent class for all error or exception classes.

Exceptions are two types.

1. Predefined exception/errors
2. User defined exception/errors



All predefined error types are used by python and python libraries.

Try block

Try block contains the statements which have to be monitored for exception handling (OR) the statements which generate error during runtime are included inside try block.

Syntax:

```
try:  
    statement-1  
    statement-2
```

except block

If there is an error within try block, it is handled by except block. Except block is error handler block. Try block followed by one or more than one except block.

Syntax1: try with one except	Syntax2: try with multiple except
<pre>try: statement-1 statement-2 except <error-type>: statement-3</pre>	<pre>try: statement-1 statement-2 except <error-type>: statement-3 except <error-type>: statement-4</pre>

