

Exception Handling

Python Programming

Dr.Selva Rani B

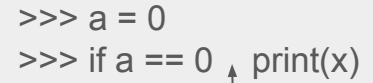
What is an exception?

★ is a special type of Python object that can be used to pass information when a failure occurs

★ Types of errors:

- Syntax Errors - violation of rules of Python while writing the coding

```
>>> a = 0  
>>> if a == 0 print(x)
```



- Logical Errors - program executes but results in incorrect results

- Divide by zero
- Accessing an item in a list outside its boundary

Some predefined/standard exceptions

ZeroDivisionError	raised when dividing a number by 0 (zero)
NameError	raised when a variable is not found
TypeError	raised when an operation is applied on an object which is of inappropriate
ImportError	raised when an import statement fails
IndexError	raised when trying to access an element from a sequence with out of boundary
ValueError	raised when a function receives a valid argument but an inappropriate value
IOError	raised when an I/O operation fails

Handling Exceptions

- With the help of *try* block and *except* block
- A critical execution statement which may raise exception can be placed inside the *try* block
- The action that handles the exception can be placed in the *except* block
- Syntax:

```
try:  
    statement(s)  
  
except ExceptionName:  
    statement(s)
```

Multiple `except` Blocks

- Can have multiple `except` blocks for a single `try` block
- The `except` block which matches the exception thrown from the `try` block will be executed

```
try:  
    statement(s)  
except Exception-1:  
    statement block-1  
except Exception-2:  
    statement block-2  
except Exception-3:  
    statement block-3  
else:  
    statement block-else
```

Multiple Exceptions in a Single Block

- Can have multiple **exceptions** in a single **except** block as a tuple
- if the exception raised from the **try** block matches any one of the **exceptions**, the same **except** block will be executed

```
try:  
    statement(s)  
except (Exception-1, Exception-2, Exception-3):  
    statement(s)  
else:  
    statement(s)
```

except Block without Exception

- Can have an **except** block without any exception
- In larger programs, it will be difficult to produce all types of exception handling
- Can be done with only one exception which can serve as a wildcard

```
try:  
    statement(s)  
except:  
    statement(s)  
else:  
    statement(s)
```

finally block

- *try* block may have an optional block *finally*
- Can be used to define clean-up actions that should be executed under all circumstances
- Will always be executed before leaving the *try* block
- Irrespective of whether exception occurs or not, the statements inside the *finally* block are executed

```
try:
    statement(s)
except ExceptionName:
    statement(s)
finally:
    statement(s)
```