Exception Handling

```
# ZeroDivisionError without try except blocks
a = 4
b = 2
c = 0
print(a/b) # 2.0
print('Statement 01') # Statement 01
print(a/c) # ZeroDivisionError: division by zero
print('Statement 02')
```

```
ZeroDivisionError: division by zero
2.0
Statement 01
```

```
# ZeroDivisionError without try except blocks
try:
    a = 4
    b = 2
    c = 0
    print(a / b) # 2.0
    print('Statement 01') # Statement 01
    print(a / c)
    print('Statement 02')
except ZeroDivisionError:
    print('Expect Block') # Expect Block
print('End of the Code') # End of the Code
```

```
2.0
Statement 01
Expect Block
End of the Code
```

```
# IndexError without try and except blocks
l = [1,2,3,4,5]
print('Statement 01') # Statement 01
print(l[6]) # IndexError: list index out of range
print('Statement 02')
```

```
IndexError: list index out of range
Statement 01
```

```
# IndexError with try and except blocks
try:
    l = [1, 2, 3, 4, 5]
    print('Statement 01') # Statement 01
    print(l[6])
    print('Statement 02')
except IndexError:
    print('Expect Block') # Expect Block
print('End of the code') # End of the code
```

```
Statement 01
Expect Block
End of the code
```

```
# Base Exception
try:
    a = 4
    b = 2
    c = 0
    print(a / b) # 2.0
    print('Statement 01') # Statement 01
    print(a / b) # 2.0
    print('Statement 02') # Statement 02

    l = [1, 2, 3, 4, 5]
    print('Statement 03') # Statement 03
    print(l[6]) # error
    print('Statement 04')

except BaseException:
    print('Expect Block') # Expect Block

else:
    print('Else Block')
print('End of the Code') # End of the Code
```

```
2.0
Statement 01
2.0
Statement 02
Statement 03
Expect Block
End of the Code
```

```
# Multiple except blocks
class Hello:
    try:
        s = {1,2,3,4,5,6,7,8,9,10,1,2}
        print(s) # {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

        print(s[15]) # error
        a = 10
        b = 2
        c = 0
        print(a/c) # error

    except TypeError:
        print('Expect Block IndexError')

    except ZeroDivisionError:
        print('Expect Block ZeroDivisionError')

    finally:
        print('Finally Block') # Finally Block

print('End of the Code') # End of the Code
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Expect Block IndexError

Finally Block

End of the Code
```

```
raise without exception
class EmployeeSalary(Exception):

    def CheckSalary(self, eSalary):

        if (eSalary <= 50000):
            print('Employee Salary is 50,000')
        else:
            raise EmployeeSalary('Employee Salary Out of Range')

e = EmployeeSalary()
e.CheckSalary(50000)
print('Employee Verified')
```

```
Employee Salary is 50,000
Employee Verified
```

```
raise with exception
class EmployeeSalary(Exception):

    def CheckSalary(self, eSalary):

        if (eSalary <= 50000):
            print('Employee Salary is 50,000')
        else:
            raise EmployeeSalary('Employee Salary Out of Range')

e = EmployeeSalary()
e.CheckSalary(60000)
print('Employee Verified')
```

```
File "D:\Python Github\PythonWorkspace\30_ExceptionHandling\Ex5.py", line 8, in
CheckSalary
    raise EmployeeSalary('Employee Salary Out of Range')
__main__.EmployeeSalary: Employee Salary Out of Range
```

raise

```
class CustomError(Exception):

   def d1(self):
      a = int(input("Enter a Number: "))
      if a == 10:
         print('No Error')
      else:
         raise CustomError('Number not validated')

c = CustomError()
c.d1()
```

```
Enter a Number: 20
Traceback (most recent call last):
  File "D:\Python Github\PythonWorkspace\30_ExceptionHandling\Ex6.py", line 11, in
<module>
    c.d1()
  File "D:\Python Github\PythonWorkspace\30_ExceptionHandling\Ex6.py", line 8, in d1
    raise CustomError('Number not validated')
__main__.CustomError: Number not validated
```

```
Enter a Number: 10
No Error
```

```
import builtins

print(dir(builtins))

# ['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError',
'BrokenPipeError', 'BufferError',
#  'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError',
'ConnectionRefusedError', 'ConnectionResetError',
#  'DeprecationWarning', 'EOFError', 'Ellipsis', 'EncodingWarning', 'EnvironmentError',
'Exception', 'False', 'FileExistsError',
#  'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError',
'ImportError', 'ImportWarning',
#  'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError',
'KeyboardInterrupt', 'LookupError',
#  'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError',
'NotImplemented', 'NotImplementedError',
#  'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError',
'ProcessLookupError', 'RecursionError',
#  'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning',
'StopAsyncIteration', 'StopIteration', 'SyntaxError',
#  'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError',
'UnboundLocalError',
#  'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError',
'UnicodeWarning', 'UserWarning',
#  'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError',
#
#  '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__',
'__package__', '__spec__',
#  'abs', 'aiter', 'all', 'anext', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable',
'chr',
#  'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod',
'enumerate', 'eval',
#  'exec', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help',
'hex',
#  'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max',
'memoryview',
#  'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr',
'reversed', 'round',
#  'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```