Q1. Explain why we have to use the Exception class while creating a Custom Exception.

Because Exception class is super class, all Exception comes under a Exception so We Have to inherit Exception for Custom Exception.

For example, You are creating your own list data type in Python that only stores integer. In such cases, it is better to define a custom Exception class that provides a better understanding of the errors that users can understand and relate.

Q2. Write a python program to print Python Exception Hierarchy.

In [1]:

```python
import inspect

def treeClass(cls, ind = 0):
    print ('-' * ind, cls.__name__)
    for i in cls.__subclasses__():
        treeClass(i, ind + 3)

print("Hierarchy for Built-in exceptions is : ")

inspect.getclasstree(inspect.getmro(BaseException))
treeClass(BaseException)
```

```
Hierarchy for Built-in exceptions is :
 BaseException
--- Exception
------ TypeError
--------- FloatOperation
--------- MultipartConversionError
--------- UFuncTypeError
------------ UFuncTypeError
------------ UFuncTypeError
------------ UFuncTypeError
--------------- UFuncTypeError
--------------- UFuncTypeError
------ StopAsyncIteration
------ StopIteration
------ ImportError
--------- ModuleNotFoundError
--------- ZipImportError
------ OSError
--------- ConnectionError
```

Q3. What errors are defined in the ArithmeticError class? Explain any two with an example.

ArithmeticError is thrown when an error occurs while performing mathematical operations. These errors include attempting to perform a bitshift by a negative amount, and any call to intdiv() that would result in a value outside the possible bounds of an int.

zerodivisionerror : You can't divide by zero If you don't specify an exception type on the except line, it will cheerfully catch all exceptions.

In [3]:

```
try:
  arithmetic = 5/0
  print(arithmetic)
except ArithmeticError as e:
  print('You have just made an Arithmetic error',e)
```

```
You have just made an Arithmetic error
```

overflowerror : An OverflowError exception is raised when an arithmetic operation exceeds the limits to be represented. This is part of the ArithmeticError Exception class.

In [6]:

```
import math

try:
     print(math.exp(1000))

except OverflowError as e:
    print("After overflow", e)
```

```
After overflow math range error
```

Q4. Why LookupError class is used? Explain with an example KeyError and IndexError.

LookupError Exception is the Base class for errors raised when something can't be found. The base class for the exceptions that are raised when a key or index used on a mapping or sequence is invalid: IndexError, KeyError. An IndexError is raised when a sequence reference is out of range.

IndexError : His error occurs when an attempt is made to access an item in a list at an index which is out of bounds.

In [10]:

```
try:
    list1 = [2,3,5,4,6,7,6,8,7,9,9]
    list1[12]
except IndexError as e:
    print(e)
```

```
list index out of range
```

KeyError : The Python KeyError is an exception that occurs when an attempt is made to access an item in a dictionary that does not exist.

In [12]:

```
try:
    dict1 = {'key1':'hiren','key2':'khokhar'}
    dict1['key3']
except KeyError as e:
    print("it is key error ",e)
```

```
it is key error  'key3'
```

Q5. Explain ImportError. What is ModuleNotFoundError?

importError : mportError is raised when a module, or member of a module, cannot be imported.

In [16]:

```
import sys
try:
    from cv import math
except Exception as e:
    print("improper module",e)
```

```
improper module No module named 'cv'
```

Q6. List down some best practices for exception handling in python.

-> Catch multiple exceptions in one except block

-> Handling multiple exceptions with one except block

-> Re-raising exceptions in Python

-> Make use of finally clause

-> Use the As keyword to catch specific exception types

In [ ]:

```

```