Coding & Solutioning

Exception Handling

Date	19September 2022
Team ID	PNT2022TMID43270
Project Name	Signs with smart Connectivity For Better Road Safety
Maximum Marks	2 Marks

Difference between Syntax Error and Exceptions

Syntax Error: As the name suggests this error is caused by the wrong syntax in the code. It leads to the termination of the program.

```
# initialize the amount variable

amount = 10000


# check that You are eligible to

# purchase Dsa Self Paced or not

if(amount > 2999)

print("You are eligible to purchase Dsa Self Paced")
```

OUTPUT:

```
  File "/home/ac35380186f4ca7978956ff46697139b.py", line 4
    if(amount>2999)
                  ^
SyntaxError: invalid syntax
```

Exceptions: Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

Example:

- Python3

```
# initialize the amount variable

marks = 10000



# perform division with 0

a = marks / 0

print(a)
```

OUTPUT:

```
Traceback (most recent call last):
  File "/home/f3ad05420ab851d4bd106ffb04229907.py", line 4, in <module>
    a=marks/0
ZeroDivisionError: division by zero
```

In the above example raised the ZeroDivisionError as we are trying to divide a number by 0.

Note: Exception is the base class for all the exceptions in Python. You can check the exception hierarchy

Try and Except Statement – Catching Exceptions

Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

Example: Let us try to access the array element whose index is out of bound and handle the corresponding exception.

- Python3

```
# Python program to handle simple runtime error
```

```
#Python 3



a = [1, 2, 3]

try:

    print ("Second element = %d" %(a[1]))



    # Throws error since there are only 3 elements in array

    print ("Fourth element = %d" %(a[3]))



except:

    print ("An error occurred")
```

```
Second element = 2
An error occurred
```

Catching Specific Exception

A try statement can have more than one except clause, to specify handlers for different exceptions. Please note that at most one handler will be executed. For example, we can add IndexError in the above code. The general syntax for adding specific exceptions are –

```
try:
    # statement(s)
except IndexError:
    # statement(s)
except ValueError:
```

```
    # statement(s)
```

Example: Catching specific exception in Python

```
# Program to handle multiple errors with one
# except statement
# Python 3

def fun(a):
        if a < 4:

                # throws ZeroDivisionError for a = 3
                b = a/(a-3)

        # throws NameError if a >= 4
        print("Value of b = ", b)

try:
        fun(3)
        fun(5)

# note that braces () are necessary here for
# multiple exceptions
except ZeroDivisionError:
        print("ZeroDivisionError Occurred and Handled")
except NameError:
        print("NameError Occurred and Handled")
```

ZeroDivisionError Occurred and Handled

Finally Keyword in Python
Python provides a keyword finally, which is always executed after the try and except blocks. The final block always executes after normal termination of try block or after try block terminates due to some exception.
Syntax:
try:

```
    # Some Code....


except:

    # optional block

    # Handling of exception (if required)


else:

    # execute if no exception


finally:

    # Some code .....(always executed)
```

Example:

- Python3

```
# Python program to demonstrate finally


# No exception Exception raised in try block

try:

    k = 5//0  # raises divide by zero exception.
```

```
    print(k)


# handles zerodivision exception
except ZeroDivisionError:
    print("Can't divide by zero")


finally:
    # this block is always executed
    # regardless of exception generation.
    print('This is always executed')
```

Output:

```
Can't divide by zero
This is always executed
```