Exception in Python

Exception

An exception can be defined as an unusual condition in a program resulting in the interruption in the flow of the program.

Python has many **built-in exceptions** that enable our program to run without interruption and give the output. These exceptions are given below:

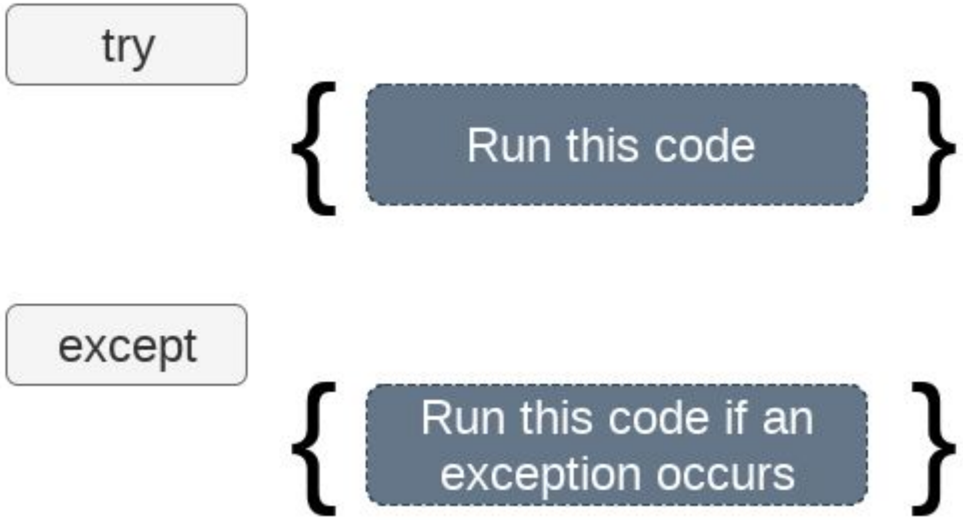
Common Exceptions

1. **ZeroDivisionError:** Occurs when a number is divided by zero.
2. **NameError:** It occurs when a name is not found. It may be local or global.
3. **IndentationError:** If incorrect indentation is given.
4. **IOError:** It occurs when Input Output operation fails.
5. **EOFError:** It occurs when the end of the file is reached, and yet operations are being performed.

The try-expect statement

If the Python program contains suspicious code that may throw the exception, we must place that code in the try block.

The try block must be followed with the except statement, which contains a block of code that will be executed if there is some exception in the try block.



