Q1. Describe three applications for exception processing.?

Ans: Exception Processing is important to find exceptions that causes the runtime error. As runtime errors Halt the program execution when exception occurs.

Exception Processing is used in Various Applications of which few examples are:

1. Checking Appropriate use of input in an application
2. Checking for Arithmetic exceptions in mathematical executions
3. Checking File I/O exceptions during File handling

Q2. What happens if you don't do something extra to treat an exception?

Ans: If Exceptions are not handled flow of program will be broken during the run time which might lead to a abnormal termination of the program. Inshort inability of program to handle exceptions will result in crashing of program.

Q3. What are your options for recovering from an exception in your script?

Ans: Python provides **try** and **except** statements for recovering from an exception in your script.

Q4. Describe two methods for triggering exceptions in your script.

Ans: **raise** and **assert** are two methods that can be used to trigger manual exceptions in your script.

- **raise** method triggers an exception if condition provided to it turns out to be True.
- **assert** will let the program to continue execution if condition provided to it turns out to be True else exception will be raised

In [1]:
```
# Example of raise
x = 10
raise Exception(f'X Value Should not exceed 5 The Provided Value of X is {x}')
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
<ipython-input-1-3bda0e4dbc13> in <module>()
      1 # Example of raise
      2 x = 10
----> 3 raise Exception(f'X Value Should not exceed 5 The Provided Value of X is {x}')

Exception: X Value Should not exceed 5 The Provided Value of X is 10
```

In [2]:
```
# Example of assert
assert(2==4), "2 is not equal to 4"
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-2-b3cb1796e5b9> in <module>()
      1 # Example of assert
----> 2 assert(2==4), "2 is not equal to 4"

AssertionError: 2 is not equal to 4
```

Q5. Identify two methods for specifying actions to be executed at termination time, regardless of whether or not an exception exists.?

Ans: Python Provides **else** and **finally** blocks for specifying actions to be executed at termination time, regardless of whether an exceptions exists or not.