**Exception and Error Handling: -**

In Python, an exception is an **error** that occurs during the execution of a program. When an exceptional condition

* such as division by zero or
* different datatypes or
* specific value of file not found or
* trying to access an index that doesn't exist

arises, Python raises an exception. Exceptions disrupt the normal flow of the program and, if not handled properly, can lead to program termination.

**Types of Exceptions:**

Python has a hierarchy of built-in exception classes, each representing different types of errors that can occur during program execution.

Examples include

* **TypeError,**
* **ValueError,**
* **ZeroDivisionError,**
* **IndexError,**

and many more.

**How to handle Exceptions or Errors:?**

The error can be handled by using the below keywords

**(*try, except, finally, else)***

* *The****try****block lets you test a block of code for errors.*
* *The****except****block lets you handle the error.*
* *The****finally****block lets you execute code, regardless of the result of the try- and except blocks.*
* *You can also use the****else****keyword to define a block of code to be executed if no errors were raised:*

*There are different combination using you can handle the error.*

***try:***

***except***

***try:***

***except:***

***finally***

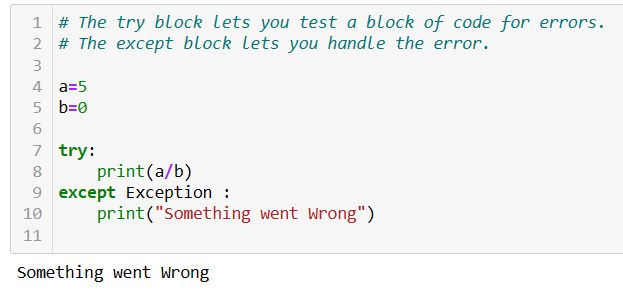
***try:***

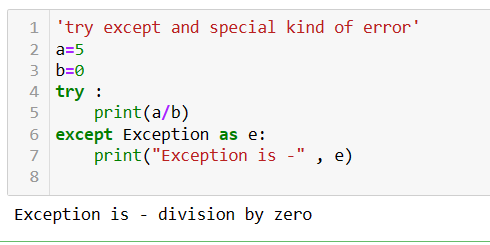
***except:***

***else:***

***try:***

***except***

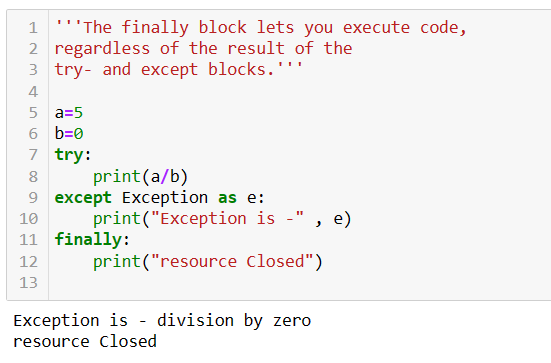


**

***try:***

***except:***

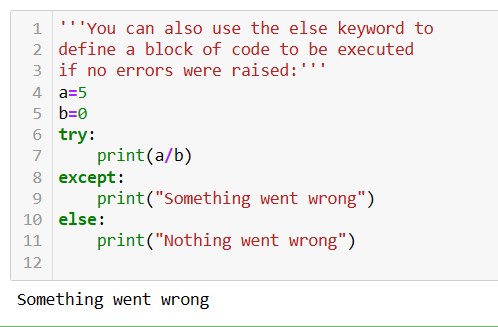
***finally***

**

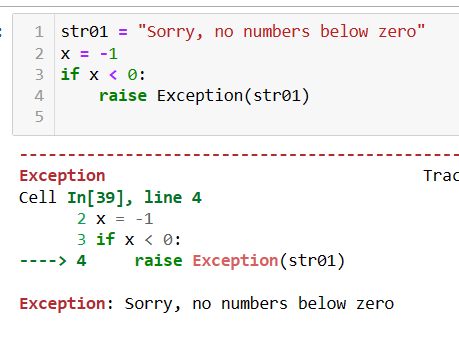
***try:***

***except:***

***else:***

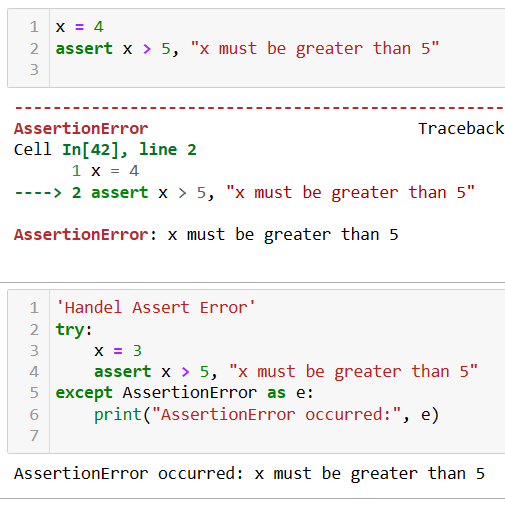


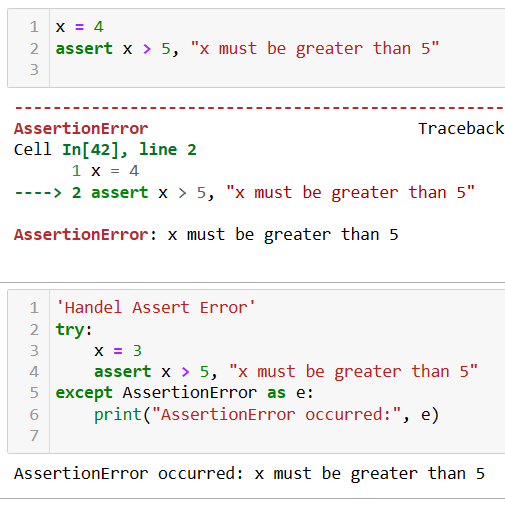
## *Raise an exception As a Python developer, you can choose to throw an exception if a condition occurs. To throw (or raise) an exception, use the raise keyword.*



**Assert:**

*The* ***assert*** *statement is used to check conditions during the program's execution. If the condition is* ***False****, it raises an* ***AssertionError****. You can also include an optional error.*

**

**

**Different types of Errors:**

In Python, many built-in exceptions cover various error scenarios. Here's a list of some common exceptions and their meanings:

**SyntaxError:** Raised when there is a syntax error in the code.

**IndentationError:** Raised when indentation is not properly aligned.

**NameError:** Raised when a local or global name is not found.

**TypeError:** Raised when an operation or function is applied to an object of an inappropriate type.

**ValueError:** Raised when a function receives an argument of the correct type but with an inappropriate value.

**ZeroDivisionError:** Raised when the second operand of a division or modulo operation is zero.

**IndexError:** Raised when a sequence subscript is out of range.

**KeyError:** Raised when a dictionary key is not found.

**FileNotFoundError:** Raised when a file or directory is requested but cannot be found.

**PermissionError:** Raised when trying to access a resource without the required permissions.

**ImportError:** Raised when an import statement fails to find the module definition.

**AttributeError:** Raised when an attribute reference or assignment fails.

**AssertionError:** Raised when an assert statement fails.

**RuntimeError:** Raised when an error is detected that doesn't fall into any of the other categories.

These are just a few examples of the built-in exceptions in Python. Each exception serves a specific purpose and helps to diagnose and handle errors effectively during program execution.