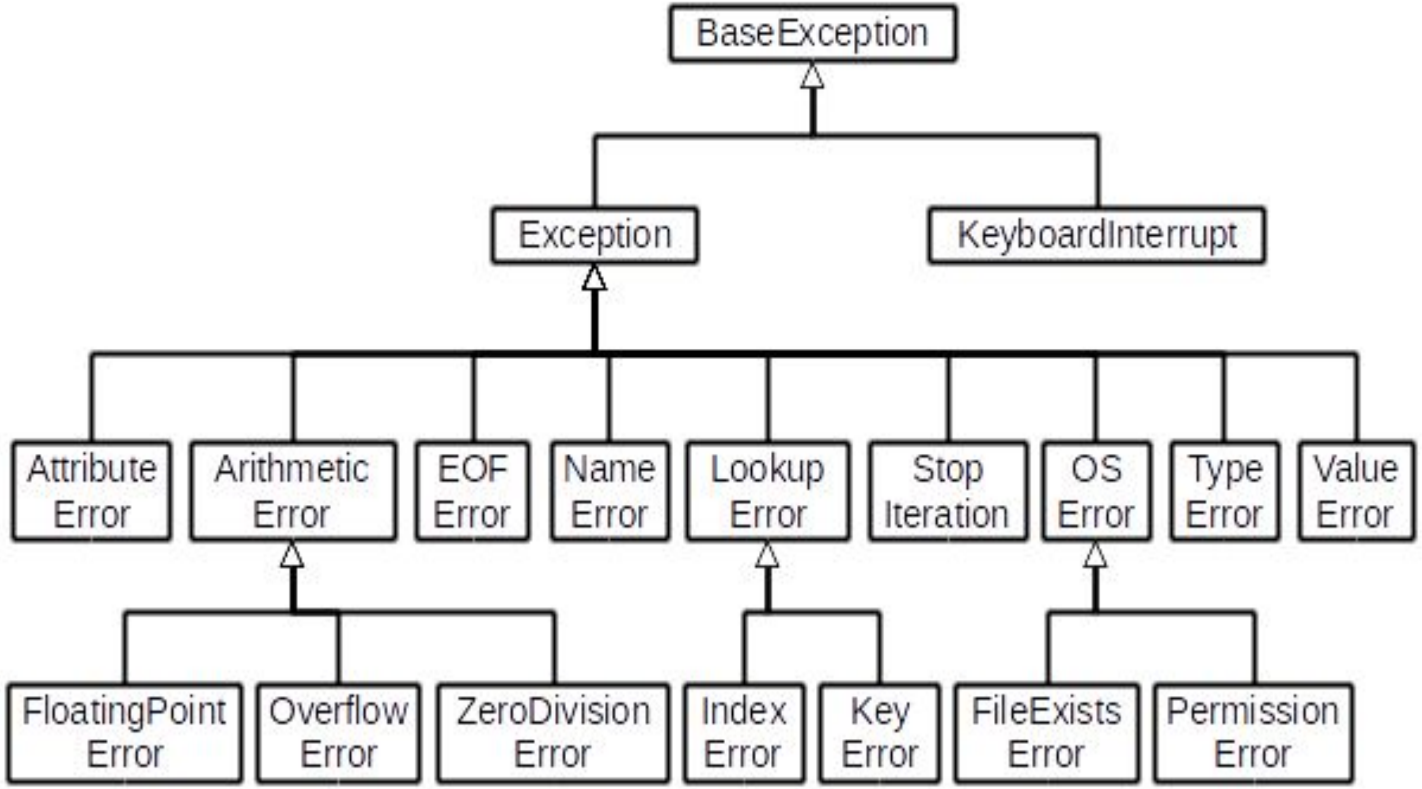
Syntax

```
try:
    #block of code

except Exception1:
    #block of code

except Exception2:
    #block of code

#other code
```

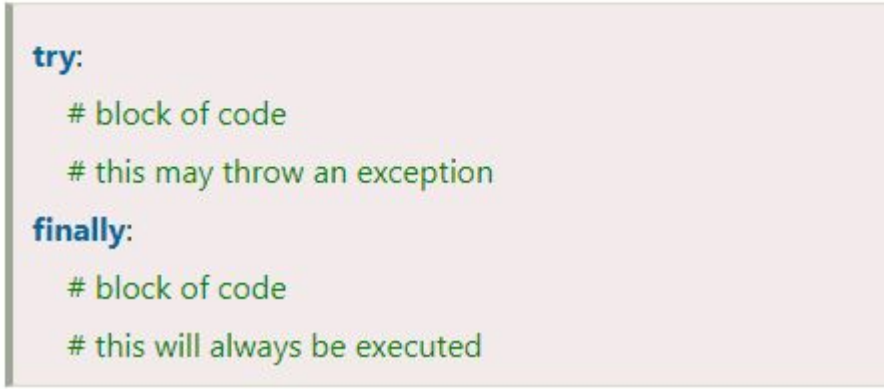

BaseException
Exception
KeyboardInterrupt
Attribute Error
Arithmetic Error
EOF Error
Name Error
Lookup Error
Stop Iteration
OS Error
Type Error
Value Error
FloatingPoint Error
Overflow Error
ZeroDivision Error
Index Error
Key Error
FileExists Error
Permission Error

Python provides the optional **finally** statement, which is used with the **try** statement. It is executed no matter what exception occurs and used to release the external resource. The finally block provides a guarantee of the execution.

