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


"""Hierarchy and Organization
Error Handling
User-Defined Situations"""
```

```
'Hierarchy and Organization\nError Handling\nUser-Defined Situations'
```

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


def print_exception_hierarchy(exception_class, indent=0):
    print(' ' * indent + str(exception_class))
    for subclass in exception_class.__subclasses__():
        print_exception_hierarchy(subclass, indent + 4)

print_exception_hierarchy(BaseException)
```

```
<class 'BaseException'>
    <class 'Exception'>
        <class 'TypeError'>
            <class 'email.errors.MultipartConversionError'>
            <class 'decimal.FloatOperation'>
            <class 'numpy.core._exceptions.UFuncTypeError'>
                <class 'numpy.core._exceptions._UFuncBinaryResolutionError'>
                <class 'numpy.core._exceptions._UFuncNoLoopError'>
                <class 'numpy.core._exceptions._UFuncCastingError'>
                    <class 'numpy.core._exceptions._UFuncInputCastingError'>
                    <class 'numpy.core._exceptions._UFuncOutputCastingError'>
            <class 'matplotlib.units.ConversionError'>
        <class 'StopAsyncIteration'>
        <class 'StopIteration'>
        <class 'ImportError'>
            <class 'ModuleNotFoundError'>
                <class 'importlib.metadata.PackageNotFoundError'>
            <class 'zipimport.ZipImportError'>
        <class 'OSError'>
            <class 'ConnectionError'>
                <class 'BrokenPipeError'>
                <class 'ConnectionAbortedError'>
                <class 'ConnectionRefusedError'>
                <class 'ConnectionResetError'>
                    <class 'http.client.RemoteDisconnected'>
            <class 'BlockingIOError'>
            <class 'ChildProcessError'>
            <class 'FileExistsError'>
            <class 'InterruptedError'>
            <class 'subprocess.ExecutableNotFoundError'>
            <class 'IsADirectoryError'>
            <class 'NotADirectoryError'>
```

Saved successfully!

```
                <class 'InterruptedError'>
                    <class 'zmq.error.InterruptedSystemCall'>
                <class 'PermissionError'>
                <class 'ProcessLookupError'>
                <class 'TimeoutError'>
                <class 'io.UnsupportedOperation'>
                <class 'signal.itimer_error'>
                <class 'shutil.Error'>
                    <class 'shutil.SameFileError'>
                <class 'shutil.SpecialFileError'>
                <class 'shutil.ExecError'>
                <class 'shutil.ReadError'>
                <class 'socket.herror'>
                <class 'socket.gaierror'>
                <class 'ssl.SSLError'>
                    <class 'ssl.SSLCertVerificationError'>
                    <class 'ssl.SSLZeroReturnError'>
                    <class 'ssl.SSLWantWriteError'>
                    <class 'ssl.SSLWantReadError'>
                    <class 'ssl.SSLSyscallError'>
                    <class 'ssl.SSLEOFError'>
                <class 'urllib.error.URLError'>
                    <class 'urllib.error.HTTPError'>
                    <class 'urllib.error.ContentTooShortError'>
                <class 'gzip.BadGzipFile'>
                <class 'socks.ProxyError'>
```

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


# The ArithmeticError class is a subclass of the Exception class in Python. It serves


try:
    result = 10 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
```

```
 Error: division by zero
```

```
try:
    result = 10.0 / 0.0
except FloatingPointError as e:
    print(f"Error: {e}")
```

Saved successfully!

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-9-8b68f3b81d1f> in <cell line: 1>()
      1 try:
----> 2     result = 10.0 / 0.0
      3 except FloatingPointError as e:
```

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

```
ZeroDivisionError: float division by zero
```

```
# The LookupError class is a base class for exceptions that occur when a key or index
# KeyError:

# KeyError is raised when you try to access a dictionary with a key that doesn't exist


my_dict = {'a': 1, 'b': 2, 'c': 3}

try:
    value = my_dict['d']  # 'd' is not a key in the dictionary
except KeyError as e:
    print(f"Error: {e}")
```

```
 Error: 'd'
```

```
# IndexError:

# IndexError is raised when you try to access an index in a sequence (like a list) tha


my_list = [1, 2, 3]

try:
    value = my_list[3]  # Index 3 is out of bounds for a list of length 3
except IndexError as e:
    print(f"Error: {e}")
```

```
Error: list index out of range
```

```
# Q5. Explain ImportError. What is ModuleNotFoundError?


# ImportError is an exception in Python that is raised when the interpreter encounters
# This can happen for various reasons, such as if the module does not exist, if there
# ModuleNotFoundError is a subclass of ImportError. It is a more specific exception th
```

Saved successfully!

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

```
"""Specific Exception Handling
Avoid Empty except Blocks
Use finally for Cleanup
Avoid Bare except Blocks
Avoid Overusing try-except Blocks"""
```

```
'Specific Exception Handling\nAvoid Empty except Blocks\nUse finally for Cleanup
\nAvoid Bare except Blocks\nAvoid Overusing try-except Blocks'
```