Exception is a type or class, generating exception is nothing but creating object of exception class and giving to PVM (python virtual machine). Exception is generated by a function or method, when given wrong input.

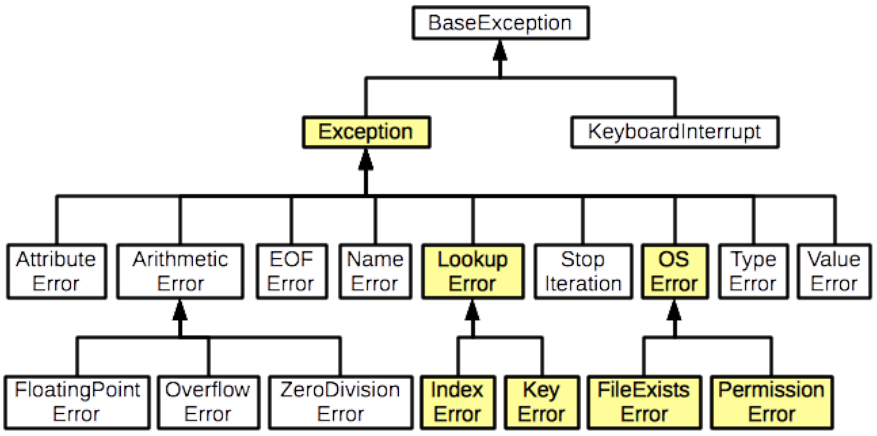
Exception types or Error types are 2

1. Predefined exceptions OR predefined error types
2. User defined exceptions OR user defined error types

Predefined exceptions are predefined error classes, these error classes or types used by python libraries or functions.

The error types build by programmer are called user defined error types.

Hierarchy of exception classes



Keywords used for handling errors/exceptions

1. try
2. except
3. finally
4. raise
5. assert

try keyword or try block

try block contains statements which has to be monitored for error handling or exception handling (OR) try block contain statements which generate error during runtime.

Syntax:

```
try:
    statement-1
    statement-2
    statement-3
```

except block or except keyword

if there is an error inside try block, that error is handled by except block.
Except block is error handler block.
Try block followed by one or more except blocks.

Syntax:

```
try:
    statement-1
    statement-2
except  <error-type>:
     statement-1
except <error-type>:
     statement-1
```

Example:

```
# Division of two numbers

while True:
    try:
        a=int(input("Enter first number "))
        b=int(input("Enter second number "))
        c=a/b
        print(f'division of {a}/{b}={c:.2f}')
        break
    except ZeroDivisionError:
        print("cannot divide number with zero")
```

Output

```
Enter first number 5
Enter second number 0
cannot divide number with zero
```

```
Enter first number 7
Enter second number 2
division of 7/2=3.50
```

try block with multiple except blocks

if try block generates multiple types of exceptions, it is handled using multiple except blocks.

Example

```
# Division of two numbers

while True:
    try:
        a=int(input("Enter first number "))
        b=int(input("Enter second number "))
        c=a/b
        print(f'division of {a}/{b}={c:.2f}')
        break
    except ZeroDivisionError:
        print("cannot divide number with zero")
    except ValueError:
        print("values must be integer type")
```

Output

```
Enter first number 5
Enter second number 0
cannot divide number with zero
Enter first number 9
Enter second number 1.5
values must be integer type
Enter first number abc
values must be integer type
Enter first number 5
Enter second number 3
division of 5/3=1.67
```

if there is no error in try block, python does not execute except block.

Except block with multiple types of exceptions

One except block is able to handle multiple types of exceptions.

Syntax:

```
try:
    statement-1
    statement-2
except (exception-type, exception-type,..):
    statement-1
```

Example:

```
# Division of two numbers

while True:
   try:
       a=int(input("Enter First Number "))
       b=int(input("Enter Second Number "))
       c=a/b
       print(f'division of {a}/{b}={c}')
       break
   except (ZeroDivisionError,ValueError):
       print("'values must be integer and divisor
should not be zero"')
```

Output

```
Enter First Number 5
Enter Second Number abc
values must be integer and divisor should not
                be zero
Enter First Number 6
Enter Second Number 2
division of 6/2=3.0
```

Generic except block

Except block without type is called generic except block. This except block is able to handle any type of error.

Syntax:

```
try:
    statement-1
    statement-2
```

```
except:
    statement-3
```

Example:

```
# Division of two numbers
import sys
while True:
   try:
      a=int(input("Enter First Number "))
      b=int(input("Enter Second Number "))
      c=a/b
      print(f'division of {a}/{b}={c}')
      break
   except:
      t=sys.exc_info()
      print(t[0])
      print(t[1])
```

Output

```
Enter First Number 5
Enter Second Number 0
<class 'ZeroDivisionError'>
division by zero
Enter First Number 8
Enter Second Number abc
<class 'ValueError'>
invalid literal for int() with base 10: 'abc'
Enter First Number 9
Enter Second Number 2
division of 9/2=4.5
```

sys.exc_info()

This function returns the old-style representation of the handled exception. If an exception e is currently handled (so exception() would return e), exc_info() returns the tuple (type(e), e, e.__traceback__)

finally keyword or finally block

finally block is not an exception handler.
Finally block is used to de-allocate resources allocated within try block

Finally block contains statements which are executed after execution of try block and except block.

Syntax-1:	**Syntax-2:**	**Syntax-3 (Wrong)**

try:
  statement-1
  statement-2
except <error-type>:
  statement-1
finally:
  statement-1
	try:
	  statement-1
	  statement-2
	finally:
	  statement-1
		try:
		  statement-1
		  statement-2
		except:
		  statement-1
		finally:
		  statement-1
		finally:
		  statement-1