# EXCEPTION HANDLING

**In any programming language there are 2 types of errors are possible.**

1) **Syntax Errors**
2) **Runtime Errors**

## 1) <u>Syntax Errors:</u>

**The errors which occur because of invalid syntax are called syntax errors.**

**<u>Eg 1:</u>**

```
x = 10
if x == 10
  print("Hello")

 SyntaxError: invalid syntax
```

**<u>Eg 2:</u>**

```
print "Hello"
SyntaxError: Missing parentheses in call to 'print'
```

**<u>Note:</u> Programmer is responsible to correct these syntax errors. Once all syntax errors are corrected then only program execution will be started.**

## 2) <u>Runtime Errors:</u>

- **Also known as exceptions.**
- **While executing the program if something goes wrong because of end user input or programming logic or memory problems etc then we will get Runtime Errors.**

**<u>Eg:</u>**

1) **print(10/0) → ZeroDivisionError: division by zero**
2) **print(10/"ten") → TypeError: unsupported operand type(s) for /: 'int' and 'str'**
3) 
```
x = int(input("Enter Number:"))
print(x)

D:\Python_classes>py test.py
Enter Number:ten
ValueError: invalid literal for int() with base 10: 'ten'
```

**<u>Note:</u> Exception Handling concept applicable for Runtime Errors but not for syntax errors**

## What is Exception?

**An unwanted and unexpected event that disturbs normal flow of program is called exception.**

**Eg:**

- **ZeroDivisionError**
- **TypeError**
- **ValueError**

## try:

**Read Data from Remote File locating at London.**
**except FileNotFoundError:**
**use local file and continue rest of the program  normally**

### Without try-except:

```
print("stmt-1")
print(10/0)
print("stmt-3")
```

**Output**

```
stmt-1
ZeroDivisionError: division by zero
```

**Abnormal termination/Non-Graceful Termination**

### With try-except:

```
print("stmt-1")
try:
    print(10/0)
except ZeroDivisionError:
    print(10/2)
print("stmt-3")
```

**Output**

```
stmt-1
5.0
stmt-3
```

**Normal termination/Graceful Termination**

## How to Print Exception Information:

**try:**

```
print(10/0)
except ZeroDivisionError as msg:
    print("exception raised and its description is:",msg)
```

**Output** exception raised and its description is: division by zero

## finally Block:

- It is not recommended to maintain clean up code(Resource Deallocating Code or Resource Releasing code) inside try block because there is no guarentee for the execution of every statement inside try block always.
- It is not recommended to maintain clean up code inside except block, because if there is no exception then except block won't be executed.
- Hence we required some place to maintain clean up code which should be executed always irrespective of whether exception raised or not raised and whether exception handled or not handled. Such type of best place is nothing but finally block.
- Hence the main purpose of finally block is to maintain clean up code.

```
try:
    Risky Code
except:
    Handling Code


finally:
    Cleanup code
```

The speciality of finally block is it will be executed always whether exception raised or not raised and whether exception handled or not handled.

**Case-1:** If there is no exception

```
try:
    print("try")
except:
    print("except")
finally:
    print("finally")
```

**Output**

```
try
finally
```

**Case-2:** If there is an exception raised but handled

```
try:
    print("try")
    print(10/0)
except ZeroDivisionError:
    print("except")
finally:
    print("finally")
```

**Output**

```
try
except
finally
```

**Case-3:** If there is an exception raised but not handled

```
try:
    print("try")
    print(10/0)
except NameError:
    print("except")
finally:
    print("finally")
```

**Output**

```
try
finally
ZeroDivisionError: division by zero(Abnormal Termination)
```

*** **Note:** There is only one situation where finally block won't be executed ie whenever we are using os._exit(0) function.

Whenever we are using os._exit(0) function then Python Virtual Machine itself will be shutdown.In this particular case finally won't be executed.

```
imports
try:
    print("try")
    os._exit(0)
except NameError:
    print("except")
finally:
    print("finally")
```

**Output:** try

**Note:**
**os._exit(0)**
**Where 0 represents status code and it indicates normal termination**
**There are multiple status codes are possible.**