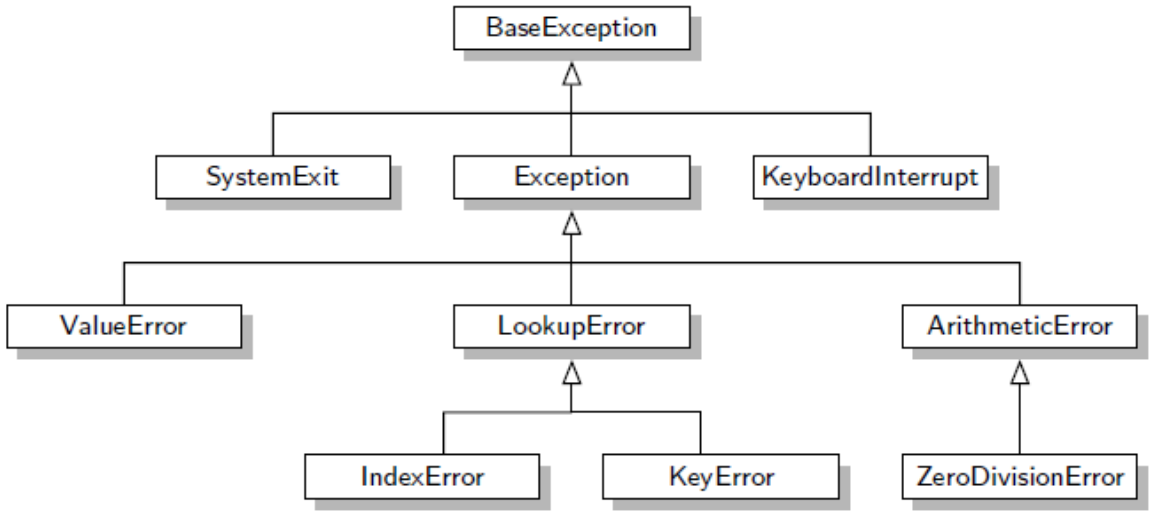
Lesson-15: Exception Handling

What is the difference b/w Compile-Time Error, Logical Error, Run-Time Error:

NOTE:

- Run-time errors are called exceptions. The process of handling run-time errors is called **Exception Handling**.
- The advantage of Exception Handling is the separation of exception detection code from the exception handling code.



Notice The Output:

1)
```
>>> n, x = 5, 6, "Jack"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: too many values to unpack (expected 2)
>>>
```

2)
```
>>> l = [1,2,3]
>>> l[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>>
```

3)
```
>>> b = 10
>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>>
```

4)
```
>>> import subhash
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'subhash'
>>>
```

5)
```
>>> 2 + '3'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

Program 1:

```
import math

#Enter negative number to check what happens
num = int(input('Enter a  number to compute factorial of:'))
print(math.factorial(num))
```

Output:

```
C:\> py -3 Handling_Exception.py

Enter a number to compute factorial of: -5
Traceback (most recent call last):
  File "Handling_Exception.py", line 6, in
<module>
    print(math.factorial(num))
ValueError: factorial( ) not defined for
negative values
C:\>
```

Program 2:

```
import math

#Enter negative number to check what happens
num = int(input('Enter number to compute factorial of:'))
try:
    print(math.factorial(num))
except ValueError:
    print('Cannot compute the factorial of negative numbers')
```

Output:

```
C:\> py -3 Handling_Exception.py
Enter number to compute factorial of:-5
Cannot compute the factorial of negative
numbers

C:\>
```

Program 3:

Output:

```
import math

#Enter negative number to check what happens
num = int(input('Enter number to compute factorial of:'))
valid_input = False

while not valid_input:
    try:
        print(math.factorial(num))
        valid_input = True
    except ValueError:
        print('Cannot compute the factorial of negative numbers')
        num = int(input('Please re-enter: ' ))
```

```
C:\> py -3 Handling_Exception.py
Enter number to compute factorial of:-5
Cannot compute the factorial of negative
numbers
Please re-enter: -5
Cannot compute the factorial of negative
numbers
Please re-enter: 5
120
C:\>
```

Program 4:

```
def getMonth( ):
    month = int(input("Enter current month(1-12):"))

    if month < 1 or month > 12:
                raise ValueError
    return month

valid = False
month_name = ("January","Febraury", "March", "April", "May", "June", "July", "August", "September",
"October", "November", "December")
while not valid:
    try:
        month = getMonth( )
        print("The month you entered is ", month_name[month - 1])
        valid = True
    except ValueError:
        print('Invalid Month Entry\n')
```

Output:

```
C:\> py -3 Handling_Exception.py
Enter current month(1-12):13
Invalid Month Entry

Enter current month(1-12):1a
Invalid Month Entry

Enter current month(1-12):2
The month you entered is  Febraury
C:\>
```

Program 5:

