13 feb ass

March 2, 2023

[ ]:
```
Q1. Explain why we have to use the Exception class while creating a Custom␣
 ↪Exception.
Note: Here Exception class refers to the base class for all the exceptions.
```

ANS -

[ ]:
```
You are creating your own list data type in python that only stores interger.␣
 ↪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.using built-in exception classes may
not be very useful in such scenarios
```

[ ]:

[ ]:

[ ]:

[ ]:
```
Q2. Write a python program to print Python Exception Hierarchy.
```

ANS -

[ ]:
```
Exception hierarchy can serve an important purpose: when you create such a␣
 ↪hierarchy the user does not have to
know all the specific exception. Instead, it enough to know and catch the␣
 ↪genreal exception it will make it possible to catch the
all exception that inherit from it . reinforces this recommendation, but warns␣
 ↪not to make such a hierarchy too complicated.
```

[ ]:
```
class LoadError(Exception):
    """General Exception to be used when there is an error with truck load."""
```

[ ]:
```
class Over loadTruckerError(LoadError):
    """The truck is overloaded."""
```

[ ]:
```
class NoLoadOnTruckError(LoadError):
    """The Truck is Empty."""
```

[ ]:

[ ]:

[ ]:
```
Q3. What errors are defined in the ArithmeticError class? Explain any two with␣
 ↪an example.
```

ANS -

[ ]:
```
The Arithmetic error occurs when an error is encountered during numeric␣
 ↪calculations in python. THIS
includes ZeroDivisionError and FloatingPointError. in addition , zero division␣
 ↪error is raised when
when you divide a numeric value by zero
```

[1]:
```
try:
    10/0
except ZeroDivisionError as e:
    print (e)
```

```
division by zero
```

[2]:
```
try :
    20/0
except ZeroDivisionError as e:
    print("This is mt zero division error i am handling" , e)
```

```
This is mt zero division error i am handling division by zero
```

[ ]:

[ ]:

[ ]:
```
Q4. Why LookupError class is used? Explain with an example KeyError and␣
 ↪IndexError.
```

ANS -

[ ]:
```
The LookupError Exception in python forms the base class for all exceptions␣
 ↪that are raised when an index
or a key is not found for a sequence or dictionary respectively. You can use␣
 ↪LookupError exception class
to handle both indexError exception classes.
```

[3]:
```
try :
    a = {1: [1,5,7,4,8] , "key" : "shadab" , "key2" : "zishan" }
    a ["key5"]
except KeyError as e :
```

```
    print ("This is my key error" , e)
```

```
This is my key error 'key5'
```

[5]:
```
try:
    l = [3,8,9,5,6]
    l[10]
except IndexError as e :
    print ("This is my index error" , e)
```

```
This is my index error list index out of range
```

[ ]:

[ ]:

[ ]:
```
Q5. Explain ImportError. What is ModuleNotFoundError?
```

ANS -

[ ]:
```
This error generally occurs when a class cannot be imported due to one of the␣
 ↪following resons:
    They imported class is in a circular dependency. The imported class is␣
 ↪unavailable or was not
    created. The imported class name is misspelled.
```

[ ]:
```
The module not found error is a syntax error that appears when the static␣
 ↪import statement cannot find the
file at the declared path. This commomn syntax error is caused by letter-␣
 ↪casing inconsistencies that are
present in your filename(s) between your repository and local machine , or both.
```

[ ]:

[ ]:

[ ]:
```
Q6. List down some best practices for exception handling in python.
```

ANS -

[6]:
```
#print always a valid msg

try:
    120/0
except ZeroDivisionError as e:
    print("This is mt zero division error i am handling" , e)
```

```
This is mt zero division error i am handling division by zero
```

[7]:
```
# always try to log

import logging
logging.basicConfig(filename = "error.log" , level = logging.ERROR)
try:
    60/0
except ZeroDivisionError as e:
    logging.error("This is my zero division error is am handling {}".format(e))
```

[8]:
```
# always avoid to write a multiple exception handling
try:
    20/0
except FileNotFoundError as e:
    logging.error("This is my file not found error {}".format(e))
except AttributeError as e:
    logging.error("This is my attribute error{}".format(e))
except ZeroDivisionError as e:
    logging.error("This is my zero division error i am handling{}".format(e))
```

[9]:
```
# clean up all resources

try:
    with open("test.txt" , "w") as f:
        f.write("this is my msg to a file")
except FileNotFoundError as e:
    logging.error("this is my file not found{}".format(e))
```

[ ]:
```
# prepare a proper documentation
```

[ ]:
```

```

[ ]:
```

```