Assignment 13: 13 Feb 2023

QI. 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.
Q2. Write a python program to print Python Exception Hierarchy.
Q3. What errors are defined in the ArithmeticError class? Explain any two with an example.
Q4. Why is the LookupError class used? Explain with an example KeyError and IndexError.
Q5. Explain ImportError. What is ModuleNotFoundError?
Q6. List down some best practices for exception handling in python.

QI. 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: Python is known for its readability and user-friendliness; using custom exceptions can help us improve this even more, especially when we are designing our package. If you decide to avoid, at all costs, custom exceptions by using only built-in ones, you risk decreasing Python's readability.

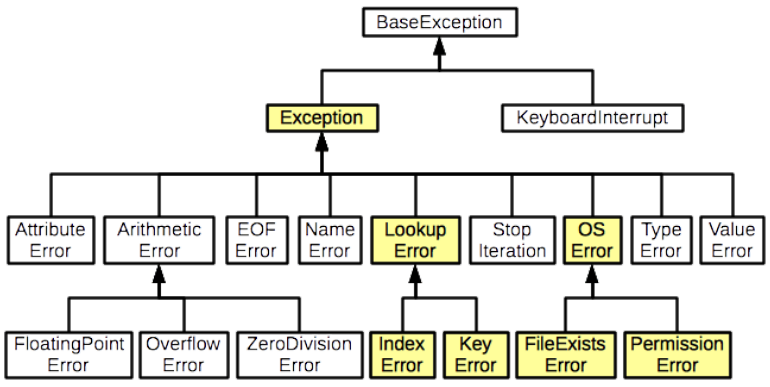
Q2. Write a python program to print Python Exception Hierarchy.

Ans:

```
# import inspect module
import inspect
# our treeClass function
def treeClass(cls, ind = 0):
        # print name of the class
        print ('-' * ind, cls.__name__)
        # iterating through subclasses
        for i in cls.__subclasses__():
                treeClass(i, ind + 3)
print("Hierarchy for Built-in exceptions is : ")
# inspect.getmro() Return a tuple
# of class cls's base classes.
# building a tree hierarchy
inspect.getclasstree(inspect.getmro(BaseException))
# function call
treeClass(BaseException)
```

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

1. FloatingPointError
2. OverFlowError
3. ZeroDivisionError

```
# Example of FloatingPointError
import sys
import math
import fpectl
try:
print 'Control off:', math.exp(700)
fpectl.turnon_sigfpe()
print 'Control on:', math.exp(1000)
except Exception as e:
print e
print sys.exc_type
# Example of OverFlowError
j = 5.0
for i in range(1, 1000):
   j = j**i
   print(j)
```

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

Ans: LookupError Exception is the Base class for errors raised when something can't be found.

```
# Example of LookupError with KeyError
pylenin_info = {'name': 'Lenin Mishra',
               'age': 28,
               'language': 'Python'}
user_input = input('What do you want to learn about Pylenin==> ')
try:
   print(f'{user_input} is {pylenin_info[user_input]}')
except LookupError as e:
   print(f'{e}, {e.__class__}')
# Example of LookupError with IndexError
# lists
x = [1, 2, 3, 4]
try:
   print(x[10])
except LookupError as e:
   print(f"{e}, {e.__class__}")
>>> list index out of range, <class 'IndexError'>
# strings
x = "Pylenin"
try:
   print(x[10])
except LookupError as e:
   print(f"{e}, {e.__class__}")
```

```
>>> string index out of range, <class 'IndexError'>
# tuples
x = (1, 2, 3, 4)
try:
    print(x[10])
except LookupError as e:
    print(f"{e}, {e.__class__}")
>>> tuple index out of range, <class 'IndexError'>
```

Q5. Explain ImportError. What is ModuleNotFoundError?

Ans: ImportError generally occurs when a class cannot be imported due to one of the following reasons: The imported class is in a circular dependency. The imported class is unavailable or was not created. The imported class name is mis-spelled.
ModuleNotError occurs when you're trying to access or use a module that cannot be found.

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

Ans: This list describes best practices for handling exceptions

1. **Never use exceptions for flow-control**
    a. Exceptions exist for exceptional situations: events that are not a part of normal execution.
    b. Consider 'find' on a string returning -1 if the pattern isn't found, but indexing beyond the end of a string raises an exception. Not finding the string is normal execution.
2. **Handle exceptions at the level that knows how to handle them**
    a. The best place is that piece of code that can handle the exception. For some exceptions, like programming errors (e.g. IndexError, TypeError, NameError etc.) exceptions are best left to the programmer / user, because "handling" them will just hide real bugs.
    b. Always ask "is this the right place to handle this exception?" and be careful with catching all exceptions.
3. **Document the exceptions thrown by your code**
    a. thinking about which exceptions your code may throw will help you write better, safer and more encapsulated code