Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

EXCEPTION HANDLING - 4

Guido Van Rossum

Agenda:

1. Important Interview Question
2. Control flow in try-except-finally blocks
3. Nested try-except-finally blocks
4. Control flow in nested try-except-finally

12. Important Interview Question

Q. What is the difference between Destructor and finally block?

- finally block meant for maintaining clean up code.
- Destructor also meant for maintaining clean up code.

Seems to be both are same, Where is the difference????????

finally block

- Used for clean up activities related to try block (i.e., Whatever resources we opened as the part of try block will be closed inside finally block).

Destructor

- Used for clean up activities related to object (i.e., Whatever resources associated with the object should be deallocated inside destructor, which will be executed before destroying the object by the garbage collector).

13. Control flow in try-except-finally blocks

Consider the following case,

```
try:
    stmt-1
    stmt-2
    stmt-3
except:
   stmt-4
finally:
   stmt-5
stmt6
```

Case-1: If there is no exception

- 1,2,3,5,6 Normal Termination

Case-2: If an exception raised at stmt-2 and the corresponding except block matched

- 1,4,5,6 Normal Termination

Case-3: If an exception raised at stmt-2 but the corresponding except block not matched

- 1,5 Abnormal Termination

Case-4:If an exception raised at stmt-4 then it is always abnormal termination but before that finally block will be executed.

Note:

- Any statement outside the try block (i.e., No except block) raised an exception, which always leads to an Abnormal termination.

Case-5: If an exception raised at stmt-5 or at stmt-6 then it is always abnormal termination

14. Nested try-except-finally blocks

- We can take try-except-finally blocks inside try or except or finally blocks.(i.e., nesting of try-except-finally is possible).

```
try:
        ----------
        ----------
        ----------
    try:
        -------------
            --------------
            --------------
    except:
        --------------
        --------------
        --------------
    ------------
except:
        -----------
        -----------
        -----------
```

Key Points:

- General Risky code we have to take inside outer try block and too much risky code we have to take inside inner try block.
- Inside Inner try block if an exception raised then inner except block is responsible to handle. If it is unable to handle then outer except block is responsible to handle.

Demo Program 1:

```
try:
    print("outer try block")
    try:
        print("Inner try block")
    except ZeroDivisionError:
        print("Inner except block")
    finally:
        print("Inner finally block")
except:
    print("outer except block")

finally:
    print("outer finally block")
```

```
outer try block
Inner try block
Inner finally block
outer finally block
```

Demo Program 2:

```
try:
    print("outer try block")
    try:
        print("Inner try block")
        print(10/0)  #Inner try raising an exception
    except ZeroDivisionError:
        print("Inner except block")
    finally:
        print("Inner finally block")
except:
    print("outer except block")

finally:
    print("outer finally block")
```

```
outer try block
Inner try block
Inner except block
Inner finally block
outer finally block
```

Demo Program 3:

```
try:
    print("outer try block")
    try:
        print("Inner try block")
        print(10/0)  #Inner try raising an exception
    except ValueError:
        print("Inner except block")
    finally:
        print("Inner finally block")
except:
    print("outer except block")

finally:
    print("outer finally block")
```

```
outer try block
Inner try block
Inner finally block
outer except block
outer finally block
```

Demo Program 4:

```
try:
    print("outer try block")
    print(10/0) # Outer try raising an exception
    try:
        print("Inner try block")

    except ValueError:
        print("Inner except block")
    finally:
        print("Inner finally block")
except:
    print("outer except block")

finally:
    print("outer finally block")
```

```
outer try block
outer except block
outer finally block
```

Important Conclusions with respect to Demo Program 4:

1. If the control not entered in the try block, then corresponding finally block won't be executed.

2. Once control entered in the try block, compulsory the corresponding finally block will be executed.

Demo Program 5:

```
try:
    print("outer try block")
    try:
        print("Inner try block")
        print(10/0)
    except ZeroDivisionError:
        print(10/0)
        print("Inner except block")
    finally:
        print("Inner finally block")
except:
    print("outer except block")

finally:
    print("outer finally block")
```

```
outer try block
Inner try block
Inner finally block
outer except block
outer finally block
```

15. Control flow in nested try-except-finally

Consider the following case,

```
try:
        stmt-1
        stmt-2
        stmt-3
        try:
                stmt-4
                stmt-5
                stmt-6
        except X:
                stmt-7
        finally:
                stmt-8
        stmt-9
except Y:
        stmt-10
finally:
        stmt-11
stmt-12
```

Case 1: If there is no exception

- 1,2,3,4,5,6,8,9,11,12 Normal Termination

Case 2: If an exception raised at stmt-2 and the corresponding except block matched

- 1,10,11,12 Normal Termination

Case 3: If an exception raised at stmt-2 and the corresponding except block not matched

- 1,11,Abnormal Termination

Case 4: If an exception raised at stmt-5 and inner except block matched

- 1,2,3,4,7,8,9,11,12 Normal Termination

Case 5: If an exception raised at stmt-5 and inner except block not matched but outer except block matched

- 1,2,3,4,8,10,11,12,Normal Termination

Case 6: If an exception raised at stmt-5 and both inner and outer except blocks are not matched

- 1,2,3,4,8,11,Abnormal Termination

Case 7: If an exception raised at stmt-7 and corresponding except block matched

- 1,2,3,.,.,.,8,10,11,12,Normal Termination (4,5,6 may or may not executed)

Case 8: If an exception raised at stmt-7 and corresponding except block not matched

- 1,2,3,.,.,.,8,11,Abnormal Termination

Case 9: If an exception raised at stmt-8 and corresponding except block matched

- 1,2,3,.,.,.,.,10,11,12 Normal Termination

Case 10: If an exception raised at stmt-8 and corresponding except block not matched

- 1,2,3,.,.,.,.,11,Abnormal Termination

Case-11: If an exception raised at stmt-9 and corresponding except block matched

- 1,2,3,.,.,.,.,.,8,10,11,12,Normal Termination

Case-12: If an exception raised at stmt-9 and corresponding except block not matched

- 1,2,3,.,.,.,.,.,8,11,Abnormal Termination

Case-13: If an exception raised at stmt-10 then it is always abnormal termination but before abnormal termination finally block(stmt-11) will be executed.

Note:

- Statement 10 is not part of try block, so there is no except block. If there is no except block means, it leads to Abnormal termination only. But before that the corresponding finally block(i.e., stmt -11) will be executed.
- In the above case, to reach stmt - 9, multiple options are there, anywhere is exception (i.e.,from stmt -1 to stmt - 9), control reach to statement 10.

Case-14: If an exception raised at stmt-11 or stmt-12 then it is always abnormal termination.

Note:

- If the control entered into try block then compulsory finally block will be executed. If the control not entered into try block then finally block won't be executed.

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You