Full stack web development using python

Exception handling

Saurabh Shukla (MySirG)

Agenda

1. Exception
2. Types of Exceptions
3. Builtin exception
4. Handling exception

Exception

Even if a statement is syntactically correct it may cause an error during execution, called exceptions.

After exception occurs in the program , program terminate and not execute below code.

Examples

1. `a = int(input("Enter a number"))`
   if input is `'abc'`
   ValueError
2. IndexError
3. AttributeError
4. ZeroDivisionError
5. KeyError
6. NameError
7. TypeError

these all error are define in the python iin the individual class.

Two types of Exceptions

1. Built-in Exceptions
2. User defined Exception

Built-in Exceptions

There are several built-in exceptions in Python that are raised when error occurs.

These built-in exceptions can be viewed using the local() built-in functions as follows:

```
locals()['__builtins__']
```

If we don't handle implictly exception error, then python handle it and after exception occurs ,it terminate the program.It is the problem that's why we require to hndle the exception

```
for x in locals()['__builtins__'].__dict__:
    print(locals()['__builtins__'].__dict__[x])
```

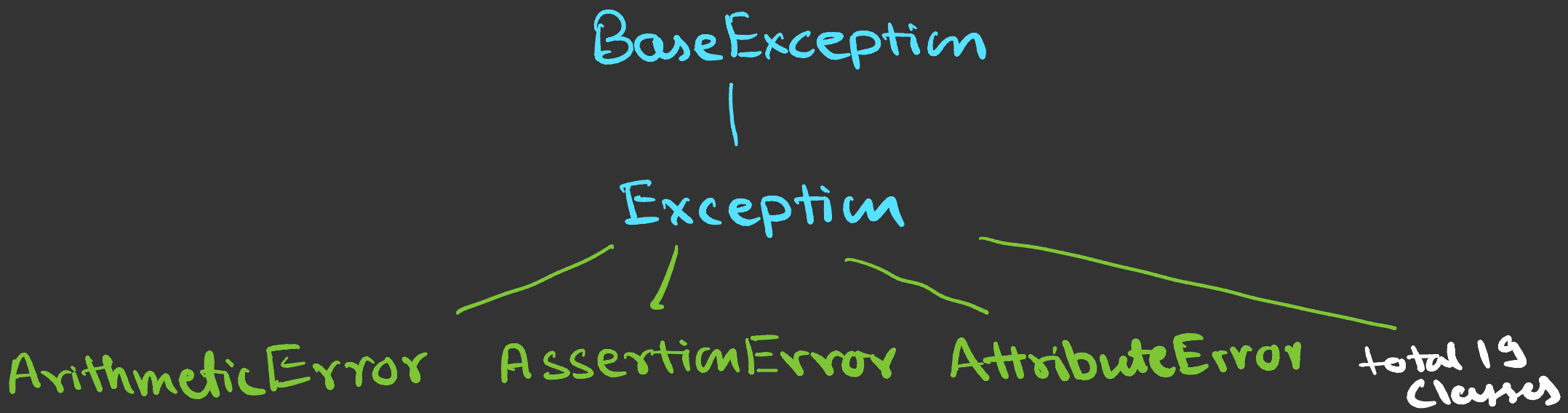
You will see all the builtin methods, classes and exception classes.

BaseException

BaseException is the base class for all built-in exceptions.

It is not meant to be directly inherited by user defined classes.

For user-defined exception, class Exception is used.


BaseException
Exception
ArithmeticError
AssertionError
AttributeError
total 19
Classes

Handling Exception

```
try:
    a = int(input("Enter a number"))
    print(a)
except ValueError as e:
    print(e)
```

- put the code in try block, which you think may cause an exception
- If no exception occurs, then the except block is skipped.

```
try:
    ===
except ExceptionClass :
    ==
except ExceptionClass :
    ==

finally:
    ==
```

- If an exception occurs, the rest of the try clause is skipped, then if its type matches the exception named after the except keyword, the except clause is executed and then execution continues after try statement.

```
try:
    ==
    ==
except ExceptionClass :
    ==
except ExceptionClass :
    ==
    ==
finally:
    ==
```

```
try:
    ==
except ExceptionClass :
    ==
except ExceptionClass :
    ==
finally:
    ==
```

If an exception occurs which doesnot match the exception named in the except clause, it is passed on to the next except clause. This will go on

If none of the except matches it proceeds with finally block and then Python error message displayed and program terminates.

If any except matches, execute it and skipping all other except clauses, jumped to finally, then rest of the script continues.

```
try:
    ===
except ExceptionClass :
    ===
except ExceptionClass :
    ===

finally:
    ===
```

with one try block, you can have any number of except and one finally.

except and finally are optional with try block, but at least one must be followed by try block

finally always executed regardless of exception occurrence

---->>>>> Finally run always

```
try:
    ==
except ExceptionClass :
    ==
except ExceptionClass :
    ==
except :
    ==
finally:
    ==
```

Default except clause must be the last

if we can't be able to hadle all exception, then write default it handle all except

```
try:
    ===
except (ExceptionClass1, ExceptionClass2):
    ===
except ExceptionClass :
    ===
except:
    ===
finally:
    ===
```

An except clause may name multiple exceptions as a parenthesized tuple.

```
try:
    ==
except (ExceptionClass1, ExceptionClass2):
    ==
except ExceptionClass :
    ==
except:
    ==
else:
    ==
finally:
    ==
```

The try except statement has an optional else clause, which when present, must follow all except clauses, else will execute when try doesn't raises an exception

```
try:
    ==
except (ExceptionClass1, ExceptionClass2):
    ==
except ExceptionClass :
    ==
except:
    ==
else:
    ==
finally:
    ==
```

finally always executed, else only executed when no exception raised