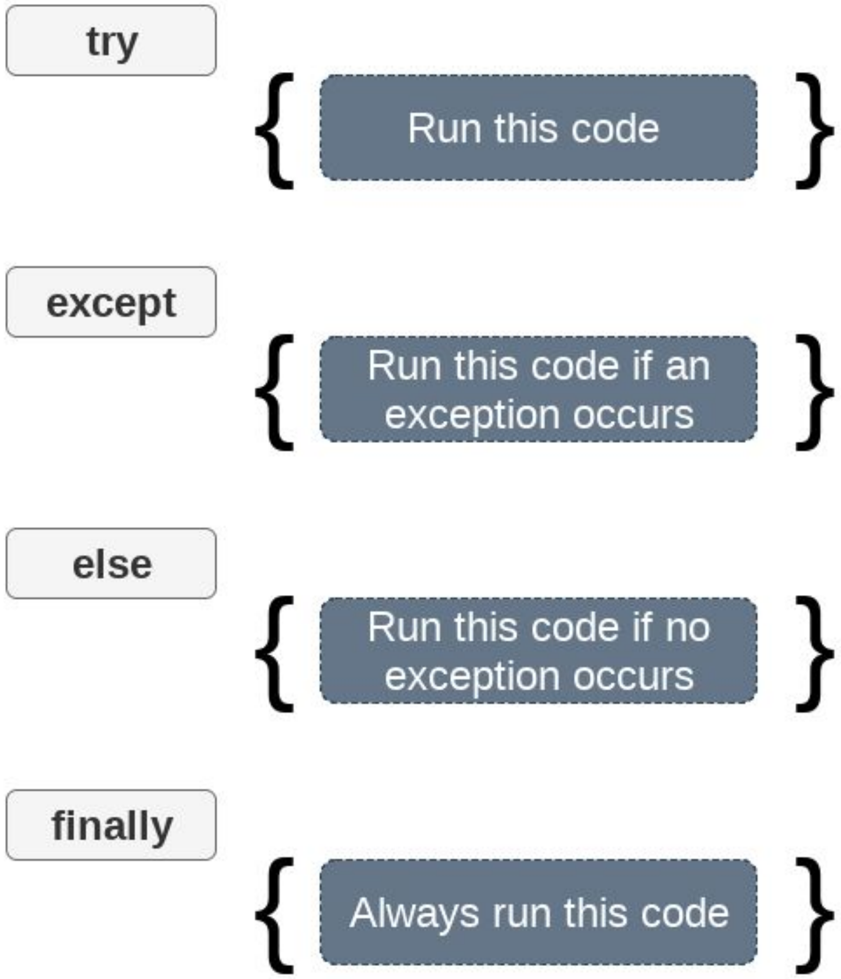
We can use the finally block with the try block in which we can pace the necessary code, which must be executed before the try statement throws an exception.

The syntax to use the finally block is given below.

Syntax



```
try:
    # block of code
    # this may throw an exception
finally:
    # block of code
    # this will always be executed
```


try
Run this code
except
Run this code if an exception occurs
else
Run this code if no exception occurs
finally
Always run this code

The try...finally block

Python provides the optional finally statement, which is used with the try statement. It is executed no matter what exception occurs and used to release the external resource. The finally block provides a guarantee of the execution.

We can use the finally block with the try block in which we can pace the necessary code, which must be executed before the try statement throws an exception.

Syntax

```
raise Exception_class,<value>
```

Points to remember

1. To raise an exception, the raise statement is used. The exception class name follows it.
2. An exception can be provided with a value that can be given in the parenthesis.
3. To access the value "**as**" keyword is used. "**e**" is used as a reference variable which stores the value of the exception.
4. We can pass the value to an exception to specify the exception type.

Example

```
try:
    age = int(input("Enter the age:"))
    if(age<18):
        raise ValueError
    else:
        print("the age is valid")
except ValueError:
    print("The age is not valid")
```