Python Programming

Exception Handling

Agenda :

- Exception Handling
- Syntax Errors
- Built in exceptions
- Raise Exception
- Handling exceptions

Exception Handling :

Sometimes while executing a Python program, the program does not execute at all or the program executes but generates unexpected output or behaves abnormally.

These occur when there are syntax errors, runtime errors or logical errors in the code.

Syntax Errors :

Syntax errors are detected when we have not followed the rules of the particular programming language while writing a program.

These errors are also known as parsing errors.

On encountering a syntax error, the interpreter does not execute the program unless we rectify the errors, save and rerun the program.

Exceptions :

Even if a statement or expression is syntactically correct, there might arise an error during its execution. (opening a file that does not exists)

Such types of errors might disrupt the normal execution of the program and are called exceptions.

Exceptions need to be handled by the programmer so that the program does not terminate abnormally.

Therefore, while designing a program, a programmer can address them by including appropriate code to handle that exception.

Built-in Exceptions:

Commonly occurring exceptions are usually defined in the compiler /interpreter.

These are called built-in exceptions.

Built-in exceptions in Python :

SyntaxError: raised when there is an error in the syntax of the Python code.

ValueError : when a built-in method or operation receives an argument that has the right data type but mismatched or inappropriate values.

IOError : raised when the file specified in a program statement cannot be opened.

KeyboardInterrupt : raised when the user accidentally hits the Delete or Esc key.

ImportError : raised when the requested module definition is not found.

EOFError : raised when the end of file condition is reached without reading any data by input().

ZeroDivisionError : raised when the denominator in a division operation is zero.

IndexError : raised when the index or subscript in a sequence is out of range.

NameError : raised when a local or global variable name is not defined.

IndentationError : raised due to incorrect indentation in the program code.

TypeError : raised when an operator is supplied with a value of incorrect data type.

OverFlowError : raised when the result of a calculation exceeds the maximum limit for numeric data type.

The raise Statement :

The raise statement can be used to throw an exception.

The syntax of raise statement is:

raise exception-name[(optional argument)]

The argument is generally a string that is displayed when the exception is raised.

Eg - raise Exception(“Oops! An exception has occurred”)

– Exception: Oops! An exception has occurred

Handling Exceptions:

This is done by writing additional code in a program to give proper messages or instructions to the user on encountering an exception. This process is known as exception handling.

Process of Handling Exception:

When an error occurs, Python interpreter creates an object called the exception object.

This process of creating an exception object and handing it over to the runtime system is called **throwing** an exception.

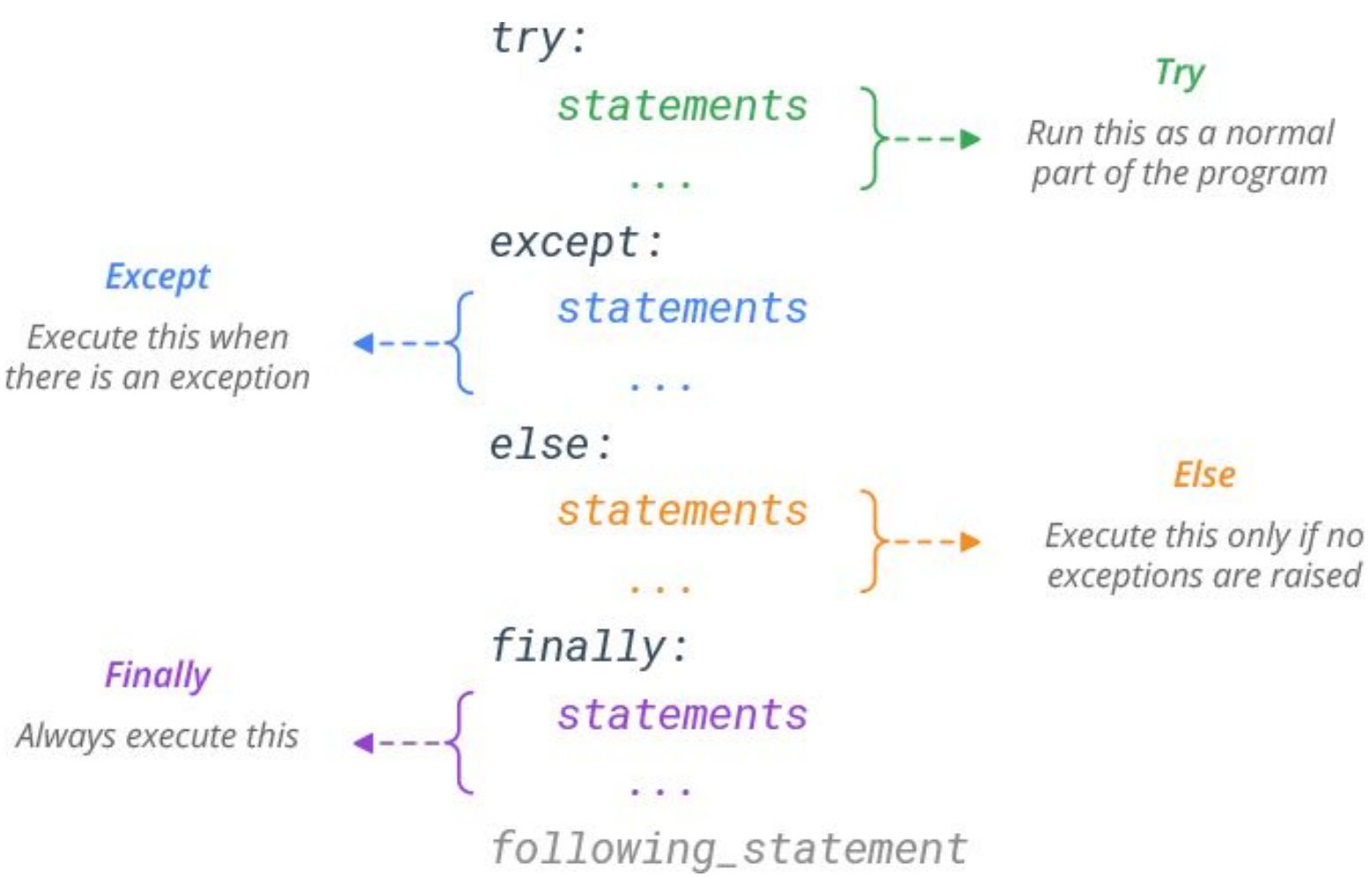
The runtime system searches the entire program for a block of code, called the exception handler that can handle the raised exception.

Hierarchical search in reverse order continues till the exception handler is found. This entire list of methods is known as call stack.

When a suitable handler is found in the call stack, it is executed by the runtime process.

This process of executing a suitable handler is known as catching the exception.

If the runtime system is not able to find an appropriate exception after searching all the methods in the call stack, then the program execution stops.


try:
statements
...
Try
Run this as a normal part of the program
except:
statements
...
Except
Execute this when there is an exception
else:
statements
...
Else
Execute this only if no exceptions are raised
finally:
statements
...
Finally
Always execute this
following_statement

Catching Exceptions :

An exception is said to be caught when a code that is designed to handle a particular exception is executed. Exceptions, if any, are caught in the try block and handled in the except block.

Finally Clause:

The statements inside the finally block are always executed regardless of whether an exception has occurred in the try block or not

Thank You