```
def getMonth( ):
    month = int(input("Enter current month(1-12):"))

    if month < 1 or month > 12:
        raise ValueError("Invalid Month Value")
    return month

valid = False
month_name = ("January","Febraury", "March", "April", "May", "June", "July", "August", "September",
"October", "November", "December")
while not valid:
    try:
        month = getMonth()
        print("The month you entered is ", month_name[month - 1])
        valid = True
    except ValueError as err_msg:
        print(err_msg)
```

Output:

```
C:\> py -3 Handling_Exception.py
Enter current month(1-12):1
The month you entered is  January
C:\> py -3 Handling_Exception.py
Enter current month(1-12):23
Invalid Month Value
Enter current month(1-12):1a
invalid literal for int() with base 10: '1a'
Enter current month(1-12):1
The month you entered is  January
C:\>
```

Program 6:

```
#This is a program for self-analysis. Reading and understanding this code slowly and steadily is worth 10
classes of someone explaining this code to you. So try to go through this code and analyze it line by line.
Make use of pen and a paper.

def getFile( ):
    input_file_opened = False
    while not input_file_opened:
        try:
            file_name = input("Enter input file name (with extension): ")
            input_file = open(file_name, "r")
            input_file_opened = True
        except IOError:
            print("Input file not found - please re-renter\n")

    return (file_name, input_file)

def countWords( input_file, search_words ):
```

```
    space = ' '
    num_occurrences = 0
    word_delimiters = (space, "'", ";", ":", ".", "\n", "", '"', "(", ")")

    search_word_len = len(search_word)

    for line in input_file:
        end_of_line = False
        #convert line read to all lower case chars
        line = line.lower()

        #scan line until end of line reached
        while not end_of_line:
            try :
                #search for word in current line
                index = line.index(search_word)

                #if word at start of line followed by a delimiter
                if index == 0 and line[search_word_len] in word_delimiters:
                    found_search_word = True
                #if search word within line, check chars before/after
                elif line[index-1] in word_delimiters and line[index + search_word_len] in word_delimiters:
                    found_search_word = True

                #if found within other letters, then not search word
                else:
                    found_search_word = False

                #if search word found, increment count
                if found_search_word:
                    num_occurrences = num_occurrences + 1

                #reset line to rest of line following search word
                line = line[index + search_word_len:len(line)]

            except ValueError:
                end_of_line = True

        return num_occurrences
#start program

print("This program will display the number of occurrences of a specified word within a given text file\n")

#open file to search

file_name, input_file = getFile( )

#get search word

search_word = input("Enter word to search: ")
search_word = search_word.lower()

#count all occurrences of search word

num_occurrences = countWords(input_file, search_word)

#display results

if num_occurrences == 0:
```

```
    print("No occurences of word", "'" + search_word + "'", "found in file", file_name)
else:
    print("The word", "'" + search_word + "'", "occurs", num_occurrences, "times in file", file_name)
```

Output:

```
C:\> py -3 Handling_Exception.py
This program will display the number of occurrences of a specified word within a given text file

Enter input file name (with extension): hello.txt
Enter word to search: the
The word 'the' occurs 3 times in file hello.txt
C:\>
```

Program 7:

```
class UserDefinedException(Exception): #mandatory to be inherited
    def __init__(self, message):
        self.message = message

try:
    s = input("Enter name:")
    if(s == " "):
        raise UserDefinedException("No Empty String Allowed")
    else:
        print(s)

except UserDefinedException as msg:
    print("Error Message:", msg)
```

Output:

```
C:\> py -3 Handling_Exception.py
Enter name:
Error Message: No Empty String Allowed
C:\>
```

Guess The Output:

1.

```
try:
    a = 5
    b = 5 / 0
except ZeroDivisionError:
    print("ZDE")
except BaseException:
    print("BE")
except Exception:
```

```
    print("E")
```

2.

```
try:
    a = 5
    b = 5 / 0

except BaseException:
    print("BE")
except Exception:
    print("E")
except ZeroDivisionError:
    print(“ZDE")
```

3.

```
try:
    a = 5
    b = 5 / 0
except ZeroDivisionError:
    print("ZDE")
except BaseException:
    print("BE")
except Exception:
    print("E")
else:
    print(“ELSE”)
```

4.

```
try:
    a = 5
    b = 5 / 5
except ZeroDivisionError:
    print("ZDE")
except BaseException:
    print("BE")
except Exception:
    print("E")
else:
    print(“ELSE”)
```

5.

```
try:
    a = 5
    b = 5 / 0
except (BaseException, Exception, ZeroDivisionError):
    print("An Exception Occured”)
```

6.

```
try:
    a = 5
    b = 5 / 5
except:
    print("Hi")
else:
    print("Hello")
```

7.

```
try:
    a = 5
    b = 5 / 5
else:
    print("Hello")
```

8.

```
try:
    a = 5
    b = 5 / 0
except:
    print("Hi")
else:
    print("Hello")
finally:
    print("Namaste")
```

9.

```
try:
    a = 5
    b = 5 / 5
except:
    print("Hi")
else:
    print("Hello")
finally:
    print("Namaste")
```

10.

```
try:
    a = 5
    b = 5 / 0
finally:
    print("Namaste")
```